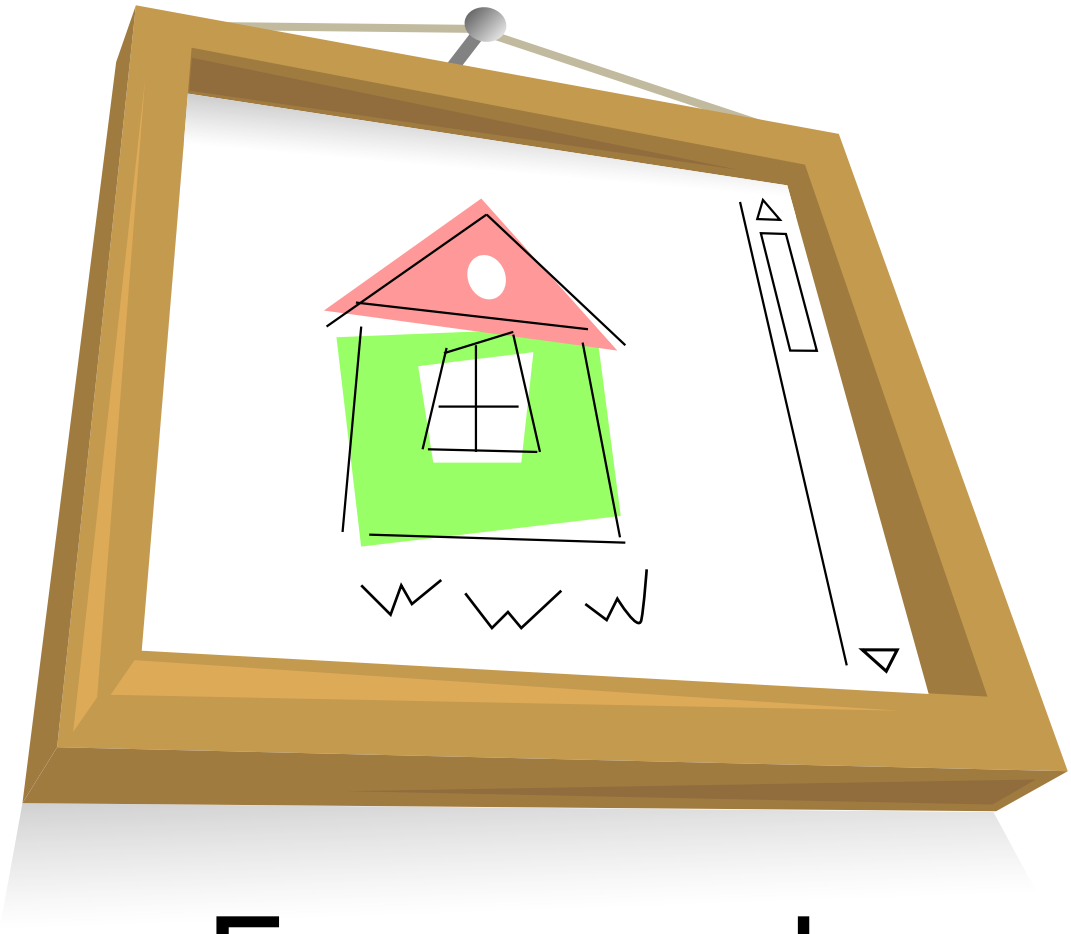
Март '05 № 11

php{INSIDE}

электронный журнал для веб-разработчиков



Frameworks...

Zend Platform: подробности

Где моя CVS, чувак?

издается с февраля 2004 года, www.phpinside.ru

Содержание

Анонс

Конференция в Киеве

PHPClub (http://phpclub.ru) совместно с компанией Миротел (http://mirotel.net/) и редакцией журнала «PHP Inside» (http://www.phpinside.ru/) проводят **4-ю международную конференцию «Современные технологии эффективной разработки веб-приложений с использованием PHP»**.

Киев, Украина 12-13 мая 2005 г.

http://www.phpconf.ru/2005/

Конференция, проводимая под эгидой PHP-Клуба в последние два года, является уникальным мероприятием, собирающим со всего постсоветского пространства ведущих веб-программистов, PHP-энтузиастов и других талантливых программистов, чья профессиональная деятельность в той или иной степени связана с веб-технологиями.

