

Веб-Аналитик.ИНФО

www.web-analitik.info

ХОСТИНГ

Выбор хостинга
Шпаргалки по MySQL

ВЕБ-РАЗРАБОТКИ

Обзор сервиса купли и продажи ссылок
TeamCity: распределенное управление сборкой проектов
KenForo: краткий обзор архитектуры нового форумного движка
Подход к оптимизации приложения на примере популярной CMS
Типографика и дизайн

NET

Быстрый запуск блога на базе Orchard
Обзор бета версии ASP.NET MVC 3
Ликвидатор велосипедов
Распараллеливание задач

ЖЕЛЕЗО

Ридер Amazon Kindle 3
Обзор планшета Zenithink zt-180

АВТОРСКАЯ РУБРИКА

Наш опыт выступления на DEMO Fall 2010
Несанкционированный трафик
Синхронизатор данных
Шифрование и эллиптические кривые

ИНТЕРВЬЮ

Андрей Роговой
Компания Parallels



**Журнал выходит ежемесячно
и распространяется бесплатно**

Издательская группа

ООО «Издательство «Стангор»
www.stangor.ru
Журнал Веб-Аналитик.ИНФО
www.web-analitik.info

Издатель и руководитель проекта
Станислав Горнаков sg@stangor.ru

Зам. главного редактора
Игорь Редько ir@stangor.ru

Арт-директор
Светлана Петрова
art-dis@stangor.ru

Редакторы
Сергей Рубан
Ирина Войкова
Андрей Коваль
Василий Сеняевский
Марат Ягудин
Игорь Войков
Игорь Периодов

Корректоры
Ксения Рубова
Максим Веров
Роман Ветрин
Максим Злобин

Авторы
Владислав Раструсный
Андрей Ребров
Владимир Юнев
Максим Петрунин
Денис Баталин
Андрей Цветков
Федор Аксенов
Алексей Дорогов
Василий Аксенов
Александр Ильин
Максим Камашев
Александр Стаханов
Денис Юсупов
Алексей Бузин
Максим Балдин
Ольга Рунова
Алексей Цуцулис
Станислав Сидристый
Сергей Дмитриев
Денис Терехов
Михаил Ивановский
Семен Приходько
Алексей Айларов

Отдел маркетинга и рекламы
Руководитель отдела
Виктор Прудников pr@stangor.ru

Реклама в журнале
siteweb@web-analitik.info

Реклама на сайте журнала
siteweb@web-analitik.info

Издание зарегистрировано в Комитете
Российской Федерации по Печати
Свидетельство № ФС 77 - 39005

Журнал издает
ООО «Издательство «Стангор»

**Для пресс-релизов и информации о
пресс-конференциях**
siteweb@web-analitik.info

Авторам
www.web-analitik.info/authors/

За достоверность рекламной информации
ответственность несут рекламодатели. Ре-
кламные материалы не редактируются и
не корректируются. Редакция ждет ваших
откликов и писем читателей. Фотографии,
рукописи и другие печатные материалы
не редактируются и не корректируются.
При цитировании или перепечатывании
материалов журнала ссылка на сайт www.
web-analitik.info и название журнала Веб-
Аналитик.ИНФО обязательна. Полное или
частичное воспроизведение материала
журналов возможно только с письмен-
ного разрешения Издательства Стангор.
Мнение редакции журнала может не со-
впадать с мнением авторов статей публи-
куемых в журнале. Все товарные знаки
принадлежат их владельцам.

© ООО «Издательство «Стангор»
© WEB-ANALITIK.INFO
© ВЕБ-АНАЛИТИК.ИНФО



ХОСТИНГ

Интервью с Андреем Роговым [Компания Parallels].....	05
Выбор хостинга с упором в облака и с прицелом на развертывание Rails 3 app.....	08
Шпаргалки по MySQL.....	10

КАТАЛОГ

Хостинг-компании.....	12
-----------------------	-----------

ВЕБ-РАЗРАБОТКИ

#404fest 2010: впечатления.....	14
Кадры решают все.....	16
Обзор сервиса купли и продажи ссылки MainLink.RU.....	20
TeamCity: распределенное управление сборкой проектов.....	22
Проклятье веб-сайтов и волшебный UX design.....	25
XenForo: краткий обзор архитектуры нового форумного движка.....	27
Перехват и редактирование файлов http-трафика на примере торрента.....	31
Подход к оптимизации приложения на примере популярной CMS.....	32
Типографика и дизайн.....	35
Разработка web-приложений на языке Common Lisp.....	36
MooTools 1.3.....	38
Inline-blocks: изобретаем снова.....	40

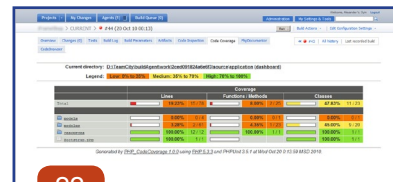


КАТАЛОГ

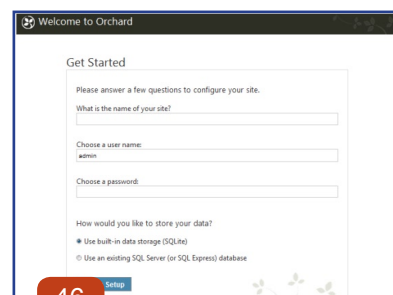
Веб-студии..... **44**

NET

Быстрый запуск блога на базе Orchard..... **46**
 Обзор бета версии ASP.NET MVC 3..... **48**
 Ликвидатор велосипедов: часть 1..... **51**
 Распараллеливание задач. Случай «идеальной параллельности»..... **52**



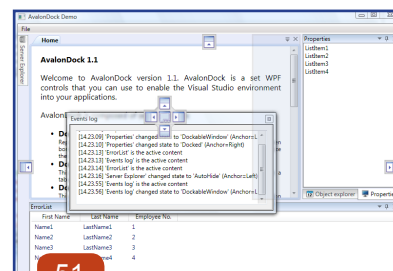
22



46

ЖЕЛЕЗО

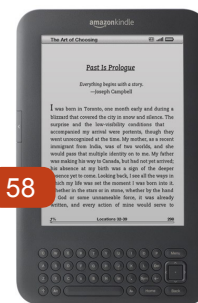
Ридер Amazon Kindle 3..... **58**
 Обзор с точки зрения русскоязычного пользователя..... **58**
 Обзор планшета Zenithink zt-180..... **62**



51

АВТОРСКАЯ КОЛОНКА

Наш опыт выступления на DEMO Fall 2010..... **65**
 Осторожно: несанкционированный трафик..... **67**
 Синхронизатор данных..... **68**
 Шифрование и эллиптические кривые..... **72**



58



62

Хочешь быть в курсе всех событий?
 Подпишись на нашу RSS-ленту новостей или Twitter
 Мы вещаем круглые сутки семь дней в неделю



Свежие новости
 в Twitter'e



RSS-лента
 новостей



Санкт-Петербург
25 по 27 ноября 2010 г.

ХОСТИНГ
Обзор

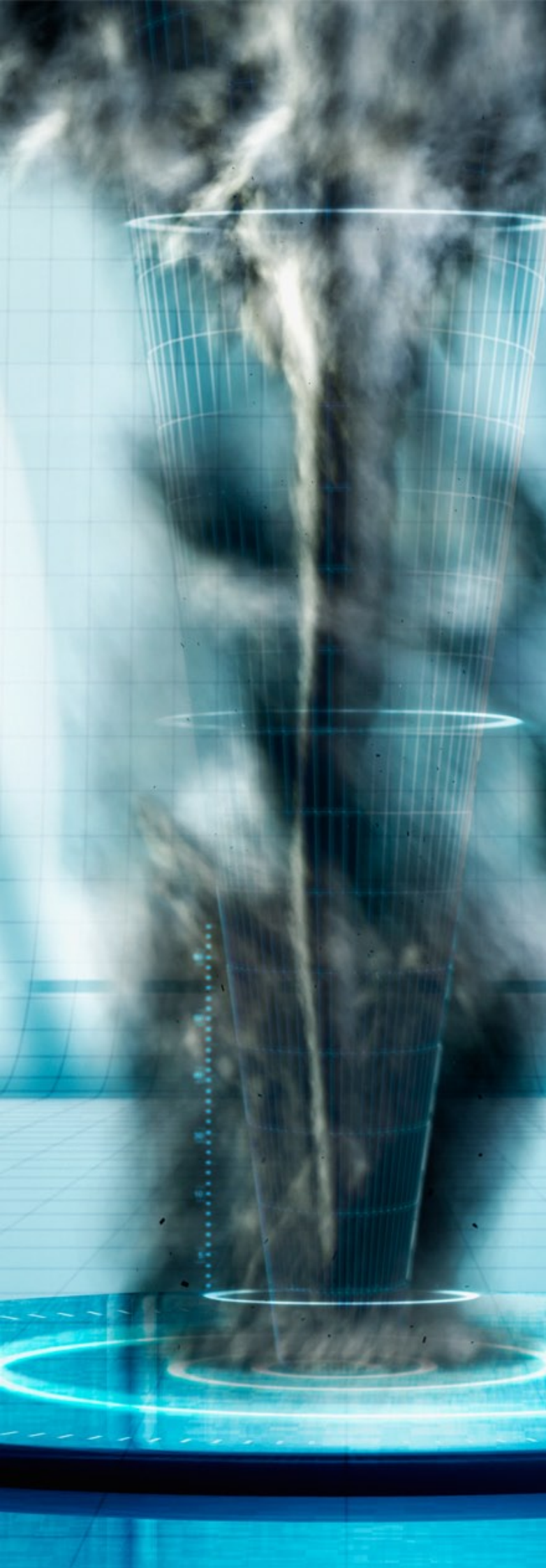
XVI Всероссийский форум хостинг-провайдеров

ХостОбзор.ру
АНО «Координационный центр национального
домена сети Интернет»



Партнеры форума





ХОСТИНГ

**Интервью с Андреем Роговым
[Компания Parallels]**

**Выбор хостинга с упором
в облака и с прицелом
на развертывание Rails 3 app**

Шпаргалки по MySQL



Интервью с Андреем Роговым

[Компания Parallels]

«Хостер может зарабатывать 60-70% с каждого SaaS-приложения»

Сейчас много говорят об облачных вычислениях и в этом контексте упоминают Parallels. Какова ваша роль на облачном рынке?

Сегодня вся шумиха связана либо с облаками для массового потребительского рынка, либо с облаками для крупных предприятий. Гораздо меньше говорится о возможностях, которые облака несут предприятиям малого бизнеса. Между тем, на малый бизнес приходится 40% всех мировых затрат на ИТ (рис. 1), 73,5 млн. малых и средних бизнесов (SMB), плюс еще более 100 млн. так называемых домашних офисов, а также примерно 60 млн. неучтенных малых предприятий (в основном в Китае) – все они потенциальные пользователи облачных сервисов.

Роль Parallels на глобальном и на российском рынках сводится к тому, чтобы помочь хостинг-провайдерам и телекомам. С одной стороны, предложить этому множеству клиентов подходящий набор облачных приложений, а с другой – помочь сервисным компаниям максимально эффективно обслуживать огромное число абонентов. Обе цели достигаются с использованием разрабатываемого в Parallels программного обеспечения. Это хостинговые продукты Parallels Plesk Panel, Parallels Small Business Panel, Parallels Virtuozzo Containers, а также линейка продуктов для автоматизации сервис-провайдеров Parallels Automation. Сервис-провайдеры используют решения Parallels для автоматизации всех этапов предложения, доставки, управления и биллинга облачных услуг. С помощью

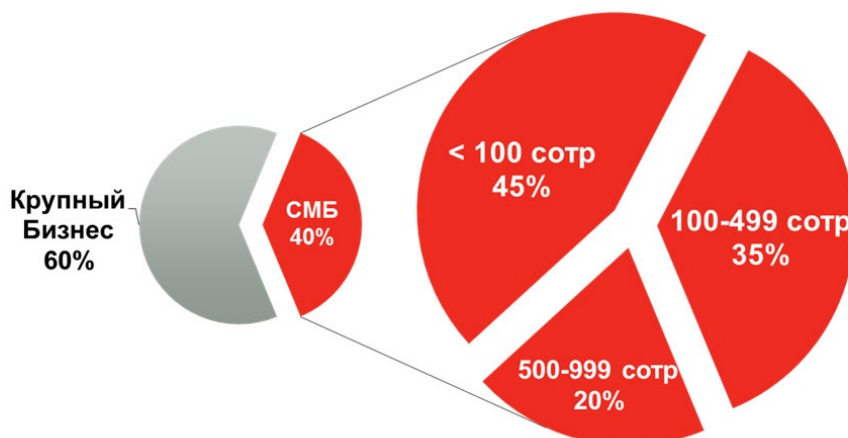
нашего программного обеспечения в мире работает уже более 5 тыс. провайдеров разного масштаба, которые на фоне роста гигантов Google, Microsoft, Amazon получают возможность оставаться конкурентоспособными. Наше ПО ставит их на один уровень с лидерами облачной индустрии.

То есть вы считаете, что хостинг-провайдеры, телекомы имеют хорошие шансы на облачном рынке?

Хостинговые компании оказывают облачные услуги для обеспечения рядовых ИТ-потребностей малого бизнеса начиная с конца 1990-х. На инфраструктуре хостеров в настоящее время ▶

О структуре ИТ рынка или кто покупает больше?

Рис. 1. Соотношение расходов



размещается более 150 млн. доменов малого бизнеса и 50 млн. веб-сайтов. Разве это не большой кусок рынка облачных услуг?

Телекомы, в свою очередь, заслужили доверие SMB, продавая услуги доступа в интернет, услуги телефонии и услуги виртуальных частных сетей. Эти сервисы для SMB уже стали привычными: платить за Сеть, связь, поддержку корпоративной почты, сайта или colocation так же естественно, как платить за аренду офиса, за электроэнергию и другие коммунальные услуги. SMB уже отдали хостерам и телекомам часть важной информации о своем бизнесе. Для них капитальные затраты на покупку внутренних ИТ давно превратились в операционные расходы. Психологически им будет легче купить у хостера, например, хранение данных (storage management) или облачную CRM-систему и «завести» туда клиентскую базу – ведь доверие к сервис-провайдеру с их стороны уже есть. Кстати, поэтому разработчики, которые продают бизнес-софт через собственное облако, имеют меньше шансов на успех. Во-первых, у стартапов (а именно они все чаще выходят на рынок с облачными продуктами), как правило, нет клиентской базы. Во-вторых, к услугам из чужого облака у клиентов меньше доверия – по крайней мере, первое время, пока компания не на слуху.

Вы учитываете системных интеграторов? У них ведь есть ИТ-инфраструктура и клиентская база.

Да, удачные примеры, когда системный интегратор начинал продавать облачные услуги, существуют. Американская компания Apptix ранее обслуживала хостинг-провайдеров и (опосредованно) их клиентов. Но ее владельцы в самом начале облачного тренда разглядели потенциал cloud-бизнеса. Apptix купил нескольких хостинг-провайдеров с их дата-центрами, объединил их, и через платформу Parallels Automation начал продавать SaaS-приложения от Microsoft, ориентированные на малый бизнес. Провайдер верно выбрал нишу: продукты Microsoft пользуются спросом в рамках любой модели потребления. Сейчас у Apptix большая клиентская база, на один только сервис электронной почты подписано более 300 тыс. пользователей.

Из российских компаний можно упомянуть Softline. Поставщик программного обеспечения пока не отказался от профильного бизнеса, но открыл дочерний проект Softcloud – полноценную площадку по продаже облачных приложений.

Представители Parallels во время своих выступлений на конференциях всячески подчеркивают, что SaaS станет для хостеров и телекомов еще и инструментом позиционирования. Каким образом?

В этом нет ничего удивительного. Когда появляется риск снижения выручки на фоне роста частных облачных платформ (Google и т.п.), многие задумываются о диверсификации бизнеса.

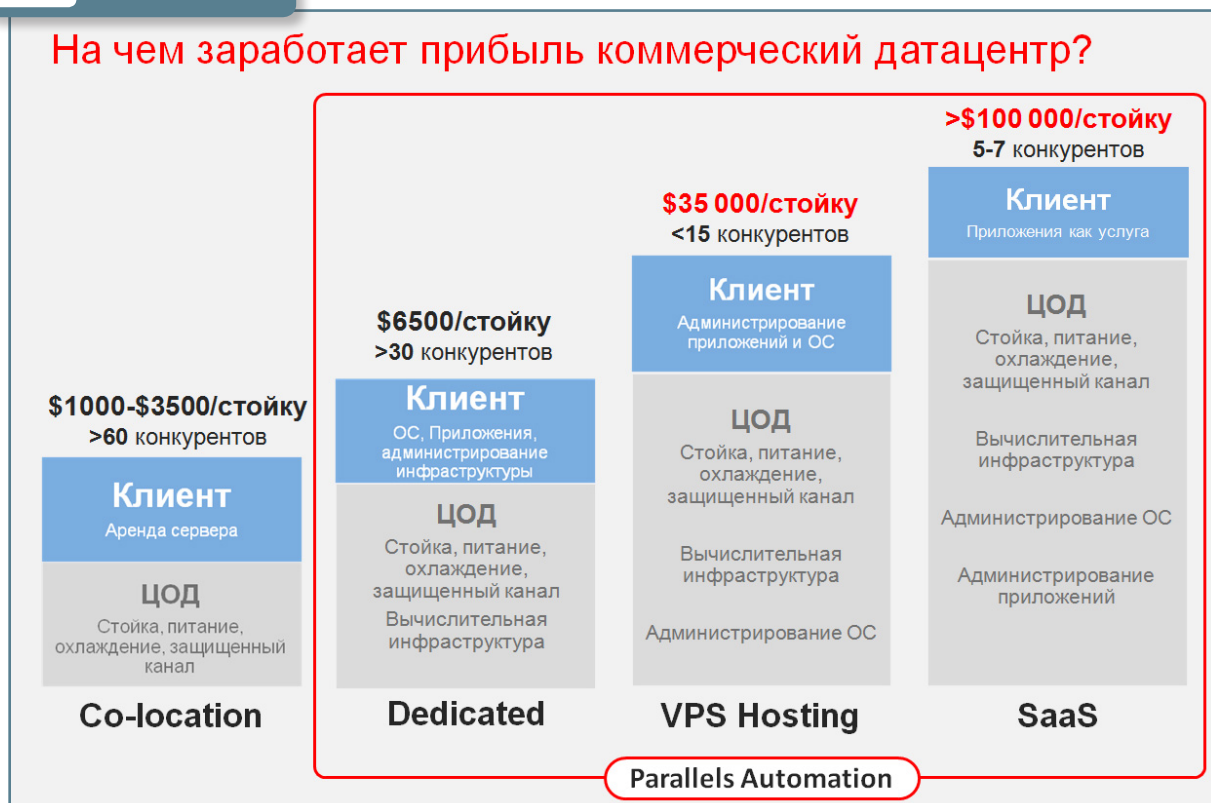
Стандартные сервисы вроде аренды стоек и традиционный виртуальный хостинг уже перестали быть интересными в связи с низкой рентабельностью. (рис. 2) Virtual Private Server (VPS) тоже не обеспечивает уже необходимой прибыли. Новые идеи лежат в области SaaS. По нашим данным, рентабельность ввода новой услуги достигается, если на нее подписать 10-15% клиентов. Если добавлять облачные сервисы в стандартные тарифные планы, можно сегментировать пользователей и выгодно позиционироваться относительно конкурентов.

Вы можете привести примеры компаний из России, которым интересна аналогичная смена профиля?

Теоретически это интересно независимым разработчикам ПО. Например, российский вендор Asoft наряду с коробочной версией, продвигает облачный вариант своей CRM-системы через проприетарное облако. Плюс разработчик стремится попасть в канал продаж сервис-провайдеров: для этого Asoft запаковал CRM-систему в Application Packaging Standard (APS) и выставил ее в маркетплейсы нескольких известных cloud-провайдеров. Вендор сегментирует аудиторию, чтобы не допустить оттока пользователей, он предлагает каждому клиенту получать сервис тем способом, который он считает самым удобным – из коробки, из своего облака или от сервис-провайдера.

Другой пример: российский хостинг- ➤

Рис. 2. Прибыль со стойки



провайдер КОМТЕТ, который предпочел самостоятельно начать локализацию популярных сервисов для включения в свои тарифные планы. При покупке услуг хостинга сайта его клиент получает собственно хостинг и необходимый для создания сайта комплект софта, запаксованный по спецификациям APS. Веб-приложения доставляются и устанавливаются бесплатно круглосуточной службой технической поддержки.

Кстати, на сайте Parallels уже есть краткий анонс Plesk 10. Что принципиально нового можно ожидать от этого решения?

Основополагающее изменение заключается в том, что с некоторых пор при разработке функциональности продуктов для сервис-провайдеров нас интересует только непосредственное их влияние на выручку нашего партнера. Если требуемые «галочки» или новая строка в интерфейсе интересны ограниченному числу хостеров и не имеют для них экономических последствий, мы не будем ничего существенно менять в продукте.

Мы, естественно, знаем, что первые версии продуктов в линейке Parallels Plesk Panel не отличались совершенством. Но ситуация изменилась. Изменилась она, прежде всего в наших собственных головах и в отношении к продуктам автоматизации со стороны самой компании, поскольку они (в том числе и Parallels Plesk Panel) приносят Parallels 65-70% доходов.

Если вкратце говорить о функционале десятой версии, то он расширен за счет ряда важных дополнений. Панель дает возможность в рамках одного аккаунта назначать разным доменам индивидуальные сервисы и ре-

сурсы. Интерфейс панели разработан с учетом замечаний системных администраторов. Добавлена функция мониторинга нагрузки на сервер. Пожалуй, самый существенный шаг сделан в области автоматизации бизнес-процессов. Десятая версия получила полный набор бизнес-инструментов – биллинг, выставление счетов, настройку витрины SaaS-приложений, возможность регистрации доменов, SSL-сертификаты.

Вы упомянули про схему включения приложений ISV в маркетплейсы провайдеров. Как разработчик и провайдер будут делить доходы? Сколько может заработать хостер?

Все компании абсолютно разные, у каждой свои представления о том, как должны распределяться доходы между вендором и хостером. Мне известны случаи, когда разработчик готов отдавать провайдеру до 70% от цены приложения, указанной в маркетплейсе. Разработчики сознательно действуют так агрессивно. Они понимают, что более-менее значительные продажи возможны только через канал сервис-провайдера и что хостера можно заинтересовать только большими процентами. В отношении телекомов это правило справедливо вдвойне. Их не заинтересует рентабельностью в 20% – они привыкли зарабатывать гораздо больше. Не нужно думать, что хостеры и телекомы это какие-то монстры с неумными аппетитами. В обычной, «коробочной» рознице ситуация аналогичная. Розничные сети заинтересованы в продуктах вендора, только если вендор дает большой процент с продаж, либо если продукт вендора по-настоящему уникален и пользуется огромным спро-

сом, либо если вендор готов заплатить «входной билет».

Российские хостеры хотят видеть у себя крупных клиентов, поскольку малый бизнес и домашние офисы – это всегда счета на маленькие суммы и, соответственно, высокие накладные расходы на ведение клиентов. Внедрение облачных услуг только усугубит ситуацию: документов и счетов станет еще больше. Как быть?

Модель потребления SaaS формировалась с оглядкой на обслуживание большого числа малых клиентов. Коммерческая успешность облачного провайдера зависит от количества абонентов. В этой связи важно построить и оптимизировать команду и инфраструктуру поставщика услуг на обслуживание максимального числа клиентов за минимальное время. Практика показывает, что «бутылочным горлышком» в подобных ситуациях становятся люди. Администраторы увязают в огромном количестве пустяковых, нудных манипуляций с выставлением счетов, проведением платежей, подключением услуг и еще десятком функций. Выходом станет автоматизация рутинных бизнес-процессов.

Пока в данной области есть всего одна типовая разработка, рассчитанная на автоматизацию бизнеса сервис-провайдеров. Это Parallels Automation. Конкуренцию ей составляет самодельный софт, написанный специалистами хостинговых компаний. Но доморощенные решения имеют ряд ограничений, связанных, прежде всего, с временными затратами на разработку и с невозможностью применить их в рамках других компаний.

Намного ли отстала Россия от облачного Запада?

Возможно, в России не так широко, как в Америке или Европе, используются облачные сервисы, но это вопрос времени и восприятия самой темы ИТ-услуг в аренду. Я думаю, российские SMB готовы потреблять облачные услуги, но пока мало предложений адекватного сервиса. Структура спроса в России во многом похожа на западную. Потребителей интересуют базовые услуги – присутствие в интернете (хостинг сайта), коммуникация (электронная почта, VoIP, сервисы мгновенного обмена сообщениями), специализированные приложения (CRM-системы, ERP-системы, бухгалтерский и кадровый софт).



Автор

Максим Петрунин
Компания «Таксовик»
г. Москва
taxovik.ru

Выбор хостинга с упором в облака и с прицелом на развертывание Rails 3 app

Несмотря на наличие «Rails 3» в названии, и рассмотрении в теле статьи специфичных для Rails 3 аспектов, она может быть интересна всем остальным из-за освещения аспектов общего характера.

Мои изначальные требования к хостингу:

- Возможность развернуть Rails 3 приложение.
- В моем приложении существуют несколько «роботов», связанных на сгон, поэтому сгон нужен.
- В будущем понадобится возможность работы с поддоменами.
- Пригодится SSL.
- Как любой стартапер, я ожидаю как рост посещаемости, так и пиковые высокие нагрузки. Хочется, чтобы хостинг мог легко их выдержать, поэтому прицел именно в облака.
- По возможности низкая цена. Я собираюсь хостить два независимых проекта на rails 3, по одному из них не планируется почти никакой посещаемости, там интересует минимальная цена среди возможных. Но мне лень разбираться в тонкостях двух хостингов, поэтому под два проекта будет выбираться один хостинг-провайдер.
- Надежность – SLA и uptime 99.9%.

Поскольку рассматриваются зарубежные хостинги, существенным критерием становится время доступа к их серверам, как из мира, так и из России. Оценка этого параметра производилась с помощью сервиса host-tracker.

Время доступа – это время загрузки главной страницы хостинга без лишней обвески (картинки, стили, скрипты), со всех возможных серверов. Добавка «по России» означает из датацентров Москвы и Питера (там иногда выскакивает Уфа). Добавка «по миру» – из датацентров мира (Европа, Америка в основном).

Понятно, что критерий не объективный. Влияет куча факторов: размер страницы (обычно копейки), кэшированность, нагрузка в данный конкретный момент. Но если хостер не оптимизировал главную, забыл закэшироваться или не сбалансировал нагрузку, то на за чем он такой нам нужен? Что касается размера страницы, то тут, конечно, как повезет. На всякий случай я пишу его в скобках.



1gb.ru рассматривается как эталонный хостинг. Потому что ровно на нем хостится наша компания Таксовик. С этим хостингом мы успешно выдержали Вести-, Хабра-, Экспер-, Рамблер- и прочие эффекты. Так же это эталон скорости доступа по России; в отличие от всех остальных тестов скорости, здесь будет скачиваться главная страница Таксовика.

Скорость taxovik.ru (10кб):

host-tracker.com/check_res_ajax/5512923-0/
0.85 сек – по миру, < 0.1 по России

Цены: VDS (при оплате на год), VZ-Профи, Linux/OpenVZ, 768 Мб, 20 Гб, 2400МГц. – 490 руб. Одна из самых лучших цен, мое мнение.

Личный опыт:

- + Хорошие сервера, куча предустановленных функций (типа email-, mysql- и т. п.)
- + Отличная панель управления
- + Хорошие и уменьшающиеся со временем цены
- + Отзывчивая, вменяемая, русскоговорящая техподдержка. Выполняют все почти все просьбы, которые могут возникнуть.
- + Из москвы пинг 4 мс.
- Никакого тебе SLA и uptime 99.9% (т. е. не более 8-9 часов простоя в год). Лежит примерно в 2-3 раза больше. Очень портит впечатление
- Старенький Gentoo из-за чего не смог воспользоваться ebuild'ом: Ruby EE + Passenger + Nginx

Выводы: Очень неплохой русский хостинг.

HostingRails.com

Тест скорости (32кб): host-tracker.com/check_res_ajax/5512973-0/
В среднем по Москве/Питеру около 0.7 секунды, что нормально. В среднем по миру ~0.5.

Цены VPS: \$4-6 – малая нагрузка, ~\$7-10 – средняя

VDS: Virtuozzo Virtualization Technology, 256 MB, 30 GB – \$27.95.

Достоинства/недостатки:

- + VPS заточен под Rails, в частности под Rails 3
- + Нормальная цена за VPS
- VDS дорого
- Нет облаков

Выводы: сойдет для малобюджетного Rails-проекта.



Говорят Rackspace этот тот же SliceHost, только с ценами пониже. Утверждается, что там даже одни сервера.

Тест скорости (30кб): host-tracker.com/check_res_ajax/5512968-0/
по России: 1 сек, по миру 0.55 сек.

Цены: 512 MB, 20 GB, 20 Mbps – \$21.90 (минималка, 256 RAM для второго сайта от \$11).

Достоинства/недостатки:

- + Нормальное полноценное облако
- + Нормальные цены для облака
- + Хвалят, что адский приток посещаемости в 40к посетителей стоил всего 7 дополнительных баксов
- + Есть приложение для iPhone ;)
- + Говорят очень удобная веб-панель управления и высокая скорость развертывания. Т. е. от окончания регистрации до ssh ~1 минута
- Не нашел автомасштабирования нагрузки
- По принципу VDS в отличие от VPS и heroku настраивать все надо самому

Выводы: лучший из вариантов по соотношению цена/качество. ►



Тест скорости (10кб): host-tracker.com/check_res_ajax/5512935-0/
Россия – 0.75 сек, Мир – 0.58 сек

Имеется система преднастроенных контейнеров, к Media-temple есть контейнер по типу VDS для Rails (есть и для Django и еще кучи всего). Если большие нагрузки на базу, то подключается контейнер отдельно для MySQL. Все по 20\$.

По заверениям тех. поддержки (личная переписка) Ruby-контейнеры не поднимут Rails 3. У них там настроен mongrel, а он, как известно не дружит с Rails 3.

Цены на контейнеры:

256MB Lite – \$20

512MB Pro – \$40

цены на VDS: 512MB, 20GB – 50\$

Достоинства/недостатки:

- + Auto-Scaling MySQL, хотя тут пишут, что SQL не масштабируешь
- + Система контейнеров
- + Фактически облако
- + Надежное решение, но дорогое
- + Написал письмо с вопросом про Rails 3, ответ пришел очень быстро
- + Существует плагин для автомасштабирования нагрузки
- Цены
- Не работают с Rails 3 на уровне контейнеров

Выводы: выглядит классно, мне понравилось, но нет поддержки Rails 3, для Rails 3 надо брать VDS за 50\$.



Тест скорости (10кб): host-tracker.com/check_res_ajax/5513147-0/
Россия – 0.5 сек, мир – 0.48 сек (круто, ничего не скажешь).

Heroku – это облако по типу PaaS (Platform as a Service). Базируется на Amazon Web Services.

Цены, минималка: 15\$ за 1 dyno, 1 dyno по обещаниям обрабатывает 10-100 запросов в секунду.

Достоинства/недостатки:

- + Облачнее не бывает
- + Адская заточка под Rails
- + Классная скорость доступа
- + Есть бесплатный тариф, но он пригодится, разве что для домашней странички моей собаки
- За все берут денежку, к примеру за почасовой (а не подневной) cron – \$3, за SSL – 5\$, за поддержку доменов вида *.mydomain.ru – 5\$.
- Дорого
- Cheap and dealy, которые хостятся на хероку, писали, что упали от рамблерэффекта. Как можно упасть от рамблерэффекта с сайтом их типа на облачном хостинге, заточенном под такого рода задачи для меня загадка, но сам факт примечателен
- Нет родной поддержки MySQL. Либо мигрирую на PostgreSQL, либо плати от \$25 за Amazon RDS
- Задал им вопрос про cron через форму на сайте пару дней назад, до сих пор не ответили.

Выводы: дорого, но круто :)



Тест хосттрекера выдает адскую статистику (20.09.10 ~21:00, 11 кб): host-tracker.com/check_res_ajax/5536253-0/
Мир – 20.32 сек, Россия – 20 сек.

Цены, которые мне были озвучены, отличаются от заявленных на сайте примерно до двух раз в лучшую сторону:

VDS (при оплате на год): RAM 512, 30 GB – 510 р., RAM 1024M – 935 р., RAM 2048 – 1955 р.

Из заявленных преимуществ: зарубежный датацентр с русскоязычным саппортом с хорошими каналами на РФ (пинг 8мс) и остальной мир, возможность работы с юр. лицами, сертификат Tier2 (uptime 99,749%, 22 часа простоя).

Выводы: цены неплохие, но похуже чем 1gb, подходит тем, кто хочет непременно зарубежный хостинг (боится маскишоу, например), но с хорошим каналом до России. Но судя по тормозам главного сайта, тоже не без проблем.



Вот на них бы хотелось вылить немного поноса. Несмотря на то, что justhost.com крупная американская хостинговая компания с кучей звездочек, которая заявляет, что поддерживает Ruby on Rails на своем VPS, она доставила массу хлопот мне абсолютно без результата.

Цена VPS: \$6.95 в месяц.

Недостатки:

- 1) Ужасная cpanel.
- 2) Около восьми часов включали мне Rails и Shell. То есть я оплатил, решил выкатиться, а не могу. Надо делать ручную заявку в поддержку.
- 3) Включение ssh у них стоит \$19.95 (!!!). Иначе только через cPanel. Кстати, у них написано: «No Hidden Fees».
- 4) У них нет git, а так же закрыт доступ к gcc. Я не смог подключить gem'ы, которые требуют компилирования модулей.
- 5) Качество поддержки как будто кричит: «купи платную поддержку», более того, они пишут об этом в каждом втором письме в подписи.
- 6) Когда я попросил их скомпилировать мне gem, они предложили скомпилировать мне ruby.
- 7) Решали проблемы по 1-3 дня на одну проблему.
- 8) Они так и не смогли до конца настроить сервер на работу с Rails 3 приложением. По умолчанию у них стоит именно Rails 3.
- 9) На все попытки настройки ушло более 14 дней.
- 10) Обещали ответить на последний вопрос за 24 часа, ответа нет уже более 72.



Тест скорости (14 кб): host-tracker.com/check_res_ajax/5541635-0/
Россия – 0.36-0.59 сек, мир – 0.95 сек. Скорость на Россию поражает.

Цены: AMD Athlon 64 3700+, 1024 MB DDR400 RAM, 2 x 160 GB SATA HDD (Software-RAID 1) – 29 евро в месяц. **Мои вы-**

Выводы: выделенный сервер по цене виртуального без русской поддержки с отличным каналом на Россию. ■

Шпаргалки по MySQL

Автор

Максим Камашев
Компания «Нотамедиа»
г. Москва
root@uk0.us
<http://uk0.us/>

Часто при разработке сайта можно заметить, как наступают на одни и те же грабли, проектируя базу данных. И ведь порой совсем не нужно всю информацию хранить в голове. Порой достаточно посмотреть на шпаргалку, маленькую заметку и все сразу становится ясно. Именно такими шпаргалками я хочу поделиться.

Работа с бекапами

Делаем бекап:

```
mysqldump -u USER -pPASSWORD DATABASE > /path/to/file/dump.sql
```

Создаем структуру базы без данных:

```
mysqldump --no-data -u USER -pPASSWORD DATABASE > /path/to/file/schema.sql
```

Если нужно сделать дамп только одной или нескольких таблиц:

```
mysqldump -u USER -pPASSWORD DATABASE TABLE1 TABLE2 TABLE3 > /path/to/file/dump_table.sql
```

Создаем бекап и сразу его архивируем:

```
mysqldump -u USER -pPASSWORD DATABASE | gzip > /path/to/outputfile.sql.gz
```

Создание бекапа с указанием его даты:

```
mysqldump -u USER -pPASSWORD DATABASE | gzip > `date +/path/to/outputfile.sql.%Y%m%d.%H%M%S.gz`
```

Заливаем бекап в базу данных:

```
mysql -u USER -pPASSWORD DATABASE < /path/to/dump.sql
```

Заливаем архив бекапа в базу:

```
gunzip < /path/to/outputfile.sql.gz | mysql -u USER -pPASSWORD DATABASE
```

или так:

```
zcat /path/to/outputfile.sql.gz | mysql -u USER -pPASSWORD DATABASE
```

Создаем новую базу данных:

```
mysqladmin -u USER -pPASSWORD create NEWDATABASE
```

Удобно использовать бекап с дополнительными опциями, например `-Q -c -e`, т. е.:

```
mysqldump -Q -c -e -u USER -pPASSWORD DATABASE > /path/to/file/dump.sql
```

где:

- `-Q` оборачивает имена обратными кавычками
- `-c` делает полную вставку, включая имена колонок
- `-e` делает расширенную вставку. Итоговый файл получается меньше и делается он чуть быстрее

Для просмотра списка баз данных можно использовать команду:

```
mysqlshow -u USER -pPASSWORD
```

А так же можно посмотреть список таблиц базы:

```
mysqlshow -u USER -pPASSWORD DATABASE ►
```

Для таблиц InnoDB следует добавлять параметр `--single-transaction`, что гарантирует целостность данных бекапа. Для таблиц MyISAM это не актуально, т. к. они не поддерживают транзакционность.

Общие факты

- Полезно под каждую базу на боевом сервере создавать своего пользователя.
- Кодировка базы может быть любой, если она UTF8.
- В большинстве случаев лучше использовать движок InnoDB.
- Используя php, стоит забыть про сильно устаревшее расширение mysql, и по возможности использовать расширение pdo или mysqli.
- Новую копию MySQL всегда можно настроить и оптимизировать.
- Без особой на то нужды, не стоит открывать MySQL наружу. Вместо этого можно сделать проброс портов:

```
ssh -fNL LOCAL_PORT:localhost:3306 REMOTE_USER@REMOTE_HOST
```

Работа с данными



Числа

- На 32-битных системах практически нет смысла ставить для типа `INTEGER` свойство `UNSIGNED`, так как такие большие числа в php не поддерживаются. Но на 64-битных системах, php поддерживает числа огромного размера, вплоть до `BIGINT` со знаком.
- Связанные таблицы через «foreign keys» (да и без внешних ключей) должны иметь полное сходство по структуре ключей. Т. е. если у нас на одной таблице для поля указано «`INTEGER UNSIGNED DEFAULT 0 NOT NULL`» то и на другой должно быть указано аналогично.
- Для хранения булевых значений, нужно использовать `TINYINT(1)`.
- А деньги лучше хранить в `DECIMAL(10, 2)`, где первое число обозначает количество всех знаков, включая запятую, а второе – количество знаков после запятой. Итого, у нас получится что `DECIMAL(10, 2)` может сохранить 9999999,99.



Строки

- В старых версиях (до 5.0.3) `VARCHAR` была ограничена 255 символами, но сейчас можно указывать до 65535 символов.
- Помните, что тип `TEXT` ограничен только 64 килобитами, поэтому чтобы сохранять «Войну и Мир» пользуйтесь «`LONGTEXT`».
- Самая правильная кодировка для вашей БД – UTF8.



Даты

Пожалуйста, не забывайте, что:

- `DATE`, `TIME`, `DATETIME` – выводятся в виде строк, поэтому поиск и сравнение дат происходит через преобразование,
- `TIMESTAMP` – хранится в виде `UNIX_TIMESTAMP`, и можно указать автоматически обновлять колонку,
- Сравнивая типы данных `DATETIME` и `TIMESTAMP`, не забывайте делать преобразование типов, например:

```
SELECT * FROM table WHERE `datetime` = DATE(`timestamp`)
```



Перечисления

- Для небольших списков, для которых совсем не хочется создавать отдельную таблицу, можно использовать тип `ENUM`.
- Правильно пишется так: `ENUM('мама', 'мыла', 'раму')`.
- Можно ставить значение по умолчанию как и для любой строки.
- В базе, поле с перечислением хранится как число, поэтому скорость работы потрясюще высокая.
- Количество таких перечислений в списке может быть до ~ 65 тысяч.



Отладка

- Если запросы тормозят, то на время можно включить лог для медленных запросов в файле конфигурации. Обычно, это `/etc/mysql/my.cnf`,
- А потом оптимизировать запросы с помощью оператора `EXPLAIN`,
- И наблюдать за запросами удобно через программу `mytop`. ■



Каталог Хостинг-компаний

Superhost.com.ua

Организация: SARBASH Lab.
Страна: Украина
Город: Донецк
Дата-центр: McLean data center, DCA2
Сайт: <http://superhost.com.ua>
Телефон: 380622521461
E-mail: info@superhost.com.ua

Инстант Хостинг

Организация: Inst-Host.ru
Страна: Germany
Город: Nuremberg
Дата-центр: HETZNER Online
Сайт: <http://inst-host.ru/>
E-mail: reg@inst-host.ru

xTremeHost.ru

Организация: xTremeHost.ru
Страна: Казахстан
Город: Кокшетау
Дата-центр: США
Сайт: <http://xtremehost.ru>
Телефон: +77770397355
E-mail: support@xtremehost.ru

КОМТЕТ

Организация: ООО «КОМТЕТ»
Страна: Россия
Город: Москва, Пенза
Дата-центр: Москва, М10
Сайт: <http://komtet.ru>
Телефон: 8-800-200-2511
E-mail: support@komtet.ru

Мирекс

Организация: Мирекс
Страна: Россия
Город: Красноярск, Москва
Дата-центр: Квиклайн, М9, Веб Дата-центр -
Дата Центры в Москве.
Сайт: <http://mirex.su>
Телефон: 8 (391) 215-29-35
E-mail: hosting@mirex.su

Партизанск Телеком

Организация: ИП Нецадим А.С.
Страна: Россия
Город: Партизанск
Дата-центр: Burst, Netdirect, SuperB
Сайт: www.p-telecom.ru
Телефон: +7 42363 69663
E-mail: company@p-telecom.ru

Хостинг Хутор

Организация: HostingHutor.com
Страна: Украина
Город: Одесса
Дата-центр: Hetzner, Utel, Wnet
Сайт: <http://hostinghutor.com>
Телефон: +3 048 7017698
E-mail: [info\[at\]hostinghutor.com](mailto:info[at]hostinghutor.com)

HOST-FOOD

Организация: ИП Седов А.Н.
Страна: Россия
Город: Саратов
Дата-центр: «Инфобокс» в СПб и
«Datacheap» в Мск
Сайт: <http://www.host-food.ru>
Телефон: +7 927-279-3758
E-mail: support@host-food.ru

Казахстанский Хостинг

Организация: Уральск Web
Страна: Казахстан
Город: Уральск
Дата-центр: АО «Казахтелеком», г.Уральск
Сайт: www.solidhost.kz
E-mail: support@solidhost.kz

InstantHost

Организация: Instant
Страна: Россия
Город: Братск
Дата-центр: Майами
Сайт: <http://instanthost.biz>
Телефон: +79501171041
E-mail: iskander019@ya.ru

Iphosters

Организация: Iphosters.com
Страна: USA
Город: New York
Дата-центр: New York
Сайт: <http://iphosters.com>
Телефон: +1-877-745-6763
E-mail: support@iphosters.com

KILOBYTE

Организация: KILOBYTE.COM.UA
Страна: Украина
Город: Киев
Дата-центр: собственная площадка
Сайт: <http://kilobyte.com.ua>
Телефон: +380443600206
E-mail: [sales\[at\]kilobyte.com.ua](mailto:sales[at]kilobyte.com.ua)

StoreHost.ru

Организация: СторХост Лтд
Страна: Россия
Город: Москва
Дата-центр: «Оверсан-Меркурий»
Сайт: <http://storehost.ru>
Телефон: +7 495 9164790
E-mail: support@storehost.ru

SVAI

Организация: ФОП Полудень В.М.
Страна: Украина
Город: Черкасы
Дата-центр: Mhost, Megastyle, Ukrtelecom
Сайт: svai.net
Телефон: +38 0472 569026
E-mail: billing@svai.net

Amberhost.net

Организация: Amberhost.net
Страна: Украина
Город: Киев
Дата-центр: Мхост
Сайт: <http://amberhost.net>
E-mail: support@amberhost.net

KirHost

Организация: СПД Кунев Д.В.
Страна: Украина
Город: Кировоград
Дата-центр: Воля, SteepHost
Сайт: www.kirhost.com
E-mail: roman@kirhost.com

Luckhost.net

Организация: ТОО «КУЛЬСЫН»
Страна: Казахстан
Город: Тараз
Дата-центр: Hetzner
Сайт: <http://luckhost.net/>
E-mail: info@luckhost.net

ProGoldHost.Net

Организация: ProGoldHost.Net
Страна: Россия
Город: Ставрополь
Дата-центр: США - DataJersey / Германия -
Hetzner
Сайт: <http://progoldhost.net>
E-mail: info@progoldhost.net

ISPserver

Организация: ISPserver
Страна: Россия
Город: Москва, Иркутск
Дата-центр: Веб-дата центр (WebDC.ru)
Сайт: www.ispserver.com
E-mail: sales@ispserver.com

Компания iHead

Организация: Компания iHead
Страна: Россия
Город: Киров
Дата-центр: Multinex, Agava
Сайт: www.ihead.ru
E-mail: info@ihead.ru

FirstVDS

Организация: ЗАО «Первый»
Страна: Россия
Город: Москва, Иркутск
Дата-центр: Веб-дата центр (WebDC.ru)
Сайт: www.firstvds.ru
E-mail: sales@firstvds.ru

Добавить хостинг-компанию в каталог бесплатно

ДОБАВИТЬ

Публикуемые в журнале компании взяты из каталога сайта издания. Данные о компаниях публикуются по мере их добавления в каталог. Добавить в каталог данные о компании может любая организация. Добавление данных в каталог сайта осуществляется бесплатно на добровольной основе самой организацией. Редакция журнала Веб-Аналитик.ИНФО не занимается добавлением данных в каталог сайта и не несет ответственности за предоставляемую информацию от организаций.



ВЕБ-РАЗРАБОТКИ

#404fest 2010 - впечатления

Кадры решают все

**Обзор сервиса купли и продажи
ссылок MainLink.RU**

**TeamCity: распределенное
управление сборкой проектов**

**Проклятье веб-сайтов
и волшебный UX design**

**XenForo: краткий обзор архитекту-
ры нового форумного движка**

**Перехват и редактирование
файлов http-трафика
на примере торрента**

**Подход к оптимизации приложе-
ния на примере популярной CMS**

Типографика и дизайн

**Разработка web-приложений на
языке Common Lisp**

MooTools 1.3

Inline-blocks: изобретаем снова



#404fest 2010

Впечатления

Автор

Андрей Ребров
г. Самара
andrebrov.blogspot.com
andrebrov@gmail.com

В начале октября в Самаре прошел фестиваль веб-деятелей – Фестиваль 404. Имея опыт посещения подобных мероприятий, я не стал приезжать к началу регистрации, а приехал уже почти под ее конец. Тем не менее, мне и другим участникам пришлось прождать еще полчаса, пока все не началось. Именно эта задержка привела к тому, что многие доклады прошли скомкано, были сдвинуты или лишены вопросов зрительского зала.

Но шоу началось, и началось оно со вступительного слова Димы Фитискина и Дениса Картунова. Они рассказали об истории мероприятия, о его символе и немного обо всем сразу, после них пошли доклады.

День первый

Начал первый день замечательный человек из Microsoft – Билл Бакстон, руководитель отдела исследований. На мой взгляд, это был самый лучший доклад на всей конференции, потому что он был максимально живой, веселый, интересный, содержательный и информативный. Билл вел свой доклад на английском языке, но вся аудитория была с ним на одном дыхании и все понимала на лету. Билл говорил о многих вещах, и вот те ключевые моменты, которые выцепил я:

- надо максимально использовать имеющий опыт, чтобы переносить его в создаваемые интерфейсы;
- будущее будет не через 100 лет, оно наступит в следующем году, поэтому надо мыслить на несколько шагов вперед;
- все вокруг нас способно становиться (и уже становится) интерактивным, пора этим пользоваться;
- прототипирование должно быть всегда и везде, и как можно раньше, потому что только оно может показать, насколько удобно создаваемое творение.

Час пролетел незаметно, и хотелось еще и еще. К сожалению, Билл уехал достаточно быстро, поэтому буду теперь читать его сайт и статьи.

Затем я направился в секцию Организации работы, чтобы послушать Никиту Шермана. Из его доклада удалось вычленить следующие мысли:

- если хотите делать свой бизнес,



найдите то, что уже сделано и работает;

- когда найдете то, что уже работает, сделайте это лучше;
- имейте план «Б».

Все остальное крутилось вокруг этого и больше ничего интересного я не услышал.

Далее я забежал к Дамиру Халилову, чтобы послушать завершение его доклада про маркетинг в социальных сетях. Он ответил на вопросы о продвижении в ВКонтакте, про развитие социальных медиа и другое в этом ключе. Жаль, что не услышал весь доклад целиком.

После этого я болтался некоторое время по разным секциям, пытаясь найти что-то интересное для себя. Первым оказался Даниил Колесников из Яндекса. Даниил сделал замечательный доклад о том, как правильно выстраивать процесс обучения сотрудников внутри компании. Причем он коснулся не только развития джуниоров, но и развития тех людей, которые уже являются профи в своем классе. Даниил представил весь процесс, как квест из типовой компьютерной игры. Как в любом квесте, в обучении и любой поставленной задаче присутствует задание, список необходимых навыков, какие-либо артефакты, игровой мастер ▶



и прочая атрибутика. Таким образом, правильно составляя эти квесты, можно добиться достаточно быстрого профессионального роста сотрудника. Что же касается профи, то для них возможно получение «второй профессии», как выразился Даниил: у них есть вариант стать «эльфом-друидом 80-го уровня». В пример он привел свой опыт работы в Яндексе. Многие их разработчики высокого уровня получают дополнительную мотивацию для дальнейшего роста, когда перед ними открывается путь «евангелиста», как внутри фирмы, так и вне ее.

Поблагодарив Даниила за отличный доклад, я перешел в соседний зал, где выступал Илья Петров с докладом «Сила целого». Еще один интереснейший и содержательный доклад. Илья рассказал, что все надо делать до конца — если вы занимаетесь рекламой, то не забывайте ее анализировать. Если вы что-то продаете, то следите, чтобы продажи шли всегда. Понравилась и презентация и сам стиль изложения.

После одного Ильи я опять провел некоторое время блуждая между секциями в ожидании доклада от другого Ильи — Ильи Бирмана. С ним мы позже поболтали на афтерпати, но об этом ниже. Итак, Илья рассказал про то, что интерфейсы зло. Причем полнейшее и абсолютное. Какой идеальный интерфейс телефона? Никакой! Потому что не хочется искать в списке имен нужное имя, нажимать кнопки и так далее, а хочется просто: раз — и позвонить. Затем последовал «вброс», что капчу надо выкинуть, а веб-технологи просто ничего не понимают и решают ею чисто свои проблемы.

Оставшись в этом зале, я послушал доклад Андрея Шапира. Андрей рассказал о 10 камнях преткновения пользователя в путешествии по сложному интерфейсу. Он показал нам как не надо делать, как можно делать и как делать надо. К сожалению записей я не вел, доклад можно найти видео, но он был замечательный.

Завершил вечер докладов Андрей Сикорский, рассказав о том, как можно математически точно подсчитать на-



сколько успешен тот или иной дизайн. Некоторые вещи были, конечно, спорны, но в целом идея очень интересна, особенно понравилась мысль со 118 карточками ощущений.

На этом кончился первый день, и началось афтерпати!

Афтерпати

В прошлом году я не пошел на АП, о чем сильно потом жалел. Поэтому в этом году я смело потратил 1500 рублей и нисколько не пожалел. Мероприятие прошло уже можно сказать по традиции в Бирхаусе на набережной. Народу собралось очень и очень много, так что пиво и разговоры лились рекой. Общение в такой атмосфере зачастую намного более полезно, чем сама конференция, потому что люди более открыты, позитивны, дружелюбны и легко можно переходить от одного собеседника к другому. Так что завидуйте те, кто не попал.

День второй

Второй день начался с секции Юзабилити, на которой Алексей Иванов рассказал о проектировании сервисов для небольших экранов. Неплохое практическое руководство, но в итоге все све-

лось к рекламе приложения от Афиши.

На этой же секции был крайне интересный доклад от Алексея Копылова и Дмитрия Жукова, которые рассказали и показали отличные и отвратительные интерфейсы под iPad. Основная мысль — будьте логичными, чтобы людям не надо было придумывать какие-то специальные движения при работе с вашим приложением. Особо понравилось, что ребята показывали все интерфейсы тут же, транслируя их через вебку с айпада.

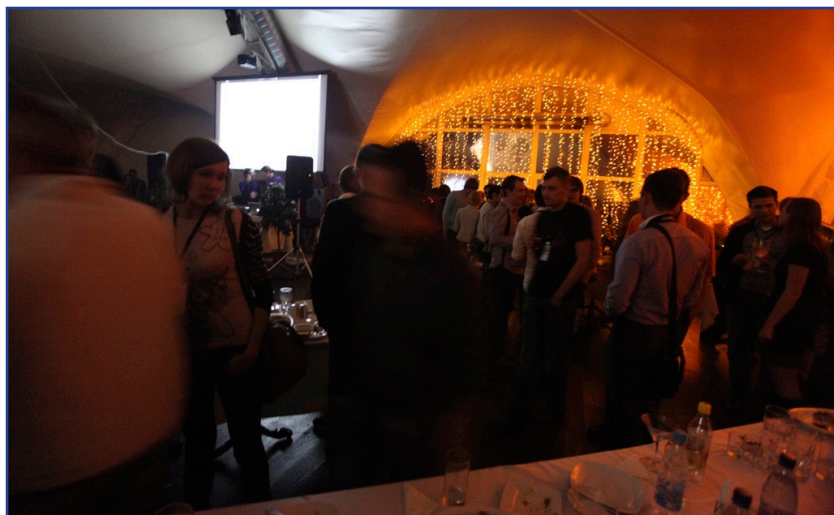
От Apple я перешел к Microsoft, а именно на доклад Гайдара Магданурова, который веселил зал и заставлял отжиматься модератора. Доклад Гайдара в конечном итоге свелся к рекламе продукции мелкомягких, но так как я сейчас много работаю с SharePoint, то я, пожалуй, часть этих продуктов даже и попробую.

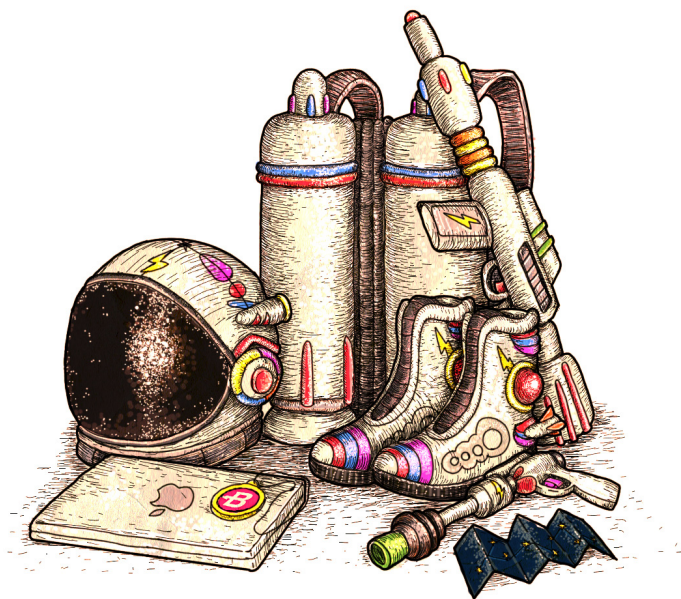
И, наконец, звезда второго дня Сергей «Зайка» Чикуненко, который рассказал обо всем кривом что есть в интернете, как он борется с этим в своем проекте «аймоби́лко» и поделился парой воспоминаний из прошлой работы с Темой Лебедевым. Так вот, «аймоби́лко» действительно крут, очень понравилась система моментального платежа.

Еще Сергей много говорил об оптимизации, о том, что она важна, но не стоит с нею перебарщивать. Даже если с точки зрения дизайнера интерфейс стал крутым, но непонятным для простого юзера, то такой интерфейс надо выкинуть. Этот доклад относится к таким, которые интересно пересматривать на видео.

Эстафету принял Николай Руфеев с самым ярким докладом в плане эффектов — Николай рассказывал про дополненную реальность. Эта технология существует уже около 20 лет, но только в последнее время ей стали уделять все больше и больше внимания. Тут собственно нечего рассказывать, надо смотреть ролики в интернете, которых существует уже великое множество.

Вот так насыщенно я провел эти два дня и могу сказать только одно: хочу еще! ■





Кадры решают ВСЕ

Автор

Денис Баталин
Генеральный директор
Digital-агентства
Batalline

На сегодняшний день на IT-рынке четко прослеживается несколько основных кадровых стратегий, связанных с подбором персонала как для выполнения разовых, так и постоянных задач по оказанию SEO-ориентированных услуг. Но прежде чем говорить о стратегиях подбора персонала, необходимо установить типы компаний, которые этими стратегиями пользуются. Таких типов, на мой взгляд, можно выделить три:

1. Компании, которые позиционируют себя как компании, занимающиеся исключительно SEO и интернет-маркетингом (к таким компаниям стоит отнести и обособленные подразделения крупных компаний, основным и главным направлением деятельности может быть любая другая деятельность в сфере IT).

2. Компании, которые занимаются SEO и интернет-маркетингом наряду с основным IT-ориентированным видом деятельности.

3. Компании, не отказывающиеся от любой возможности заработать денег, берущиеся за любые проекты и для которых указанный вид деятельности является совершенно новым.

С компаниями, которые занимаются исключительно SEO-ориентированными услугами, в принципе, все довольно понятно: являясь, в основном, старожилками рынка с богатой историей и опытом, имея выстроенную систему по подбору и расстановке кадров и, самое главное, обладая значительными финансовыми ресурсами, такие компании не испытывают хоть сколько-нибудь серьезных кадровых трудностей. С рын-

ка предложений трудовых ресурсов кадровики таких компаний (имеющие, как правило, более чем общее представление о том, какие соискатели им нужны и важны) выхватывают любые светлые головы. Идет кропотливая и неторопливая работа по переманиванию специалистов у коллег по рынку, а выстроенная система подготовки и обучения SEO-специалистов позволяет в кратчайшие сроки подготовить не только много новых рядовых исполнителей по проектам, но и линейных руководителей. Такая быстрота подготовки кадров часто обусловлена глубокой и узкой специализацией сотрудников в таких компаниях. Весь процесс работы над SEO-проектами в таких компаниях со стороны похож на конвейер. Любую из технологических цепочек такого конвейера можно заменить или быстро подготовить замену, в случае нарушения ее работы.

Вместе с тем, что касается компаний, для которых SEO-услуги являются либо дополнительным профилем к основной деятельности, либо компаний, которые вообще не занимаются оказанием SEO-услуг и рассматривают такие проекты как возможность дополнительного привлечения прибыли или дальнейшей диверсификации своего бизнеса — здесь не все так однозначно.

Очень часто в таких компаниях опытом поисковой оптимизации и знаниями в сфере интернет-маркетинга, а также всеми задачами, связанными с оказанием SEO-услуг занимается либо руководитель компании, либо сотрудник, совмещающий такую компетенцию вместе с основной и профильной. Именно эти сотрудники, в предвкушении солидных доходов от невесты откуда свалившегося на них SEO-проекта берутся за решение кадрового вопроса. Если опыт работы над проектом оказался удачным, то, как правило, для компании это становится новым направлением ее работы. Довольно много «историй успехов» свидетельствует именно о таком опыте развития их бизнеса: молодое

digital-агентство считало, что только за счет глубокой специализации можно успешно конкурировать на рынке, но почувствовав запах «больших денег» (а сегодня вряд ли найдется хоть кто-то, кто осмелится предположить, что оказание SEO-услуг бизнес низкомаржинальный) срочно открывало новое направление своего бизнеса.

О пяти стратегиях решения кадрового вопроса такими компаниями нам бы и хотелось поговорить. Кроме того, мы возьмем на себя смелость определить наиболее слабые и сильные стороны рассмотренных стратегий и, кроме того, приведем общие рекомендации по выбору одной из них.

Рассмотрим наиболее типовую ситуацию. Вы руководитель направления (или единоличный руководитель небольшой web-студии, главной и единственной специализацией которой является создание интернет-магазинов), и есть благодарный клиент, довольный сотрудничеством с вами по созданию интернет-магазина, который хочет еще делегировать вам поисковую оптимизацию и продвижение созданного вами проекта. Отказаться от возможности принять такое предложение — значит отказаться от нового опыта и вероятных прибылей. Возникает много вопросов, но первостепенный из них — кто всем этим будет заниматься?

Наверняка можно говорить о нескольких базовых стратегиях подбора SEO-персонала:

- 1) найм лучших за большие деньги,
- 2) найм нескольких «середняков» за деньги умеренные,
- 3) найм множества начинающих специалистов за меленькие деньги, ►

4) передача проекта субподрядчику,

5) найм специалистов, не имеющих вообще никакого опыта, их быстрое обучение и «запуск в дело».

Все указанные кадровые стратегии в той или иной мере использовались в рамках решения SEO-задач компанией Bataline, так что можно смело утверждать, что каждая из них проверена в действии.

№1 Стратегия «нанять лучших за большие деньги»

Если вы можете позволить себе вести ту же кадровую политику, что и Доналд Трамп (вспомним его знаменитый пассаж о подборе кадров «Работаю с лучшими»), скорее всего вы не только не читаете эту статью, но и читать вам ее совсем необязательно. У вас и так уже все в порядке.

Однако очевидно, что у всех «продвинутых» руководителей, так или иначе, велик соблазн «подкопить и купить». Как правило, покупка такого специалиста является эмоциональной. Мы интуитивно верим в большой потенциал человека, успешно выхваченного нами из контекста его накопившихся неудовлетворенностей текущим работодателем или просто переманенного большими деньгами. При этом очень редко в таких случаях мы задаемся такими важными вопросами, как: действительно ли он настолько профессионален, сколь себя позиционирует? И если это так, какими мотивами он руководствуется, покидая текущее место работы? Даже если мы абсолютно уверены в компетенциях такого сотрудника и в его искреннем желании просто «продаться подороже», важно не забывать о том, что у любой медали две стороны.

Действительно, такого сотрудника совершенно не нужно обучать. Более того, можно рассчитывать, в некотором смысле, на его желание передать свой опыт вашей компании, широкий круг знакомств, существующие ресурсы, влияющие на внешние факторы SEO-процессов. Кроме того, специалисты очень часто работают с собственными парами копирайтеров и эникейщиков. К сожалению, опыт работы компании Bataline по ведению такой кадровой стратегии нельзя назвать успешным. Очень часто случалось так, что при найме на работу, «дорогие» сотрудники, позиционирующие себя как крепкие профессионалы в поисковой оптимизации и продвижении, на деле оказывались либо «пустышками», единственной ценностью которых является запись в трудовой книжке о предыдущем месте работы, либо специалистами, твердо решившими встать на путь самостоятельного ведения бизнеса и единственное, что их отделяло от этого шага – отсутствие собственного пула клиентов и первоначального капитала. Ценности, которые они «благородно» получали в нашей компании. Если структурировать общие фразы, то

можно выделить несколько сильных и слабых сторон использования подобной кадровой стратегии.

+	знания привлекаемых специалистов самодостаточны
	возможны бонусы в виде собственных ресурсов
	могут являться руководителями проекта
	в случае успешного кадрового выбора можно быть уверенным в отличных результатах работы
-	дорого
	велика вероятность, что подобные сотрудники надолго в компании не задержатся
	риски связанные с делегированием большого количества компетенций и доступа к «телу клиента»

№2 Стратегия «нанять «среднячков» за умеренные деньги»

«В большом бизнесе в обрез не кроют» – важное замечание Френка Каупервуда из «Финансиста», может прийти весьма кстати при формировании команды для решения конкретных задач. Часто возникает вопрос о том, зачем делать ставку на одного конкретного профессионала, если можно поработать сразу с несколькими.

Такая стратегия кажется оправданной – как правило, лишённые лидерских амбиций, считающие себя «среднячками» в отрасли, такие сотрудники при должной мотивации (а она может быть, естественно, не только финансовой) могут сделать даже больше, чем от них ожидают. Коллективный разум, а также разделение отдельных участков ответственности работы над проектом смогут стать реальной движущей силой. Кроме того именно из таких специалистов можно растить руководителей направлений. Общая история и выполненные в рамках компании проекты могут стать «якорем» для таких специалистов, а своевременные доверительные отношения и дополнительные полномочия, делегируемые со временем – повышать лояльность к компании, в которой они работают.

Вместе с тем, появляются и некоторые опасности при выборе такой кадровой стратегии. Основной из них является опасность размытия ответственности как за отдельные участки работы, так и за весь проект. В то же время делать «тимлидом» одного из таких специалистов чревато, прежде всего тем, что такие сотрудники могут оказаться именно замечательными специалистами, но не хорошими руководителями. Готовы ли вы делегировать ответственность хорошему специалисту? А готовы

ли вы получить неудовлетворительный результат, наконец, поняв, что такая ситуация была ожидаема, учитывая отсутствие управленческого бэкграунда у конкретного, выделенного вами специалиста?

Кроме того, существует также опасность постоянных внутренних склоков, связанных с тем, что такие специалисты могут в определенный момент выступить не в лице коллективного разума, но в контексте басни «лебедь, рак и щука».

+	понимание ситуации
	коллективный разум
	возможность подготовить руководителя направления
	отсутствует необходимость обучения
-	размывание ответственности за проект
	дополнительно необходим руководитель

№3 Нанять много специалистов низкого уровня за мелкие деньги

Под специалистами начального уровня, в этой стратегии понимаются фрилансеры. В некотором смысле работа с такими сотрудниками может быть более чем удобна. Во-первых, можно упростить отношения работодатель-работник до рамок гражданско-правового договора или иных форм, позволяющих часть бремени по оплате обязательных налогов возложить на плечи самого сотрудника.

Во-вторых, очень часто такие сотрудники работают дома и, соответственно, снижают расходы работодателя на рабочие места и оргтехнику. В-третьих, часто за плечами таких специалистов большой бэкграунд, связанный с выполнением совершенно разных проектов по продвижению. Что так же может рассматриваться как преимущество. В-четвертых, как и в предыдущей стратегии, можно тешить себя мыслью о том, что среди кучки низкооплачиваемых «специалистов» может найтись кто-то, кто сможет преподнести вам на блюде наиболее удачное решение, при этом стоить такое решение может сущие копейки.

Другое дело, что и минусов при таком подходе довольно много. Это и необходимость создания большого центра по работе с такими специалистами (не стоит забывать, что не все фрилансеры работают в одинаковых часовых поясах) и обязательная предварительная смарт-аттестация (кто и чем будет заниматься), и, естественно, контроль фактически выполненных работ.

К сожалению, многие такие сотрудники часто не воспринимают свою деятельность как «настоящую» работу, от-

сюда огромное количество ошибок в работе, и проблемы с трудовой дисциплиной. Компания Bataline с такими специалистами работает по системе тикет-контроля, используя для этого как самостоятельно разработанную CRM-систему, так и Basecamp от 37signals.

Конечно же, задача найти множество низкоуровневых специалистов за скромные деньги может показаться непосильной, но в условиях нашей огромной страны, при различии уровня доходов в зависимости от конкретного региона такая задача успешно решается.

+	минимальный зарплатный бюджет
	каждый отвечает за конкретный небольшой участок работы
	оптимизация налоговых расходов
-	стратегия невозможна без выстроенной автоматизированной системы работы
	необходим руководитель

№4 Нанять субподрядчика

Идея нанять субподрядчика может оказаться не такой уж «нерассматриваемой», как может показаться на первый взгляд. Учитывая высокую маржинальность бизнеса, а также присутствие на рынке крупных компаний, которые давно и успешно работают с проектами по продвижению, можно избежать многих рисков и проблем, при этом совершенно легально заработав на «посреднических» услугах.

Как правило, у таких компаний не только хорошо оптимизирован сам процесс продвижения, но и бизнес-процесс, связанный с работой на субподряде. Очень часто, за счет больших оборотов такие компании имеют дополнительные преференции и скидки при работе с биржами ссылок и компаниями, зани-

мающимися контекстной рекламой.

Естественно, всегда остается риск того, что компании станут договариваться с вашим клиентом «напрямую». Кроме того сильно усложняется процесс коммуникации между непосредственным исполнителем и заказчиком. Решение любого, даже самого тривиального вопроса без вас невозможно. Важно так же понимание ответственности в контексте таких отношений. Готовы ли вы принять на себя ответственность понимая, что не сможете влиять на работу конечного исполнителя?

+	возможность заработать без инвестиций в непрофильное направление деятельности
-	повышенная ответственность
-	возможность потерять клиента

№5 Нанять много новичков, быстро обучить их и запустить в дело

Всегда сложно заниматься чем-то конкретным в первый раз. Но тут, как говорится, главное начать. Желаящих получить специальные знания, связанные с комплексным продвижением и интернет-маркетингом, как известно, немало. Если умело воспользоваться желанием таких соискателей можно не только успешно решить свои задачи, но и обеспечить безбедное существование таким желающим.

Понятно, что об обучении новичков можно говорить только в том случае, если за руководителем есть личный серьезный SEO-бэкграунд, либо если таковой имеется не у него, а у ближайших помощников. Именно в этом случае получается замечательный коктейль под названием «отстроить именно так (конечно же, воспользовавшись при этом самым положительным опытом), как сам считаешь правильным».

За счет такого подхода можно сде-

лать обучение и дальнейший процесс действительно специализированным (кого-то «посадить» плотно на контекст, кого-то на внутренние, а кого-то на внешние факторы оптимизации, кого-то на работу с директами...). Важным плюсом является то, что можно сразу же определить новичка с задатками лидера и управленца, и вырастить из него руководителя целого направления.

Главным и серьезным минусом в таком подходе является невозможность сразу же «ввязаться в драку» и приступить к работе над проектом. А так же безусловная вероятность совершения «доморощенными» специалистами некоторых ошибок, большое личное участие и трудозатраты либо самого руководителя, либо его заместителей (в случае, если к процессу обучения не будет привлечен «специалист со стороны»).

+	вы сами растите необходимую вам команду
	низкие стартовые финансовые затраты
	социальная миссия
-	большие трудозатраты
	возможность того, что человек, получив знания уйдет
	множественные косяки на старте
	долгий запуск проекта

Рассматривая все перечисленные стратегии, мы совершенно не ставили целью открыть глаза на проблему или сделать настоящий материал дискуссионный. Скорее это структурирование описанных стратегий, желание поделиться полученным в процессе работы компании Bataline опытом, помочь начинающим компаниям не наступать на уже сломанные грабли и позволить выбрать наиболее успешную стратегию в зависимости от стоящих перед организацией задач. ■



FastVPS.ru

VPS от 129 рублей

- 100 MB RAM
- 300 MHz CPU
- 6 GB HDD

**Dedicated от
2199 рублей**

- Intel® Core™ i7-920
Quad-Core incl
- 8 GB DDR3 RAM
- 2 x 750 GB HDD



- *один из лучших
дата-центров Германии*
- *служба поддержки
на русском языке*
- *уникальная система
восстановления*
- *удаленный reboot/reinstall*
- *установка сервера
в течение 24 часов*

Автор

Алексей Бузин

Обзор сервиса купли и продажи ссылок MainLink.RU

В современном мире невозможно вести какой-либо бизнес без сайта компании в интернете. Любой желающий может зайти на сайт и посмотреть каталог продукции, прайсы, прочитать договор о сотрудничестве, узнать о способах оплаты товара. То есть узнать всю исчерпывающую информацию о вашей компании не выходя из дома, и принять решение — покупать данный товар у вас или еще у кого-то. Сайт — это лицо компании. Вопрос создания сайта понятен — чем дороже будет сайт, тем красивее будет дизайн и тем больше он привлечет внимание посетителей.

Но вот как привести посетителей на сайт? С тех пор как поисковые машины стали считать авторитетность сайтов в поисковой выдаче на основе ссылок, которые ссылаются на эти сайты, получение ссылок стало необходимым элементом при оптимизации веб-сайтов и главным фактором в поисковом продвижении. Логичным продолжением процесса получения ссылок стало их платное получение с помощью бирж ссылок.

Ссылка стала главным орудием для человека, который продвигает свой сайт и желает видеть его в первой десятке поисковой выдачи. Для облегчения процесса покупки ссылок и для избегания разногласий вебмастеров и рекламодателей стали появляться биржи по купле-продаже ссылок.

Одним из первых на рынке, в 2005 году появился сервис купли-продажи ссылок MainLink, и продолжает занимать одну из лидирующих позиций в этой области. В системе было создано огромное количество функций под самые разные задачи: от оценки ссылок, до полностью автоматического проведения массовых покупок. В 2009 году в MainLink был произведен ребрендинг и полностью изменен интерфейс. Над интерфейсом работали лучшие специалисты в области юзабилити и дизайна. Интерфейс прост в использовании и реализован на русском и английском языках. Вести покупку ссылок в системе может как человек, хорошо знакомый с азами продвижения сайтов, так и новичок. Для этого в системе был реализован сервис «Быстрый старт», который позволяет запустить покупку ссылок в 3 шага: нужно ▶

MainLink
сервис купли-продажи ссылок

Продвижение сайтов с помощью текстовых ссылок
Получение дохода от продажи ссылок на сайтах

Служба поддержки
+7(495) 620-08-70

О нас Рекламодателю Вебмастеру Клубы Справка

Купить ссылки

- Новый удобный интерфейс
- Автоматическое и ручное продвижение сайта
- 295K сайтов
- 230M страниц
- 44K проектов
- Оценка бюджета продвижения
- Удобная система персональных настроек
- Различные формы оплаты
- Белые и черные списки сайтов и страниц
- Отзывчивые и ответственные менеджеры

Регистрация для покупки

Продать ссылки

- Новый удобный интерфейс
- Размещение ссылок руками или скриптом
- 1907K проданных ссылок ежедневно
- Стабильный и постоянный доход
- Цензура, стоп-слова и словари
- Черные и белые списки
- Круглосуточная служба поддержки
- Любые формы вывода средств
- Свои ссылки на своих сайтах бесплатно
- Помощь в установке кода

Регистрация для продажи

Авторизация

логин:

пароль:

☐ запомнить меня

[Регистрация](#) **Войти**

Дополнительные возможности

- Курсы обучения** - Мы проводим семинары и курсы обучения по SEO для наших пользователей, на которых можно получить полный объем знаний по продвижению сайта. [Ближайший курс обучения состоится 29 октября](#)
- Toolbar MainLink** - Тулбар MainLink - бесплатное приложение для Mozilla, которое предоставляет неограниченные возможности по анализу качества приобретаемых ссылок, что позволяет повышать эффективность работы со ссылками.

Новости
[Поиск релевантных страниц](#)
22 октября 2010г.

ввести «урл» продвигаемого сайта, продвигаемые запросы и зафиксировать бюджет, все остальное система выполнит самостоятельно. Вам нужно будет только вовремя пополнять баланс на аккаунте, и следить за ростом позиций вашего сайта в поисковике.

Помимо быстрого старта вы можете добавить проект по одному из заранее заготовленных шаблонов: **коммерческий сайт, Интернет-магазин, спателлит**. Для каждого шаблона уже есть встроенные сценарии покупки ссылок, это позволит сэкономить массу времени и сразу запустить проект в работу.