В рамках конференции у вас будет отличная возможность познакомиться с современными тенденциями разработки веб-приложений, обсудить со специалистами интересующие вас вопросы, найти оптимальные решения для вашего бизнеса, да и просто пообщаться с единомышленниками в теплой и неформальной обстановке. Предыдущие PHP конференции показали, что материалы, информация и опыт, полученные на конференции, участники с успехом применяют на практике, значительно повышая эффективность своей работы.

Место проведения - Киев, величественная красота его золотых куполов, раскинутых на правом берегу Днепра, цветущие каштаны, которые каждую весну превращают город в сказочное королевство. Приглашаем вас своими глазами увидеть достойный удивления и восхищения неповторимый облик Киева.

Команда номера

Авторы и переводчики
Денис Баженов
Денис Попель
Андрей Олищук
Дмитрий Шейко
Данил Миронов

Редакционная коллегия
Александр Смирнов
Александр Войцеховский
Андрей Олищук [nw]
Антон Чаплыгин
Дмитрий Попов
Елена Тесля

Выпуск номера
Андрей Олищук [nw]
Антон Чаплыгин
Денис Зенькович
Алексей Волков

Контактные данные
http://phpinside.ru
nw@phpinside.ru

Тематика конференции включает следующие направления*

- XML Sapiens как универсальная концепция сайтостроения в разрезе XML/PHP
- Две стороны документирования - PHPdocumentor и DocBook
- Поддержка нескольких СУБД в проекте.
- Свой проект свободно распространяемого Программного Обеспечения
- Вебсервисы на примере Xforms
- Разработка современной CMS
- Оптимизация PostgreSQL
- MySQL: индексы и оптимизация
- Влияние тестирования на дизайн PHP кода (TDD в PHP)
- Платежные системы - это не страшно (мастер-класс)
- XML в PHP5

С подробной программой конференции можно познакомиться по адресу: http://phpconf.ru/index.php?m=4

Труды конференции будут опубликованы в виде полных текстов принятых статей в специальном номере журнала PHPInside (http://phpinside.ru).

Время и место проведения PHP конференции

12-13 мая 2005 г. КиевЭкспоПлаза, павильон №2, зал №1 метро Нивки

Стоимость регистрационного взноса

560 грн. (3000 руб.) В стоимость регистрационного взноса входит:

- Возможность участия во всех мероприятиях конференции (2 дня)
- Получение раздаточных материалов
- Кофе-брейки и обеды
- Участие в лотерее участников конференции

P.S: Участие докладчиков в конференции - бесплатно (+ небольшой бонус)

Внимание! Скидки для групп от 3-х участников - с одного члена группы 500грн. (2600руб).

Контактные данные

Оргкомитет конференции http://www.phpconf.ru/

E-mail: 2005@phpconf.ru

Телефон: +380 44 494 03 50

Факс +380 44 494 03 51

Председатель оргкомитета:

Андрей Зинченко

+380 67 440 49 41

2005@phpconf.ru

Информационный партнер в Москве:

Александр Смирнов

phpclub@rambler.ru

7-095-783-2659

В фокусе

Каркасы, каркасы, каркасы…

Автор: Денис Баженов

В девятом выпуске PHPInside была размещена моя статья «Теория разработки Framework систем». После ее публикации в форуме phpclub'а прошло маленькое обсуждение. Направление этого обсуждения привело меня к пониманию того, что некоторые моменты требуют пояснения (хочу поблагодарить критиков за замечания и советы по доработке статьи). Так же я решил сделать описание нескольких уже существующих framework – систем.

Сразу хочу оговориться, что анализ, приведенный в данной статье никак нельзя считать исчерпывающим, так как на полное сравнение и выявление плюсов/минусов потребовалась бы ни один месяц, потому что истинная сущность таких систем познается, что называется «в бою». Данный анализ является, в большинстве своем, теоретическим (хотя для выявления некоторых особенностей использовалась и практика), однако даже с помощью такого поверхностного анализа можно выявить особенности разных реализаций.

От читателя данной статьи я ожидаю знаний понятия паттерна как такового, а также знание архитектуры паттерна MVC в частности. Излагать этот материал я не вижу смысла, так как все уже давно написано и выверено. Любой желающий может найти достаточно материала по данному вопросу.

Описание существующих решений

php.MVC

Вообще phpMVC – это порт знаменитой «Jakarta Struts Framework» на PHP. Java, наверное довольно неплохая платформа для WEB-приложений. Однако о phpMVC я подобного сказать не могу.

Объем распакованных файлов более полутора мегабайт, что заставляет задуматься о серьезности проекта.

То количество классов, которые используется в системе, поражает. Реализация паттернов слишком запутанная. Разобраться в такой канонаде классов будет непросто, особенно человеку не профессионалу, который не знаком с понятием паттерна и основными реализациями.

Структурного контроля системы я не увидел. Приложения интегрируют код системы на этапе выполнения.

Контроля ошибок выполнения тоже нет. Исключительные ситуации, генерируемые во время выполнения приложения – обрабатываются стандартным обработчиком ошибок.

Стандартные (тестовые) приложения, которые поставляются с системой (и, по сути являются ее визитной карточкой) реализованы явно не в рамках паттерна MVC. Разделения бизнес-логики и представления я не увидел. PHP-код идет вставками в шаблонах. Придется искать верстальщика со знанием PHP или постоянно отвлекать программиста, иначе есть риск того, что верстальщик испортит код приложения. В системе присутствует популярный шаблонизатор Smarty/2.5, что наталкивает на мысль о том, что какое-то разделение все-таки существует, однако примеров его использования или описания как использовать я не нашел. Наверное, все делается через банальный «include», а далее документация по Smarty в руки и вперед.

Поддержка БД реализована в виде драйверов к механизмам абстракции PEAR::DB и ADODb. Никакой унификации интерфейсов для этих двух пакетов я не увидел. Поэтому смысла в этих драйверах тоже не вижу. С таким же успехом можно использовать PEAR или ADO напрямую.

Исходный код отлично документирован, так что необходимую информацию все же найти можно. Но отсутствие структурной документации все же затрудняет этот процесс.

Ну и, наконец, время генерирования пустой страницы. Из всех представленных framework'ов у phpMVC самый низкий результат. 0,191-0,209 секунды. Не тяжело понять, что один поток сервера, на моем компьютере, более четырех-пяти страниц за секунду не обработает. Даже для моей, пусть не серверной, платформы (AMD 2.5GHz/512MB RAM) это довольно низкий результат.

В общем, у меня осталось довольное серое впечатление об этом framework'е.

LIMB (WACT based)

Этот пакет разрабатывается нашими соотечественниками. В качестве базиса используется другой framework, WACT. Это единственный известный мне случай, когда на основе одного framework'а строят другой, судя по названию, совсем отличный от базового.

На самом деле ребята взялись лишь за высокоуровневые интерфейсы приложения. Интерфейс БД и шаблонизатор были взяты из WACT'а.

Система обладает довольно интересной и, что на мой взгляд немаловажно, оптимальной архитектурой. Я советую изучить ее, перед тем как браться за что-то свое. Несмотря на относительную сложность структуры (которая, в принципе, схожа с phpMVC), она довольно прозрачна и вполне достойна, быть ориентиром для людей, которые испытывают сложности в проектировании. Разобраться в «потрохах» limb'а не доставит вам больших трудностей. А если доставит, то всегда можно обратиться к разработчикам, которые любезно отвечали на вопросы во время моей работы с этим пакетом. Надо заметить, что вопросов было немного.

Тем временем...

В MySQL существует функция SUBSTRING_INDEX(str,delim,count), которая возвращает часть строки str состоящую из count количества подстрок разделенных делимитерами delim.