В системе реализован удобный контроль за позициями запросов. Можно смотреть позиции в поисковых системах Яндекс и Google. Есть возможность сбора позиций по любым регионам в Яндексе, количество регионов в отличие от других систем, в MainLink не ограничено. Позиции обновляются регулярно, и это экономит массу времени на ручной проверке. Сервис подскажет какая у запроса частотность, гео-зависимый он или нет, найден он по тексту или же по ссылке Яндексом, совпадает ли продвигаемая страница с той, которую Яндекс считает релевантной – эта статистика будет незаменимой при продвижении сайта.

Совсем недавно подключена новая возможность для рекламодателей – поиск релевантных страниц по запросам. Теперь можно не ломать голову над тем, какой запрос на какую страницу продвигать, а подставить ключевые слова и система сама опросит поисковые машины на предмет релевантности страниц, и затем добавит запросы в соответствующие страницы.

Для того чтобы упростить и ускорить процесс написания текстов ссылок, в сервисе реализован **генератор текстов**, который сам сгенерирует нужное количество уникальных текстов ссылок, причем тексты действительно делаются абсолютно уникальными, т. к. собираются из поисковых систем.

Модуль оценки стоимости продвижения рассчитает ежемесячный бюджет для продвижения запросов и позволит контролировать расход бюджета в соответствии с видами поисковых запросов (высокочастотные, среднечастотные и низкочастотные). Бюджет можно изменять на свое усмотрение.

Очень удобная «фишка» есть для владельцев интернет-магазинов. Можно импортировать YML – файл из Яндекс-маркета, оттуда будут автоматически импортированы страницы и запросы, а также под них будут созданы сразу же сценарии закупки ссылок. Робот сам выполнит всю работу, станет следить за каждым запросом в поисковых системах и включать/отключать покупку ссылок в зависимости от позиции запроса.

Модули автопокупки, автопродления, а также модули контроля за индексацией и списания непроиндексированных ссылок интегрированы в управление проектами и проводят всю рутинную работу за оптимизатора.

Если вам не понравился какой-то сайт, то его можно отправить в **Черный список** и размещение ссылок на нем больше не будет предлагаться, также есть **Белый список** – туда можете отправлять понравившиеся сайты, а затем среди них проводить покупку или дозакупку.

В системе поддерживаются две валюты – рубли и доллары: западные рекламодатели работают преимущественно в долларах, отечественные – в рублях. Пополнять баланс можно без комиссий через WebMoney, Яндекс-деньги, Альфа-банк и Банк Открытие. Также есть возможность работать по договору, используя безналичную форму оплаты.

Каждый пользователь может участвовать в **Клубах общения**, где можно завести блог, хранить ссылки на полезные ресурсы, а также заводить новые знакомства с другими пользователями сервиса.

В MainLink развита система безопасности, которая фиксирует нарушения и иногда блокирует аккаунты за нарушения или подозрения в мошенничестве. Рекламодателю возвращаются средства за убытки, связанные с нарушениями веб-мастеров.

Кстати в сервисе MainLink есть круглосуточная служба поддержки, даже по телефону, чего нет в других аналогичных сервисах. Если что-то не получается или что-то не понятно, можно написать или позвонить в службу поддержки и вам помогут решить проблему в кратчайшие сроки. Сервис развивается и идет в ногу со временем. ■



TeamCity: распределенное управление сборкой проектов

Автор

Александр Ильин
г. Санкт-Петербург
alexander.ilyin.tel
alexander@ilyin.eu

Разработчик: JetBrains Inc.

Сайт: <http://www.jetbrains.com/teamcity/>

Русский перевод: нет

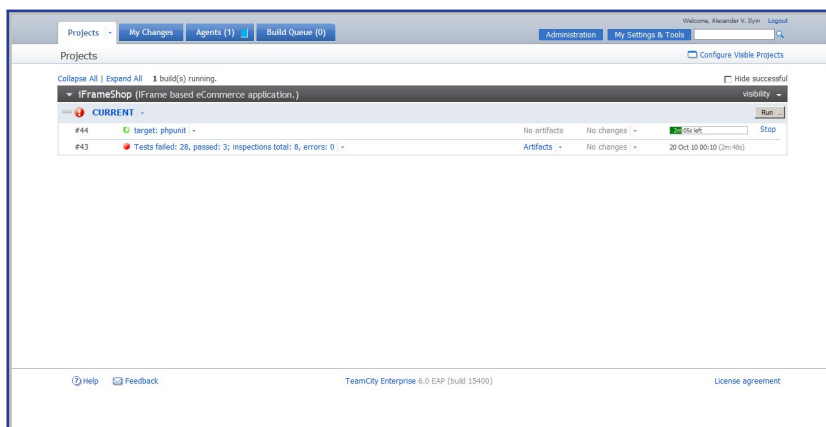
Стоимость: бесплатная версия / платная версия \$1999

Непрерывная интеграция (англ. Continuous Integration) – это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем. В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ. Переход к непрерывной интеграции позволяет снизить трудоемкость интеграции и сделать ее более предсказуемой за счет наиболее раннего обнаружения и устранения ошибок и противоречий.

В настоящее время существует несколько популярных решений, среди них:

- Atlassian Bamboo
- Cruise Control (существует специальная надстройка phpUnderControl для PHP проектов)
- Hudson
- Xinc
- TeamCity

Я попробовал их все, но приглянулся мне лишь последний. Bamboo возможно будет более полезен при интеграции с другими продуктами Atlassian, к примеру если вы уже используете JIRA для сообщения об ошибках. Cruise Control, даже с надстройкой phpUnderControl, показался мне несколько ограниченным в функциональности, Hudson имеет устаревший, с моей точки зрения, дизайн, но тем не менее пользуется достаточно большой популярностью среди разработчиков. Xinc и вовсе обладает минимумом функционала. Тем не менее для самостоятельного знакомства, при условии интеграции с PHP проектами наиболее



подходящими решениями являются Xinc и phpUnderControl.

Мы же рассмотрим непрерывную интеграцию с использованием TeamCity, так как она обладает дружелюбным современным интерфейсом, предоставляет достаточно большой набор дополнительных возможностей, а так же имеет одну исключительную особенность – предварительное тестирование коммитов. В процессе сборки используются следующие составные элементы: билд-утилита (в нашем случае это будет Apache Ant, поставляемый в комплекте с дистрибутивом), агент сборщик (бесплатная версия TeamCity позволяет использовать до трех агентов), система управления версиями – SVN (но поддерживаются почти все популярные включая Git), непосредственно проект, а так же утилиты для тестирования исходного кода и сбора статистических данных:

- **phpDocumentor** – популярный инструмент для создания документации о проекте на основании комментариев в исходном коде.
- **PHP CodeCoverage** – собирает информацию о покрытии кода тестами.
- **PHP Copy/Paste Detector** – анализатор идентичного кода в проекте.
- **PHP Depend** – отображает статистическую информацию о проекте.
- **PHP PMD** – второй инструмент для сбора статистической информации о коде.

- **PHP CodeSniffer** – позволяет следить за соблюдением стандартов кодирования.

- **PHPUnit** – популярный инструмент для создания и запуска модульных тестов, так же поддерживает работу с Selenium RC.

- **PHP CodeBrowser** – позволяет просматривать исходный код, а так же может интегрировать отчеты от других утилит.

- **phploc** – утилита для быстрого отображения общей информации о проекте.

- **PHP Dead Code Detector** – уведомляет о неиспользованных методах/функциях.

Бесплатная версия в отличие от платной не предоставляет возможности настройки индивидуального доступа пользователей к проектам, но текущая версия (6.0) TeamCity EAP в течении двух месяцев работает в Enterprise режиме, а так же обладает некоторым количеством улучшений – автоматической очисткой рабочей директории от артефактов и прочих файлов созданных в процессе сборки, и предоставляет возможность инициализации нескольких шагов сборки с использованием различных билд-утилит.

Прежде всего, вам потребуется скачать и установить подходящую версию TeamCity для вашей ОС, на данный момент поддерживаются Windows, Linux и Mac. Процесс установки освещен в ▶

документации и не требует каких-либо особых советов, за одним исключением. При установке EAP (Early Access Program) на Windows платформе с существующей инсталляцией предыдущего релиза, были обнаружены конфликты с правами администратора в TeamCity, но как обычно удаление, перезагрузка и повторная установка решили проблему. При установке можно оставить все параметры со значениями по умолчанию, если это не вызовет конфликтов в вашей системе, к примеру порт для доступа к GUI по HTTP.

Для увеличения производительности при большом количестве проектов или же для тестирования сборки в различных окружениях вы можете подключить дополнительные агенты для сборки. Несколько агентов так же могут быть запущены и в одном окружении, но их рабочие директории не должны пересекаться, подробнее этот момент освещен в документации.

Первоначальный запуск сервера и агентов может занять некоторое время, но как только система будет сконфигурирована, вы перенаправитесь на экран создания учетной записи администратора. После этого сможете с помощью удобного визарда в несколько простых шагов сконфигурировать свой первый проект и конфигурацию сборщика для него. Опять же большинство параметров можно оставить по умолчанию, основные настройки мы будем производить с помощью конфигурационного файла для Apache Ant и сервисных сообщений TeamCity.

При создании конфигурации сборщика, на первом шаге вам будет необходимо указать имя, а так же правила для формирования номера сборки, на втором – указать настройки для подключения к системе управления версиями. С помощью несложных правил можно настроить получение исходных кодов только из определенной ветки, так же TeamCity позволяет создавать теги на основе успешных сборок. К примеру, мне было достаточно несложного правила:

```
+ :trunk=>source
```

В результате весь исходный код из trunk будет находиться в поддиректории source рабочей директории. На третьем шаге предстоит выбрать утилиту для сборки и месторасположение конфигурационного файла для сборщика. В нашем случае это будет Apache Ant, а конфигурационный файл мы храним в корне проекта в репозитории (/trunk/build.xml). Соответственно после применения условий описанных выше, путь к конфигурационному файлу будет иметь следующий вид:

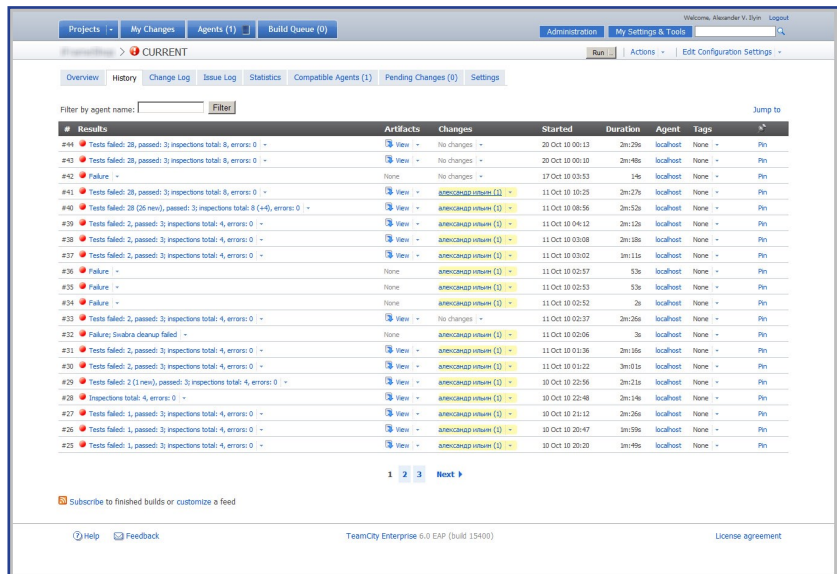
```
source/build.xml
```

Так же в TeamCity EAP 6.0 доступен инструмент под названием Swabra, который вкратце был освещен в начале статьи, он позволяет легко следить за чистотой в рабочей директории при сборке проекта. После этих трех шагов вы будете иметь первый сконфигурированный проект. Но для большего комфорта вам еще будет необходимо настроить автоматизацию запуска сборки (Build configuration/Build Triggering). Я предпочел запустить сборку проекта при обнаружении изменений в системе управления версиями. Так же при наличии нескольких проектов мы можете настроить зависимости сборки между ними.

Далее перейдем к созданию build.xml который собственноручно и будет руководить сборкой. К сожалению он достаточно объемный, поэтому полную версию вы можете получить по этой [ссылке](#). Мы же рассмотрим основные секции.

Build.xml представляет собой стартовый конфигурационный XML файл для Apache Ant:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Application" default="build"
```



```
basedir=".." />
```

```
<!-- ... -->
```

```
</project>
```

В примере выше стоит обратить на свойство default – оно определяет target по умолчанию, так как в TeamCity мы не указали какого-либо конкретного target. В принципе используя один конфигурационный файл вы можете настроить различные варианты сборки, вызываемыми target'ами. Например, при автоматическом обнаружении изменений будут запускаться только тесты и анализаторы кода, а при «ночной» сборке так же будет генерироваться документация.

Второй момент – это свойство basedir, которое собственно и определяет базовую директорию относительно самого файла.

Далее рассмотрим подготовку рабочей директории к сборке и target по умолчанию:

```
<target name="build" depends="prepare,phpunit,phpmd,phpcpd,phpdoc,phpcb,phpcd,phploc,phpcs"/>
```

```
<property name="applicationDir" value="${basedir}/source" />
```

```
<property name="buildDir" value="${basedir}/build" />
```

```
<target name="prepare">
```

```
<mkdir dir="${buildDir}/logs" />
```

```
<mkdir dir="${buildDir}/html/phpdoc" />
```

```
<mkdir dir="${buildDir}/html/phpcb" />
```

```
<mkdir dir="${buildDir}/html/phpcov" />
```

```
<mkdir dir="${buildDir}/html/phpunit" />
```

```
</target>
```

В target по умолчанию мы определили зависимые задачи, а так же их очередность. Для большего удобства мы определили пути к исходным файлам и директории для хранения результатов сборки в виде переменных. Если же вы пользуетесь более ранними версиями TeamCity или по каким-то причинам отказались от использования Swabra, то вам необходимо настроить очистку директории для сборки либо в виде отдельной задачи, либо в начале задачи prepare.

Рассмотрим настройку запуска сторонних утилит во время сборки проекта на примере PHPUnit:

```
<target name="phpunit" description="PHPUnit">
```

```
<property name="phpunitLogs" value="${applicationDir}/tests/logs" />
```



```

<property name="junit" value="\${phpunit-
Logs}/phpunit.xml" />

<exec dir="\${basedir}" executable="phpunit"
failonerror="false" osfamily="unix" output=
"\${buildDir}/logs/phpunit.txt">

  <arg line="--configuration\${applicationDir}/
tests/phpunit.xml"/>

</exec>

<exec dir="\${applicationDir}\tests" exe-
cutable="cmd" failonerror="false" osfamily="win-
dows" output="\${buildDir}\logs\phpunit.txt">

  <arg line="/c phpunit.bat" />

  <arg line="--configuration \${applicati-
onDir}\tests\phpunit.xml" />

</exec>

<echo message="##teamcity[importData ty-
pe='junit' path='\${junit}']" />

<zip destfile="\${phpunitLogs}/coverage.
zip" basedir="\${phpunitLogs}/coverage"/>

<delete dir="\${phpunitLogs}/coverage"/>

<echo message="##teamcity[publishArtifa-
cts '\${phpunitLogs}']" />

</target>

```

Опять же для большего удобства мы определим некото-
рые пути в виде переменных `phpunitLogs` и `junit`, а так
же добьемся кросс-платформенного запуска PHPUnit как на
Linux, так и на Windows агентах, за что отвечает свойство
`osfamily` элемента `exec`. Для уменьшения размера `build-
xml`, а так же для удобства запуска PHPUnit из IDE/консоли
при «ручном» выполнении тестов, мы выделили настройки
PHPUnit в отдельный файл. Полную версию вы можете по-
лучить по [ссылке](#), а мы рассмотрим часть, непосредственно
относящуюся к сборке:

```

<logging>

  <log type="coverage-html" target="./logs/
co-verage" charset="UTF-8" yui="true" high-

```

```

light="true" lowUpperBound="35" highLowerBo-
und="70"/>

  <log type="coverage-clover" target="./logs/
coverage.xml"/>

  <log type="json" target="./logs/phpunit.json"/>

  <log type="tap" target="./logs/phpunit.tap"/>

  <log type="junit" target="./logs/phpunit.
xml" logIncompleteSkipped="true"/>

  <log type="testdox-html" target="./logs/
testdox.html"/>

  <log type="testdox-text" target="./logs/
testdox.txt"/>

</logging>

```

Теперь при сборке после вызова задачи `phpunit` в `source/`
`tests/logs` будут созданы артефакты, которые позднее будут
опубликованы с помощью сервисных сообщений TeamCity:

```

<echo message="##teamcity[importData type=
'junit' path='\${junit}']" />

<!-- ... -->

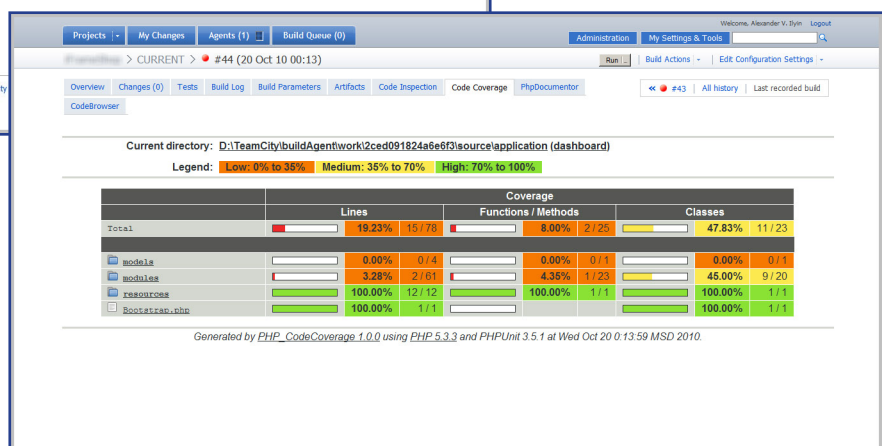
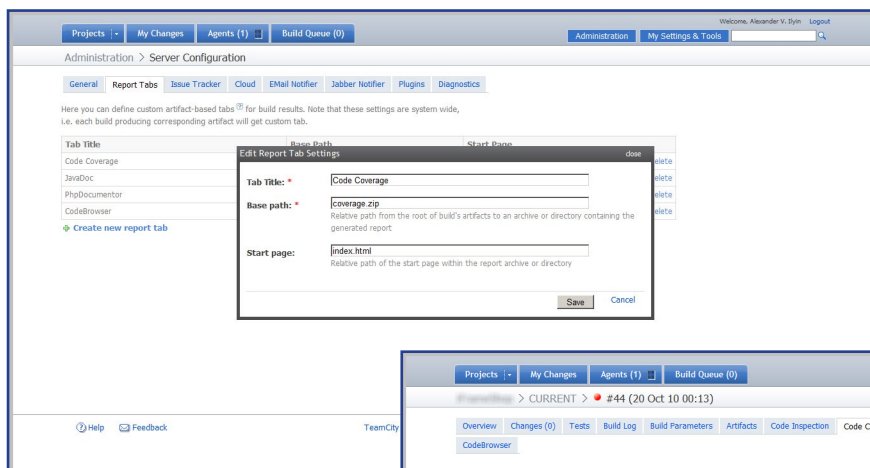
<echo message="##teamcity[publishArtifacts
'\${phpunitLogs}']" />

```

Нам осталось настроить корректное отображение содер-
жимого артефактов (`coverage.zip`) в TeamCity. Для этого не-
обходимо перейти в раздел **Administration => Server Con-
figuration => Report Tabs**, нажать **Create New Report Tab** и
указать имя новой вкладки (**Code Coverage**), путь к директо-
рии или архиву с артефактами (`coverage.zip`) и индексную
страницу (`index.html`). Аналогичным способом создаются
вкладки для результатов PHP CodeBrowser и PhpDocumentor.

TeamCity обладает большим количеством вариантов уве-
домления пользователей о сборках, среди них:

- по Email,
- в IDE в виде дополнений,
- Jabber,
- приложение для панели задач Windows. ■



Проклятье веб-сайтов и волшебный UX design

Автор

Федор Аксенов
Консультант по UX дизайну
и проектированию сайтов
г. Москва
wagthedog.ru
faxenoff@gmail.com

Представьте себе обычного директора, который решил сделать обычный сайт для своей не менее обычной компании. Он нашел ближайших веб-разработчиков с честными глазами, и в двух словах на одной салфетке описал им стандартные пожелания обычного корпоративного сайта. А может быть он поступил проще — указал на похожий сайт и сказал: «Мне надо тоже самое, только круче!». Разработчики привычно сами себе написали техническое задание и с честью его выполнили. Потом как смогли наполнили сайт контентом и помолясь запустили. За месяц работы на сайт зашло десяток роботов и несколько случайных посетителей. Директору рассказали о SEO, на которое ему, скрепя сердце, пришлось выделить на три месяца по 50 тыс. руб. — «чтобы попробовать» и «чтобы наверняка». Благодаря раскрутке сайт начал появляться в поисковой выдаче, и счетчик стал показывать почти тысячу посетителей в день. За это время к менеджерам звонило по 5-10 человек в месяц, которые говорили, что они смотрят сайт и у них есть вопросы по товарам. Реальным клиентом стал только один, да и то, по мелочи.

Убитый печалью директор отменил все это бессмысленное SEO и с негодованием отказался от предложений потратить еще денег на услуги «партизанского» и «социального» маркетинга со «100% результатом». Попытки выяснить причину неудачи ни к чему не привели — все эксперты только разводили руками и опять советовали потратить денег на SEO, контекстную рекламу и маркетинг. Или заказать новый сайт у новых разработчиков, вдруг на этот раз он получится лучше. И вот поиск причин неудачи перешел из области разума в область метафизики. Ведь неизвестно, может у разработчиков сайта отрицательная карма, или хостинг сайта расположен на старом индийском кладбище, или сервер не окропили святой водой, или слгазили по ftp, или это конкуренты заплатили денег эне-

гоинформационному экстрасенсу и он с помощью генератора акустического резонанса (модель бубен) взял, да и сайт весь ПРОКЛЯЛ!

Такие же необъяснимые случаи происходят не только с корпоративными сайтами, но и со всеми остальными, а особенно с интернет-магазинами и веб-сервисами. Казалось бы, одни хорошие люди заказали сайт у других хороших людей, те добросовестно сделали им хороший сайт, а в результате получилось нехорошо. Кто виноват? Что делать?

Давайте разберемся без мистики. Допустим, профессионализм веб-разработчиков и качество их работы мы проверили у независимых экспертов. Установили, что в этом месте проколов нет. Что еще может быть? Плохая фирма, плохой товар? Проверяем — обыкновенная фирма, обыкновенный товар. Есть хуже, есть лучше. Некоторые их тех, которые «хуже», вполне замечательно живут и всюю торгуют через интернет.

Догадались в чем проблема?

Давайте отмотаем назад, на момент первого разговора директора и веб-разработчиков. Смотрите, директор рассказывает веб-разработчикам что он хотел бы видеть на сайте, а те прилежно это записывают... Стоп! Вот то самое место. Директор рассказывает что, по его мнению, должно быть на сайте. Вопрос: «А откуда он знает?»

В том, что он знает, что надо от сайта для его бизнеса — нет никаких сомнений. А вот решать каким образом это должно быть реализовано на сайте, директор абсолютно некомпетентен. Да и откуда ему быть компетентным? У него нет никакого опыта в веб-технологиях. Он не знает о проектировании веб-сайтов совершенно ничего, но в этот момент фактически занимается именно проектированием своего сайта, причем на профессиональном

уровне ниже дилетантского. Поэтому совершенно неудивительно, что результат оказывается в лучшем случае «никаким», а в худшем смешным и нелепым. Для руководителя нет ничего зазорного поручить это дело профессионалам. Он же точно так же доверяет ведение бухгалтерии — бухгалтерам, написание договоров — юристам, а ремонт унитазов — сантехникам.

«Позвольте, уважаемый. — Скажете вы. — Но ведь это явно область компетенции веб-разработчиков, которые должны, даже просто обязаны заниматься всесторонним проектированием для заказчика. Им же за это деньги и платят, в само-то деле!». Оно было бы так, кабы ни этак. Процесс проектирования при участии заказчика очень противоречив, и основной его проблемой, как правило, является сам заказчик и его убеждения. Спорить с заказчиком и самостоятельно закладывать в проект решения, которые усложняют его разработку могут позволить себе лишь супер авторитетные веб-разработчики (например «Студия Артемия Лебедева») у которых это получается, ну или просто идиоты, которые тем самым себе только вредят. Нормальная веб-студия зарабатывающая деньги, прежде всего, заинтересована сделать сайт как можно проще в разработке, и иметь как можно меньше конфликтных тем с заказчиком. Вот поэтому проектированием занимаются специальные люди в штате заказчика, а если таковых нет, то приглашаются независимые консультанты со стороны.

Само проектирование называется User Experience Design (UX design) или по-русски «Проектирование взаимодействия с пользователем». Как можно понять из названия, оно сфокусировано конкретно на пользователе и представляет собой набор различных методик, сделанных из смеси инженерных дисциплин, дизайна, маркетинга, социологии и психологии. Звучит сложно, но на самом деле все проще. С одной стороны определяются цели и задачи

системы, с другой стороны – цели и потребности пользователей. И ставится задача спроектировать систему так, чтобы подружить их между собой.

Работа происходит примерно в такой последовательности:

- Определяются цели и задачи системы (продукты и клиенты в случае маркетинга).
- Определяются пользователи, их роли (на основе их целей) и сценарии их поведения.
- На основе этого строятся сценарии взаимодействия пользователей с системой.

Для реализации сценариев взаимодействия требуется описать необходимую функциональность, которая будет это взаимодействие осуществлять.

Все это моделируется и собирается в виде диаграмм взаимодействия и плавно переходит к непосредственному проектированию структурных схем страниц (wireframes). На их основе уже делается последний штрих – юзабилити и дизайн.

Если посмотреть на это со стороны заказчика, то все выглядит следующим образом. Приходит UX дизайнер или консультант на встречу с заказчиком/автором проекта и пытается наиболее полным образом понять цели и задачи проекта (товар-услугу и ее особенности). А так же сформировать образы пользователей (или клиентов-покупателей). В ходе обсуждения UX дизайнер исходя из собственного опыта, предлагает свои решения для организации взаимодействия пользователей и обсуждает их с заказчиком/автором проекта. Узнав всю необходимую информацию, UX дизайнер приступает к проектированию (по шагам из предыдущего списка) и периодически синхронизирует свои наработки с видением заказчика/автора проекта. Результатом работы является готовый проект в виде исчерпывающего опи-

сания всех аспектов взаимодействия, функциональности и с предварительным блочным дизайном, что подойдет для любых веб-разработчиков в качестве техзадания. И пожалуй лучшего и точного техзадания, из тех, которые они получают обычно.

UX дизайн не является панацеей от всех бед. Но если вам надо сделать сайт (интернет-магазин, веб-сервис) комфортным для людей, то вам нужен только он. Им можно повысить конверсию посетителей в клиентов/покупателей/пользователей. Есть хорошая фраза: «UX дизайн – это научное обоснование здравого смысла».

Стоит сказать немного о различиях UX дизайна и юзабилити, а то у многих возникает ощущение что это одно и то же. Юзабилити входит в область методик UX дизайна и отвечает за удобство, логичность, последовательность и непротиворечивость визуального интерфейса. Другими словами, юзабилити отвечает только на вопрос «Как делать?», а UX дизайн, прежде всего, начинает с вопроса «Что делать?».

Пример: вы дали юзабилитисту задание сделать форму регистрации нового пользователя максимально простой и понятной. Он честно занялся этой головоломкой и выдал проработанное решение. Правда в нем все равно были все необходимые для регистрации поля, вопросы и контролы. Если бы вы поставили задачу улучшить сайт перед UX дизайнером, то он, скорее всего, придумал бы как вообще отказаться от регистрации и тем самым убрать лишние шаги для пользователей. Если бы это был мастер UX дизайна, то он бы мог (в некоторых случаях) придумать, как можно отказаться и от всего сайта вообще, а стоящие перед ним цели и задачи успешно решить, многократно увеличив конверсию (например, интеграцией в социальную платформу).

Если у вас возникает ощущение, что ваш сайт накрыла подобная печать проклятия или вы хотите поднять конверсию посетителей, или планируете сделать новую прекрасную версию своего сайта – теперь вы знаете, что вам поможет и к кому надо обращаться. ■



XenForo:

краткий обзор архитектуры

нового форумного движка

В этой статье я постараюсь вкратце описать архитектуру нового форумного движка XenForo, нового конкурента VBulletin и IPB.

Размеры кода

Размер основной части кода XenForo 3,24 Мб (3 405 312 байт). Еще 8,31 Мб (8 724 429 байт) занимает Zend Framework (далее ZF), который поставляется вместе с кодом XenForo. ZF включен в поставку целиком, хотя используется из него только небольшая часть.

MVC и модель контента

XenForo полностью основан на MVC архитектуре. Есть отдельные классы для моделей, видов и контроллеров. Корневой `index.php` имеет размер 467 байт и заканчивается, как и положено в таких случаях:

```
$frc = new XenForo_FrontController(new XenForo_Dependencies_Public());
$frc->run();
```

Разумеется, в ходу автозагрузчик классов. Система именования классов как в ZF.

Сама MVC пока внутри использует классы ZF, но то, что `XenForo_Controller` не унаследован от соответствующих классов ZF, говорит о том, что эта часть, скорее всего, будет переписана.

MVC работает следующим образом. Все, что идет через `index.php`, обрабатывается классами `XenForo_Controller_Public_*`. Все, что через `admin.php` — `XenForo_Controller_Admin_*`. Соответствующий класс-обработчик выбирается исходя из префикса URL. Скажем, если запрашивается `www.example.com/forum`, запрос передается на обработку `XenForo_Controller_Public_Forum`. Все префиксы роутинга задаются в административной панели и могут добавляться допол-

нениями (addons). Все классы MVC обеспечивают полный SEO из коробки (например, для каждого префикса роутинга существует соответствующий класс `XenForo_Route_Prefix_*`, который отвечает за красивые URL для данного префикса и за выбор контроллера для данного префикса).

Контроллер может запросить какие-то данные из модели, а потом отрисовывает вид (о шаблонах см. далее), передавая ему необходимые параметры (например, список сообщений раздела форума).

Система контента XenForo чем-то слегка напоминает Drupal. Во всяком случае, здесь тоже есть понятие Node, правда выражение «Everything is a node» здесь уже не работает.

Пока в XenForo типов контента четыре: категория (что-то вроде папки), раздел форума (в котором могут быть сообщения), страница (HTML) и ссылка (обычный редирект). Каждый тоже называется Node (узел). Причем экземпляр узла может быть предком/потомком экземпляра любого другого типа. Пока не знаю, как эта гремучая смесь, например, из страницы и дочернего ей раздела форума, будет отображаться, этого я не пробовал :).

Некоторые типы контента можно слегка расширять. Так, например, для страницы помимо собственно ее HTML содержимого, указываемого в виде шаблона XenForo, можно указать PHP обработчик, который, например, получит из БД дополнительные данные для отображения, или вообще «плюнет» в пользователя ошибкой. PHP обработчики везде указываются не кодом на PHP, как этого можно было ожидать, а именами класса и статического метода-обработчика. При активации класс будет загружен автолоадером (соответственно, должен быть правильно расположен в файловой системе).

Типы контента пока заданы жестко, самому добавить их не получится. Даже их названия в админке почему-то

Автор

Владислав Раструсный
Программист, системный администратор
г. Москва
<http://www.fractalizer.ru>
FractalizeR@yandex.ru

не локализованы.

Модель данных делится на две половинки:

- **модель для чтения** — классы, унаследованные от `XenForo_Model`; используются для удобного чтения информации из базы данных,
- **модель для записи** — классы, унаследованные от `XenForo_DataWriter`; используются для создания и изменения содержимого.

Например, модель разделов форума состоит из `XenForo_Model_Forum` + `XenForo_DataWriter_Forum`. Хотя из этого правила есть небольшие несимметричные исключения вроде `XenForo_Model_Post` + `XenForo_DataWriter_DiscussionMessage_Post`. Впрочем, не будем забывать, что XenForo пока еще в бете и, несмотря на то, что API уже устоялся, небольшие изменения все равно возможны.

Использование сторонних библиотек

XenForo использует Sabre и Zend Framework. Причем наличие в коде мест, подобных

```
/*require_once('Zend/Loader/Autoloader.php');
```

```
$autoloader = Zend_Loader_Autoloader::getInstance();
```

```
$autoloader->pushAutoloader(array($this, 'autoload'));
```

```
spl_autoload_register(array($this, 'autoload'));
```

скорее всего, означает, что от лишних зависимостей на протяжении разработки продолжат избавляться, заменяя их ►

на что-то свое более легковесное и более специализированное.

Из ZF используется совсем немного классов. В их числе `Zend_Registry` (в качестве реестра для синглтон-подобных объектов вроде экземпляра `XenForo_Db`), `Zend_Config`, `Zend_Cache` (много чего кеширует, в частности используется в базовом классе моделей `XenForo_Model`), `Zend_Service_ReCaptcha`, `Zend_Http_Client`, `Zend_Mail`, `Zend_Validate` и некоторые другие.

Для работы с базой данных используется часть комплекта `Zend_DB` (`Zend_Db_Adapter_Abstract` и подобное), завернутая в класс `XenForo_Db`, что опять говорит о том, что и эту часть в последствии перепишут. Результат выполнения запроса к базе представляет собой обычный массив (как `mysql_fetch_assoc`, практически), так что особых проблем здесь не будет.

Конструктор запросов не используется, они встроены прямо в код, как обычно, что, конечно, несколько затруднит поддержку нескольких баз данных. Но с другой стороны, запросы сконцентрированы внутри модели, так что...

Архитектура дополнений

Для расширения форума предназначено несколько событий, на которые можно повесить обработчики. Обработчики указываются в админке, как уже было сказано, через имя класса и метода, что позволяет использовать акселераторы вроде APC для кеширования кода дополнений. Список обработчиков вместе с базовой информацией о плагине помещается в *.xml файл. Писать его руками не обязательно. Достаточно активировать отладочный режим на форуме, и в админке появятся дополнительные возможности. PHP файлы дополнения, разумеется, надо будет сначала распаковать в соответствующую папку. Классы `XenForo` лежат в `/library/XenForo`, стало быть ваши будут лежать в `/library/VasyaCorp`. Меня это порадовало, поскольку довольно часто я занимался поиском остатков плагина в файловой системе при его удалении из `VBulletin`.

Поскольку практически каждое дополнение должно иметь настройки, их можно создавать прямо в админке и экспортировать в *.xml вместе с информацией о хуках. Располагаются они там же, где и настройки самого `XenForo` и выглядят неотличимо. Встраивать их можно в произвольные места раздела общих настроек `XenForo`. Устроено все примерно также, как это было в `VBulletin`. Очень удобно и практически нет необходимости писать код для их поддержки, все настраивается через интерфейс. Хотя, конечно, при желании дополнение может встроить свою отдельную страницу в административную панель. Это делается довольно просто. Из дополнений к настройкам можно обращаться так:

```
XenForo_Application::get('options')->discussionsPerPage
```

Меня немного удивило количество хуков. Я полагаю, все помнят оглушительное количество хуков в `VBulletin` (в версии форума 4.0.3 их было 996 и это без хуков блогов и CMS!), которое даже в голове нельзя было удержать. В `XenForo` хуков всего... 17! Такое количество объясняется их назначением. Вот их имена:

```
container_admin_params
container_public_params
controller_pre_dispatch
front_controller_post_view
front_controller_pre_dispatch
front_controller_pre_route
front_controller_pre_view
init_dependencies
load_class_bb_code
load_class_controller
load_class_datawriter
load_class_model
load_class_route_prefix
load_class_search_data
load_class_view
navigation_tabs
visitor_setup
```

Небольшая инспекция кода показала, что семейство хуков `load_class_*` предназначено для динамического расширения соответствующей группы классов `XenForo`. Вы просто пишете свой класс, расширяющий один из базовых специальных классов системы, а затем внутри хука указываете его имя. Вот и все. Методы классов `XenForo` спроектированы именно для такого устройства работы дополнений, поэтому пользоваться данным способом очень удобно.

А несколько неожиданное для меня использование хука `visitor_setup` для изменения всего чего угодно на странице показано *my*. Практически через любой хук вы можете залезть в сердце системы.

В общем, малое число хуков, похоже, вполне компенсируется их мощностью.

Система шаблонов

Как и во всех современных форумах в `XenForo` есть КУЧА шаблонов. Синтаксис довольно мощный (см. отрывки внизу), причем, этим уже никого не удивишь. Спецтеги пишутся по-хожим на *PHPTAL* («неймспейсированный» XML/HTML). Один шаблон может содержать несколько файлов. Например, шаблон `forum_list` содержит `forum_list`, `node_list` (он вставляется в `forum_list`), `node_list.css`, `sidebar.css`, `sidebar_online_users`. CSS в шаблоне подключается через `xen:require`, что позволяет кешировать все CSS файлы как угодно.