Пример использования: mysql> SELECT SUBSTRING_INDEX ('www.mysql.com', '.', 2);
-> 'www.mysql'

Информацию по другим строковым функциям MySQL можно найти по адресу: http://dev.mysql.com/doc/mysql/ru/string-functions.html

WACT обладает неплохим интерфейсом работы с базами данных. Вы можете работать с ADODb, с PEAR::DB либо через классы прямого доступа к БД. В отличие от phpMVC, интерфейс абстрагирован, и вам не понадобиться менять код приложений при смене сервера баз данных или смене интерфейса

Шаблонный движок WACT'а тоже достоин рассмотрения. Шаблоны похожи на XML-документы и частично реализуют идею ASP.NET. Однако сносной документации по движку мне найти не удалось, как в принципе и по системе в целом. Разработчики утверждают, что скоро будет выпущена структурная документация на русском языке. В целом шаблонный движок показался мне немного сложным. Разработчики сделали отображение более приоритетным. Данные достаются отображением из модели. Это усиливает нагрузку на верстальщика, который в принципе и не должен знать, что такое модель или паттерн. На мой взгляд, идеальным является случай, когда верстальщик и программист принимают соглашение об именовании переменных, программист заполняет их, а верстальщик их выводит в нужном месте.

Никакого контроля ошибок времени выполнения, к сожалению, опять таки нет.

На мой взгляд, очень неплохая платформа для реализации своих приложений. Было бы идеально присутствие в системе XSLT шаблонизатора, но такого в ближайшее время не предвидится.

Framework-системы. Путь к величию или самообман?

Framework система – это «важнейшая часть интеллектуальной собственности и даже корпоративной культуры компании занимающейся web-разработками всерьез» (http://phpclub.ru/talk/showthread.php?threadid=62146 – Автор: mariroz)

Framework… Когда я слышу это слово, меня переполняет целая масса эмоций. Что такое framework??? Это вопрос, на который еще не было дано должного и исчерпывающего ответа. Мою позицию подтверждает окружающая действительность. Ведь куда не посмотришь везде framework'и, а остановить свой взгляд на чем-то конкретном и не получается. Есть отличная литература, в которой рассматривается архитектура подобных систем, но четких ответов на интересующие вопросы нет. Каждый волен думать и поступать, так как ему хочется.

Мне же кажется, что обсуждение систем подобного плана надо вести в другом ракурсе. В форуме мне не раз указали на такие вещи как programming pattern и прочее. Да, это все хорошо. И в принципе без этого теперь немыслима ни одна каркасная система. Но мы с вами говорим не о том. Рассмотрение таких фундаментальных вещей как MVC, singleton, ORM, которые находят прямое применение во framework'ах - это тема, как минимум, для одной книги и нельзя сказать, что маленькой. Да и надо ли этим заниматься?

По этой тематике существует немалое количество качественной литературы. Да, реализация этих технологий тоже, порой, нелегкое дело. Но ведь мы с вами, на самом деле, не сделали и более простых вещей. А именно ядра этой системы. Наверное, в прошлой статье я не очень ясно выразил свое мнение по отношению к ядру этих систем. Если вы работали с какими-то конкретными framework системами, то задайтесь вопросом: «А где в моей framework системе ядро?». Сможете ли вы дать четкий и ясный ответ на этот вопрос? В большинстве случаев вряд ли. Почему? В одном форуме было дано такое определение framework системе: «это обычно просто набор стандартных библиотек». С технологической точки зрения примерно так все и обстоит. Но, скажите мне, тогда и PEAR – это framework система? Я так не считаю. Возможно, вы уже задались вопросом: «А зачем вообще нужно ядро как таковое»? Давайте подумаем…

А как должно быть?

Конечно же, мои слова не претендуют на роль последней истины – это было бы слишком самоуверенно с моей стороны. Но я хочу изложить свой взгляд на порядок вещей в нашем вопросе.

Framework-система – это не просто набор стандартных библиотек. Это система, которая определяет некоторые стандарты кодирования и обладает интерфейсом работы с модулями. Модули – это ключевое понятие для CMF системы! Ядро обладает лишь системной функциональностью и даже не подозревает о том, что такое базы данных или паттерн MVC. Вся необходимая функциональность реализуется в модулях. Зачем это надо? Поймите меня правильно, я люблю MVC, я уважаю XML и все XML-подобное и не имею ничего против множества других полезных технологий. Но иногда мне нужно писать чистый PHP код без контроллеров, модели, отображения и всех прочих благ, которые в большинстве других случаев, очень ценю. Иногда у меня возникает необходимость писать целые приложения без использования паттерна MVC или шаблонизатора как такового. Например, группа скриптов для работы с графиками (построение графиков по данным, по функциям) или с изображениями (генерирование картинки предпросмотра с маленьким разрешением и сохранение ее в кэше сервера). Поэтому я хочу иметь под рукой такую систему, от которой я бы смог отключить все ненужное и подключить все нужное одним взмахом руки над клавиатурой. И в итоге получить приложение максимально эффективное по соотношению удобство программирования/быстродействие (а как вы знаете, эти величины находятся в обратно-пропорциональной зависимости). К сожалению многие framework системы этого не позволяют, так как данная функциональность (вроде интерфейса MVC паттерна или интерфейса к БД) в них «вшита». Если быть точнее, то система как раз и заключается в этой функциональности и не существует сама по себе. Для реализации же того, о чем я говорю, необходимо абстрактное модульное ядро. Ядро, загрузчик, модуль и приложение - вот ключевые компоненты системы. Проведите аналогию с родственными терминами из теории Операционных Систем и, возможно, вам многое станет яснее.

Тем временем...

Для перекодирования строковых значений из одной кодировки в другую, в PHP существуют несколько функций. К примеру, для перекодировки из кириллической кодировки windows-1251 в другую кириллическую KOI8-R, можно использовать функцию: convert_cyr_string($string,'w','k') Подробную информацию по жданной функции смотрите здесь: http://ru.php.net/manual/ru/function.convert-cyr-string.php

Если одна из кодировок не является исключительно кириллической (или обе), то лучше всего использовать функцию iconv(). Вот пример перекодирования из KOI8U в UTF-8: iconv("KOI8-U", "UTF-8", $string) Подробнее о функции iconv() можно прочитать здесь: http://www.php.net/manual/ru/function.iconv.php

Я пришел к пониманию того, что framework система должна обладать хорошо выраженной структурой с централизованным управлением с помощью этого самого ядра. Ядро может выполнять разные функции, но главная из них коммуникативная! На основании API этого ядра должна создаваться вся остальная функциональность. Это в некоторой мере облегчает создание более высокоуровневых компонентов, так как определенная функциональность уже выполняется ядром. К этой функциональности можно отнести, например, функции обработки ошибок и работы с конфигурационным хранилищем.

Тем временем...

Чтобы получить информацию не только о браузере пользователя, но и о его возможностях (например, поддержка JavaScript и CSS), можно воспользоваться функцией get_browser(), которая возвращает всю полезную информацию в виде PHP-массива.

Более подробно о функции get_browser() можно узнать здесь: http://ru.php.net/manual/ru/function.get-browser.php

Проблемы отрасли

Да. У этой отрасли инженерного знания, конечно же, есть проблемы. И вы их, наверное, хорошо понимаете. Отсутствие хорошей теоретической информации именно по построению ядра таких систем, а не высокоуровневых компонентов. Наверное, это обусловлено тем, что та интегрирующая (коммуникативная) функция ядра, о которой мы говорим, почти заканчивается на операторе «include». Однако надо помнить, что «качественное» ядро должно уметь и много другого. Например, оно должно обладать удобными механизмами генерирования URL'ов. Все эти вопросы остаются на усмотрение программиста. Наверное, это и есть самое страшное.

Еще одна проблема, которую я уже высказывал – это неверное, на мой взгляд, понимание задач. Например, из всех мною виденных framework-систем, только несколько перехватывали ошибки времени выполнения. Разработчики зря обходят этот вопрос стороной. Ведь PHP предоставляет отличные средства для отладки. Для примера взгляните на отладочную информацию, которую генерирует моя framework-система в случае исключительной ситуации. С точки зрения средств языка – ничего экстраординарного.

Необходимо рассматривать framework-систему как «черный ящик», о задачах которого ничего неизвестно. И наращивать функциональность слоями - это способствует модульности системы и более низкому уровню связанности модулей.