```
<xen:require css="node_list.css" />
<xen:if hascontent="true">
<fieldset>
<ol class="nodeListsectionMain" id="forums">
<xen:contentcheck>
<xen:foreach loop="$renderedNodes" value="$node">{xen:raw $node}</xen:foreach>
</xen:contentcheck>
</ol>
</fieldset>
</xen:if>
```

```
<xen:edithint template="node_link.css" />
<xen:contentcheck>
<xen:foreach loop="$onlineUsers.records" value="$user">
<xen:if is="{ $user.is_moderator } OR { $user.is_admin }">
<li>
<xen:avatar user="$user" size="s" img="true" />
<a href="{xen:link members, $user}" class="username">{xen:helper richUserName, $user}</a>
<div class="muted">{xen:helper userTitle, $user}</div>
</li>
</xen:if>
</xen:foreach>
</xen:contentcheck>
```

Обратите внимание на использование хелперов в шаблонах. Хелпер, как и в *Symfony*, это просто метод класса. Например, вот код хелпера `helperUserLink`:

```
public static function helperUserLink(array $user) ▶
```

```
{
    return '<a href="'
        . XenForo_Link::buildPublicLink('members', $user)
        . '" class="username">'
        . htmlspecialchars($user['username'])
        . '</a>';
}
```

Шаблоны можно вставлять друг в друга через `xen:include`.

Поисковая машина

Поисковая машина в XenForo реализована в виде отдельного класса (в отличие от ненавидимой мною реализации в VB4, в которой, по-моему, сами разработчики давно уже запутались, просто не сознаются в этом), унаследованного от `XenForo_Search_SourceHandler_Abstract`. В настоящий момент реализация только одна – `MySqlFt.php` (класс `XenForo_Search_SourceHandler_MySqlFt` extends `XenForo_Search_SourceHandler_Abstract`), реализация для *Sphinx* уже доступна [дополнением](#) от стороннего разработчика на официальном форуме.

Система локализации

Система локализации практически полностью повторяет свой аналог из VBulletin с тем исключением, что фразовые группы отсутствуют вообще. Вот начало файла фраз английского языка:

```
<?xml version="1.0" encoding="utf-8"?>
<phrases>
    <phrase title="1_more_message" global_cache="0" version_id="1000017" version_string="1.0.0 Alpha 7"><![CDATA[1 more message]]></phrase>
    <phrase title="about" global_cache="0" version_id="1000015" version_string="1.0.0 Alpha 5"><![CDATA[About]]></phrase>
```

Каждое дополнение может иметь собственные фразы. Языки можно экспортировать. Переводить дополнения можно прямо в админке.

При внутренней компиляции шаблонов, фразы соответствующего языка вставляются прямо в PHP вариант, поэтому на загрузку фраз в них использованных, время не расходуется.

Если в своем дополнении вы часто используете фразу вне шаблонов (например, при каждой загрузке часто открываемой страницы), можно добавить ее в глобальный кеш. Если фраза используется редко (например, сообщение о критической ошибке), ее можно непосредственно загрузить из базы данных вызовом:

```
$myPhraseString = new XenForo_Phrase('phrase-name')
```

Кстати, русская локализация XenForo уже [практически готова](#).

Система стилей

Я не дизайнер и вообще у меня плохой вкус. Все, что касается дизайна и его разработки меня пугает. Но в XenForo мы видим древовидную систему стилей аналогичную VB, в которой мы можем взять какой-то стиль за основу и изменить часть его настроек или шаблонов, породив дочерний стиль.

Кир с Майком реализовали самый крутой редактор CSS, который мне приходилось вообще доводилось в жизни видеть. Dreamweaver отдыхает :). В XenForo настраивается все и настраивается вполне визуальное. Забудьте о редактировании CSS напрямую, о листе с двумя тысячами перемен-

ных, который какой-то нехороший человек реализовал в VB. Здесь все очень грамотно сгруппировано так, что 99% секций помещаются на одном экране. Если вы не такой лох в плане дизайна как я, и имеете художественный вкус, наверное, вы сможете уловить своего дизайнера, поскольку стиль сможете сделать и сами (через пару месяцев, как наиграетесь с редактором).

Все настройки стиля XenForo компилирует в соответствующие CSS файлы, которые подключаются по мере необходимости.

Разумеется, все настройки экспортируются и импортируются в виде *.xml файлов, так что в ближайшее время для XenForo появится большое количество стилей.

Система аутентификации

У вас есть база данных, скажем, от WordPress? Импортируйте пользователей из WP в базу XenForo с сохранением их паролей! Для того, чтобы потом все заработало, вам придется всего лишь расширить класс `XenForo_Authentication_Abstract` чтобы подсказать XenForo каким образом проверять пароль. Данные пользователей с вашими системами аутентификации связывает таблица `xf_user_authenticate`.

Службы связи

Уже достаточно давно на форумах позволяет указывать свои контакты в различных системах коммуникации типа ICQ. В XenForo существует легко расширяемая система контактных служб. Службы можно добавлять в панели управления. Класс поддержки служб выглядит примерно так:

```
<?php
class XenForo_Model_IdentityService_Icq extends XenForo_Model_IdentityService_Abstract
{
    protected function _getIdentityServiceId()
    {
        return 'icq';
    }

    static public function verifyAccountName(&$accountName, &$error)
    {
        if (!preg_match('/^\d+$/', $accountName))
        {
            $error = new XenForo_Phrase('please_enter_valid_icq_uin_using_numeric_characters_only');
            return false;
        }
        return true;
    }
}
```

Теперь уже никто не сможет ввести в поле ICQ своего профиля строчку «Продам носки шерстяные» :)

Система прав и привилегий

Каждая группа пользователей системы имеет права по умолчанию. Плюс права каждой группы можно дополнительно настроить в любом типе узла. Разумеется, привилегии наследуются.

Типов разрешений четыре: Inherit, Allow, Revoke, Deny. Revoke от Deny отличается исключительно тем, что если, например, в разделе форума для группы пользователей указано Deny, а в его подразделе Allow – доступ все равно будет запрещен. Deny нельзя переопределить в дочернем наборе прав, а ▶

Revoke можно. В остальном все похоже на VBulletin и другие движки.

После переключения форума в режим отладки, вы сможете добавлять в систему новые типы привилегий (дополнения тоже могут их добавлять). В коде они проверяются примерно так:

```
$userCanCleanSpam = (XenForo_Permission::hasPermission($visitor['permissions'], 'general', 'cleanSpam'));
```

Что же плохого?

На эту тему следует говорить с большой осторожностью, поскольку обзор касается исключительно самой-самой первой бета версии продукта.

Самым большим недостатком сейчас мне показалось отсутствие шаблонных хуков. Если дополнению требуется изменять шаблоны форума, администратор должен проделать это вручную, либо разработчик должен сделать это путем `str_replace/preg_replace` по коду шаблона, переопределив класс нужного контроллера. Оба способа не очень-то привлекательно выглядят. Кир сказал, что у него есть некоторые мысли по работе дополнений с шаблонами, но он не уверен, что успеет их реализовать в 1.0. Впрочем, это настолько важно, что я думаю, эту доработку можно ожидать в ближайшие месяцы три-четыре.

Еще один недостаток – отсутствие возможности расширять некоторые функции системы без редактирования исходников. В глаза сейчас бросается как минимум два таких момента: невозможность изменить поисковый движок (например, на Sphinx) и невозможность встроить иные платежные системы для премиум-подписок вместо PayPal, который в России крайне непопулярен.

Не существует стандартного механизма добавления новых BB-кодов в систему, как это сделано во всех форумных движках. Добавление BB-кодов возможно через дополнения, а удобный редактор реализован только для добавления BB-кодов медиаконтента вроде YouTube.

Пока нет удобного способа добавить свои кнопки на панель редактора сообщений. Впрочем, в качестве редактора использован *TinyMCE*, а он легко расширяется. Главное, найти место в XenForo куда можно вставить код расширения (боюсь, это место в шаблонах, которые пока автоматически не отредактировать).

Разработчики ждали автоустановку дополнений прямо из архивов с автоматической распаковкой файлов и импортом файла настроек дополнения, но она пока не реализована, как

и автоматическое обновление дополнений. Думаю, эти функции отложены на приличный срок. Авторам есть над чем пока потрудиться.

Не реализованным остался специализированный с особым функционалом раздел сайта XenForo для дополнений. Сейчас авторы публикуют их просто в виде постов в выделенном для этого разделе форума, прикрепляя как вложения.

Еще раз напоминаю, что авторы полностью в курсе изложенных отрицательных моментов и, скорее всего, исправят ситуацию в самое ближайшее время. Все же вышла пока только Beta 1.

Общие впечатления

Вообще код XenForo произвел на меня приятное впечатление. Он красивый, понятный, гармоничный, полон комментариев и ООП. Сначала посмотрел на количество классов, я собирался испугаться и отложить статью до лучших времен, но теперь, подбираясь к концу, я могу сказать, что уже почти все ОК, и я немного въехал, как тут все устроено. Надеюсь, что и вы немножко тоже.

Конечно, при внимательном взгляде заметны некоторые шероховатости, но в целом архитектура системы позволяет их ликвидировать с минимальными трудозатратами.

Кстати, я очень ждал выхода PHPBB3, хотел им заняться серьезно. Но его архитектура для меня выглядит гораздо менее четкой и логичной, чем архитектура XenForo. Честно признаться, я так и не въехал в Olympus. Будем ждать четверку на Symfony2...

Вот, пожалуй, и все. На *официальном форуме* уже есть несколько релизов (51 на текущий момент), можно скачать и посмотреть. Сам я пока лицензию на движок не покупал, не было повода. Поэтому за предоставленный для изучения код благодарю моих друзей, уже купивших официальную версию.

Дополнительные материалы

К сожалению, объем журнальной статьи ограничен, а еще так много хочется рассказать. Если разработка под XenForo Framework вас заинтересовала, отсылаю вас к *разделу разработчиков* на форуме Российского сообщества XenForo. Там будут публиковаться все дополнительные материалы, примеры и прочее. Если при прочтении этой статьи у вас возникли вопросы, вы можете задать их там же в *соседнем разделе*.

Я буду благодарен любым комментариям по поводу этой статьи. Присылайте ваши мнения и пожелания на email. ■



Home Buy Customers Community

A compelling community experience.

Intuitive. Social. Engaging. Fast. XenForo brings a fresh outlook to forum software.

[Buy XenForo Now](#)[Join the Community](#)



Social Engagement

Keep your users coming back by letting them earn trophies for reaching milestones. An intuitive "like" system makes users feel appreciated for their contributions.

Recent Activity Stream

Allow people to easily see all the recent happenings on your forum, beyond just the messages posted. Members can follow others to get their own personalized news feed.

Alerts

Make it easy for users to stay up to date with updates that are applicable to them. They'll receive alerts when someone quotes their post or responds to a status update, when they receive a new trophy, and more.

Easy Styling

You can use XenForo's simple color changer to easily change the look to suit your needs. You can make further changes through an extensive style property editor. No code changes necessary!

Have a question? [Contact us.](#)

©2010 XenForo Ltd.

[License Agreement](#) [Contact Us](#) [Home](#) [Top](#)

Веб-Аналитик.ИНФО ноябрь 2010

30

Автор

Денис Юсупов
Сетевой инженер
г. Санкт-Петербург
_dyr.livejournal.com
yusdyr@gmail.com

Перехват и редактирование файлов http-трафика на примере торрента

Пару лет назад возникла идея сделать локальный bittorrent-ретрекер для пользователей нашей «домашней» городской сети, чтобы и пользователи быстрее скачивали, и у нас меньше трафика было. Установкой самого ретрекера дело только начиналось, необходимо было как-то анонсировать его для скачиваемых торрентов. В процессе выяснения способов и механизмов анонса я пришел к достаточно общему и универсальному алгоритму, с которым и предлагаю познакомиться.

Что делать

Есть три способа объявить о существовании трекера в локальной сети:

- Использовать добавляемыми некоторыми трекерами адрес `retracker.local`.
- Анонсировать локальный трекер посредством механизма `isp.bep22`.
- Перехватывать скачиваемые торрент-файлы и редактировать их, добавляя адрес нашего торрента «на лету».

Каждый из них имеет свои преимущества и недостатки, соответственно:

- Использование зоны `.local` противоречит черновику RFC «Multicast DNS» и вызывает проблемы в работе `zeroconf`-сервисов Linux и Apple; добавляется лишь немногими трекерами.
- `isp.bep22` работает, насколько мне известно, лишь в клиенте `µTorrent` и то выключен по умолчанию.
- О перехвате трафика нет никаких упоминаний с *success stories*, за исключением единственного опыта дружественной сети.

Для начала сделали поддержку первых двух вариантов, благо никаких особых усилий для этого не требовалось: добавить в DNS несколько записей это просто (`retracker.local IN A` и `retracker.smarthome.spb.ru _SRV_`).

Для работоспособности .local у пользователя и нашего ретрекера нужно создавать зону не .local, а .retracker.local, что как раз и позволяет совместить оба варианта

Но все же наиболее интересным и заманчивым выглядел третий вариант, так что я решил поискать информацию об общей технике изменения скачиваемых файлов на лету. Требования к серверу были просты:

- работа на FreeBSD,
- open-source,

- прозрачность (незаметность) работы для клиента,
- определение трафика 7 уровня и восстановление файлов из http-потока,
- передача этих файлов редактирующему скрипту и получение их обратно,
- передача отредактированных файлов клиенту.

Чем делать

К своему удивлению я обнаружил, что техник и open-source программ для такого перехвата и редактирования файлов практически нет. По большому счету, их всего две: это Squid с экспериментальными модулями ICAP/ECAP и некий фильтрующий прокси под названием «MiddleMan», последний релиз которого вышел в далеком 2004 году, но который продолжает поддерживаться в портах.

От использования Squid я отказался практически сразу: несмотря на наличие сразу двух экспериментальных модулей для работы с проходящим трафиком, решение оказалось чрезвычайно «кривым» и неустойчивым даже в установке и настройке, не говоря уж о работе.

Перешел к MiddleMan. Поразительно, но факт — старая программа оказалась функциональней и удобней современного монстра Squid. В сущности, он удовлетворяет всем требованиям, кроме полной прозрачности для пользователя — `source ip` пользователя переприсваивается на `ip` сервера с прокси. Замечу, что возможность оставлять `source ip` есть только у Squid с модулем `TProxy` под Linux. Более того, у него есть уникальная опция — при превышении настроиваемого таймаута ожидания, прокси отдает пользователю неизменный исходный файл.

Как делать

1. Определение самых популярных торрент-серверов

Для начала я написал небольшой *perl'овый скрипт*, который через `rsync` слушает 80 порт и собирает `ip`-адреса, на которые идут запросы с «Content-Type: application/x-bittorrent». Нужно это для того, чтобы перехватывать не весь `http`-трафик, а только тот, который принадлежит крупным трекерам. Затем путем нехитрых манипуляций эти `ip`-адреса заносятся в таблицу `ipfw`, используемую при перенаправлении на наш прокси:

зуюмую при перенаправлении на наш прокси:

```
{ipfw} add fwd ${proxy_ip},
${proxy_port} tcp from $lan_
customers to 'table(15)' dst-
port 80 in via ${int_if}
```

2. Конфигурация прокси-сервера middleman

Секция, ответственная за отдачу файлов редактирующему скрипту, называется `external` в *mmn.xml*.

3. Редактирующий «внешний» скрипт

Скрипту *mypatcher.pl* передаются файлы с `mime`-типом «`application/x-bittorrent`», он добавляет в него записи локального трекера (убирая при этом `retracker.local`, если он там есть, что решает проблемы клиентов с `zeroconf`) и передает содержимое обратно прокси, попутно сохраняя файл еще и на диск, в таком виде:

```
#for i in `find /home/torrents/
patched/ -type d`; do echo -n «$i»
&& ls -l $i | wc -l; done | awk
'{print $2} «$1»' | sort -rn | head
-n 10

24103 /home/torrents/patched/
dl.ru.tracker.org
7817 /home/torrents/patched/dl.tor-
rents.ru
6184 /home/torrents/patched/tfile.ru
3744 /home/torrents/patched/kino-
zal.tv
2928 /home/torrents/patched/ru-
tor.org
2872 /home/torrents/patched/tor-
rents.thepiratebay.org
2583 /home/torrents/patched/www.
tfile.ru
```

Итог работы: на сервере, который выполняет NAT, шейпинг и роутинг сети из 3000 пользователей, загрузка `mmn` вообще не ощутима. В день сейчас таким образом редактируется порядка 200-400 файлов. Нареканий за почти год работы не было, все довольны. ■

«Сохранение файлов на диск» служит исключительно для сбора обезличенной статистики в виде, указанном выше. Ну и попросту я забыл отключить этот механизм отладки скрипта :)

**Автор**

Максим Балдин
Руководитель отдела разработки
программного обеспечения
ООО «Все для принтера»
г. Санкт-Петербург
www.fprints.ru/
deusmodus.habrahabr.ru/
maximebaldin@gmail.com

Подход к оптимизации приложения на примере популярной CMS

Статья может помочь вашему живому проекту не тормозить или тормозить меньше, либо стать отправной точкой для исследования стороннего продукта. Например, у вас стоит задача понять, что же происходит внутри «Самописная система 3.14» и, опять же, помочь ей не есть по 100 мегабайт RAM на одного клиента.

Об исследуемой программе

WebAsyst Shop-script – это вторая попытка ребят из ООО «Артикус» сделать «хорошо». Первой попыткой был дырявый и приносящий многим и по сей день проблемы Shop-script Premium.

Строго говоря, WebAsyst – это целый комплекс программ типа блокнота, календаря и менеджера проектов, но человеку, который не первый день в интернет-разработке или бизнесе эти решения вряд ли будут интересны (basesamp же). О том, увенчалась их попытка успехом или нет, я могу сказать так – совсем недавно мы отпраздновали 666-ю ревизию альтернативной ветки и это не конец.

Цели

Целью является выявление наиболее ресурсоемких операций и определение состояния системы при критических объемах данных. Под данными я подразумеваю количество категорий и товаров. В некоторых случаях будут даваться рекомендации по оптимизации, но видя источник беды, нейтрализовать ее уже не так сложно.

Подготовка

Структура директорий и зависимости

Совсем недавно я открыл для себя included расширение, которое дружит с Graphviz, что и вам советую, иначе понять что практически самый главный компонент программы находится по адресу
`\published\SC\html\scripts\modules\test\class.test.php`
и зачем нужен
`\published\SC\html\scripts\modules\test\_methods\search_simple.php` будет нудно и неинтересно.

Я пошел по пути грер, так как у меня было время и необходимость познать суть, но больше так не хочу и вам не советую. Особенно учитывая их эксперименты с callback, от которых мне до сих пор дурно.

Наполнение содержимым

Глупо было бы спорить, что тестирование (если это только не alpha стадия разработки) на малом объеме данных не имеет смысла. Поэтому в первую очередь наполните вашу CMS содержимым под завязку. Если нет реальных данных, то напишите спамера, который пишет в БД пока может или пока вы этого хотите. WebAsyst SS совершенно по-разному

ведет себя на каталогах с 450-ю и 4500-ю товаров.

Перегрузка стандартных функций

Я буду говорить не о перегрузки с точки зрения ООП, а перегрузке в лоб. Метод прост – проходимся по всем файлам в поисках стандартной функции, заменяем ее на `ov_{original_name}` и тут уж все карты открыты. Хотим – логируем все запросы к базе, хотим – смотрим кто и когда стучится на `fopen`, `fwrite`, `file_get_contents` или пытаемся воспользоваться черной магией, вроде `eval`. Самым полезным оказывается логирование именно `mysql_query`, так как обычно производительность упирается в бэкенд.

У меня используется что-то такое:

```
function ov_mysql_query($text) {  
    $debug=false;  
    if($debug){  
        $source=debug_backtrace();  
        $src_array_count=count($source);  
        $what=array('\r\n','=',' ','?','&','+',',',  
            '(', '/', ')');  
        $to=array('\r\n','_','_','_','_','_','_','_','_','_');  
        $filename=str_replace($what,$to,$_SERVER['REDIRECT_URL']);  
        static $function_counter_m = 0;  
        $function_counter_m++;  
        $oldDir=getcwd();  
        chdir($_SERVER['DOCUMENT_ROOT']);  
        $fp=fopen('logs/'.$filename.'.log','a');  
        fwrite($fp,$function_counter_m.'  
' . str_replace('\r\n',' ',trim($text))."\r\n");  
        for($i=0;$i<$src_array_count;$i++) {  
            fwrite($fp,'DEBUG INFO: '.$source[$i]  
                ['file'].' | '.$source[$i]['line']."\r\n");  
        }  
        fwrite($fp,""\r\n");  
        fclose($fp);  
        chdir($oldDir);  
    }  
    $q=mysql_query($text);  
    return $q;  
}
```

Как результат работы, в папке `www/logs` сохраняется файл с отладочной информацией (стек вызовов) о запросе и сам запрос.

xDebug

Если честно, то мне сложно назвать отладкой попытку разобратся в чужом механизме. Скорее это препарирование. Однако в зависимости от того, стоит ли у вас отладчик, напрямую зависит сможете ли вы выявить узкие места и оптимизировать систему. Если вы пишете программы на php, то вам необходим xDebug, он бесплатен и его поддерживают все хоть чуть-чуть уважающие себя редакторы php кода.

Отладчик генерирует в установленной вами директории специальные дампы, в которых хранятся различные данные (опционально). Так как основной ОС у меня служит Windows, у вас может быть преимущество на этом шаге, так как Linux'овский `kcachegrind` намного удобнее `wincachegrind` (обе программы позволяют просматривать эти дампы, хотя по правде, это обычные `txt` файлы, прочесть которые можно и через блокнот при должной суровости).

Начнем же тормозить зомби.

Тестовый стенд

Используемая версия WebAsyst:
287 (чистая, без патчей и модов)
Количество товаров в БД: 4461
Количество типов характеристик в БД: 144
Количество значений характеристик в БД: 44170
Количество категорий в БД: 354
Количество картинок в БД: 3516

Немного о начальной конфигурации

Результаты работы чистого движка на дефолтном одном товаре и одной категории. В программе присутствуют штатные механизмы кеширования, но насколько они себя оправдывают, вы можете судить сами (табл. 1).

Если у вас еще не встали волосы дыбом, читайте дальше и не забывайте, что здесь были всего один товар и одна категория.

Конфигурация с данными

Настало время использовать наш тестовый стенд с несколькими тысячами товаров и мощной иерархией каталогов (табл. 2).

Если главная страница еще хоть как-то остается при своих 64-ех запросах (конструкциями `IN(a,b,c,d,...,z)`), то категорию немного колбасит, а подбор по характеристикам уничтожит не то что обычный хостинг, но и VPS. Но вы же не думаете, что отключение расширенного поиска вам поможет? В данном программном продукте есть несколько недокументированных возможностей, которые в руках конкурентов

Таблица 1

Страница*	Запросов к БД с дефолтным кешем	Запросов к БД без дефолтного кеша	Скорость загрузки с дефолтным кешем**	Скорость загрузки без дефолтного кеша**
Главная	64	73	10,304	17,011
Категория	83	90	10,616	19,457
Товар	100	107	15,010	28,731
Поиск (успешный)	69	76	10,507	18,209

*ссылка ведет на скриншот страницы, чтобы было понятно, соразмерны ли запрашиваемые объемы данных с отображаемыми

**cumulative time из Wincachegrind

Таблица 2

Страница	Запросов к БД с дефолтным кешем	Запросов к БД без дефолтного кеша	Скорость загрузки с дефолтным кешем	Скорость загрузки без дефолтного кеша
Главная	64	73	12,323	19,404
Категория	186	193	20,333	29,881
Товар	108	115	16,156	30,100
Поиск (успешный)	69	76	20,733	25,162
Подбор по характеристикам (расширенный поиск)	900	907	43,216	50,242

Таблица 3

Страница	Запросов к БД с дефолтным кешем	Запросов к БД без дефолтного кеша	Скорость загрузки с дефолтным кешем	Скорость загрузки без дефолтного кеша
Категория (/all/)	241	248	430,049	439,102

могут осложнить жизнь. Об этих возможностях можно узнать копаясь в классе, отвечающем за обработку URL (`class.furl.php`). Например, можно безостановочно долбить запросом `store.ru/category/category_with_lot_products/all/`. У меня в такой категории в верхнем уровне 113 страниц (табл. 3).

Промежуточный итог

На текущей стадии исследования мы знаем:

- Есть возможность создавать потенциально высокую нагрузку
- Количество запросов к базе данных с кешем и без слишком велико

Также, если посмотреть на дамп, созданный отладчиком при загрузке страницы `store.ru/category/category_with_lot_products`, можно с уверенностью выделить две самые прожорливые операции:

```
foreach ($Interfaces as $_Interface) {
    ModulesFabric::callInterface($_Interface);
}
```

и

```
print $smarty->fetch($CurrDivision->MainTemplate); ▶
```


Помимо них очень много ресурсов тратится на получение дерева категорий, 95 с лишним тысяч раз вызывается `is_object`, 70 тысяч раз программа спрашивает `LanguagesManager::getInstance` и 28 с лишним тысяч раз считает длину строки, а вызовы `LanguagesManager::ml_isEmpty` составляет 2/3 самой медленной операции – `getExtraParametrs`.

Варианты решения проблем



Легкий

Если у вас не так много посетителей, но программа тормозит, то можно воспользоваться файловым кешированием с минимальным временем интеграции.

Я предлагаю такую схему:

1. Находим тяжелую функцию.
2. Определяем, не зависит ли она от каких-либо глобальных переменных.
3. Переименовываем ее во что-то вроде `{original_function}_cached`.
4. Создаем `{original_function}`, в теле которой вызывается через специальную функцию `{original_function}_cached`.

На ранних стадиях оптимизации, когда нужно было чтобы программа работала быстро и времени не хватало я использовал такое решение:

```
function cache_function($buildCallback, array
$args = array(), $timeoutHours = 5){
    $oldDir=getcwd();
    chdir($ _SERVER['DOCUMENT_ROOT']);
    // Устанавливаем имя кеш-файла
    if(is_array($buildCallback)){
        $cacheKey = get_class($buildCallback[0])
        . ':: ' . $buildCallback[1];
    }
    else{
        $cacheKey = $buildCallback . ' : ' . serialize($args);
    }
    $cacheKey .= ' : ' . serialize($args);
    if(!file_exists('functions_cache/' . $buildCallback . '/')) {
        @mkdir('functions_cache/' . $buildCallback . '/');
    }
    $file_path = 'system_cache/' . $buildCallback . '/ ' . sha1($cacheKey);
    if(!file_exists($file_path) OR filemtime($file_path) < (time() - $timeoutHours*60)){
        $result = call_user_func_array($buildCallback, $args);
        file_put_contents($file_path, serialize($result), LOCK_EX);
    }else{
        $result = unserialize(file_get_contents($file_path));
    }
    chdir($oldDir);
    return $result;
}
```



Получаем:

```
function original_function($arg1,$arg2){
    return cache_function('original_function_cached',array($arg1,$arg2),10);
}
```

Как результат в директории `www/functions_cache/original_function_cached` появится сериализованный результат выполнения функции `original_function_cached` и будет использоваться 10 часов.



Трудный

Как бы мы не кешировали результаты выполнения функций, у нас все равно останется ресурсоемкий `fetch`, который собирает из десятка шаблонов, использующих десятки контроллеров и плагинов единую страницу. Тут я бы предложил оптимизировать количество шаблонов, создать для них нормальную иерархию (по умолчанию все шаблоны хранятся скопом) и начать двигаться в сторону блочного кеширования. Таким образом, на самых посещаемых страницах мы увидим немаленький прирост скорости.



Очень трудный

Но если у вас, как и у меня нет иного выбора, кроме как работать с WA и работать с ним вы будете еще долго, то все это полумеры. Необходима оптимизация, переписывание алгоритмов (посмотрите на досуге как у них реализована пагинация) и не хаковое кеширование. Вообще, мне в этом плане проще, так как я знаю, что новый контент добавляется автоматически в определенное время, и в это время я могу себе позволить сбросить весь кеш. Вам же, чтобы бороться с инвалидацией цен, характеристик, придется настраивать группы кешей и очень много менять (от формирования URL, до реструктуризации директорий). В большинстве проблем, конечно, можно справиться с помощью smarty, но работу программы со smarty придется здорово перестроить, так как сам WebAsyst SS кажется и не собирался использовать его механизмы кеширования (все говорит об этом).

Например: мы закешировали всю страницу с товаром, и установили время жизни 5 часов. Предполагается, что цена может поменяться и раньше, а сбрасывать кеш не очень хочется. Можно создать smarty-plugin, который обратится к нужному методу нужной модели (скажем `$productModel->getPrice($pID)`) и вернет цену. На страницу с товаром мы получаем 1 запрос к БД. Кеш представления не перестраивается.

Заключение

Надеюсь, что готовые решения и рекомендации из данной статьи натолкнут вас на что-то новое (будь то `included` или `xDebug`, или правило не верить на слово разработчикам, которые говорят, что все обращения к БД у них идут через класс) или помогут развить старые идеи. ■

Типографика и



Автор

Ольга Рунова
Программист
Компания Netrika
г. Санкт-Петербург
<http://spbnet.ru>
olga.runo@gmail.com

Когда человек, не разбирающийся в искусстве, смотрит на картину, он видит только то, что на ней нарисовано и не понимает, как это нарисовано. Если на картину смотрит художник или искусствовед, он видит отдельные детали, мельчайшие оттенки и мазки краски, возможно даже представляет себе, как и в какой последовательности они наносились.

Это относится и к шрифтам. Многие люди просто читают текст и не обращают внимания на собственно шрифт. Когда тот же самый текст видит дизайнер, он замечает детали шрифта, ширину символов, «капли» на знаках, засечки, контрастность линий, рисунок каждой буквы и многое другое.

Использование определенного шрифта должно быть чем-то обусловлено, каждый шрифт используется для своей цели: текстовый шрифт, самый незаметный, для удобочитаемости при больших объемах текста, чтобы глаз читателя не уставал. А заголовочные и декоративные шрифты, заметные — для акцентного выделения отдельных фраз и заголовков, чтобы привлечь внимание читателя, но не рассчитанные на восприятие больших кусков текста.

Многие ли люди задумываются над отображением шрифтов? Многие ли умеют их различать и отличать семейства шрифтов? А многие ли люди понимают, какой шрифт, где и зачем используется?

Казалось бы, шрифт — это то, что мы видим очень часто в повседневной жизни, но тем не менее, шрифту уделяется мало внимания, поскольку кажется, что он незаметен и не играет роли. Поэтому для привлечения внимания зрителя часто делают акцент не на шрифт, а на других графических элементах, картинках и цветовой гамме. Однако можно создать красивый эффект оформления используя только шрифт. Например, всем известный шрифт Helvetica.



Иногда встречаются варианты оформления с большим количеством различных шрифтов на странице, которые не только не сочетаются между

собой, но и не воспринимаются глазом. Важно знать, что грамотное использование шрифтов, выбор шрифта, размещение текста на странице, определение интервалов между элементами текста и текстовыми блоками очень сильно влияет на воспринимаемость информации. При неправильном подборе этих параметров глаз читателя устанет, а грамотный типограф испытает раздражение и недовольство.

Использование шрифтов связано с историей развития печатного дела. Многие из тех шрифтов, которые используются сейчас, появились не просто так, они имеют давнее происхождение и за долгое время модернизировались.

Например, происхождение моноширинных шрифтов связано с тем, что раньше использовались печатные машинки, у которых была одинаковая ширина кегельной площадки символов. Происхождение динамических шрифтов связано с рукописным письмом, при котором наблюдается контрастность штрихов за счет разной силы нажима на перо при письме, и разной толщине самого пера. Применение различных шрифтов создает определенную стилистику оформления.

Сейчас шрифты часто используются для создания декоративных элементов, и в частности, при создании логотипов. Например, на логотипе этой компании можно увидеть слово FLOOK на картинке в изображении рыбы.



А здесь в качестве изображения логотипа использована одна буква «а», которая напоминает пингвина.



Использование различных гарнитур следует учитывать при проектировании дизайна, так как в разных операционных системах есть свой набор стандартных шрифтов, и если в .css файле не указана одна из тех гарнитур, отображаемых на данном компьютере, то

система выберет одну из своих гарнитур по умолчанию, и результат отображения будет не таким, каким его задумывал дизайнер.

Можно выделить несколько удачных на сегодняшний день шрифтовых гарнитур.

Шрифт **Georgia** (был разработан Matthew Carter в 1993 г.). Данный шрифт одинаково хорошо смотрится в мелких и в крупных кеглях, на мониторах с любым разрешением, а также как с применением сглаживания, так и без него.

Шрифт **Lucida Grande** (был разработан Charles Bigelow и Kris Holmes для компании Apple). Данный шрифт входит в большое семейство Lucida, используемых во всех основных компонентах системы MacOS X.

Шрифт **Myriad Pro** (был разработан специалистами компании Adobe (Robert Slimbach и Carol Twombly)). Данный шрифт имеет очень богатый набор начертаний, а также отдельные версии для капители, лигатур и альтернативных символов. Myriad является коммерческим, и не входит в стандартные поставки операционных систем, зато прилагается ко всем ключевым продуктам компании Adobe.

Шрифт **Trebuchet MS** (был разработан в 1996 году дизайнером компании Microsoft Vincent Connare). Данный шрифт является неплохой альтернативой другим гротесковым шрифтам Microsoft. Он одинаково хорошо смотрится как в заголовках, набранных крупными кеглями, так и в качестве основного текста.

Совсем недавно компания Паратайп выпустила новый текстовый шрифт **PT Sans**, который был спроектирован А. Корольковой, автором вышеупомянутой книги. Гарнитура PT Sans является частью проекта по созданию комплекта бесплатных общедоступных российских шрифтов с открытой пользовательской лицензией. PT Sans — это шрифт класса гротесков, в своей стилистической основе базирующийся на рубленых шрифтах.

Гарнитура состоит из 8 начертаний: 4 стандартных начертания; 2 начертания с расширенными пропорциями для мелкокегельного набора; 2 узких начертания для экономного набора. ■

PT Sans Regular	PT Sans Narrow
PT Sans Italic	PT Sans Narrow Bold
PT Sans Bold	PT Sans Caption
PT Sans Bold Italic	PT Sans Caption Bold

Автор

Семен Приходько
AlterMoby
<https://www.altermoby.com/>
г. Винница, Украина
proza.ru/avtor/semion
semion.ababo@gmail.com

Разработка web-приложений на языке Common Lisp

Данный обзор является небольшим путеводителем для тех, решил(и) доверить этому чудесному языку будущее своего стартапа. Несмотря на то, что основной акцент будет ставиться на web-разработке, я постараюсь осветить также и более общие темы, так или иначе связанные с Common Lisp. Материал почерпнут из собственного опыта разработки web-сервиса AlterMoby. Мы рассмотрим web-сервер Hunchentoot, разберем его архитектуру и базовые возможности. Кроме того, затронем некоторые смежные вопросы, в частности, генерацию HTML/XML.

Выбор web-сервера

После развертывания минимальной Lisp-системы нужно выбрать web-сервер. Этот вопрос не очевиден – тот же CLiki дает ссылки на семь разных библиотек. Когда я выбирал web-сервер для работы, то исходил из нескольких факторов. Во-первых, он должен быть известным в CL-сообществе, поддерживаться широким кругом энтузиастов и находиться в актуальном состоянии вплоть до нынешнего времени. Во-вторых, на нем должен базироваться хотя бы один перспективный web-фреймворк. Кроме того, желательно удостовериться в наличии коммерчески успешных сайтов на его основе. Исходя из этих критериев, в поле моего зрения попало всего два продукта: Portable AllegroServ и Hunchentoot.

Portable AllegroServ является версией AllegroServ – оригинального web-сервера, написанного специально для Allegro CL. Исходный код последнего не совместим с Common Lisp, поскольку использует нестандартные расширения родного компилятора. Дело в том, что автор AllegroServ стремился достичь максимальной эффективности, потому и был вынужден отходить от строгого стандарта. В итоге получился быстрый и элегантный web-сервер, который, наверное, и по сей день является лучшим выбором для Allegro CL. Поскольку исходный код AllegroServ выпущен под лицензией LGPL, нашлись энтузиасты, адаптировавшие его к стандартному Common Lisp. Получившейся продукт назвали Portable AllegroServ. По мне-

нию некоторых лисперов, последний уступает оригиналу как в производительности, так и в надежности.

Главный герой нашего обзора, web-сервер со странным названием Hunchentoot изначально ориентировался на стандартный Common Lisp. Потому он «прямо из коробки» может работать с большинством популярных компиляторов. Для меня существенным аргументом в пользу Hunchentoot являлся также тот факт, что на его основе базируется перспективный web-фреймворк Weblocks. Несмотря на то, что я отказался в итоге от использования этого фреймворка в своем проекте, Hunchentoot меня несколько не разочаровал. Далее рассмотрим структуру и функционал этого прекрасного продукта.

Знакомство с Hunchentoot

Прежде всего, удостоверимся в том, что мы поставили Hunchentoot:

```
(asdf:oos 'asdf:load-op :hunchentoot)
```

Теперь создадим и запустим экземпляр класса `acceptor` – приемщика `http`-запросов:

```
(hunchentoot:start (make-instance 'hunchentoot:acceptor :port 8080))
```

Чтобы Hunchentoot мог принимать `https`-запросы, нужно заменить `'acceptor` на `'ssl-acceptor` (его потомка), не забыв указать файлы сертификата и ключа.

Рассмотрим простейший обработчик, взятый из справки по Hunchentoot:

```
(hunchentoot:define-easy-handler (say-yes :uri </yo>) (name)
  (setf (hunchentoot:content-type*) «text/plain»)
  (format nil «Hey~@[ ~a~]!» name))
```

Данный обработчик называется `say-yes`, привязывается к URL `/yo` и принимает единственный параметр `name` (через GET или POST). Обработчик генерирует текстовый документ, содержащий «Hey имя!», где `имя` – значение параметра `name`, или же просто «Hey!», если параметр `name` не задан (т. е. равен `nil`). Если вас смущала замысловатость форматирования строки, познакомьтесь с директивами функции `format`. Проверьте работу данного обработчика, перейдя на `/yo` и `/yo?name=Vasia`.

Вместо текста обработчик может отдавать нужный файл:

```
(hunchentoot:define-easy-handler (get-file :uri </file>) (name)
  (handle-static-file (format nil «/home/me/folder/~a» name)))
```

В принципе, этого функционала должно хватить для на-



писания небольшой любительской поделки. Более серьезная работа потребует отказаться от макроса `define-easy-handler`, разобравшись в механизме диспетчеризации запросов. Обрисуем основные черты последнего.

Каждому экземпляру класса `acceptor` (а значит и класса `ssl-acceptor`) соответствует диспетчер запросов. Диспетчер запросов представляет собой функцию, принимающую экземпляр класса `request` (`http-запрос`) и возвращающую выходной `http-поток` (`HTML/XML` или бинарные данные). Поскольку выходной поток зависит от заданного `URL`, на практике диспетчер запросов его не генерирует, а занимается поиском соответствующего обработчика, которому и делегирует эту роль. По умолчанию диспетчер запросов просматривает список `*dispatch-table*`, содержащий функции диспетчеризации. Каждая функция диспетчеризации принимает объект запроса, анализирует его согласно своим критериям и в случае соответствия возвращает обработчик – функцию, генерирующую выходной поток (при несоответствии возвращается `nil`). Таким образом, стандартный диспетчер запросов по очереди запускает функции диспетчеризации до тех пор, пока одна из них не вернет обработчик. В конце `*dispatch-table*` обычно располагается `default-dispatcher` – функция диспетчеризации, всегда возвращающая обработчик (по умолчанию это сообщение о найденной странице).

Вышеописанная схема диспетчеризации может показаться несколько переусложненной, однако на практике такая структура позволяет добиваться большой гибкости. В частности, подход с возвратом обработчика позволяет писать элегантные функции диспетчеризации. Диспетчеризация обработчиков, созданных макросом `define-easy-handlers`, использует стандартную схему: `*dispatch-table*` содержит функцию диспетчеризации `dispatch-easy-handlers`. Последняя реализует свой упрощенный диспетчер, ищущий в собственном внутреннем списке. Этот список содержит описания всех обработчиков, определенных с помощью `define-easy-handlers`. Таким образом, диспетчеризацию можно разбивать на множество независимых веток с собственной логикой выбора обработчика.

Hunchentoot определяет несколько генераторов стандартных функций диспетчеризации, а также некоторые стандартные обработчики. Например, `create-prefix-dispatcher` по заданному префиксу `URL` и обработчику, генерирует функцию диспетчеризации, проверяющую `URL` запросы на соответствие заданному префиксу. Функция `create-regex-dispatcher` аналогична предыдущей, но генерирует функцию диспетчеризации, сопоставляющую `URLs` заданному шаблону регулярного выражения. Функция `create-folder-dispatcher-and-handler` принимает префикс `URL` и путь к директории, возвращая функцию диспетчеризации, раздающую файлы из заданной директории. К стандартным обработчикам относится `handle-static-file`.

Гибкость Common Lisp позволяет налету переопределять существующие функции и методы, что не может не способствовать гибкости Hunchentoot. Так можно не только создать

новый потомок класса `acceptor`, но и переопределить диспетчер по умолчанию, полностью отказавшись от вышеописанной схемы поиска обработчиков, или полностью изменить логику обработки запросов. Например, в процессе работы над своим проектом я заставил Hunchentoot проверять формат запросов `multipart/form-data` еще в процессе закидки данных, а не после того, как они будут полностью приняты. Это позволяет избежать ситуации, когда при `DoS-атаке` множество машин постанет на сервер многомегабайтный мусор.

Являясь мощным и современным web-сервером, Hunchentoot поддерживает множество стандартных функций. К ним относятся поддержка `cookies`, удобные отладка и логирование, и многое другое. Описание этого функционала вы сможете найти на его web-странице.

Генерация HTML/XML

В заключение обзора рассмотрим библиотеку `CL-WHO`, определяющую простой и удобный DSL для генерации `HTML/XML`. Как водится, лучше один раз увидеть:

```
(push :tag3 *html-empty-tags*)

(with-html-output-to-string (http-stream)

  (:tag1 :attr1 1 :attr2 «2»

    (:tag2 :attr3 (+ 1 2))

    (loop for i from 1 to 3 do

      (with-html-output (http-stream)

        (:tag3 :attr4 (when (oddp i) «odd»))))))
```

Результат выполнения этой конструкции не противоречит интуиции:

```
«<tag1 attr1='1' attr2='2'><tag2 attr3='3'></tag2>

<tag3 attr4='odd' /><tag3 /><tag3 attr4='odd' /></tag1>»
```

Как видите, макрос `with-html-output-to-string` строит XML-конструкцию из открывающих и закрывающих тегов, соответствующих данному s-выражению. Первая команда добавляет тег `tag3` в список одиночных тегов. Для такого тега при отсутствии вложений не будет генерироваться закрывающий тег. Далее вызывается макрос `with-html-output-to-string`, являющийся оберткой для `with-html-output` – основного макроса `CL-WHO`. Из примера видно, что допускаются произвольные управляющие конструкции для автоматизации генерации кода.

Результирующая строка была создана в режиме `XML`, в этом режиме стоит генерировать `XML/XHTML` документы. Для работы с `HTML` есть режим `SGML`, в котором пустые теги не содержат закрывающий слеш, а атрибуты могут быть без значений. ■

СЛУШАЕМ ПОДКАСТЫ НА ВЕБ-АНАЛИТИК.ИНФО

www.web-analitik.info/podcast/

- РУНЕТОЛОГИЯ с Максимом Спиридоновым
- ВРЕМЯ НОВОСТЕЙ с Владом Филатовым и Сергеем Болисовым
- PODCAST9 с Петром Диденко и Михаилом Черномордиковым
- APPLE МАНИЯ с Владом Филатовым и Сергеем Болисовым
- UBUNTU с Валентиной Мухамеджановой (Ubuntu.ru)
- Новости Hi-tech
- Новости IT



ДОБАВИТЬ СВОЙ
ПОДКАСТ





Автор

Сергей Дмытрив
Lemberg Solutions
Должность: Team Lead
lemborg.co.uk
г. Львов, Украина
slik.jay@gmail.com

Вот и вышел долгожданный релиз MooTools 1.3. Расскажу немного об изменениях в самом фреймворке, и в его «окружении». Между версиями 1.2 и 1.3 довольно много различий, но обо всем по порядку.

Первое что надо сказать, это то, что существует сборка версии 1.3, которая 100% совместима с 1.2, можно не бояться обновляться.

Slick

И так, об отличиях. Самая вкусная вещь в новой версии – это новый css selector engine Slick. Он полностью независим от MooTools, может использоваться отдельно. Из коробки поддерживает все мыслимые и немыслимые селекторы/псевдоселекторы, а если вам и этого покажется мало, ничего страшного, всегда есть возможность создать новый.

Примеры использования вне MooTools:

```
Slick.search(document, «div>p.moo:not(.foo)»);
// возвращает набор (массив) элементов, даже если
под выборку попал только один, если не было вы-
брано ни одного элемента, возвращает null. Первым
параметром идет контекст поиска, вторым селек-
тор.
```

```
Slick.find(document, «div > p.moo:not(.foo)»);
// почти то же самое, только теперь возвращается
только один, первый попадающий под селектор эле-
мент
```

```
Slick.match(node, «p.moo»); // проверяет на со-
ответствие node указанному селектору
```

```
Slick.contains(node, node2); // проверяет вклю-
чен ли node2 в node
```

Внутри MooTools же селекторы используются как обычно.

А теперь лаконично:

- Pure-javascript
- Быстрее на 50% чем старый
- Поддерживает любой селектор, о котором только можно подумать
- Гибкий, позволяет создавать свои псевдо-селекторы
- Поддерживает поиск по XML

Изменения в API

Ядро

\$методы теперь в прошлом:

```
$chk(value) => (value != null)
```

```
$A(item) => Array.from(item)
$splat => Array.from
$clear => придется использовать нативные clearTimeout
или clearInterval
$defined => (value != null)
$each => Array.each / Object.each в зависимости
от контекста
$empty => удалено, надо использовать function() {}
$extend(source, extended) => Object.append(so-
urce, extended)
$merge({}, a, b) => Object.merge({}, a, b)
$mixin(a, b) => Object.merge(a, b)
$lambda => Function.from
$random => Number.random
$time => Date.now
$type => typeof с одним изменением, для пустых объ-
ектов теперь возвращается строка "null"
$unlink => Array.clone / Object.clone
$arguments => удалено
$pick => Array.pick или [a, b, c].pick()
$exec => Browser.exec
```

А так же:

Native теперь именуется Type

Array.type / String.type / Number.type / и т.д. =>
Type.isArray / Type.isString / Type.isNumber / и т.д.

Hash и \$H перенесены в MooTools More.



Browser detection

```
Engine detection => user-agent detection.
Browser.Engine => удалено
Browser.Engine.trident => Browser.ie
Browser.Engine.gecko => Browser.firefox
Browser.Engine.webkit => Browser.safari или
Browser.chrome
Browser.Engine.presto => Browser.opera
Browser.Platform.ipod => Browser.Platform.ios
```



Array

```
Array.extend => Array.append ▶
```

**Function**

```
myFn.bind(this, [arg1, arg2, arg3]) => myFn.
bind(this, arg1, arg2, arg3) ИЛИ myFn.pass([arg1,
arg2, arg3], this)

$try => Function.attempt

myFn.run(args, bind) => myFn.apply(bind, Ar-
ray.from(args));

myFn.create => используйте .pass, .bind, .delay,
.periodical

myFn.bindWithEvent => устарело
```

**Element**

```
element.injectInside, .injectBefore, .inject-
After, etc => element.inject(context, where);

element.grabTop,... => element.grab(context,
where)

element.hasChild(item) => element.contains(item) &&
item != element

Selectors.Pseudo=>Slick.definePseudo(name, fn)

Element.get => больше не поддерживает второго пара-
метра (это используется для анимации, например element.
get('tween', options) теперь element.set('tween',
options).get('tween'))
```

С помощью Slick создавать новые элементы стало про-
ще:

```
new Element(<input>, {<id>: <someID>, <class>:
<someClass1>, <disabled>: true}); // Было

new Element(<input#someID.someClass1[disa-
bled=true]>); // Стало
```

Новые возможности

Type.from (Array.from, String.from, Function.
from и т. д.) => преобразование типов

Fix => унифицированный таймер для всех эффектов

**Окружение**

MooTools Test Runner => теперь тестирование проходит в
автоматическом режиме не только в каждом браузере, но и
на сервере

Packager => новый инструмент для сбора MooTools под
свои нужды.

**Packager**

Packager был разработан для управления проектами.
Работает он следующим образом: вы создаете код, в нача-
ле каждого файла создаете специальный комментарий, где
указываете название, автора, зависимости, так же создаете
конфигурационный файл пакета (проекта), где указываете на-
звание, описание, список участвующих в сборке файлов и т. д.
Packager работает из терминала:

```
$ packager register /path/to/project
```

```
$ packager build ProjectName/* > myproject.js
```

При сборке учитываются внешние зависимости. Под внеш-
ними я имею ввиду зависимости от других проектов, которые
так же зарегистрированы в Packager. Штука очень удобная,
использую для своих проектов уже полгода.

Резюме

В целом было переписано около трети кода. Так же много из-
менений в MooTools More, например появилась поддержка
псевдо-селекторов для событий:

```
element.addEvent('click:once', function(){});
// после клика обработчик будет автоматически
удален
```

```
element.addEvent('click:relay(.some-selec-
tor)', function(){}); // а тут у нас делегиро-
вание
```

```
element.addEvent('keydown:keys(ctrl+s)',
function(){}); // думаю здесь все ясно ■
```



a compact javascript framework

[Home](#) | [Download](#) | [Docs](#) | [Forge](#) | [Blog](#) | [Demos](#)

MooTools is a compact, modular, Object-Oriented JavaScript framework designed for the intermediate to advanced JavaScript developer. It allows you to write powerful, flexible, and cross-browser code with its elegant, well documented, and coherent API.

MooTools code respects strict standards and doesn't throw any warnings. It's extensively documented and has meaningful variable names: a joy to browse and a snap to understand.

Open Source License

MooTools is released under the Open Source [MIT license](#), which gives you the possibility to use it and modify it in every circumstance.

Browser Compatibility

MooTools is compatible and fully tested with [Safari 3+](#), Internet Explorer 6+, [Firefox 2+](#) (and browsers based on gecko), [Opera 9+](#), and Chrome 4+.

Developer Tools

[Core Builder](#)
[More Builder](#)

Development

[GitHub Repository for Core](#)
[GitHub Repository for More](#)
[Tickets @ LightHouse](#)

Resources

[Wear MooTools! \(US, EU\)](#)
[IRC Channel](#)
[MooTools on Twitter](#)
[MooTools Users Group](#)
[Unofficial MooTools Forum](#)
[The MooTorial](#)

People

[MooTools Developers](#)

MooTools Essentials

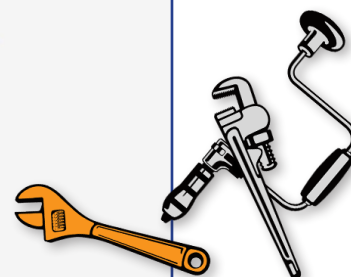
Pick up a copy of [MooTools Essentials](#) from Aaron Newton, published by [Apress](#)

MooTools 1.2 Beginner's Guide

Pick up a copy of [MooTools 1.2 Beginner's Guide](#) from Jacob Gube and Garrick Cheung, published by [Packt](#)

Some applications using MooTools

[ape](#) - [ajax](#) [push engine](#) [w3c](#) [cnet](#) [boxee](#) [bing](#) [joomla](#) [vimeo](#) [gamespot](#) [macheist](#)
[ferrari](#) [formulat](#) [jeep](#) [powerset](#) [paris.envies](#) [netvibes](#)





Автор

Василий Аксенов
СКБ Контур,
разработчик web-ui
skbkontur.ru
г. Екатеринбург
outring@gmail.com

Inline-blocks: изобретаем снова

Способ кроссбраузерной реализации инлайн-блоков давно известен, но этот тип отображения нечасто используется и поэтому не все знают, что данная реализация не такая уж и кроссбраузерная. Более того, она не позволяет полностью пользоваться всеми возможностями инлайн-блоков. Чтобы разобраться в проблемах данной реализации, для примера сделаем меню, пункты которого равномерно распределены по всей ширине экрана.

Обычно в разметке меню выглядит так:

```
<ul class="menu">
  <li class="menu-item">
    <a href="/news">Новости</a>
  </li>
  <li class="menu-item">
    <a href="/swen">Старости</a>
  </li>
  <li class="menu-item">
    <a href="/profit">Всякая информация</a>
  </li>
</ul>
```

А css для этого примерно таков:

```
.menu-item {
  display: inline-block;
}

/** Эмуляция inline-block для IE6 */
* html .menu-item {
  display: inline;
  zoom: 1;
}

/** Эмуляция inline-block для IE7 */
*+html .menu-item {
  display: inline;
  zoom: 1;
}
```

Вот как это будет выглядеть во всех браузерах:

[Новости](#) [Старости](#) [Всякая информация](#)

А вот как это выглядит в Internet Explorer 6 и 7:

[НовостиСтаростиВсякая информация](#)

При такой разметке получается, что каждый пункт меню это «слово» в строке (независимо от того, сколько слов внутри пункта, т. к. инлайн-блок – неделимая единица со своим контекстом форматирования). Но Internet Explorer 6 и 7 склеили наши «слова» так, будто бы между ними нет про-

белых символов. Можно сказать, что в этих браузерах такие инлайн-блоки «слишком блочные» и пробелы между ними не влияют на их отображение. Можно решить эту проблему простым способом – заставить пункты меню быть «более инлайновыми», для этого нужно обернуть каждый из них в инлайн-элемент:

```
<ul class="menu">
  <li class="menu-item_wrap">
    <div class="menu-item">
      <a href="/news">Новости</a>
    </div>
  </li>
  <li class="menu-item_wrap">
    <div class="menu-item">
      <a href="/swen">Старости</a>
    </div>
  </li>
  <li class="menu-item_wrap">
    <div class="menu-item">
      <a href="/profit">Всякая информация</a>
    </div>
  </li>
</ul>
```

```
.menu-item_wrap {
  display: inline;
}

.menu-item {
  display: inline-block;
}

* html .menu-item {
  display: inline;
  zoom: 1;
}

*+html .menu-item {
  display: inline;
  zoom: 1;
}
```

Но результат останется прежним. Первая причина: инлайн-блочный элемент все еще «слишком блочный» из-за того, что он изначально блок. Вторая причина: пробельные символы в закрывающей последовательности тегов инлайн-элементов как бы разваливают элемент, не дают ему быть целостным. Если убрать все пробелы закрывающей последовательности ►

и сделать инлайн-блочный элемент из инлайн элемента, то будет достигнут необходимый результат.

[Новости](#) [Старости](#) [Всякая информация](#)

Теперь необходимо, чтобы слова растягивались на всю ширину строки. Для этого нужно использовать свойство `text-align` со значением `justify`:

```
.menu {
    \text-align: justify;
}

.menu-item_wrap {
    display: inline;
}

.menu-item {
    display: inline-block;
}

* html .menu-item {
    display: inline;
    zoom: 1;
}

*+html .menu-item {
    display: inline;
    zoom: 1;
}
```

Но результата это не даст, т. к. `justify` не влияет на последнюю строку. Чтобы решить эту проблему в конец нужно вписать длинное слово, которое бы переносилось на новую строку, тем самым «растянув» слова на предыдущей строке. Писать слово – плохой вариант, достаточно просто сделать еще один пункт меню и растянуть его на 100% ширины родительского элемента (если сделать ширину меньше 100%, можно добиться интересного эффекта):

```
<ul class="menu">
    <li class="menu-item_wrap">
        <span class="menu-item">
            <a href="/news">Новости</a></span>
        </li>
    <li class="menu-item_wrap">
        <span class="menu-item">
            <a href="/swen">Старости</a></span>
        </li>
    <li class="menu-item_wrap">
        <span class="menu-item">
            <a href="/profit">Всякая информация</a></span>
        </li>
    <li class="menu-item_sizer">&nbsp;</li>
</ul>
```

```
.menu {
    text-align: justify;
}

.menu-item_wrap {
    display: inline;
}

.menu-item {
```

```
display: inline-block;
}

* html .menu-item {
    display: inline;
    zoom: 1;
}

*+html .menu-item {
    display: inline;
    zoom: 1;
}

.menu-item_sizer {
    display: inline-block;
    width: 100%;
}

* html .menu-item_sizer {
    display: inline;
    zoom: 1;
}

*+html .menu-item_sizer {
    display: inline;
    zoom: 1;
}
```



Получим необходимый результат, который выглядит и работает одинаково во всех браузерах:

[Новости](#)

[Старости](#)

[Всякая информация](#)

Теперь представим, что у каждого пункта может быть подменю, которое показывается при наведении или клике. Чтобы обозначить этот факт добавим к правому краю пункта иконки стрелки и спозиционируем ее абсолютно, отталкиваясь от правого края пункта:

```
<ul class="menu">
    <li class="menu-item_wrap">
        <span class="menu-item">
            <span class="menu-item-icon">&darr;</span>
            <a href="/news">Новости</a></span>
        </li>
    <li class="menu-item_wrap">
        <span class="menu-item">
            <span class="menu-item-icon">&darr;</span>
            <a href="/swen">Старости</a></span>
        </li>
    <li class="menu-item_wrap">
        <span class="menu-item">
            <span class="menu-item-icon">&darr;</span>
            <a href="/profit">Всякая информация</a></span>
        </li>
    <li class="menu-item_sizer">&nbsp;</li>
</ul>
```

```
.menu {
    text-align: justify;
}

.menu-item_wrap {
```

```

display: inline;
}
.menu-item {
display: inline-block;
padding-right: 15px;
position: relative;
text-align: left; /** Возвращаем на место */
}
* html .menu-item {
display: inline;
zoom: 1;
}
*+html .menu-item {
display: inline;
zoom: 1;
}
.menu-item-icon {
position: absolute;
right: 6px;
}
.menu-item_sizer {
display: inline-block;
width: 100%;
}
* html .menu-item_sizer {
display: inline;
zoom: 1;
}
*+html .menu-item_sizer {
display: inline;
zoom: 1;
}

```

Отображение будет одинаковым во всех браузерах:

[Новости](#)↓ [Старости](#)↓ [Всякая информация](#)↓

Но во всех версиях Орега можно увидеть, что иконка находится левее, чем должна быть:

[Новости](#) [Старости](#) [Всякая информация](#)

Связано это с тем, что если в inline-block элементе находится неблочный или нетекстовый элемент, то происходит нарушение границ позиционирования. Исправить такое поведение можно только добавлением еще одной внутренней блочной обертки.

```

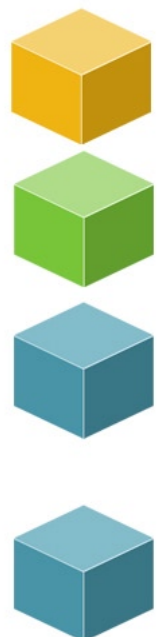
<ul class="menu">
<li class="menu-item_outerWrap">
<span class="menu-item_innerWrap">
<span class="menu-item">
<span class="menu-item-icon">&darr;
</span>
<a href="/news">Новости</a></span></span>
</li>
<li class="menu-item_outerWrap">
<span class="menu-item_innerWrap">

```

```

<span class="menu-item ">
<span class="menu-item-icon">&darr;</span>
<a href="/swen">Старости</a></span></span>
</li>
<li class="menu-item_outerWrap">
<span class="menu-item_innerWrap">
<span class="menu-item">
<span class="menu-item-icon">&darr;</span>
<a href="/profit">Всякая информация</a></span></span>
</li>
<li class="menu-item_sizer">&nbsp;</li>
</ul>
.menu {
text-align: justify;
}
.menu-item_outerWrap {
display: inline;
}
.menu-item_innerWrap {
display: inline-block;
}
* html .menu-item_innerWrap {
display: inline;
zoom: 1;
}
*+html .menu-item_innerWrap {
display: inline;
zoom: 1;
}
.menu-item {
display: block;
padding-right: 15px;
position: relative;
text-align: left;
}
.menu-item-icon {
position: absolute;
right: 6px;
}
.menu-item_sizer {
display: inline-block;
width: 100%;
}
* html .menu-item_sizer {
display: inline;
zoom: 1;
}

```




```
*+html .menu-item_sizer {
    display: inline;
    zoom: 1;
}
```

Теперь все выглядит и работает как нужно во всех браузерах:

[Новости↓](#)
[Старости↓](#)
[Всякая информация↓](#)

Таким образом «формула» кроссбраузерного инлайн-блока выглядит так:

```
inline>inline-block>block
```

Это решение очень удобно применять для создания собственных элементов управления, т. к. их настоящие аналоги как раз имеют такой тип отображения. Но нужно не забывать, что нельзя «включать» `hasLayout` для блочных элементов, т. к. это приведет к «распираанию» элемента на всю ширину родителя в Internet Explorer 6. Если же «включить» `hasLayout` необходимо, то нужно сделать это для блочной обертки и задать ей фиксированную ширину. Если нужно, чтобы ширина рассчитывалась по содержанию, то она должна быть равна нулю и свойство `overflow` должно иметь значение `visible`. Все это нужно применять только для Internet Explorer 6.

Само меню получилось не совсем хорошо, из-за того, что приходится создавать дополнительный элемент. Самым простым вариантом является замена последнего элемента на CSS псевдоэлемент `:after` для всех браузеров, которые его поддерживают, а для тех, которые не поддерживают (а это Internet Explorer 6 и Internet Explorer 7) есть куда более простое и логичное решение. Решение это – свойство `text-justify`, которое было введено Microsoft еще с выпуском шестой версии своего браузера (спасибо Владиславу Шкодину и Кириллу за информацию) и которое теперь попало в спецификацию CSS3. Несмотря на наличие этого свойства в новой спецификации, активно разрабатываемые браузеры не торопятся его поддерживать, делая упор на более «востребованные» (модные) свойства. Данное свойство имеет 7 возможных значений по спецификации и 9 возможных значений в реализации от Microsoft. В нашем случае нам поможет значение `distribute-all-lines`, которое включает растягивание всех строк, включая последнюю. Хотя есть и другие интересные значения, например `newspaper`, при использовании которого можно было бы избежать создания дополнительных инлайн оберток, но т. к. меню является не главной целью, оно намеренно не было использовано. В итоге код меню будет выглядеть так:

```
<ul class="menu">
  <li class="menu-item_outerWrap">
    <span class="menu-item_innerWrap">
      <span class="menu-item">
        <span class="menu-item-icon">&darr;</span>
        <a href="/news">Новости</a></span></span>
      </li>
      <li class="menu-item_outerWrap">
        <span class="menu-item_innerWrap">
          <span class="menu-item">
            <span class="menu-item-icon">&darr;</span>
            <a href="/swen">Старости</a></span></span>
          </li>
          <li class="menu-item_outerWrap">
            <span class="menu-item_innerWrap">
              <span class="menu-item">
                <span class="menu-item-icon">&darr;</span>
                <a href="/profit">Всякая информация</a></span></span>
```

```
ция</a></span></span>
  </li>
</ul>
```

```
.menu {
    text-align: justify;
    text-justify: distribute-all-lines;
}
```

```
.menu:after {
    content: '';
    display: inline-block;
    width: 100%;
}
```

```
.menu-item_outerWrap {
    display: inline;
}
```

```
.menu-item_innerWrap {
    display: inline-block;
}
```

```
* html .menu-item_innerWrap {
    display: inline;
    zoom: 1;
}
```

```
*+html .menu-item_innerWrap {
    display: inline;
    zoom: 1;
}
```

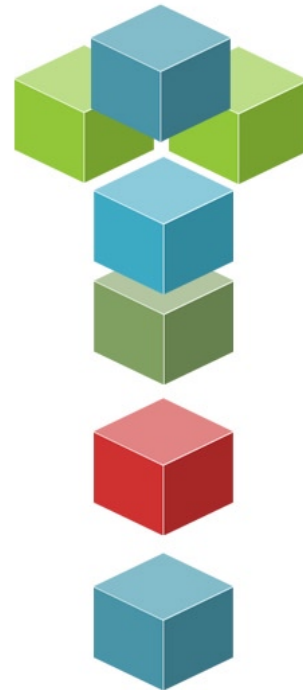
```
.menu-item {
    display: block;
    padding-right: 15px;
    position: relative;
    text-align: left;
}
```

```
.menu-item-icon {
    position: absolute;
    right: 6px;
}
```

```
.menu-item_sizer {
    display: inline-block;
    width: 100%;
}
```

```
* html .menu-item_sizer {
    display: inline;
    zoom: 1;
}
```

```
*+html .menu-item_sizer {
    display: inline;
    zoom: 1;
}
```



Каталог Веб-студий



Веб-студия ИТКЕМ

Страна: Россия
Город: Кемерово
Год основания: 2010
Сотрудников: 5
Сайт: <http://itkem.ru>



Imagz.ru

Страна: Россия
Город: Санкт-Петербург
Год основания: 2009
Сотрудников: 4
Сайт: imagz.ru



KamIT

Страна: Россия
Город: Каменск-Уральский
Год основания: 2008
Сотрудников: 2
Сайт: <http://kamit.net.ru>



Dart's

Страна: Россия
Город: Краснодар
Год основания: Краснодар
Сотрудников: 5
Сайт: www.studio-dart.ru



Shocos Design Team

Страна: Украина
Город: Николаев
Год основания: 2009
Сотрудников: 2
Сайт: www.shocos.com



Сайт Плюс

Страна: Россия
Город: Хабаровск
Год основания: 2007
Сотрудников: 5
Сайт: <http://site-plus.ru>



Delmar

Страна: Россия
Город: Москва
Год основания: 2006
Сотрудников: 15
Сайт: www.delmar.ru



Sandal

Страна: Россия
Город: Владивосток
Год основания: 2004
Сотрудников: 5
Сайт: www.sandal.ru



ArtMachine

Страна: Украина
Город: Николаев
Год основания: 2005
Сотрудников: 7
Сайт: www.artmachine.com.ua



Угол зрения

Страна: Россия
Город: Новосибирск
Год основания: 2008
Сотрудников: 8-10
Сайт: www.ugolzreniya.ru



CrazyOne

Страна: Украина
Город: Кривой Рог
Год основания: 2010
Сотрудников: 5
Сайт: <http://crazyone.net/>



SMSdesign

Страна: Украина
Город: Киев
Год основания: 2008
Сотрудников: 7
Сайт: <http://SMSdesign.com.ua>

Добавить веб-студию в каталог журнала бесплатно



ДОБАВИТЬ

Публикуемые в журнале компании взяты из каталога сайта издания. Данные о компаниях публикуются по мере их добавления в каталог. Добавить в каталог данные о компании может любая организация. Добавление данных в каталог сайта осуществляется бесплатно на добровольной основе самой организацией. Редакция журнала Веб-Аналитик.ИНФО не занимается добавлением данных в каталог сайта и не несет ответственности за предоставляемую информацию от организаций.



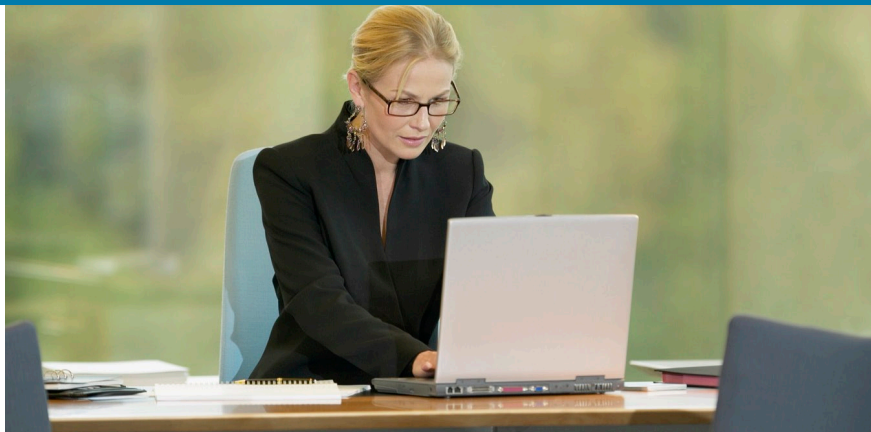
.NET

Быстрый запуск блога на базе Orchard

Обзор бета версии ASP.NET MVC 3

Ликвидатор велосипедов: часть 1

**Распараллеливание задач.
Случай «идеальной
параллельности»**



Автор

Владимир Юнев (ХаосCPS)
Веб-разработчик, Microsoft MVP
г. Екатеринбург
www.progg.ru

Быстрый запуск блога на базе Orchard

Orchard — это движок блогов с открытым исходным кодом, который разрабатывается фондом CodePlex.org при поддержке Microsoft. Orchard основан на ASP.NET MVC 2 и на данный момент использует .NET 3.5. Orchard развивается быстрыми темпами и на сегодня поддерживает следующие функции:

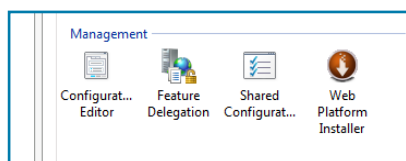
- заведение нескольких блогов,
- многопользовательский режим,
- создание дополнительных страниц,
- админка с редактированием основных функций,
- темы,
- поддержка XML-RPC, постинг в блоги через Live Writer или другое ПО.

В этой статье я расскажу, как развернуть блог на базе Orchard самыми минимальными усилиями. Поможет нам в этом Web Platform Installer.

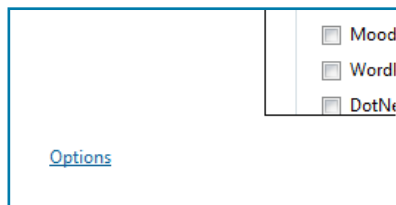
Для успешной установки Orchard вам надо иметь IIS-сервер (в статье рассмотрен IIS 7.5) и установленный .NET Framework 3.5, плюс для упрощения установки мы будем использовать Web Platform Installer 2.0, который интегрируется в панель администрирования IIS.

Установка Orchard

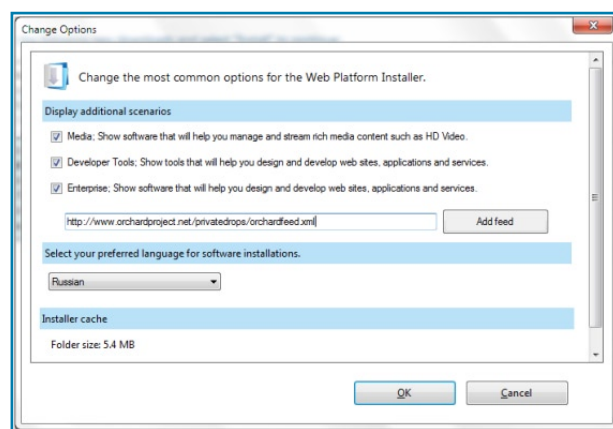
Запустите IIS manager, в нем запустите Web Platform Installer.



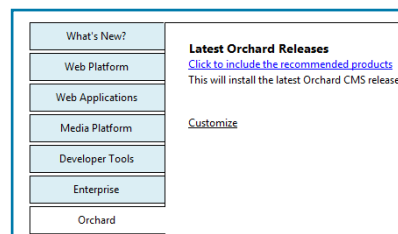
Перейдите в опции WPI, используя гиперссылку снизу.



В окне добавьте в поле рядом с кнопкой **Add Feed** значение: <http://www.orchardproject.net/privatedrops/orchardfeed.xml>. Нажатие **Add feed**. Этим вы добавите новый канал для установки последней сборки Orchard через WPI.

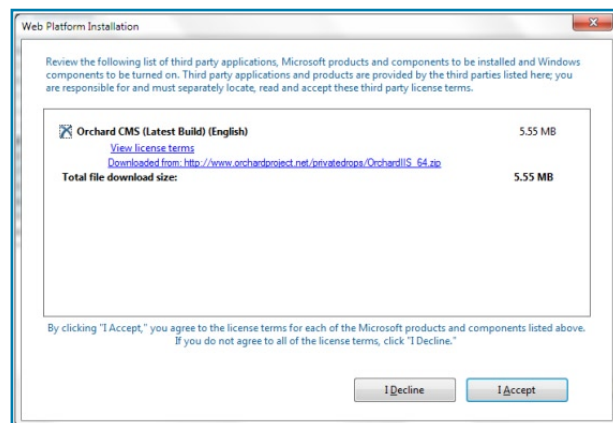


Нажмите **OK**. Вы получите новую вкладку главного окна WPI для установки Orchard.

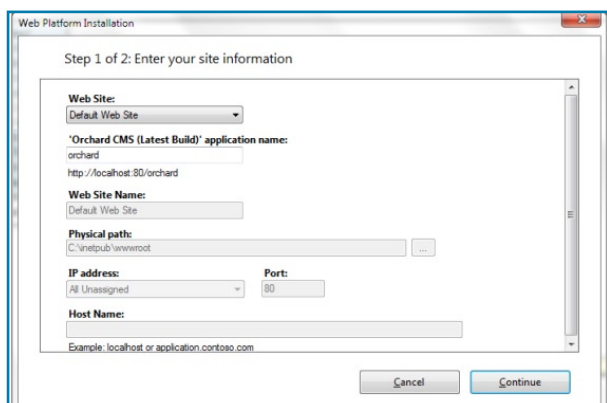


Щелкните по ссылке **Click to include...** для того, чтобы включить требуемые компоненты, которые могут быть у вас не установлены, но необходимые для запуска Orchard. Этим вы разрешите WPI автоматическую установку и конфигурирование всех нужных компонентов.

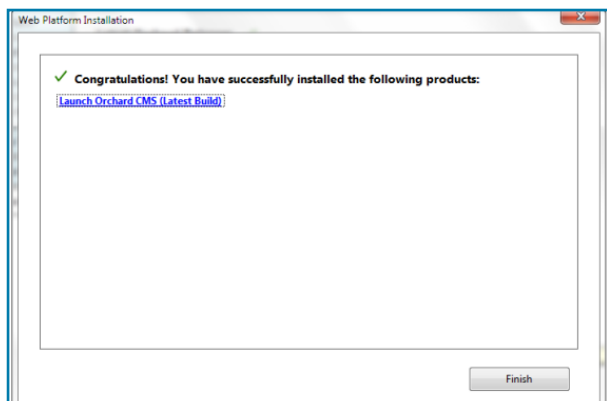
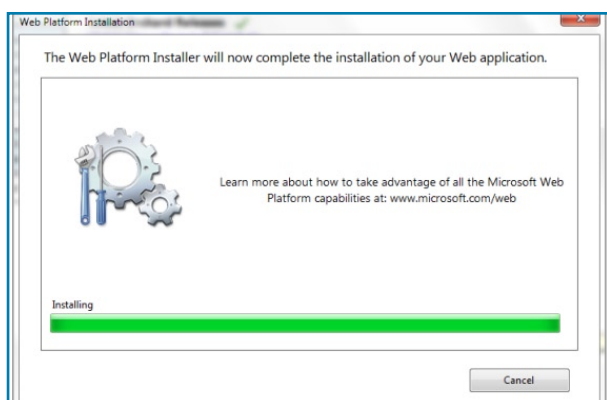
Нажмите кнопку **Install**. В окне нажмите **I Accept**, подтвердив свое желание установить выбранные компоненты. ►



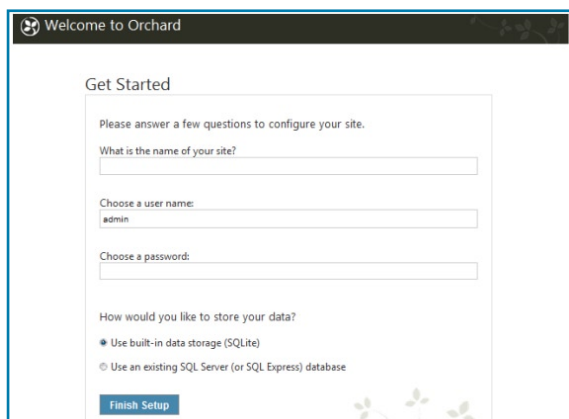
После загрузки и установки компонентов настройте необходимые параметры хостинга.



Произойдет установка Orchard в IIS. После чего вы получите сообщение об успешной установке.



Перейдите по ссылке и убедитесь, что Orchard установлен на вашем сервере. Это все! Осталось настроить параметры блога и можете работать.

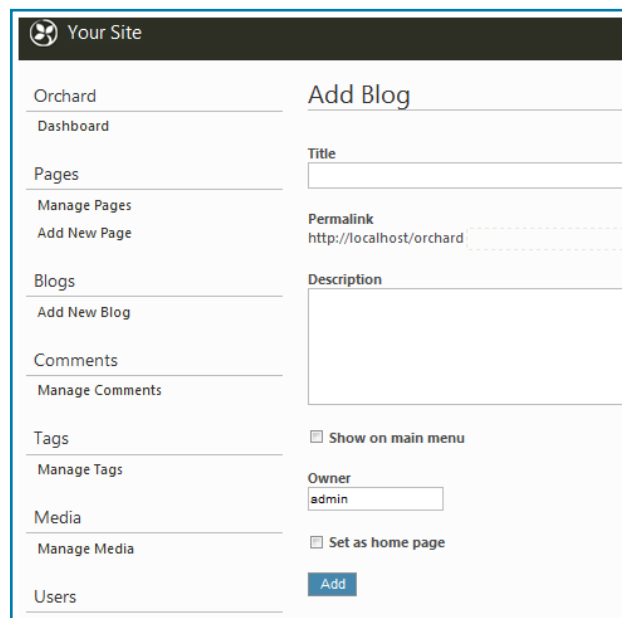


Live Writer

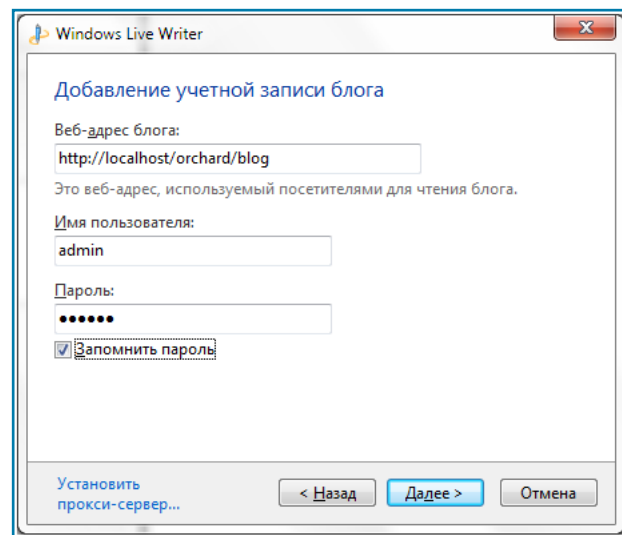
В дополнение я расскажу, как использовать Live Writer для публикации статей в блоги Orchard.

Для тех, кто не знает: Live Writer позволяет публиковать статьи во множество мест интернета из одного источника. Лично я публикую статьи в LiveJournal, GotDotNet, Spaces и в свой блог на базе Orchard.

Для того чтобы начать публиковать статьи в блог на базе Orchard, создайте его в админке.



После создания, вы можете задать в Live Writer параметры нового блога следующим образом:



Где **Веб-адрес блога** – это URL блога, который вы только что создали. Нажмите **Далее** и остальное конфигурирование Live Writer произведет сам. ■



Автор

Владимир Юнев (ХаосCPS)
Веб-разработчик, Microsoft MVP
г. Екатеринбург
www.progg.ru

Обзор бета версии ASP.NET MVC 3

Выпущена новая предварительная версия фреймворка ASP.NET MVC 3 Beta. Напомню, что предыдущая версия вышла в конце июля, подробный обзор о нововведениях в ней вы можете найти [у нас на сайте](#). В этой статье перечислены последние нововведения, которые разработчики добавили с выходом MVC 3 Beta.

Улучшенный контроль над валидацией запросов

Атрибут `ValidateInputAttribute`, представленный еще в первой версии ASP.NET MVC позволяет управлять валидацией запроса для конкретного действия. Ранее вы могли только включить или выключить валидацию, но в MVC 3 Beta добавлена новая возможность управлять валидацией для каждого параметра запроса.