Выводы

Это очень тяжелая тема. И я вам скажу мое мнение по поводу того, почему она такая тяжелая. Отнюдь не потому, что люди не могут применять паттерн MVC или уровень абстракции от БД. Она тяжелая потому, что framework-система определяет стиль кодирования, который нам, иногда, так не хочется менять (если только не мы сами его придумали). Она тяжелая потому, что как я считаю, framework-систему легче написать самому для себя, так же как легче самому составить распорядок своего дня, потому что никто не сделает это лучше вас, так как этот «кто-то» не знает СТИЛЯ вашей жизни. Тем более, что написание основы такой системы - дело, в принципе, довольно не сложное и не должно занять много времени.

Еще одним очень важным моментом является документация. Документация CMF системы – это ее жизнь. Без качественной документации система не представляет никакой ценности ни для кого, и через определенный промежуток времени даже для ее разработчика. Кто-то уже высказал идею о том, что документацию framework-системы надо писать раньше, чем саму систему. Именно этими правилами я руководствовался при создании своей framework-системы.

Я думаю, что рано или поздно мы создадим достаточно хорошую framework-систему, которая устроит если не всех, то большинство. И тогда в наш мир ворвется еще одна революционная вещь сродни .NET Framework или Delphi VCL (а ведь это тоже в определенной степени framework-система). Остается только работать и верить.

Знакомство с Freeform Framework

Автор: Денис Попель

С выходом первых версий PHP5 веб-разработчики получили богатый арсенал современных технологий написания приложений. Самым революционным шагом стало внедрение развитого и мощного объектно-ориентированного расширения, которое включает и интерфейсы, и исключения, и механизм рефлексии. С другой стороны, многие соглашаются с мыслью, что классическое использование PHP как смеси простых инструкций и HTML-кода является очень неэффективным для разработки и поддержки больших корпоративных приложений. При таком подходе проблематичными остаются редизайн, добавление новых компонентов или быстрое изменение частей кода в существующих страницах.

В таком традиционном подходе пользователю веб-приложения предлагается взаимодействовать со страницами сайта, в то время как для разработчика намного привычней создавать программные модули, которые реагируют на действия этого пользователя - переход по гиперссылке с некоторыми параметрами для приложения или заполнение формы.

Многие могут поспорить, что написание скрипта, который генерирует страницу в ответ на отправку данных в форме от пользователя не есть ничто иное, чем реакция на действия пользователя, но на самом деле разработчик в первую очередь создает программный код, который обрабатывает поступившие данные.

Таким образом, каждое веб-приложение имеет две стороны - это и обработка данных, и создание страниц для пользователя. В связи с этим при разработке больших приложений есть смысл разделить работу на две большие части - отделить обработку данных (где в первую очередь задействованы программисты) от создания результирующих страниц (где задействованы веб-дизайнеры и некоторая система, которая обрабатывает шаблоны страниц и внедряет в них данные из приложения).

На самом деле, этот подход уже давно широко используется и называется MVC-паттерн. Главная его идея - разделить обработку данных, хранение и выборку данных и подготовку результирующей страницы между разными довольно независимыми разработчиками или их командами. Наибольшее применение он нашел при разработке именно веб-приложений, существует целое множество систем, начиная от обработчиков форм и шаблонных систем до полноценных фреймворков - наборов скриптов или классов, которые позволяют быстро создавать большие приложения командой разработчиков. Конечно, поскольку PHP является самым популярным (хотя пока и не самым приемлимым для разработки корпоративных веб-приложений), создано множество MVC-фреймворков для PHP. Но с приходом PHP5 многие оказались неготовы использовать его новые преимущества, кроме того, большинство - это клоны имеющихся систем из Java/Apache Projects.

Большинство на самом деле не имеют встроенного средства обработки шаблонов, поэтому приходится кроме самой системы изучать и отдельный продукт - обработчик шаблонов. В связи с этим нами была предпринята достаточно успешная попытка создать оригинальный (не портированный) объектно-ориентированный проект, который бы действительно был легок в изучении и использовании и решал все недостатки существующих систем. В конце прошлого года была выпущены первые пробные версии Freeform Framework, нового MVC фреймворк. Основными его отличиями являются:

1. Очень четкая, понятная и расширяемая архитектура
2. Компактное абстрактное ядро
3. Мощный, доступный по умолчанию, обработчик шаблонов
4. Единый интерфейс для обработки данных форм
5. Поддержка неограниченного числа клиентов - архитектура не накладывает ограничений на обработчики шаблонов; они могут генерировать любые выходные документы - веб-страницы, XML-файлы, картинки и т.д. Кроме того, фреймворк поддерживает обработку входных данных от различных клиентов, например, данные могут приходить и с PDF-документа, или какого-либо другого источника, но они всегда будут доступны через единый интерфейс к данным запроса, поэтому код является абсолютно клиентонезависимым
6. Развитое средство эмулирования пакетов классов и файлов ресурсов - отказ разработчиков от пакетов (пространств имен) вызвали горячую дискуссию и сомнения, что этот шаг не позволит создавать с помощью PHP действительно масштабированные и управляемые библиотеки или приложения. Freeform Framework частично решает эту проблему через эмуляцию пакетов классов. Конечно, это не полноценная замена соответствующих синтаксических конструкций, поэтому разработчики должны следить за тем, чтобы все классы имели уникальные имена. С другой стороны, при использовании Freeform разработчики уже не должны беспокоиться о включении необходимых файлов классов - используя новые возможности PHP5 система сама подгружает файлы классов по мере необходимости (т.е. при первом обращении к какому-либо классу). Кроме того, в месте с пакетами можно распространять и файлы ресурсов - шаблоны страниц, мультимедиа и т.д. Интерфейс доступа к ресурсам позволяет очень просто находить такие файлы независимо от путей инсталляции самого приложения. Пакеты также поддерживают конфигурирование и доступ к конфигурационной информации через простой интерфейс.

Тем временем...

Функция phpinfo() может использоваться с параметрами, с помощью которых есть возможность выводить на экран не полную информацию, а только нужные части.

К примеру, вызов phpinfo(32) приведет к выводу на экран только секции с предопределенными переменными, включая переменные окружения.

Более подробную информацию смотрите по адресу: http://ru2.php.net/manual/ru/function.phpinfo.php

Простая архитектура

Как уже было отмечено, одной из целей разработки была простая, открытая и расширяемая архитектура. По сути, единственной стандартизированной частью является механизм обработки каждого запроса:

1. Каждый запрос обрабатывается фронт-контроллером

2. В зависимости от типа входного запроса выбирается класс-адаптер, который понимает, как разобрать запрос (например, PHP "понимает" только GET и POST запросы типа application/x-www-form-urlencoded и multipart/form-data, в то время как приложение может быть спроектировано так, что оно должно принимать данные, например, в XML-формате). Адаптер обрабатывает запрос и возвращает его данные в виде, понятном для остальной части приложения
3. Создается экземпляр класса-обработчика сессии
4. В зависимости от входящих данных строится и инициализируется экземпляр класса Action, который и является реализацией механизма обработки действия пользователя. Следуя идеологии MVC, для каждой функциональности приложения создается отдельный класс-потомок Action
5. Применяется политика безопасности и в зависимости от того, имеет ли пользователь право выполнять текущее действие, вызывается тот или другой метод экземпляра класса Action
6. Документ, который является результатом выполнения действия пользователя, возвращается ему через сеть, или пользователь перенаправляется на другую страницу

Тем временем...

Вы можете задать для веб-сервера Apache 1.3.x свой обработчик того или иного типа файлов используя директивы Action и AddHandler. Вот пример:
AddHandler my-file-type .xyz
Action my-file-type /xyzhandler.php

Дополнительно смотрите информацию по адресу: http://httpd.apache.org/docs/mod/mod_actions.html#action

Расширяемость

Используя Freeform, каждый разработчик вправе сам решать, какие шаблонную систему, менеджер сессий, систему регистрации пользователей ему использовать. Используя интерфейсы, введенные в пятую версию PHP, ядро может взаимодействовать с любыми такими системами - достаточно, чтобы они реализовывали тот или иной интерфейс. Например, разработчик может создать свой собственный обработчик сессий, который реализует интерфейс SessionHandler и изменить одну строчку в конфигурационном файле.