```
[HttpPost]
[ValidateInput(true, Exclude = "helloText")]
public ActionResult SayHello(string name, string helloText)
{
    ViewData["Name"] = name;
    ViewData["Text"] = helloText;

    return View("Index");
}
```

С помощью параметра `Exclude` вы можете указать те параметры запроса, которые не требуется проверять на безопасный ввод. При использовании `Exclude` вся ответственность на обработку пользовательского ввода ложится на плечи разработчика.

Новое правило конвертации ' _ ' в '-' в хелпер-классах

HTML5 в своих атрибутах (например, `data-*`) использует дефисы. Однако в C# для переменных использование дефисов запрещено и это приводит к невозможности использовать в хелпер-методах задание атрибутов через анонимные типы. Например, следующий код вызовет ошибку:

```
@Html.TextBox("Name", "Value", new {data-required = "true"})
```

Для решения этой проблемы, в MVC 3 Beta вводится соглашение, согласно которому имена, определенные в анонимных типах хелпер-методов с символом подчеркивания ' _ ' будут транслироваться в атрибуты с дефисом.

Код

```
@Html.TextBox("Name", "Value", new {data_required = "true"})
```

Разметка

```
<input data-required="true" id="Name" name="Name" type="text" value="Value" />
```

Дополнительная поддержка Dependency Injection

В первой превью-версии ASP.NET MVC 3 появилась встроенная поддержка внедрений зависимостей кода (Dependency Injection). В новой версии эта поддержка была расширена.

IDependencyResolver на смену IServiceLocator

Представленный в превью-версии интерфейс `IServiceLocator` был заменен на упрощенную версию `IDependencyResolver`. Теперь вместо старого интерфейса:

```
public interface IServiceLocator : ServiceProvider {
    IEnumerable<TService> GetAllInstances<TService>();
    IEnumerable<object> GetAllInstances(Type serviceType);
    TService GetInstance<TService>();
    TService GetInstance<TService>(string key);
    object GetInstance(Type serviceType);
    object GetInstance(Type serviceType, string key);
}
```

роль Common Service Locator играет следующий интерфейс:

```
public interface IDependencyResolver
{
    object GetService(Type serviceType);
    IEnumerable<object> GetServices(Type serviceType);
}
```

Соответственно, на смену статического класса `MvcServiceLocator` пришел класс `DependencyResolver`. Регистрация своего экземпляра класса реализующего интерфейс `IDependencyResolver` выглядит просто:

```
DependencyResolver.SetResolver(myResolver);
```

Например, ниже реализация `IDependencyResolver` для использования DI-контейнера Unity в MVC 3 Beta: ►

```
public class UnityDependencyResolver : IDependencyResolver
{
    IUnityContainer _container;

    public UnityDependencyResolver(IUnityContainer container)
    {
        _container = container;
    }

    public IEnumerable<object> GetServices(Type serviceType) {
        if (_container.IsRegistered(serviceType))
            return _container.ResolveAll(serviceType);
        else
            return new List<object>();
    }

    public object GetService(Type serviceType)
    {
        if (_container.IsRegistered(serviceType))
            return _container.Resolve(serviceType);
        else
            return null;
    }
}
```


Обратите внимание, методы `GetService` и `GetServices` должны возвращать `null` и пустую коллекцию (важно) соответственно, в случаях, когда невозможно вернуть тип из контейнера. Если MVC получит эти значения, то для работы будут использоваться значения объектов поддерживаемых механизмом внедрения зависимостей установленные по умолчанию. К примеру, если ваш контейнер не содержит тип для фабрики контроллеров, то он (контейнер) должен вернуть `null` на запрос с типом `IServiceLocator` и MVC будет использовать для работы класс `DefaultControllerFactory`.

Регистрация этого интерфейса выглядит следующим образом (`global.asax`):

```
var container = new UnityContainer();
DependencyResolver.SetResolver(new UnityDependencyResolver(container));
```

Хотя `IDependencyResolver` заменил `IServiceLocator`, в статическом классе `DependencyResolver` осталась возможность регистрировать экземпляры реализующие интерфейс `Common Service Locator`.

IControllerActivator

Новый интерфейс `IControllerActivator` позволяет управлять созданием конкретного экземпляра контроллера. В отличие от фабрики контроллеров, `IControllerActivator` отвечает только за инстанцирование определенного типа, тогда как фабрика занимается еще и поиском нужных типов контроллеров. Интерфейс `IControllerActivator` объявлен следующим образом:

```
namespace System.Web.Mvc {
    using System.Web.Routing;

    public interface IControllerActivator {
        IController Create(RequestContext requestContext, Type controllerType);
    }
}
```

Для того чтобы механизм `IControllerActivator` заработал, фабрика контроллеров должна поручать `IControllerActivator` инстанцировать класс контроллера. Для этого в класс `DefaultControllerFactory` (фабрика контроллеров по умолчанию) добавлен новый конструктор, который принимает параметр `IControllerActivator`:

```
public DefaultControllerFactory(IControllerActivator controllerActivator);
```

Разработчику остается зарегистрировать реализацию `IControllerActivator` в своем DI-контейнере и фабрика контроллеров будет использовать его для инстанцирования контроллеров. Ниже простейший пример реализации `IControllerActivator`, который просто возвращает экземпляры контроллеров из контейнера.

```
public class MyControllerActivator : IControllerActivator
{
    public IController Create(System.Web.Routing.RequestContext requestContext,
        Type controllerType)
    {
        return (IController)
            DependencyResolver.Current.GetService(controllerType);
    }
}
```

В целом, новый механизм позволяет внедрять зависимости в процесс создания конкретного типа контроллера без надобности вносить изменения в работу фабрики контроллеров, задача которой состоит в поиске нужного типа.

IViewPageActivator

`IViewPageActivator` предлагает функционал аналогичный `IControllerActivator`, но только применительно к представлениям.

```
namespace System.Web.Mvc {
    public interface IViewPageActivator {
        object Create(ControllerContext controllerContext, Type type);
    }
}
```

`IViewPageActivator` позволяет определить класс, который будет отвечать за создание представлений. Для под-

держки нового интерфейса у классов `WebFormView` и `RazorView` (отвечают за представления в Web Forms и Razor соответственно) появились новые конструкторы с параметром типа `IViewPageActivator`. Это нововведение позволит сделать внедрение кода в процесс создания представлений или элементов управления.

Дополнительная автоматическая поддержка внедрения кода

В дополнение к реализованной ранее поддержке внедрения кода для фабрики контроллеров, самих контроллеров, движков представлений, представлений и фильтров действий, в MVC 3 Beta добавлен еще ряд областей, где происходит автоматическая поддержка DI:

- **провайдеры валидации моделей** – классы, реализующие абстрактный класс `ModelValidatorProvider` могут автоматически использоваться при валидации модели, если их поместить в DI-контейнер;
- **провайдеры метаданных моделей** – класс, реализующий абстрактный класс `ModelMetadataProvider` может автоматически использоваться для предоставления метаданных при валидации и работе с шаблонами, если его поместить в DI-контейнер;
- **провайдеры значений** (Value providers) – классы, реализующие абстрактный класс `ValueProviderFactory` могут автоматически использоваться при связывании модели, если их поместить в DI-контейнер;
- **классы связывания модели** (Model binders) – классы, реализующие интерфейс `IModelBinderProvider` могут автоматически использоваться при создании связующих классов для модели, если их поместить в DI-контейнер.

Поддержка ненавязчивого синтаксиса при Ajax и валидации

В ASP.NET MVC есть ряд хелпер-методов для работы с Ajax-запросами, например, `Ajax.ActionLink`. Ранее, использование этих методов на странице генерировало в разметке встроенный javascript для обработки Ajax, а это не так хорошо, как если бы скрипты были вынесены в отдельный файл и использовался бы ненавязчивый (unobtrusive) подход в разметке.

С выходом MVC 3 Beta для всех Ajax-методов появляется поддержка ненавязчивого javascript. Производится это с помощью внедрения специальных атрибутов `data-ajax-*`. Например, при использовании `Ajax.ActionLink` теперь будет генерироваться следующая разметка.

Код

```
Ajax.ActionLink("Ajax-запрос", "AjaxMethod", "Home", new AjaxOptions { UpdateTargetId = "someId" })
<span id="someId"></span>
```

Разметка

```
<a data-ajax="true" data-ajax-mode="replace" data-ajax-update="#someId" href="/Home/AjaxMethod">Ajax-запрос</a>
<span id="someId"></span>
```

Всю работу по обработке новой разметки берет на себя специальный javascript-файл `jquery.unobtrusive.ajax.js`, который необходимо включать на все страницы с ajax-функционалом MVC Framework. Работает этот файл на базе jQuery, поэтому и эту библиотеку (версия 1.4.1 и выше) необходимо включить на страницы.

В MVC 3 Beta ненавязчивое употребление javascript применяется так же для клиентской валидации пользовательского ввода. Чтобы использовать ненавязчивый javascript при валидации, используйте в разметке следующие хелпер-методы:

```
Html.EnableClientValidation();
```

```
Html.EnableUnobtrusiveJavaScript();
```

Всю работу в данном случае берет на себя javascript-файл `jquery.validate.unobtrusive.js`, который основан на jQuery и тре-

будет подключения этой библиотеки (jquery-1.4.1.js), а так же скрипта jQuery-валидации (jquery.validate.js).

Необходимо отметить, что функционал ненавязчивого javascript управляется через параметр в файле web.config:

```
<appSettings>
  <add key="enableSimpleMembership" value="false" />
  <add key="ClientValidationEnabled" value="false"/>
  <add key="UnobtrusiveJavaScriptEnabled" value="true"/>
</appSettings>
```

Для новых проектов он установлен по умолчанию, однако для проектов созданных ранее его необходимо установить вручную. Здесь же вы можете включить клиентскую валидацию по умолчанию для всех форм.

Изменения в движке представлений Razor и Web Pages

По многочисленным просьбам разработчиков добавлен упрощенный вариант декларации типа модели, используемой в типизированном представлении. Если ранее тип задавался через конструкцию вида:

```
@inherits System.Web.Mvc.WebViewPage<MyModelNamespace.MyModelType>
```

то теперь введена новая директива @model:

```
@model MyModelNamespace.MyModelType
```

Поддержка кода выполняемого для всех представлений в Web Pages

В MVC 3 Beta появилась возможность задать код, который будет выполняться перед запуском всех представлений. Для этого используется новый файл _viewstart.cshtml, который необходимо расположить в папке Views. Вы можете использовать этот файл и для определенного набора представлений. Для этого расположите _viewstart.cshtml в любой подпапке с нужными представлениями. Файл _viewstart.cshtml в подпапке имеет приоритет над общим файлом, который расположен в папке Views.

Новый функционал предназначен в первую очередь для централизованной установки единой компоновки для всех представлений. В проекте по умолчанию, файл _viewstart.cshtml содержит следующее:

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Здесь для всех представлений разом задается единая модель компоновки из файла _Layout.cshtml.

Новые хелпер-классы Web Pages

К существующим хелпер-классам типа ReCaptcha, Twitter или Validation добавлен ряд новых:

- Chart – рисование графиков;
- Crypto – алгоритмы шифрования;
- WebGrid – рендеринг данных в виде таблицы;
- WebImage – изменение и отображение картинки, добавление водяных знаков;
- WebMail – работа с электронной почтой;
- Facebook – добавление Like-кнопки.

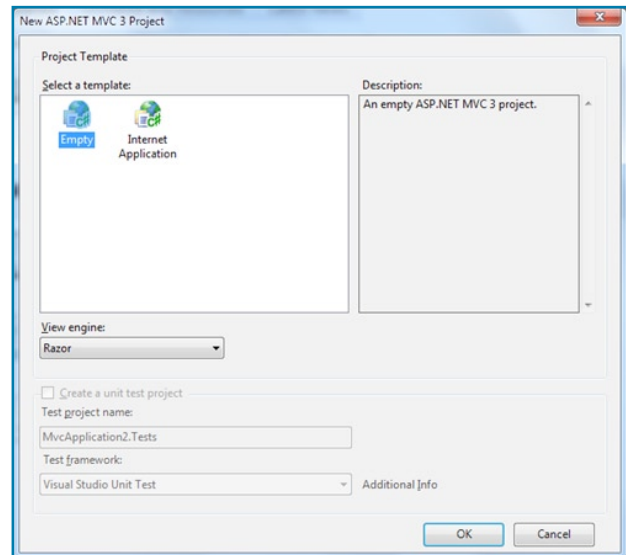
Полный список всех хелпер-классов можно посмотреть [здесь](#).

Поддержка VBHTML

В MVC 3 Beta представлена поддержка Razor для разработчиков использующих Visual Basic. Файлы с Razor-синтаксисом на Vb будут носить расширение .vbhtml. Подробности по синтаксису можно узнать по этой [ссылке](#).

Улучшенное диалоговое окно «New Project»

Окно, служащее для создания нового проекта было переработано. В него помимо возможности выбора типа приложения, добавили выбор используемого механизма представлений (view engine).

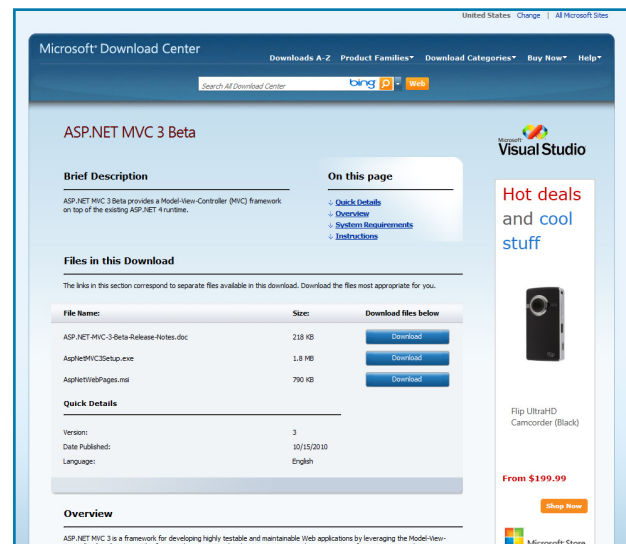


Исправленные ошибки

В MVC 3 Beta исправлены некоторые ошибки:

- шаблон для EditorFor и DisplayFor используемый по умолчанию теперь принимает во внимание параметр Order в атрибуте DisplayAttribute. То есть появилась возможность управлять порядком отображения элементов модели в шаблоне по умолчанию;
- клиентская валидация теперь поддерживает валидацию переопределенных свойств, которые объявлены с атрибутом валидации;
- фабрика JsonValueProviderFactory теперь зарегистрирована по умолчанию.

К сожалению, в этой версии MVC 3 так и не появилась подсветка синтаксиса и работа IntelliSense при использовании Razor. Но с первой проблемой можно справиться. Для загрузки доступен плагин к Visual Studio 2010, который добавляет подсветку синтаксиса в razor-представлениях. Загрузить Razor Syntax Highlighter можно [отсюда](#).



Ликвидатор велосипедов: часть 1

Автор

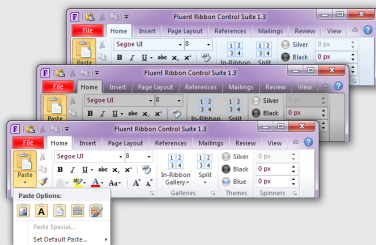
Станислав Сидристый
Лидер команды разработки
г. Санкт-Петербург
sunex.development@gmail.com

Я начинаю серию статей, которые должны помочь каждому разработчику избавиться от велосипедов в своих последующих и текущих проектах. Усилить код и увеличить функционал. Каждый из нас хочет внести в свою программу нечто, что делает ее богаче, но зачастую решение либо не находится, либо лень тратить время на поиски. А еще внутри сидит глупая уверенность, что это можно реализовать быстро и не менее функционально. В любом случае я хочу чтобы каждый пробежался глазами по спискам библиотек, которые я вам предоставлю на суд, и чтобы у каждого эти библиотеки отложились в памяти. И когда их функционал востребуется в будущем, я очень надеюсь, что вы вспомните статью и поищите эти библиотеки вместо того, чтобы писать все новые и новые велосипеды, полные палок в колесах.

Начну с оконных менеджеров.

Microsoft Office 2007-2010 - like интерфейс. Он же Ribbon, он же Fluent

В интернете он представлен множеством библиотек. Однако огромное количество библиотек – платные. Есть мнение, что они стартовали одновременно с бесплатными версиями, иначе сложно объяснить их обилие, ведь зачастую их функционал беднее бесплатных аналогов. Бесплатные аналоги представлены двумя основными проектами.



Fluent Ribbon Control Suite

Богатый функционал, грамотная реализация.

Лицензия: MS-PL

Старт проекта: November 12, 2009

Частота коммитов: несколько раз в сутки

Базируется на: WPF, .NET Framework 3.5, 4.0

Локализация: имеются готовые локализации на 22 языка

Поддержка шкурок: есть, менять можно любую компоненту

Проект: fluent.codeplex.com

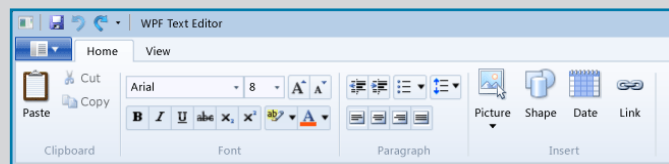
WPF Ribbon

Официальная библиотека от компании Microsoft. Теоретически на ней построена линейка MS Office, но я не проверил.

Много сказать не могу, использовал, но не долго. Удалось очень легко запустить и построить на базе компоненты плагино-расширяемое приложение (привет, MEF). Богатый функционал, реализация на высоком уровне. Основной минус – лицензия. По особому запросу и через регистрацию на сайте можно получить библиотеку, условием использования является НЕ написание программного обеспечения, конкурирующее с MS-Office линейкой.

Поддержка шкурок: да, менять можно любую компоненту

Проект: входит в состав wrf.codeplex.com



Библиотеки докирования окон

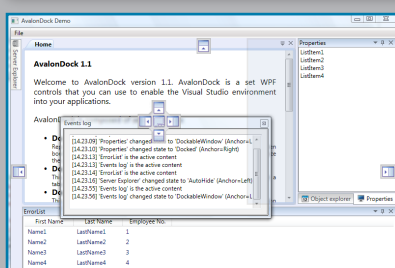
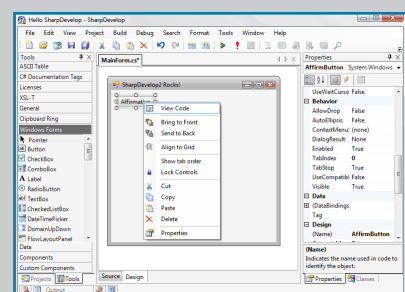
Если вы решили использовать докирование как основу UI вашего приложения, то к вам на помощь придут две библиотеки.

DockPanel Suite (Weifen Luo)

Богатая библиотека для WinForms, используемая во множестве проектов. В числе этих проектов находится такой монстр, как SharpDevelop (не включая версию 4), его скриншот я и представлю в качестве примера использования:

- Сохранение и восстановление положения окон
- Лицензия MIT
- Сделана такой, чтобы результат был неотличим от интерфейса Visual Studio
- Локализуема
- Единственный маленький минус – последний коммит 2009-05-08, однако это говорит о стабильности библиотеки

Страница проекта: sourceforge.net/projects/dockpanelsuite



AvalonDock

Шикарнейшая библиотека докирования для Windows Presentation Foundation. Используется в SharpDevelop начиная с версии 4, что говорит о том, что ее будут поддерживать в полном объеме если не авторы, то команда #develop.

Можно менять шкурки как угодно, ведь это WPF. Т. е. теоретически можно построить интерфейс а-ля VisualStudio 2010.

Лицензия: New BSD License

Страница проекта: avalondock.codeplex.com

Автор

Андрей Цветков
«Гендикс»
Архитектор программного обеспечения
MCPD for Web Development, MCAD
<http://gendix.ru>
г. Москва
TS@gendix.ru

Распараллеливание задач. Случай «идеальной параллельности»

В этой статье мы поговорим о подходах к параллельной обработке циклов в тех удачных случаях, когда между отдельными итерациями нет зависимостей, и они могут корректно выполняться параллельно.

Рассмотрим некоторый цикл, в котором обрабатываются данные:

```
for (int i = 0; i < upperBound; i++)  
{  
    // ... тело цикла  
}
```

Если логика работы тела цикла такова, что результат его вычислений на конкретной итерации никак не зависит от результата вычислений какой-либо другой итерации, то данный цикл относится к «идеально-параллельным», так как все его итерации могут быть выполнены параллельно. Такое же определение можно отнести и к аналогичному `foreach`-циклу; из него же вытекает, что порядок вычисления итераций неважен.

Подходы к реализации параллельного цикла

Попробуем реализовать сам метод для выполнения цикла параллельно, и посмотрим, с какими трудностями мы при этом столкнемся.

Итак, мы собираемся создать метод наподобие такого:

```
public static void ParallelFor(int from, int  
to, Action<int> body);
```

Он должен выполнить `body` для всех значений в интервале от `from` до `to` (не включая последнее), при этом распараллелив их выполнение на несколько потоков, чтобы добиться максимальной производительности. Насколько же возможно это распараллелить?

Определение степени параллелизма

Есть здравое суждение, что логично использовать столько потоков, сколько есть ядер у процессора на машине. При таком решении, каждое из ядер будет полностью загружено. При большем количестве потоков мы будем получать рост накладных затрат на переключение между ними, при меньшем — какие-то из ядер будут простаивать. Такой подход «по потоку на ядро» во многих случаях верен, хотя и базируется на идеалистичной модели вычислений, когда все потоки занимаются только вычислениями на процессоре. Однако иногда может быть полезно иметь и больше потоков. Например,

в случае, если потоки в своей работе половину времени проводят в ожидании ресурсов, то процессор все равно простаивает в это время, и увеличение количества рабочих потоков может привести к улучшению производительности.

Таким образом, выходит, что стоит начинать с простого варианта «один поток на ядро», но быть готовым к рассмотрению и других стратегий.

Для того чтобы получить количество доступных ядер, мы можем воспользоваться свойством `System.Environment.ProcessorCount`. Надо отметить, что оно возвращает количество ядер с учетом `Hyper-threading`, то есть, в случае его наличия, будет возвращено «логическое», уже удвоенное количество ядер.

С учетом вышесказанного, вот логичная (хотя несколько наивная) реализация:

```
public static void ParallelFor(int from, int  
to, Action<int> body)  
{  
    // определяемся с количеством потоков и раз-  
    мером блока данных для каждого потока  
    int size = to - from;  
    int numProcs = Environment.ProcessorCount;  
    int range = size / numProcs;  
    // разбиваем данные, запускаем все потоки и  
    ждем завершения  
    var threads = new List<Thread>(numProcs);  
    for (int p = 0; p < numProcs; p++)  
    {  
        int start = p * range + from;  
        int end = (p == numProcs - 1) ?  
            to : start + range;  
        threads.Add(new Thread(() => {  
            for (int i = start; i < end; i++) body(i);  
        }));  
    }  
    foreach (var thread in threads) thread.Start();  
    foreach (var thread in threads) thread.Join();  
}
```

Важным недостатком такой версии является создание/завершение новых потоков каждый раз, так как это довольно затратная операция (в частности, каждый поток резервирует

1Мб памяти в стеке, даже если в данный момент на нем не выполняется ни одной функции).

Другая проблема тоже вытекает из создания новых потоков. Предположим, что в `body` нам передали некий код, который тоже содержит вызов `ParallelFor`. В таком случае, в процессе выполнения будет создано уже не `numProcs` потоков, а в два раза больше, и может возникнуть ситуация, когда затраты на переключение между потоками будут слишком велики (чрезмерная многопоточность).

Статическое распределение итераций

Поэтому вместо создания потоков вручную мы предпочтем воспользоваться пулом потоков:

```
public static void ParallelFor(int from, int to, Action<int> body)
{
    int size = to - from;
    int numProcs = Environment.ProcessorCount;
    int range = size / numProcs;

    int remaining = numProcs;

    // объект синхронизации, для определения
    // завершения работы
    using (ManualResetEvent mre = new ManualResetEvent(false))
    {
        // создаем все задания
        for (int p = 0; p < numProcs; p++)
        {
            int start = p * range + from;
            int end = (p == numProcs - 1) ? to : start + range;

            ThreadPool.QueueUserWorkItem(delegate {
                for (int i = start; i < end; i++)
                {
                    body(i);
                }
                // проверяем, не последнее ли это задание
                // выполнилось
                if (Interlocked.Decrement(ref remaining) == 0)
                {
                    mre.Set();
                }
            });
        }
        // Ждем, пока все задания выполнятся
        mre.WaitOne();
    }
}
```

Такое решение уже лишено проблем с чрезмерной многопоточностью с затратами на создание потоков. Что еще может помешать в данной ситуации максимально быстрой отработке данного цикла?

В случае если все итерации небольшие по времени и приблизительно одинаково затратные, приведенный подход близок к оптимальному. А если мы представим себе, что некоторые итерации завершаются быстро, а некоторые во много раз дольше, то в таком случае тот поток из пула, которому не повезло быть исполнителем большего количества «долгих» итераций, будет работать в несколько раз дольше остальных. При этом так как время работы всей параллельной операции определяется временем работы самой медленной ее составляющей, то в итоге остальные потоки будут

простаивать, сделав всю «свою» работу, в то время как один «невезучий» будет задерживать всех.

Чтобы в такой ситуации обеспечить минимальное время работы, нужно:

- либо знать природу итераций, чтобы всем потокам поровну раздать «долгие» итерации,
- либо изменить вообще подход к разделению задач между потоками.

Динамическое распределение итераций

От статического распределения можно перейти к динамическому, то есть выдавать каждому потоку некоторую «порцию» работы, сделав которую он получит следующую порцию, если не сделанная работа еще останется к тому времени. Такой подход можно проиллюстрировать этим кодом:

```
public static void ParallelFor(int from, int to, Action<int> body)
{
    int numProcs = Environment.ProcessorCount;
    // количество оставшихся
    int remainingWorkItems = numProcs;
    int nextIteration = from;

    using (ManualResetEvent mre = new ManualResetEvent(false))
    {
        // создаем задания
        for (int p = 0; p < numProcs; p++)
        {
            ThreadPool.QueueUserWorkItem(delegate {
                int index;
                // отбираем по одному элементу на выполнение
                while ((index = Interlocked.Increment(ref nextIteration) - 1) < to)
                {
                    body(index);
                }
                if (Interlocked.Decrement(ref remainingWorkItems) == 0)
                {
                    mre.Set();
                }
            });
        }
        // ждем, пока отработают все задания
        mre.WaitOne();
    }
}
```

Данный код хорош для случая долгих итераций с непредсказуемым временем работы, но для быстрых итераций, в нем слишком много затрат на синхронизацию.

Сбалансированный подход

Таким образом, мы видим, что в зависимости от природы параллелизуемого цикла, может быть выигрышной как стратегия статического распределения заданий между потоками, так и динамического. Логично предположить, что удачной для промежуточных случаев может быть и некоторая сбалансиро-

рованная стратегия. Вот вариант ее кода. Идея в том, что «порцию» мы делаем несколько больше, чем просто один элемент, но меньше, чем в случае статического распределения.

```
public static void ParallelFor(int from, int to, Action<int> body)
{
    int numProcs = Environment.ProcessorCount;
    int remainingWorkItems = numProcs;
    int nextIteration = from;
    // размер «порции» данных
    const int batchSize = 3;
    using (ManualResetEvent mre = new ManualResetEvent(false))
    {
        for (int p = 0; p < numProcs; p++)
        {
            ThreadPool.QueueUserWorkItem(delegate {
                int index;
                while ((index = Interlocked.Add(ref nextIteration, batchSize) - batchSize) < to)
                {
                    int end = index + batchSize;
                    if (end >= to)
                        end = to;
                    for (int i = index; i < end; i++)
                        body(i);
                }
                if (Interlocked.Decrement(ref remainingWorkItems) == 0)
                    mre.Set();
            });
        }
        mre.WaitOne();
    }
}
```

Здесь размер порции данных задан константой, и меняя ее, мы выбираем нужный нам уровень между статическим и динамическим распределением задач.

В любом случае, оптимальные настройки для конкретного случая определяются природой вычислений. Но компромиссный вариант с динамической раздачей более-менее крупных «порций» работы может оказаться вполне приемлемым для большинства ситуаций. Именно так реализованы библиотечные методы параллельного выполнения циклов в .NET 4.

Предлагаемые решения в .NET 4

Для распараллеливания «идеальных» циклов предоставляются следующие варианты:

- класс `System.Threading.Tasks.Parallel` с методами `For()`, `ForEach()`,
- `Parallel LINQ` с методом расширения `AsParallel()`.

Методы `Parallel.For` и `Parallel.ForEach`

Начнем с рассмотрения класса `Parallel` и его методов организации циклов. Первый рассматриваемый нами метод имеет сигнатуру (это базовая, одна из многих перегрузок):

```
public static ParallelLoopResult For(int fromInclusive, int toExclusive, Action<int> body);
```

Она во многом аналогична сигнатурам наших экспериментальных методов выше, кроме типа возвращаемого значения.

Второй «в базовой комплектации» выглядит как:

```
public static ParallelLoopResult ForEach<TSource>(IEnumerable<TSource> source, Action<TSource> body);
```

Рассмотрим, что умеют методы `Parallel.For` и `Parallel.ForEach`.

Обработка исключений

При возникновении исключения в одной из итераций, новые итерации больше не стартуют. Уже стартовавшие, тем не менее, могут закончить свою работу, и уже по их завершении все исключения (изначальное и возможные последующие исключения в дорабатывавших итерациях) агрегируются в одно исключение типа `AggregateException` и оно выкидывается.

Раннее прерывание цикла

Соответствующие перегрузки метода `For` предоставляют возможность прерывать выполнение, вызывая у контекстного объекта (передаваемого в каждую итерацию) методы `Stop` или `Break`.

Отличие `Stop` от `Break` в том, что `Stop` сигнализирует о необходимости прекратить запускать новые итерации вообще, а `Break` — прекратить запускать новые итерации, следующие (по порядку) за той, в которой вызван `Break`. То есть, при вызове `Break` на 5-й итерации гарантируется, несмотря на параллельность, что итерации с 1-й по 4-ю все же будут выполнены. А при вызове `Stop` на 5 итерации, если итерация 4 еще не стартовала, то она и не будет стартовать.

Текущие итерации, которые уже были запущены на момент вызова `Stop` или `Break`, могут проверять статус прерывания цикла, и завершиться раньше времени, если узнают о прерывании всего цикла. Для этого они могут у контекстного объекта проверить соответствующие свойства: `IsStopped` и `LowestBreakIteration`.

Поддержка хранения данных на уровне потока

Некоторые перегрузки метода позволяют сохранять промежуточные результаты вычислений в хранилище данных, локальном для каждого потока. Это позволяет, например, при агрегации результата, сперва агрегировать результаты работы отдельного потока, и только потом уже агрегировать их между всеми работавшими потоками. Это избавляет от излишних затрат на синхронизацию в процессе вычислений.

Возможность конфигурирования уровня параллелизма

Можно указать максимальное количество потоков, используемое для выполнения.

Поддержка вложенности вызовов

Благодаря использованию пула потоков, чрезмерной многопоточности не возникает, как при вложенных вызовах методов `For` и `ForEach`, так и при параллельном их выполнении.

Динамическое изменение количества потоков

`Parallel.For` был спроектирован из расчета изменений во времени загрузки, и с учетом того, что некоторые элементы работы могут требовать большего количества вычислений, чем другие. Поэтому количество используемых потоков может изменяться в процессе работы метода.

Сложное управление загрузкой потоков

В методе была реализована логика балансировки загрузки потоков, которая учитывает большое количество разных факторов. Также, размер блока («порции данных» для обработки) увеличивается в процессе работы, что позволяет получить ▶

более качественную балансировку загрузки для небольшого количества итераций (предполагается, что более «тяжелых»), и меньшие затраты на синхронизацию «раздачи заданий» при большом количестве итераций.

Поддержка отмены выполнения цикла извне его

Для этого используется класс `CancellationTokenSource`. При запуске цикла необходимо передать в него свойство `Token`, и тогда для запроса отмены цикла извне необходимо просто вызвать у объекта `CancellationTokenSource` метод `Cancel()`, что предотвратит запуск новых итераций цикла, и по завершению всех текущих сгенерирует исключение `OperationCanceledException`. Текущие итерации, кстати, могут проверять статус отмены, чтобы «добровольно» завершиться раньше времени, если они узнают, что весь цикл отменен.

Parallel LINQ

Доступный еще в .NET 3.5 в качестве расширения, `Parallel LINQ` доступен в .NET 4.0 сразу в `System.Core`. Идея его использования очень проста: мы добавляем в цепочку LINQ-запросов вызов `.AsParallel()`, и все последующие вызовы в цепочке выполняются параллельно. Например:

```
var doubled = new [] {1, 2, 3, 4}.AsParallel().
Select(i => i*2);
```

Может оказаться неочевидным, что порядок следования элементов в результирующей коллекции может быть произвольным, так как они обрабатываются параллельно. Чтобы сохранить порядок следования элементов в коллекции, мы можем модифицировать наш первый фрагмент вот так:

```
var doubled = new [] {1, 2, 3, 4}.AsParallel().
.AsOrdered().Select(i => i*2);
```

Обработка все равно пройдет параллельно, но этап агрегации (сборки результатов от каждого потока) может длиться дольше, так как нужно будет дожидаться вначале потоков, считающих «первые» элементы, взять их результат, и только потом взять результат от потоков, считавших «последние» элементы.

Важно осознавать при работе с PLINQ, что есть дополнительные накладные расходы на:

1. распределение поступающих элементов исходного перечисления по потокам,
2. агрегацию вычисленных элементов в общую коллекцию.

Причем от распределения работы по потокам никуда не деться, а вот агрегацию можно попробовать отсрочить, а то и вообще обойтись без нее, если стоит задача не получить новую коллекцию, а просто обработать элементы существующей.

Метод `AsParallel()` возвращает не `IEnumerable`, а `ParallelEnumerable`, после чего все другие LINQ-методы (`Select`, `Where` и т.п.) тоже возвращают этот же тип. До тех пор, пока в цепочке передается `ParallelEnumerable`, результаты вычисления каждого из потоков без необходимости не агрегируются, и строится конвейер.

Важно понимать, что когда конвейер рвется, параллелизм прекращается и данные агрегируются в единую результирующую коллекцию. Например, вот этот код работает не так, как можно было бы ошибочно предположить:

```
List<InputData> inputData = ...;

foreach (var o in inputData.AsParallel().
Select(i => new OutputData(i)))
{
    ProcessOutput(o);
}
```

Параллельно здесь выполнится только создание объектов `OutputData`, после чего у всех потоков будут собраны

эти объекты и из них будет образована результирующая коллекция, которая последовательно будет обходиться с вызовом `ProcessOutput()` для каждого элемента.

Чтобы избежать ненужного здесь этапа агрегации, можно воспользоваться методом `ParallelEnumerable.ForEach()`:

```
List<InputData> inputData = ...;

inputData.AsParallel().Select(i => new OutputData(i)).ForEach(o =>
{
    ProcessOutput(o);
});
```

В этом случае после этапа `new OutputData(i)` этап `ProcessOutput(o)` тоже будет выполняться параллельно, причем без этапа агрегации между ними.

Надо отметить, что вызов `Parallel.ForEach()` для `inputData.AsParallel().Select(i => new OutputData(i))` будет иметь тот же недостаток, что и первый пример с обычным `foreach`. В `Parallel.ForEach()` передается `IEnumerable`, а не `ParallelEnumerable`, поэтому перед передачей коллекции в `Parallel.ForEach()` произойдет ее агрегация. Именно для избегания этого существует метод `ParallelEnumerable.ForEach()`, который и следует использовать в этом случае.

Типичные проблемы и ошибки

Рассмотрим возможные проблемы при работе с этими компонентами.



Потокобезопасность собственного кода

Прежде всего, надо понимать, что использование `Parallel.ForEach` само по себе не делает ваш код потокобезопасным — необходимо самому следить, что итерации независимы друг от друга, или если они зависимы, четко предусмотреть потокобезопасную работу с разделяемыми ресурсами.

Также при работе с `Parallel.For` непосредственно не поддерживаются нисходящие циклы и циклы с нестандартным (не равным единице) приращением счетчика. Если ваши начальные алгоритмы написаны таким образом, необходимо внимательно проанализировать их, так как часто нестандартные циклы пишут как раз из-за наличия зависимостей между итерациями (обращение к вычисленному на предыдущем этапе предыдущему элементу массива и т.п.).



Размеры тела цикла

Использование класса `Parallel` предполагает накладные расходы, как минимум,

- на вызов делегата для выполнения тела цикла,
- на синхронизацию между потоками при раздаче заданий.

Если тело цикла достаточно длительное, эти дополнительные расходы играют небольшую роль. Однако если мы распараллеливаем что-то очень простое вроде `i = i*i`, накладные расходы в этом случае превышают полезную работу. Чтобы избавиться от этого недостатка, необходимо «укрупнить» тело цикла. Это очень просто сделать, включив в него не одну, а много итераций.

Вручную это можно сделать, явно разбив входную последовательность на набор блоков, и запустить параллельный цикл по этому набору, а в теле этого параллельного цикла уже последовательно обрабатывать каждый блок. Однако тут придется принимать решение о количестве этих блоков. Его можно возложить на библиотеку и воспользоваться классом, специально созданным для создания поднаборов из входной последовательности `System.Concurrent.Collections.Partitioner`.

С его использованием цикл

```
for (int i = 0; i < length; i++) ▶
```

```
result[i] = i*i;
```

вместо наивной версии:

```
Parallel.For(from, to, i =>
```

```
{
```

```
    result[i] = i*i;
```

```
});
```

можно эффективно распараллелить вот так:

```
Parallel.ForEach(Partitioner.Create(from, to), range =>
```

```
{
```

```
    for (int i = range.Item1; i < range.Item2; i++)
```

```
{
```

```
    result[i] = i*i;
```

```
}
```

```
});
```

Partitioner.Create (from, to) создает тот самый набор блоков, по которому мы проходим параллельно, внутри тела цикла последовательно обрабатывая каждый блок. Тем самым мы обеспечиваем параллельному циклу длительное тело, и распределяем накладные расходы на большее количество полезной работы.



Обработка вложенных циклов

При обработке вложенных циклов возникает вопрос, насколько глубоко стоит проводить распараллеливание. Возьмем пример по обработке прямоугольного изображения:

```
for(int y = 0; y < screenHeight; y++)
```

```
{
```

```
    int stride = y * screenWidth;
```

```
    for (int x = 0; x < screenWidth; x++)
```

```
{
```

```
        rgb[x + stride] = calcColor(x, y); // вычисляем цвет
```

```
}
```

```
};
```

Можно заменить только внешний цикл на вызов Parallel.For, а можно и оба. Если сделать оба цикла параллельными, не окажется ли тело внутреннего слишком мало? Если оставить внутренний последовательным, точно ли мы загрузим работой все наши ядра?

Ответ на эти вопросы может дать только тестирование производительности. Вероятно, если вычисления цвета не слишком сложны, стоит оставить внутренний цикл последовательным. Но если мы работаем с широким, но невысоким изображением, вполне возможно, что одно только распараллеливание внешнего цикла загрузит не все ядра процессора.

Альтернативным путем может быть разворачивание двух вложенных циклов в один, и распараллеливание уже его:

```
int totalPixels = screenHeight * screenWidth;
```

```
Parallel.For(0, totalPixels, i =>
```

```
{
```

```
    int y = i / screenWidth, x = i % screenWidth;
```

```
    rgb[i] = calcColor(x, y);
```

```
});
```

Если calcColor () слишком проста, то чтобы укрупнить тело цикла, можно воспользоваться классом Partitioner, как в предыдущем примере:

```
int totalPixels = screenHeight * screenWidth;
```

```
Parallel.ForEach(Partitioner.Create(0, totalPixels), range =>
```

```
{
```

```
    for (int i = range.Item1; i < range.Item2; i++)
```

```
{
```

```
    int y = i / screenWidth, x = i % screenWidth;
```

```
    rgb[i] = calcColor(x, y);
```

```
}
```

```
});
```



Непотокбезопасные реализации IList

Оба механизма – и Parallel.ForEach, и PLINQ принимают на входе IEnumerable, но при этом пытаются для переданной им коллекции найти наиболее быстрый интерфейс по работе с ней. В частности, для распараллеливания интерфейс IList подходит лучше, чем просто IEnumerable так как в нем есть индексер для произвольного доступа к любому элементу. Поэтому если для переданной коллекции определен IList, то работа с ней происходит именно через этот интерфейс. Это понижает затраты на синхронизацию, но при этом библиотечный код полагается на потокобезопасную реализацию индексера.

Если же используемая коллекция не предоставляет потокобезопасный индексер, что часто может быть в случае, если элементы хранятся в сложном для индексации виде, или вообще подгружаются лениво, необходимо явным образом указать системе, что для нее не нужно использовать IList, а ограничиться IEnumerable.

Чтобы это сделать, подходят два варианта. Первый – создать для коллекции System.Collections.Concurrent.Partitioner:

```
// Здесь может быть использован IList<T>, если он поддерживается коллекцией
```

```
IEnumerable<T> source = ...;
```

```
Parallel.ForEach(source, item => { /*...*/ });
```

```
// А здесь гарантированно будет использоваться IEnumerable<T>
```

```
IEnumerable<T> source = ...;
```

```
Parallel.ForEach(Partitioner.Create(source), item => { /*...*/ });
```

Второй способ очевиден, и знаком по обычному LINQ: просто вызвать для коллекции .Select (item => item). Он подходит как для Parallel.For, так и для PLINQ:

```
// Здесь тоже гарантированно будет использоваться IEnumerable<T>
```

```
source.Select(i => i).AsParallel().Select(i => { /*...*/ });
```



Наличие средства к потоку (thread affinity) у исходной коллекции

При работе Parallel.ForEach и PLINQ каждый из рабочих потоков сам вызывает у исходной коллекции MoveNext (). Если коллекция такова, что доступ к ней возможен лишь из одного определенного потока (обладает «средством к потоку»), как, например, бывает при работе с UI-компонентами в Windows Forms или WPF, то использовать эти механизмы непосредственно для нее нельзя.

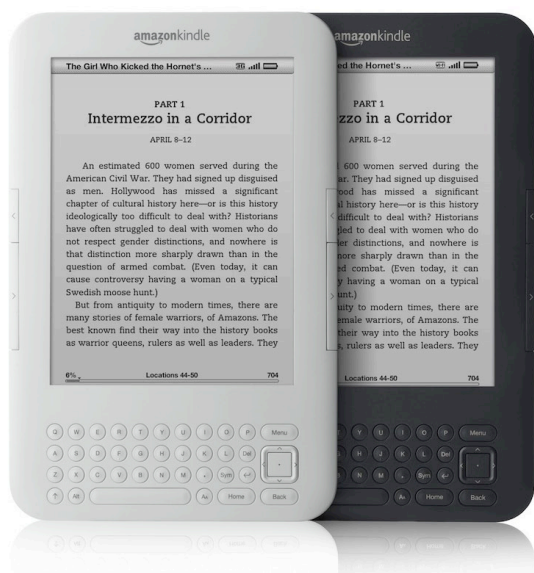
Чтобы обеспечить параллельную обработку такой коллекции, нужно воспользоваться шаблоном Источник-Потребитель (Producer-Consumer), основной «кирпичик» для которого в .NET 4.0 – это класс BlockingCollection, который выходит за рамки этой статьи. ■



ЖЕЛЕЗО

**Ридер Amazon Kindle 3:
обзор с точки зрения русскоязычного пользователя**

Обзор планшета Zenithink zt-180



Автор

Денис Теремий
г. Киев
admin@denter.kiev.ua

Ридер Amazon Kindle 3: обзор с точки зрения русскоязычного пользователя

Про технологию электронной бумаги я узнал, если не ошибаюсь, в году этак в 96-м. Готовился сдавать «тысячи» по английскому и в каком-то штатовском журнале нашел статью с описанием технологии и радужными перспективами коммерческой реализации. До рынка технология добиралась почти десять лет.

Когда же устройства наконец пошли в продажу, стоимость их, на мой совершенно субъективный взгляд, была завышена раза в два от нормальной. Ну не позволяло мне национальное зеленое земноводное выложить три сотни долларов за вещь, которая должна стоить сотню-полторы. Слишком узок функционал ридеров (особенно первых поколений) и слишком много других устройств меньшей стоимости можно использовать для чтения — от КПК и коммуникаторов, до телефонов с джавой и GPS'ов.

Начавшееся в этом году снижение цен на ридеры наконец сместило их стоимость в область, которую я лично готов называть «оправданная покупка». Так что когда Amazon объявил о предпродажной регистрации на Kindle 3-й версии за \$139 (с доставкой в Украину почти \$160), решиться было не сложно.

Проверить возможность заказа для всех стран и доступность бесплатного 3G можно на самом Амазоне в разделе «Experimental Features»

Комплектация



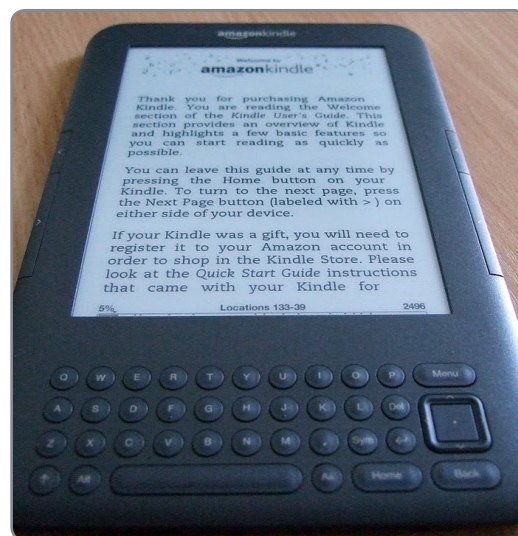
Спартанская до абсолютизма (платить \$35 за оригинальный чехол я все-таки не стал) — сам девайс, блок питания с плоской вилкой, собственно в виде вилки и реализованный, да USB кабель. К сожалению, разъем в девайсе не mini, а micro-USB.



Блок питания — верх минимализма.

Первые впечатления

Посмотрим снаружи: выглядит изящно, сделан качественно. Легкий, тонкий. Матовый корпус без шансов для отпечатков пальцев. О люфтах и скрипах даже вспоминать неловко — подогнано все идеально.



Ряд с цифрами убрали, теперь их можно набирать или с Alt и верхним символьным рядом, или через Sym.

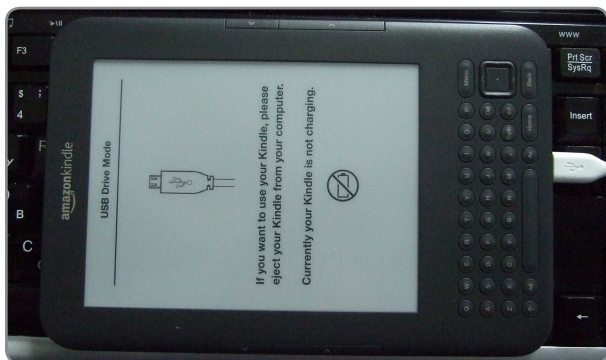
Качельки регулировки громкости, гнездо наушников, от-▶



верстие микрофона (обещают использование в следующих прошивках), micro-USB разъем и слайдер выключателя. Вид сбоку в сравнении с «боевым» HTC Touch и обычным офисным столом.



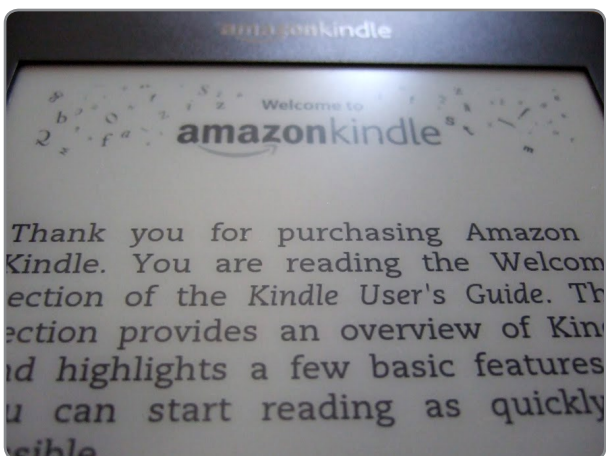
И вид сверху на стандартной клавиатуре – так, полагаю, многим будет проще оценить размеры устройства.



Кстати, при подключении к ПК ридер распознается как внешний USB накопитель и, соответственно, в это время доступа к внутренней памяти не имеет.

Включаем

Первая «холодная» загрузка длится минуты полторы. В принципе с подобной загрузкой практически сталкиваться не придется – из обычного режима ожидания устройство выходит за пару секунд.



Новое поколение e-ink, используемое в 3-й модели, по заявлению производителя дает «на 50% больше контрастности». Не знаю, как они это измеряли, но изображение дей-

ствительно очень хорошее.

При включении попадаем в окно **Home**, которое служит отправной точкой для всех функций ридера. Изначально тут будет только инструкция пользователя, естественно на английском. С ней рекомендую ознакомиться, не все функции и настройки я готов отнести к «интуитивно очевидным».

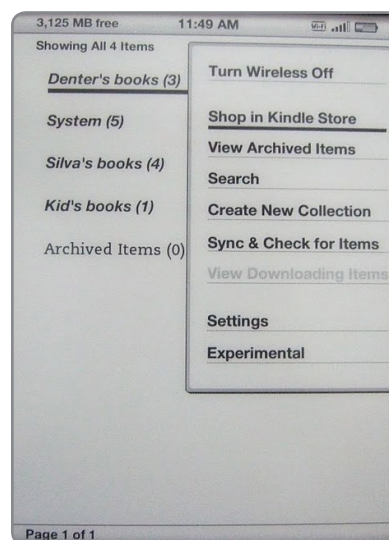
Открываем текст и получаем реально яркое впечатление – страница обновляется «в мгновение ока», на глаз около трети секунды. Как раз столько, сколько нужно взгляду, чтобы перейти с нижней части экрана на верхнюю. По сравнению с теми ридерами предыдущего поколения, что я видел, разница разительная.

В принципе скорость отрисовки экрана позволяет ридеру отображать динамические элементы типа колесика активности в браузере, линии загрузки там же, или даже мигание курсора в строке ввода Гугла. Но о браузере отдельно ниже.

Структура библиотеки

Первое что замечается – структура каталогов файловой системы в ридере не используется. Использована же система «Коллекций», в которые можно закидывать книги (газеты, блоги). Так как одна книга может входить в разные коллекции, то это, по сути, что-то типа тэгов, облегчающих группировку и поиск. С точки зрения файловой системы все книги складываются в один каталог. Это далеко не уникальное решение, но не всем оно понравится. Тем более что удобного поиска для русских названий/авторов мы лишены.

Из окна **Home** вызывается меню настроек и дополнительных функций.



Все сведено к минимуму, что и логично – устройство рассчитано на широкий круг пользователей. Американских, к тому же.

Кстати, возможность создания коллекций появляется только после регистрации устройства на Амазоне. Понимайте как знаете...

Пополнение библиотеки

Очевидно, что стоимость ридера Амазоном уменьшена за счет ожидаемых покупок электронных книг в его магазине. Столь ▶

же очевидно, что тут наши с ним интересы расходятся. Хотя бы потому, что англоязычная литература не пользуется у русскоязычных читателей коммерчески значимым спросом.

Какие же еще есть у нас варианты?

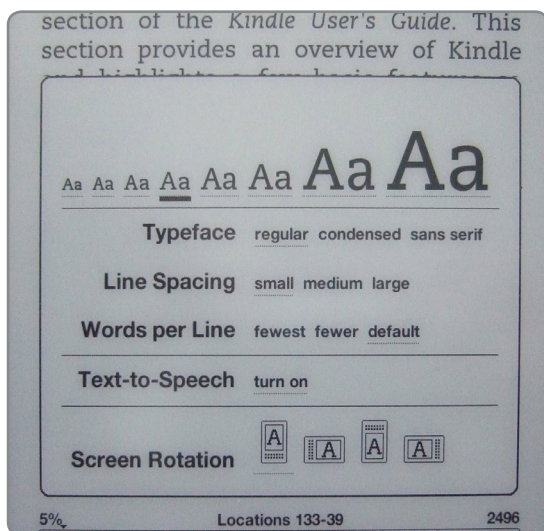
- Подключить ридер к ПК и залить напрямую книги в форматах TXT, PCR, MOBI или PDF в каталог с книгами на устройстве.

- Если у нас уже есть большая и любимая библиотека в FB2, то можно поставить на ПК программу Calibre (кроссплатформенная), которая обнаружит ридер и позволит заливать книги на него «в один клик» с автоматической конвертацией. Да, даже ваши любимые комиксы в архиве картинок она перепакует и залетит в читабельном виде.

- Зайти браузером ридера на сетевую библиотеку типа Флибусты, выбрать формат mobi и скачать книгу напрямую. Кстати, поддержку браузера ридера они прикрутили в течение дня после небольшой переписки. Есть ли такая поддержка в других библиотеках – не проверял.

Чтение текстовых книг

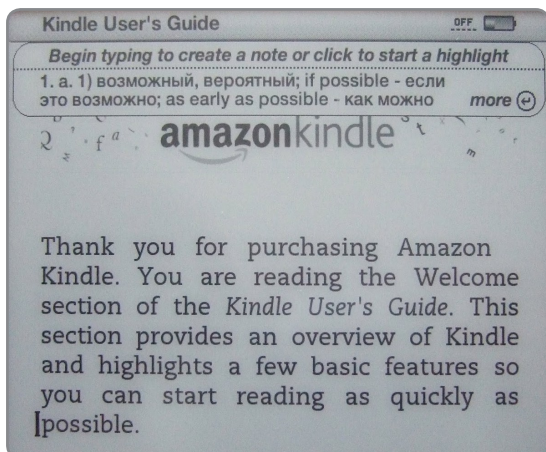
С основной функцией он справляется так, как от него и ожидается – на отлично. Возможности по настройке отображения обычных книг все на фото.



Внизу видна полоска статуса. Опять же отсутствует возможность поиска в «не английских» текстах.

Словари

Полезной функцией ридера является возможность использования словарей прямо при чтении. Джойстиком подводим курсор в тексте к нужному слову и хоп – в течение секунды сверху (или снизу, в зависимости от положения слова) появ-



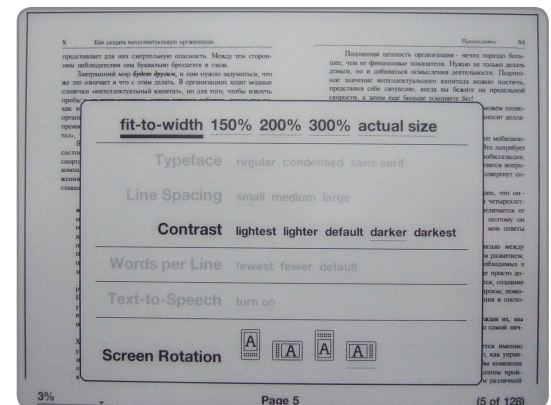
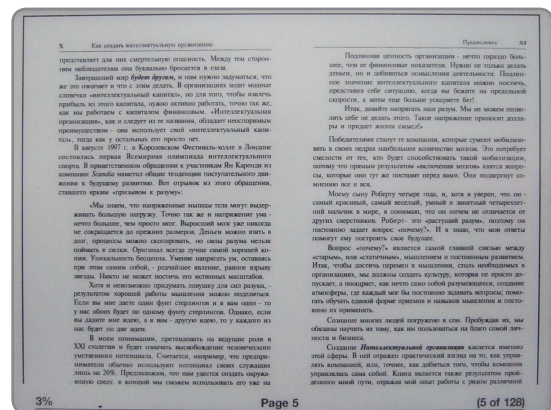
ляется пара строк со значением из текущего подключенного словаря.

По умолчанию в ридере есть пара оксфордских словарей (американский и британский), но никто не мешает найти, залить и подключить любой другой. В данном примере перевод из словаря Мюллера, найденного за 5 минут гугления.

Чтение PDF

Никаких специальных режимов просмотра PDF документа, типа разделения колонок или выделения текста, в ридере нет. Он отображает его именно в том виде, в котором тот сверстан. Но, естественно, делает он это хорошо – двухколоночный документ в ландшафтном режиме читаем вполне спокойно.

Возможности по регулировке отображения только необходимые.



Увеличенная контрастность значительно улучшает читаемость на мелком шрифте.

Так как при просмотре PDF документов джойстик управляет положением видимой части страницы, то курсор в текст (для получения информации из словаря) вызывается через меню.

Подготовка к отображению следующей страницы PDF документа занимает до 2-х секунд.

Отрисовка сканированных PDF, в том числе отконвертированных из djvu, в принципе тоже нормальна. Хотя мне присла-



ли парочку файлов (PDF, сделанный из djvu), в которых одна страница готовилась секунд 40 (да, именно сорок). Но это была проблема использованного конвертера, отконвертированные мной с помощью DjView4 (линуксовая программа) файлы открывались и листались нормально.

Звук

Кроме разъема для наушников, на нижнем торце на задней стенке ридера есть пара стереоколонок. Какой же диапазон аудио возможностей предлагает устройство:

- Проигрывание фоновой музыки. Проигрываются MP3 файлы в порядке заливки на устройство. Т.е. для проигрывания аудиокниг не применимо.
- Прослушивание аудиокниг из амазоновского магазина audible.com. Поверхностное гугление способа преобразования аудиокниг из MP3 в поддерживаемый ридером формат однозначного результата не дало.
- «Text-to-Speech». На английском, естественно. Вполне внятно и даже с претензией на выражение.

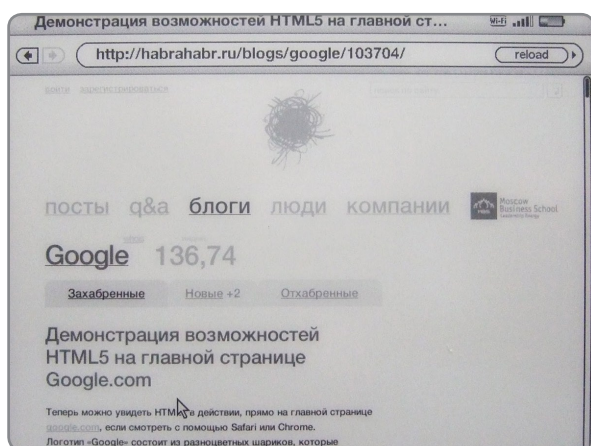
Wi-Fi и браузер

Подключение к беспроводной сети вполне тривиально – отображаем список, выбираем нужную, вводим пароль. Сети AdHoc и корпоративные сети с авторизацией на сертификатах не поддерживаются.

С включенным Wi-Fi адаптером, если верить производителю, устройство проживет в 3 раза меньше – 10 дней вместо 30-ти. Что, на мой взгляд, все равно очень много для устройства с активированным Wi-Fi, разработчики Амазона достигли серьезных успехов в оптимизации энергопотребления.

Кроме возможности подключаться через Интернет к магазину Амазона нас интересует другая, более привычная и нужная для нас возможность – браузер.

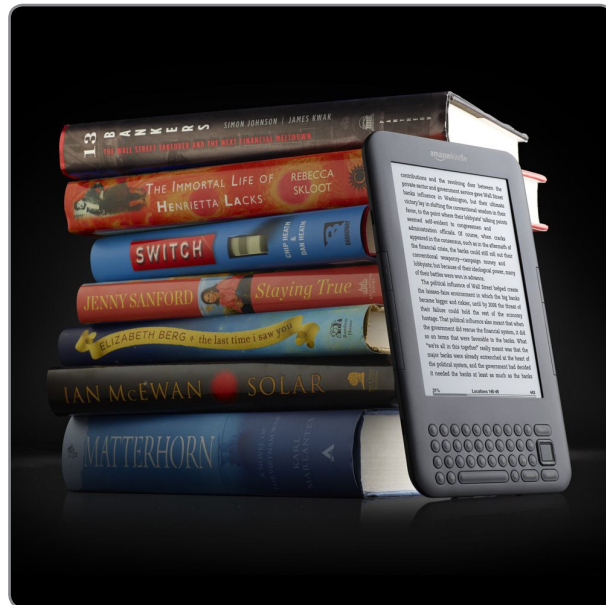
Построенный на Webkit движке браузер справляется со своими задачами в доступном ему объеме: поддерживаются Javascripts и Cookies, есть закладки и режим «Article mode», механизм которого знаком нам по новому Сафари и схожим расширениям для десктопных браузеров. Но отсутствуют вкладки, невозможность ввода русского языка радикально ограничивает диапазон решаемых задач, а управление курсором с помощью джойстика удобным ну никак не назовешь.



Зато теперь для пополнения библиотеки не обязателен компьютер – книги можно брать прямо из сети в любом месте с доступным Wi-Fi. Причем бесплатно, а не за \$9.99 за книгу.

Что осталось за кадром

Можно напомнить, что Kindle 3 – это изначально устройство для работы с магазином Amazon. И в нем есть множество функций для вытягивания денег, получения свежих мате-



риалов и социального взаимодействия. Кроме книг можно покупать подписку на периодику и блоги. Можно интегрировать свою учетную запись в Твиттере или Фейсбуке и оставлять там сообщения с понравившимися фрагментами текста. Можно отправлять тексты на почту и получать на устройстве в уже сконвертированном виде (за деньги). В любом случае – это мне не интересно ни разу. Потому простите, но этот функционал вам придется изучать самостоятельно.

Отдельно нужно упомянуть про модель с 3G. Теоретически это устройство дает владельцу бесплатный мобильный интернет. Да, в том числе и через браузер. По крайней мере, Амазон к этому стремится и заключает с мобильными операторами договора на обслуживание. Но сам Амазон не просто так отнес браузер в раздел «Experimental» – никаких гарантий того, что бесплатный 3G и далее будет доступен для браузера, а не только для покупок в их магазине, он не дает. Т.е. переплачивать или нет за данную «фичу» – решайте сами.

Обновления

Буквально через пару недель после начала продаж вышло первое обновление прошивки, которое должно убрать проблемы с перезагрузкой и зависаниями, иногда возникавшими при открытии PDF (было такое, столкнулся). К моему удивлению, автоматически по беспроводной сети его ставить нельзя, а нужно скачать на ПК (причем выбрав самостоятельно версию), зайти на ридер и уже после этого запустить обновление из меню. Надеюсь, что так коряво это было реализовано исключительно из-за срочности выпуска обновления. ■

+	Качественный экран с великолепной контрастностью и скоростью обновления
	Толщина, вес и общее качество исполнения
	Удобное использование словарей
	Wi-Fi и браузер, обеспечивающие независимость от ПК
	Значительное время работы от батарей
	Минимальная цена на рынке
-	Отсутствие ввода для языков, отличных от английского
	Отсутствие сенсорного экрана (минус условный, чувствуется только в браузере)

Автор

Алексей Дорогов
dorogoff.ru

Обзор планшета Zenithink zt-180

Ввиду нового тренда на планшетные ПК, я решил тоже обзавестись подобным устройством. Анализ рынка показал, что устройств с возможностью подключения USB флешек, а так же в том ценовом диапазоне на который я рассчитывал, не так уж и много, а лидерами здесь являются китайские производители. Поэтому я остановил свой выбор на китайском поставщике, а именно на устройстве Zenithink zt-180 с десятидюймовым экраном.

Где покупалось

До этого времени все покупки, которые касались китайских товаров заказывались на сайте dealextreme, но планшетов там крайне мало, и после недолгих поисков был найден сайт pandawill.com. Там выбор оказался достаточно широким, все было в наличии, и по не очень высокой цене. Отзывы о магазине хорошие, служба поддержки очень быстрая, в режиме реального времени, а не по почте как в DX. Так что я свой выбор остановил на этом поставщике.

Комплектация

Помимо планшета в коробке присутствовали:

- зарядное устройство с переходником
- разъем-переходник для RJ45
- наушники
- инструкция на китайском и английском

По характеристикам он превосходит остальные планшеты в этом ценовом диапазоне, что наравне с размером экрана сыграло решающую роль в покупке данного аппарата.

Внешний вид

Дизайн ничем не отличается от других устройств, экран окаймлен черными полосками, есть кнопка «назад» и светодиод, отражающий состояние планшета (включен/на зарядке). Сбоку имеются несколько кнопок – это собственно включение/отключение, качель **Меню-Домой**, разъем для подключения переходника на RJ45, а так же разъем для microSD. С другой стороны имеем два динамика. С торца выход на наушники, микрофон, перезагрузка, а так же USB и mini-USB, и разъем для зарядки.

Все подогнано ровно, никаких зазоров, ничего не скрипит, не люфтит. Единственное что не нравится, так это то, что он весь глянцевый, разводы и отпечатки остаются моментально как на экране, так и на корпусе. Хорошо, что хоть задняя крышка немаркая, серого цвета и с некоторой шероховатостью.

По внешнему виду особенно рассказывать нечего, нужно только отметить, что чувствительность экрана очень хорошая, при прокрутке или наборе текста всегда четко попадаешь в нужное место, но конечно надо в начале потренироваться. Когда первый раз пытаешься прокрутить вниз/



вверх, получается нажатие, но потом понимаешь логику и все становится прекрасно и удобно. Неудобство заключается только в том, что для кнопок **Меню** и **Домой** сделана качель — иногда просто надоедает выщипывать куда нажимать, конечно можно использовать конечно софтовые кнопки в панели задач, но это на любителя. Если заниматься только просмотром сайтов или просто каким-либо одним приложением, то проблем нет.

Программы и использование

Планшет работает под управлением Android OS 2.1. Про обновление до Frogo ничего неизвестно. Установка новых прошивок удобна: качаем файлы прошивки, запускаем программу и ждем 3-5 минут, все инструкции можно найти через поисковик. Хочу отметить, что аппарат можно спокойно прошить и windows се, нужно только указать соответствующий образ, но я не пробовал, мне нужен был аппарат с Android.

По поводу USB host — все флешки что у меня были сразу завелись, определяются как idisk и можно делать с ними что угодно. Так же в USB можно вставить 3G модем, только он должен быть разлочен или перепрошит, однако это я тоже лично не пробовал.

Wi-Fi работает сносно, имеются инструкции по апгрейду антенны, но у меня заметной разницы после проведения открывательно-паяльных работ не обнаружилось, возможно я выбрал неправильный материал для удлинения антенны. Если уж совсем будет неважно, то можно поменять Wi-Fi модель на простой USB донгл, инструкции тоже есть. Ну а в целом лежать на кровати и смотреть Youtube можно без проблем, в самом дальнем углу квартиры, без прямой видимости, с бетонными стенами ловит одну палочку. А USB донгл для десктопа показывает 36Мбит из 54 возможных, так что можно сделать вывод, что модуль в планшете обладает не очень хорошей антенной, вернее она там практически отсутствует, антенной является просто дорожка.

Про программы особо говорить нечего. Я пользуюсь vid-roid для чтения djvu, а по умолчанию стоит ireader. Проигрыватель для Youtube пришлось ставить отдельно, теперь он играет и HD видео тоже. Проигрывание видеофайлов проблем не вызывает, видео открывается в RealPlayer.

Заключение

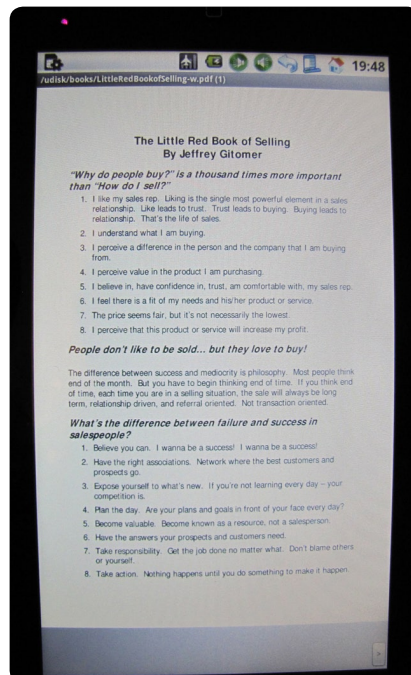
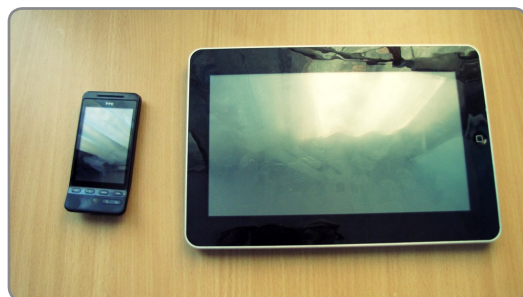
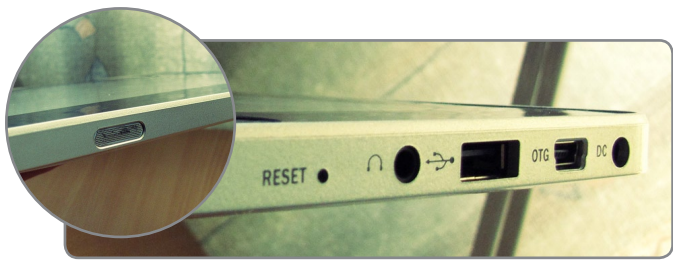
Перед покупкой я сразу определил, что мне нужна читалка перед сном, а так же удобное устройство для просмотра карты. На моем Lenovo S9 как-то все-таки это делать неудобно, и экран небольшой, и зарядку держит недолго, но самое главное это конечно размеры. С нетбука все-таки сложно читать ввиду размера экрана и его тепловыделения. Так что я получил идеальное устройство для своих нужд. На последней прошивке появилась возможность использовать Android Market, правда для этого пришлось проделать некоторые тривиальные телодвижения. Видео играет на отлично даже HD в 720p, и звук хороший. Посерфить тоже удобно.

С включенным Wi-Fi батарея садится через 4.5 часа, с выключенным немногим больше. И это с 2500mah батареей.

Покупал устройство с купоном 5% для новых заказчиков, и EMS доставкой — получилось 6472.29 руб. Добралась посылка за 12 дней, немного потертая, но целая и не помятая.

Пример отображения отсканированной книги в pdf можно посмотреть на картинке, и определить четкость текста. В сети ходит мнение, что на многих планшетах текст плывет, читать невозможно. Опровергаю — в моем случае текст читать удобно и за час глаза не устают.

Один не приятный момент связан с SD карточками — почти сразу сломался фиксатор в разьеме microSD, так что будьте аккуратны, моя карточка в полете проделала немалое количество метров в другой конец комнаты. ■



Технические характеристики

ОС: Android 2.1

Экран: 10", 1024 x 600 px, резистивный

Процессор: ARM V6L, 1GHz

ОЗУ: 256 MB

Аккумулятор: 2500 Mah

Акселерометр: есть

Wi-Fi: 802.11 b/g

Ethernet: через mini-USB

Камера: отсутствует

Порты USB: 1xUSB2.0

Слот для карты памяти: MicroSD, до 16 GB

Габариты: 270*181*16 мм

Вес: 682 г



АВТОРСКАЯ КОЛОНКА

**Наш опыт выступления
на DEMO Fall 2010**

**Осторожно: несанкционирован-
ный трафик**

Синхронизатор данных

**Шифрование и эллиптические
кривые**

DEMO

Автор

Алексей Айларов
«Zingaya Ltd.»
Менеджер проекта
www.zingaya.com
г. Москва
twitter.com/aylarov
aylarov@gmail.com

Наш опыт выступления на DEMO Fall 2010

DЕМО — это конференция, посвященная демонстрации продуктов разных компаний и стартапов. Отобранные участники на сцене рассказывают про свой продукт и демонстрируют его вживую. Причем, требуется именно живая демонстрация, а не презентация в Powerpoint. У стартапа есть 6 минут, в которые очень желательно уложиться, но этого вполне достаточно, чтобы рассказать про продукт и показать, как он работает. Благодаря конференции на стартапы обращают внимание (или не обращают), как инвесторы, так и другие компании, заинтересованные в сотрудничестве.

Мы показывали наш сервис Zingaya, который позволяет разместить на сайте click-to-call виджет для связи с посетителями, и теперь хотим поделиться опытом и нашими мыслями по поводу данного события.

В этот раз конференция проходила в Санта-Кларе, одном из городков «Силиконовой» долины. Погода была очень хорошая: солнышко, тепло +25. Конференция длилась 3 дня и проходила в здании отеля Hyatt.

Первый день прошел в ожидании прослушивания. Прослушивание позволяет команде DEMO настроить оборудование и понять, что вообще будет происходить на сцене во время вашего выступления. Надо заметить, что команда свое дело знает, и каждый отлично выполняет свою работу. На ваши плечи технические проблемы никто не свалит, участники просто должны выступить, как считают нужным. Во время прослушивания, конечно же, произошел интересный случай: после того, как я вставил аудиоджек в мак, его экран погас и, пока не стало понятно, что именно аудиоджек виноват, прошло несколько беспокойных минут в попытке понять, что же из «вставленного» в ноут могло вызвать такой эффект. После того, как джек вынули и снова вставили, все стало нормально (привет Apple!).

Мы выступали во второй день часов в 10 утра, в конце первой партии выступающих, когда подтянулось побольше людей, которые не были сонными, но и не успели устать от постоянного прослушивания выступлений. Все прошло ►



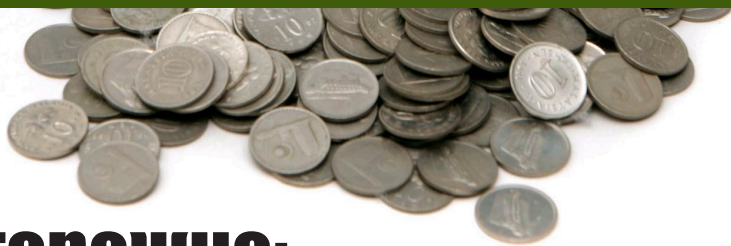
очень гладко. Интернет нас не подвел, наш софт тоже работал отлично. Во время выступления мы показывали звонок с веб-страницы интернет-магазина, торгующего досками для серфинга менеджеру по продажам. Менеджер принимал звонок на своем мобильном, сидя в машине рядом с отелем :).

Третий день был посвящен сервисам для социальных сетей и награждениям. Вручали приз зрительских симпатий и присваивали некие звания DEMOgod. Мы не заработали призов, но и цель у нас была не призы выигрывать (хотя было бы неплохо), а показать наш сервис американской аудитории. После выступлений, кстати, надо было идти в павильон к своему стенду и показывать продукт всем интересующимся.

Какие выводы можно сделать по поводу конференции:

1. Если ваш продукт классно смотрится при демонстрации вживую (wow-эффект), то DEMO подходящее место, чтобы его показать.
2. Если ваши финансовые возможности ограничены, то лучше посмотреть в сторону Techcrunch Disrupt (раньше оно, вроде, называлось Techcrunch50). Помимо платы за участие в DEMO, появляются еще различные статьи расходов, о которых вначале не подозреваешь. Кстати, официальные цены DEMO более чем обсуждаемы.
3. Несмотря на обещания про ошеломительное количество прессы и освещение события везде, где только можно, с прессой было не очень хорошо: прессы мало и работа ее с выступающими никак не организована. Тут DEMO похвастаться нечем. Основная площадка для DEMO – это сайт VentureBeat. По странному стечению обстоятельств Matt Marshall – главный организатор DEMO и одновременно CEO в VentureBeat. Больше было похоже на то, что DEMO пиарит само себя.
4. Инвесторы могут реально заинтересоваться вашей технологией и компанией, то есть они за этим событием следят. Не все, конечно, но многие. Поэтому желательно посещать DEMO не только на 3 дня, а на неделю-две (а то и больше), чтобы было время со всеми пообщаться.
5. Конференции помогают донести информацию о вашем продукте, сделать официальный «запуск», но не надо думать, что это панацея. Это просто один из способов заявить или напомнить о себе. ■





Автор

 Михаил Ивановский
www.komtet.ru

Осторожно: несанкционированный трафик

И для кого не секрет, что при подключении любого сетевого устройства к интернету генерируется трафик, который мы «не заказывали». Основные источники:

- паразитный трафик;
- нежелательный трафик от программного обеспечения;
- вирусная активность;
- несанкционированное использование подключения.

И... Google?

В статье нет пошаговых инструкций по обеспечению сетевой безопасности. Но надеюсь, пользователей материал заставит задуматься, а опытные сетевые администраторы дополнят информацию конкретикой.

Паразитный трафик

Неконтролируемая сетевая активность: даже если ваше подключение 100% закрыто, при наличии реального фиксированного IP-адреса, а не «серого» и/или динамического IP-адреса, предоставляемого провайдером для внутренней сети, из интернета передаются широковещательные, ICMP, keepalive-пакеты и разнообразный служебный трафик. В этом случае любой может обратиться к вашему IP-адресу, используя простейшую команду ping X.X.X.X, средства поиска уязвимостей, или попытаться нарушить работу подключения, сгенерировав syn flood командой hping3 (для Linux-систем). При этом генерируется и учитывается входящий трафик.

Более того, даже если ваше сетевое оборудование выключено, трафик может быть сгенерирован и учтен интернет-провайдером пока из ARP-кеша провайдера (в котором хранится соответствие IP и MAC-адресов) не удалится запись о вашем подключении. Время обновления зависит от настроек у провайдера, обычно это несколько минут (в маршрутизаторах Cisco по умолчанию 4 часа).

Если злоумышленник не ставит себе целью атаковать ваш IP-адрес, то такой трафик представляет несколько Кб в сутки, но избежать его не представляется возможным. Разумеется, интернет-провайдеры обычно предпринимают необходимые меры по защите.

Трафик от программного обеспечения

Современное программное обеспечение

установленное на вашем компьютере, часто не спрашивая вашего разрешения явным образом, обращается к внешним сервисам для передачи служебной информации, запроса обновлений и их загрузки. Т.е. данный трафик не является необходимой частью работы программы. Так, например, в интернет для проверки обновлений и регистрационной информации обращается ПО Adobe, Microsoft и т.п. А ОС Microsoft еще и по умолчанию загружает обновления, размер которых может превышать сотни Мб.

Выходом из данной ситуации может служить отключение обновлений и обращение к внешним ресурсам в настройках программного обеспечения (не всегда это возможно) и установка программного файрволла (сетевой экран), который при всех запросах на обращение к внешним ресурсам будет запрашивать ваше решение. «Режим обучения» присутствует в большинстве современных файрволлов.

Вирусная активность

Возникает за счет заражения вашего компьютера вирусами. Это худший вариант из перечисленного, т. к. кроме создания паразитного трафика, злоумышленник может получить доступ к вашему компьютеру со всеми вытекающими из этого последствиями.

Решение – установка антивирусов и регулярное обновление операционной системы и ПО. Причем это относится не только к Windows-системам. Уязвимости обнаруживаются с завидной регулярностью во всех ОС, и проверку на руткиты никто не отменял.

Существует специальное программное обеспечение, которое отслеживает выявление уязвимостей и информирует о наличии обновлений для распространенного ПО. К примеру, для Windows-систем существует Secunia Personal Software Inspector. Для Linux-систем, если программное обеспечение устанавливалось из официальных репозиторий, те же задачи выполняют «Менеджеры пакетов».

Несанкционированное подключение

При низком уровне безопасности, несанкционированное использование подключения характерно для локальных сетей предприятия, а с распространением Wi-Fi устройств и для обычных пользователей. Очень часто пользователи не заботятся о безопасности беспроводных соединений, открытых (или недостаточно защищенных) точек доступа можно обнаружить значительное количество – только с моего балкона «видно» две. Должен предостеречь, что при подключении к открытой точке доступа ее владелец может сканировать трафик на наличие в открытом виде логинов и паролей доступа.

Если с локальными сетями предприятий все понятно – надо нанимать квалифицированных системных администраторов, то владельцам домашних точек доступа настоятельно рекомендуется ознакомиться с документацией для обеспечения максимальной безопасности. На эту тему есть много информативных статей.

И Google?

Откровенным сюрпризом для меня стал просмотр логов, в которых видно, что каждые полчаса, в течение нескольких суток, с адресов Google при заблокированном компьютере проходит около 6Мб, достаточно запустить Google Chrome. Такая ситуация наблюдается, даже если открыта одна пустая вкладка. К сожалению, цели данного трафика мне выяснить пока не удалось.

Разумеется, за весь несанкционированный трафик на ваш IP-адрес придется платить. Могу заметить, что по требованию большинство провайдеров без проблем предоставляют подробную детализацию трафика. Есть случаи, когда интернет-провайдеры идут на встречу клиентам и корректируют трафик. Особенно, это касается юридических лиц. Конечно, есть и кардинальное решение – безлимитные тарифы. ■

chrome.exe [2660]

TCP OUT	192.168.1.2:4876
TCP OUT	192.168.1.2:4936
TCP OUT	192.168.1.2:4937
TCP OUT	192.168.1.2:4944
TCP OUT	192.168.1.2:4911

74.125.79.102:443	126 B {...	66 B
74.125.79.100:80	66 B	66 B
74.125.79.100:80	66 B	174 B {...
74.125.163.84:80	987,8 K...	23,8 KB ...
74.125.79.102:80	66 B	66 B



Автор

Александр Стаханов
Developer Express Inc.
www.devexpress.com

Синхронизатор данных

На работе и в повседневной жизни нам довольно часто приходится сталкиваться с различного рода синхронизацией данных. Вы синхронизируете файлы и папки своего телефона с компьютером, выполняете привычные действия в системах контроля версий, используете различного рода интернет-службы синхронизации контактов, почтовых сообщений и документов и зачастую даже не задумываетесь, как этот процесс реализован в том или ином случае.

Если вы решили написать собственный синхронизатор, то скорее всего столкнетесь с рядом вопросов. В этой статье мы поделимся опытом написания такого компонента и рассмотрим требования, предъявляемые к нему. В основу этих требований легли всевозможные пожелания, полученные от пользователей, и реальные сценарии использования синхронизатора событий планировщика XtraScheduler. Потому в качестве примеров кода будем приводить фрагменты кода от указанного продукта.

Для начала определим, какие объекты будут участвовать в процессе синхронизации. Это два набора данных (исходный и целевой/конечный) и синхронизатор, который выполняет ряд операций над этими наборами, в результате которых целевой набор должен измениться в соответствии с реализуемым сценарием.

Объект синхронизатор

Сценарий синхронизации определяет, каким функционалом будет обладать ваш синхронизатор. Если синхронизация подразумевает только одну «главную» копию и будет выполняться полным замещением содержимого другой, то подойдет простой вариант вида импортер/экспортер. Если же планируется выполнять синхронизацию наборов независимых записей, то придется делать более сложную реализацию синхронизатора.

Базовый класс

Создайте базовый класс синхронизатора, определяющий общее для всех наследников поведение и интерфейс методов, свойств и событий. Назовем такой класс, например, `SynchronizerBase`. Этот класс может определять строго определенный порядок вызовов абстрактных методов, для выполнения основных необходимых действий синхронизации в нужной последовательности. Функционал будет расширяться путем наследования. Наличие базового класса освободит вас от дублирования кода. Такие общие операции, как инициализация внутренних структур и свойств, могут быть реализованы один раз в этом классе.

Дополнительным плюсом такого подхода будет являться то, что вы получите единый стройный API для всех реализуемых впоследствии наследников.

```
public abstract class SynchronizerBase {
    //...

    protected SynchronizerBase(StorageBase sto-
```

```
rage) {...}

    public event ExchangeExceptionHandler
    OnException;

    protected internal abstract void Synchro-
    nizeCore();

    public virtual void Terminate() {...}

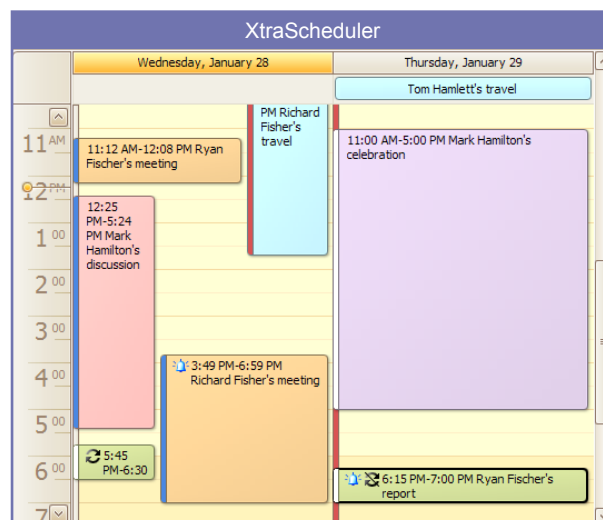
    public virtual void Synchronize() {
        Storage.BeginUpdate();

        try {
            ResetTermination();
            SynchronizeCore();
        }

        finally {
            Storage.EndUpdate();
        }
    }
}
```

Разделение обязанностей

В зависимости от сценариев работы с наборами данных, вы можете реализовать соответствующих наследников `SynchronizerBase`, которые «умеют» выполнять последовательность заложенных в сценарии действий, заранее «зная», какой набор является «главным». Такие специализированные классы будут гораздо проще в использовании, чем настройка и использование единственного, но «умеющего все и сразу». ►



Таким образом, вы можете создать несколько наследников, например, `ImportSynchronizer` и `ExportSynchronizer`, реализующих основную логику синхронизации для каждого из сценариев. Эти классы могут остаться абстрактными, если в дальнейшем вы планируете реализовывать их конкретных наследников для различных наборов данных.

Например, в `XtraScheduler` у нас получилась следующая схема базовых классов, которая изображена на рис. 1.

Выделение алгоритмов в подклассы

Чтобы не делать объект синхронизатора слишком нагруженным, имеет смысл организовать архитектуру, отделив реализацию алгоритма синхронизации от интерфейса самого синхронизатора. Выделите подкласс в классе-синхронизаторе, не забыв при этом наладить взаимодействие между этими объектами.

```
public class ImportSynchronizeManager :
ImportManager {

    public ImportSynchronizeManager(Appointmen
tExchanger synchronizer) : base(synchronizer) {

    }

    protected internal override void Perform
rmExchange() {

        Appointments.BeginUpdate();

        try {

            PrepareSynchronization();

            SynchronizeAppointments();

            DeleteNonSynchronizedAppointments();

        }

        finally {

            Appointments.EndUpdate();

        }

    }

}

public class ExportSynchronizeManager : Ex
portManager {

    public ExportSynchronizeManager(Appointmen
tExchanger synchronizer) : base(synchronizer) {

    }

    protected internal override void Perform
Exchange() {

        PrepareSynchronization();

        SynchronizeOutlookAppointments();

        CreateNewOutlookAppointments();

    }

}
```

Операции над объектами наборов данных

В зависимости от сценария синхронизации, наборы данных могут модифицироваться по-разному. Но, как правило, все действия с объектами наборов сводятся к трем основным действиям:

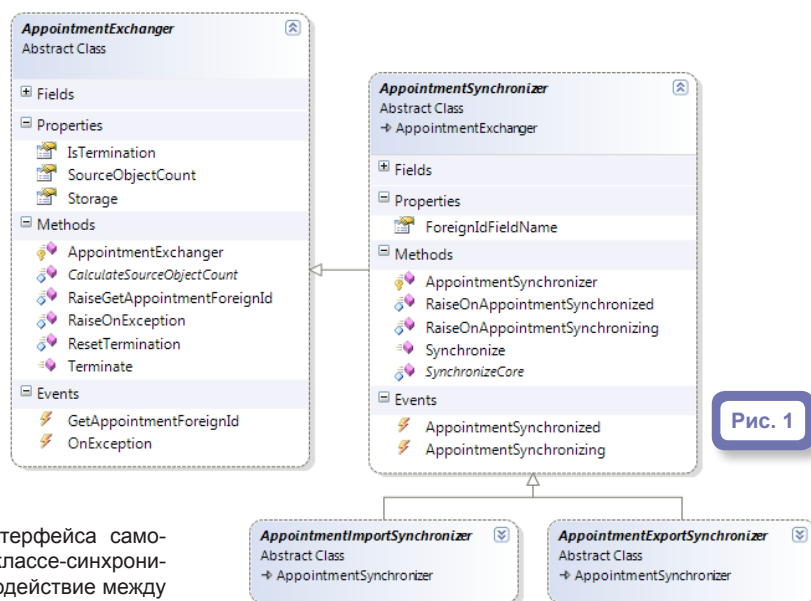


Рис. 1

- создать новую копию объекта на целевом наборе на основании объекта из исходного набора данных;
- обновить соответствующий объект в целевом наборе с учетом изменений объекта в исходном наборе;
- удалить «лишние» объекты на целевом наборе, которые отсутствуют в исходном.

Все описанные выше операции можно свести к следующей таблице:

Исходный набор	Целевой набор	Действие на целевом наборе
✓	✗	Создать
✓	✓	Обновить
✗	✓	Удалить

✓ - объект присутствует в наборе данных

✗ - объект отсутствует в наборе данных

Описанные действия должны быть реализованы в том или ином виде в каждом из наследников вашего синхронизатора. При этом необходимо учесть важный момент, какую копию данных считать «главной». В зависимости от выбора, исходный и целевой наборы могут меняться местами. Именно поэтому классы `ImportSynchronizer` и `ExportSynchronizer` будут выполнять противоположные действия над наборами данных.

Поддержка событий

Несомненно, при выполнении любого действия над объектами наборов, пользователь захочет иметь возможность получить доступ к этим объектам «до» и «после» выполнения операции. Организуйте пару событий `Synchronizing` и `Synchronized` в базовом классе.

Определите аргументы обработчика событий `SynchronizingEventArgs` и `SynchronizedEventArgs` и добавьте туда поля и свойства для соответствующих объектов из синхронизируемых наборов. В случае, когда в базовом классе это невозможно сделать сразу, воспользуйтесь наследованием аргументов и сделайте недостающие свойства в наследниках.

```
public class AppointmentEventArgs : EventArgs {

    public AppointmentEventArgs(Appointment apt) {

    }
}
```



```

    public Appointment Appointment { get { ... } }
}

public class AppointmentCancelEventArgs :
AppointmentEventArgs {

    public AppointmentCancelEventArgs(Appointment apt) : base(apt) {

    }

    public bool Cancel { get { ... } }

}

public class AppointmentSynchronizingEventArgs : AppointmentCancelEventArgs {

    public AppointmentSynchronizingEventArgs(Appointment apt) : base(apt) {

    }

    public SynchronizeOperation Operation { get { ... } }

}

public class OutlookAppointmentSynchronizingEventArgs : AppointmentSynchronizingEventArgs {

    public OutlookAppointmentSynchronizingEventArgs(Appointment apt, AppointmentItem apt) : base(apt) {

    }

    public AppointmentItem OutlookAppointment { get { ... } }

}

```

Рассмотрите необходимость наличия событий на обработку исключений. Например, вы можете реализовать событие `OnException` и перенаправлять полученные исключения туда. Дайте возможность пользователю решать – стоит ли продолжать процесс синхронизации после возникновения исключения. Дополните аргументы события всей необходимой информацией.

```

iCalendarImporter importer;

//...

importer.OnException += new ExchangeExceptionEventHandler(importer_OnException);

void importer_OnException(object sender, ExchangeExceptionEventArgs e) {

    iCalendarParseExceptionEventArgs args = e as iCalendarParseExceptionEventArgs;

    if (args != null) {

        ShowErrorMessage(String.Format("Cannot parse line {0} with text '{1}'",
            args.LineIndex, args.LineText));

    }

    else

        ShowErrorMessage(e.OriginalException.Message);

    e.Handled = true; // prevent this exception from throwing

}

}

```

Сохранность данных

Поддержите для выполнения каждой операции над объектами наборов возможность отмены. Это можно сделать путем добавления свойства `Cancel` в аргументы события `SynchronizingEventArgs`.



Будьте уверены, что это обязательно пригодится для выполнения сценария «объединения» двух независимых наборов данных, когда имеет смысл отменять **все** операции на удаление при выполнении сначала синхронизации сначала в одну сторону, а потом в другую.

```

AppointmentImportSynchronizer synchronizer;

//...

synchronizer.AppointmentSynchronizing += new AppointmentSynchronizingEventHandler(synchronizer_AppointmentSynchronizing);

void synchronizer_AppointmentSynchronizing(object sender, AppointmentSynchronizingEventArgs e) {

    // ...

    if (ShouldCancelOperation)

        e.Cancel = true;

}

```

Покрывание всех сценариев манипуляции с объектами

Предусмотрите ситуацию не только отменять предложенное «по умолчанию» действие над объектом, но и дать возможность выполнить другую допустимую операцию. Например, добавьте в аргументы дополнительное свойство `SynchronizeOperation { Create, Replace, Delete }` и дайте пользователю возможность указать желаемое значение. Тем самым, вы получите возможность удалить объект на целевом наборе данных, вместо замещения его копией из исходного набора.

Такой подход дает возможность более точно обрабатывать «конфликты правок» в наборах данных.

```

void synchronizer_AppointmentSynchronizing(object sender, AppointmentSynchronizingEventArgs e) {

    switch (e.Operation) {

        case SynchronizeOperation.Replace:

            if (ShouldDeleteInsteadReplace.Checked)

                e.Operation = SynchronizeOperation.Delete;

            break;

    }

}

```

Элементы UI

Попробуйте спроектировать синхронизатор таким образом, чтобы он не использовал выдачу сообщений, диалогов и прочих визуальных элементов. Потому что всегда найдется пользователь, которому понадобится использовать ваш ком- ▶

понтент, выполняя синхронизацию в потоке, созданном из приложения или, например, из специального сервиса.

Кэширование данных

Может возникнуть потребность загрузить набор данных, сохранив его объекты в синхронизаторе. Это имеет смысл, когда желательно избежать повторных обращений к реальному набору данных. К тому же, в процессе выполнения синхронизации целевой набор данных может меняться, что потенциально может проявиться в проблемах, связанных с итерированием по этому набору из кода синхронизатора.

Параметризация набора данных

Рано или поздно пользователь захочет синхронизировать не весь набор, а только его часть, ограниченную, к примеру, временным интервалом или специфическим параметром. Решением будет являться написание провайдера исходных данных с возможностью «подсунуть» пользователю провайдер, определяющий свою логику. При этом синхронизатор должен получать данные не напрямую, а через провайдер. В этом случае придется использовать описанное выше кэширование данных внутри синхронизатора.

```
void DoFilteredImport() {
    OutlookImport importer = new OutlookImport(
        this.schedulerStorage);

    TimeInterval interval = CalculateCurrentWeekInterval();

    ParameterizedCalendarProvider provider =
        new ParameterizedCalendarProvider(interval);

    importer.SetCalendarProvider(provider);

    importer.Import(System.IO.Stream.Null);
}
```

Завершение процесса

Предусмотрите возможность завершить процесс синхронизации по желанию пользователя. К примеру, заведите метод `Terminate` в базовом классе `SynchronizerBase`. Такая функция может быть полезна, когда при синхронизации больших наборов данных возникло исключение или объект набора не удовлетворяет неким условиям и необходимо немедленно прервать дальнейшее выполнение. В таком случае пользователю уже не придется дожидаться окончания процесса.

```
void synchronizer_OnException(object sender,
    ExchangeExceptionEventArgs e) {
    // ...

    AppointmentImportSynchronizer synchronizer =
        (AppointmentImportSynchronizer) sender;

    synchronizer.Terminate();

    e.Handled = true;
}
```

Расширяемость

Может случиться так, что пользователь захочет синхронизировать свойства, отличные от тех, которые синхронизирует ваш компонент. Предоставьте возможность определять это на уровне задания пользовательских свойств или объектов и предоставьте все необходимое API для этого.

```
void exporter_AppointmentExporting(object sender,
    AppointmentExportingEventArgs e) {
    iCalendarAppointmentExportingEventArgs
        args = e as iCalendarAppointmentExportingEventArgs;

    AddEventAttendees(args.VEvent, «test@corp.
```

```
com»);
}

private void AddEventAttendee(VEvent ev, string
    address) {
    TextProperty p = new TextProperty(«ATTENDEE»,
        String.Format(«mailto:{0}», address));

    p.AddParameter(«CN», address);

    p.AddParameter(«RSVP», «TRUE»);

    ev.CustomProperties.Add(p);
}
```

Дайте возможность пользователям наследоваться от вашего синхронизатора для переопределения какой-либо функциональности. Предусмотрите это при проектировании классов и методов.

```
public class MyOutlookImportSynchronizer : OutlookImportSynchronizer {
    public MyOutlookImportSynchronizer(SchedulerStorageBase storage) : base(storage) {
    }

    protected override OutlookExchangeManager CreateExchangeManager() {
        return new MyExchangeManager();
    }
}
```

Вспомогательные классы

Создайте методы для получения параметров инициализации синхронизатора (например каталог, указываемый на конкретный календарь) или любой другой необходимой для синхронизации информации. Для более удобного использования вынесите их в отдельные вспомогательные классы или сделайте доступными прямо в компоненте.

```
public class OutlookExchangeHelper {
    public static string[] GetOutlookCalendarNames();

    public static string[] GetOutlookCalendarPaths();

    public static List<OutlookCalendarFolder> GetOutlookCalendarFolders();
}

public class OutlookUtils {
    public static _Application GetOutlookApplication();

    public static void ReleaseOutlookObject(object obj);
}

public class iCalendarConvert {
    public static TimeSpan ToUtcOffset(string value);

    public static string FromUtcOffset(TimeSpan value);

    public static string UnescapeString(string str);
}
```

Надеюсь, что приведенный выше «сборник советов» поможет вам наилучшим образом определить объем необходимой вам функциональности перед написанием синхронизатора и даст возможность избежать ошибок в процессе его реализации. ■



Автор

Алексей Цуцулис
neverwalkaloner@gmail.com

Шифрование и эллиптические кривые

В качестве предисловия к статье позвольте напомнить, что представляют собой гибридные криптосистемы и почему они получили такое широкое распространение. Итак, *гибридной криптосистемой* принято называть способ передачи большого объема зашифрованных данных, при котором данные шифруются секретным ключом с применением симметричного алгоритма (например, AES или DES), а сам ключ передается зашифрованным асимметричным шифром (как, скажем, RSA). Такой способ получил широкое распространение за то, что он вбирает в себя преимущества как симметричной, так и асимметричной криптографии. Большой блок данных шифруется очень быстрым симметричным алгоритмом (и это избавляет от серьезных временных затрат), а ключ шифрования передается надежно зашифрованным с помощью асимметричного алгоритма (это решает проблему распределения ключей, свойственную только симметричным методам). Все это достаточно широко известно, поэтому перейдем к самому главному, а именно к вопросам реализации. И вот тут начинается много интересного.

Немного об общем

На первый взгляд наиболее оптимальным решением является комбинация RSA+блочный шифр. Однако этот способ не лишен некоторого количества подводных камней.

Самым очевидным среди них является семантическая слабость алгоритма RSA. Поэтому применяя алгоритм RSA можно только с использованием схемы дополнения, такой как RSA-OAEP. Кратко напомним, что в RSA-OAEP используется две криптографические стойкие хеш-функции, а также такие методы как добавление к строке сообщения определенного количества нулевых бит. Это делает схему RSA-OAEP не самым эффективным с точки зрения реализации асимметричным алгоритмом. Из-за такого ограничения использование RSA, как наиболее простого варианта теряет всю свою привлекательность.

Еще одним недостатком совмест-

ного использования RSA и блочного шифра является отсутствие контроля целостности передаваемых данных. И это уже камень в огород не асимметричного шифра, все-таки RSA-OAEP с задачей проверки целостности получаемых данных отлично справляется. Сейчас речь идет непосредственно о блочном алгоритме.

Об этой неприятности я хочу рассказать подробнее. Как известно все блочные шифры поддерживают несколько режимов работы. Наибольшее распространение получили режимы ECB и CBC. В режиме ECB исходные данные разбиваются на блоки $M_1, M_2, \dots, M_n$ определенного размера. Криптотекст состоит из блоков $C_1, C_2, \dots, C_n$, где $C_i = E(M_i)$, а $E(M_i)$ – функция шифрования, примененная к блоку M_i . Собственно недостаток такого режима сразу бросается в глаза. Злоумышленник может свободно подменить любой блок C_i на блок C_o . Таким образом, если при передаче данных один ключ используется несколько раз, то злоумышленник может перехватив новое сообщение, заменить некоторые данные на более старые, причем законный получатель не заметит подвоха.

Режим CBC по мнению многих способен устранить этот недостаток блочных шифров, но на самом деле это не так. В режиме CBC шифрование происходит следующим образом. Исходные данные разбиваются на блоки $M_1, M_2, \dots, M_n$ определенного размера, а криптотекст состоит из блоков $C_1, C_2, \dots, C_n$, где

$$C_i = E(M_i \oplus C_{i-1})$$

При таком способе создается иллюзия, что все блоки шифротекста связаны и замена одного блока приведет к потере остальных. На самом деле это не так. Замена отдельно взятого блока повлияет на расшифровку только одного(!) следующего за ним блока. Поэтому злоумышленник может успешно подменить несколько блоков, а законный получатель приняв эти данные и заметив, что только один блок получен неправильно вполне может и не придать этому значения.

Поэтому говоря о гибридном шифровании необходимо понимать, что

передаваемая таким способом информация должна состоять не из двух блоков, а из трех $E_{asy}(K) || E_{sym}(M) || MAC_K(M)$, где $E_{asy}(K)$ – зашифрованный асимметричным алгоритмом ключ, $E_{sym}(M)$ – зашифрованные симметричным алгоритмом с применением ключа K данные M , $MAC_K(M)$ – код аутентификации сообщения M , полученный с применением ключа K .

В связи с этим появляется еще одно небольшое неудобство, связанное с применением асимметричной криптосистемы наподобие RSA в гибридных схемах – это избыточные данные. Чтобы передать сообщение M , нам итак необходимо добавлять к криптотексту лишнюю строку бит $MAC_K(M)$. А помимо этого еще и RSA, в силу большого размера модуля шифрования (в настоящее время рекомендуется 2048 бит), увеличивает исходный размер шифруемого ключа в разы. Конечно избыточность не критична, всего 2048 бит (с учетом MAC-функции), но наряду со сложностями реализации стоящими за RSA-OAEP это все дает повод поискать другой способ гибридного шифрования и такой способ есть. Называется он DHIES.

Коротко о главном

Схема гибридного шифрования DHIES (Diffie-Hellman Integrated Encryption Scheme) была предложена тремя авторами Michel Abdalla, Mihir Bellare и Phillip Rogaway в 2001 году. Идея лежащая в основе схемы строится на трудности решения задачи дискретного логарифма. Сама схема шифрования заключается в следующем.

Предположим, что Алиса хочет отправить Бобу зашифрованное симметричным алгоритмом сообщение и ключ к этому сообщению. Пусть у Боба имеется пара открытый/закрытый ключ ($x, G, P, Y = G^x \bmod P$), где G, P, Y – открытые данные, а x – секретный ключ Боба. Т. е. набор ключей Боба соответствует стандартам DSA и ГОСТ Р 34.10-2001 (это важно, потому что избавляет от необходимости генерировать дополнительную пару ключей).

Для отправки сообщения Алиса выполняет следующие действия: ►

1. Генерирует случайное большое число k .

2. Вычисляет $U=G^k$ и $V=Y^k=G^{kx}$.

3. Определяет пару $(K1, K2)=H(V||U)$, где $H()$ – криптографически сильная хеш-функция.

4. Находит $C=E_{K1}(M)$ и $T=MAC_{K2}(C)$

5. Отправляет Бобу сообщение вида $U||C||T$.

Получив сообщение от Алисы, Боб используя свой секретный ключ x , может вычислить $V=U^x$. Зная V и U , Боб в состоянии получить пару ключей $(K1, K2)=H(V||U)$. С помощью ключа $K2$, он проверяет целостность полученного шифра $T=MAC_{K2}(C)$ и в случае если результат верен, расшифровывает текст при помощи ключа $K1$, $M=D_{K1}(C)$.

Преимущества схемы DHIES заключается в простоте реализации асимметричного шифрования. Так, чтобы получить надежно зашифрованный симметричный ключ и передать его, необходимо всего два возведения в степень. В то время как при использовании RSA-подобных систем еще необходимо будет позаботиться о схемах дополнения.

Кроме этого схема основывается на задаче дискретного логарифма, а следовательно ее легко можно перенести на эллиптические кривые. Единственный нюанс использования DHIES на эллиптических кривых заключается в том, что вычисление G^x на самом деле означает нахождение точки $x*G$, а в остальном все остается по-прежнему. Известно, что эллиптическая криптография работает на пространствах с гораздо меньшим размером (например, 256 бит вместо 2048). И тем самым, применяя схему EC-DHIES, мы избавляемся от избыточных бит (тут речь, разумеется, идет не об экономии трафика, немного уменьшить избыточность – это дело чести).

К тому же нет необходимости создавать дополнительную пару открытый/закрытый ключ для передачи зашифрованных данных. Если получатель использует цифровую подпись с алгоритмом DSA или ГОСТ Р 34.10-2001, то для отправки сообщения можно использовать открытый ключ применяемый для верификации подписи.

И совсем чуть-чуть кода

Напоследок небольшой класс для выполнения математических операций над точками эллиптических кривых, с помощью которого можно реализовать EC-DHIES.

```
//класс для умножения точек эллиптической кривой
public class ECPPoint
{
    public BigInteger x;
    public BigInteger y;
```

```
    public BigInteger a;
    public BigInteger b;
    public BigInteger FieldChar;

    public ECPPoint(ECPPoint p)
    {
        x = p.x;
        y = p.y;
        a = p.a;
        b = p.b;
        FieldChar = p.FieldChar;
    }

    public ECPPoint()
    {
        x = new BigInteger();
        y = new BigInteger();
        a = new BigInteger();
        b = new BigInteger();
        FieldChar = new BigInteger();
    }

    //сложение двух точек P1 и P2
    public static ECPPoint
    operator+(ECPPoint p1, ECPPoint p2)
    {
        ECPPoint p3=newECPPoint();
        p3.a = p1.a;
        p3.b = p1.b;
        p3.FieldChar = p1.FieldChar;

        BigInteger dy=p2.y-p1.y;
        BigInteger dx=p2.x-p1.x;

        if (dx < 0)
            dx += p1.FieldChar;
        if (dy < 0)
            dy += p1.FieldChar;

        BigInteger m = (dy *
        dx.modInverse(p1.FieldChar)) %
        p1.FieldChar;

        if (m < 0)
            m += p1.FieldChar;

        p3.x = (m * m - p1.x -
        p2.x) % p1.FieldChar;
        p3.y = (m * (p1.x -
        p3.x) - p1.y) % p1.FieldChar;
        if (p3.x < 0)
            p3.x += p1.FieldChar;
        if (p3.y < 0)
            p3.y += p1.FieldChar;
        return p3;
    }

    //сложение точки P с собой же
    public static ECPPoint
    Double(ECPPoint p)
    {
        ECPPoint p2=newECPPoint();
        p2.a = p.a;
        p2.b = p.b;
        p2.FieldChar=p.FieldChar;

        BigInteger dy = 3 * p.x
```

```
* p.x + p.a;
        BigInteger dx = 2 * p.y;

        if (dx < 0)
            dx += p.FieldChar;
        if (dy < 0)
            dy += p.FieldChar;

        BigInteger m = (dy *
        dx.modInverse(p.FieldChar)) %
        p.FieldChar;

        p2.x = (m * m - p.x -
        p.x) % p.FieldChar;
        p2.y = (m * (p.x - p2.x) -
        p.y) % p.FieldChar;
        if (p2.x < 0)
            p2.x += p.FieldChar;
        if (p2.y < 0)
            p2.y += p.FieldChar;

        return p2;
    }

    //умножение точки на число
    x, по сути своей представляет x
    сложений точки самой с собой
    public static ECPPoint
    multiply(BigInteger x, ECPPoint
    p)
    {
        ECPPoint temp = p;
        while (x != 0)
        {
            if ((x % 2) != 0)
            {
                if ((temp.x==p.x)
                || (temp.y==p.y))
                    temp=Double(temp);
                else
                    temp = temp + p;
                x = x - 1;
            }
            x = x / 2;
            p = Double(p);
        }
        return temp;
    }
}
```

Заключение

Конечно, каждый должен выбирать тот способ, который лично ему кажется наиболее удобным, но знание еще одного варианта никогда не бывает лишним. Надеюсь, данная статья была вам полезна и помогла узнать еще один метод (на мой взгляд, оптимально сочетающий удобство и надежность) сохранения ваших данных в секрете. А применять этот метод или выбрать что-то другое решать, несомненно, только вам. ■

FastVPS.ru

VPS от 129 рублей

- 100 MB RAM
- 300 MHz CPU
- 6 GB HDD

**Dedicated от
2199 рублей**

- Intel® Core™ i7-920
Quad-Core incl
- 8 GB DDR3 RAM
- 2 x 750 GB HDD



- *один из лучших
дата-центров Германии*
- *служба поддержки
на русском языке*
- *уникальная система
восстановления*
- *удаленный reboot/reinstall*
- *установка сервера
в течение 24 часов*