Также можно создать свой обработчик шаблонов, который реализует интерфейс Document - и ядро сможет получить тело выходного документа и возвратить его пользователю. Кроме того, разработчик сам определяет, какую систему хранения данных он использует - Freeform вообще не определяет никаких требований к модели данных.

Безопасность

Вопросы безопасности - ключевой момент разработки веб-приложений. В Freeform каждый запрос подвергается проверке на предмет того, может ли он быть выполнен в текущем окружении. Всякий раз, когда пользователь переходит по ссылке или отправляет данные формы, строится экземпляр класса Action, в котором определен метод getAccessController(), который должен возвратить экземпляр класса, реализующего интерфейс AccessController.

Этот интерфейс декларирует единственные метод isAccessible (), который должен сообщить, может ли данный Action быть выполнен в текущем окружении.

С помощью этого интерфейса создаются классы, ответственные за применение политики безопасности. Таким образом, создавая приложение, разработчик строит иерархию классов, которые могут быть повторно использованы в других приложениях. Кроме того, отдельные пакеты, которые реализуют некоторую стандартную функциональность (например, форум), могут быть сконфигурированы таким образом, чтобы использовать эти классы при проверке доступа к своим функциям. Таким образом, достигается очень простая интеграция программного обеспечения от третьих лиц.

Обработка форм

Обработка данных форм - это, пожалуй, самая часто использованная функция приложения. Freeform предлагает стандартный интерфейс программирования форм. По сути, форма - это своего рода фильтр, который анализирует данные запроса. Форма умеет отличать, была она отправлена или нет, которые ее поля заполнены правильно, а какие - неправильно. Формы состоят из полей - отдельных классов, которые собственно и проверяют правильность заполнения каждого из них. Кроме того, стандартная система обработки шаблонов, которая доступна в каждой инсталляции, имеет все средства для автоматизации отображения полей ввода форм.

Архитектурной особенностью программирования форм является то, что этот класс не сопряжен никаким образом с классом Action, что дает возможность использовать несколько форм на одной странице или использовать одну и ту же форму на нескольких страницах. Создав подкласс этого класса, можно в нем определить, какие поля форма имеет, и какой Action будет ее обрабатывать.

Система обработки шаблонов

В каждой инсталляции Freeform имеется пакет html, который является примером реализации системы подготовки выходных документов. Основной класс пакета, HTMLDocument реализует интерфейс Document и позволяет создавать сложные HTML документы на основе XML шаблонов. Отличительной особенностью этого пакета является возможность создавать собственные тэги - классы, которые отвечают за обработку одноименных тэгов шаблона.

Такие классы используются для циклического или условного отображения частей шаблона, создания меню или деревьев в результирующем документе, вставки гиперссылок и отображения полей форм. Так, создание ссылок на другие страницы теперь происходит без изучения всех адресов и параметров дизайнерами - специальные тэги сами определят, какие URL отображать на основе данных их приложения.

Также нет необходимости прописывать путь к обработчику формы или метод ее отправки, не говоря о списках выбора и остальных полях - все данные заполняются автоматически обработчиком шаблона. В последней версии имеется более десяти специализированных тэгов, нацеленных на упрощение и ускорение создания выходных документов.

Стандартные классы

В Freeform также имеется отдельный пакет - набор вспомогательных классов и интерфейсов, которые могут использоваться в любых ситуациях. Здесь, например, содержатся интерфейсы Iterable и PaginatedIterable, которые используются для создания итераторов с постраничным доступом, абстрактный класс Validator, который используется для создания классов для проверки правильности значения, а также классы IterablePearDbResult и IterableADORecordSet, которые позволяют организовать вывод данных, полученных при помощи библиотек Pear::DB и ADOdb, средствами пакета html.

Вспомогательные приложения

На данный момент создано одно вспомогательное приложение - Freddy - средство просмотра документирующих комментариев в файлах классов и пакетной документации. Это отдельный пакет, который в режиме реального времени генерирует документацию для всех установленных пакетов и классов. Он также позволяет использовать отдельные файлы документации, распространяемые вместе с ресурсами пакетов, а также вставлять ссылки на пакеты, классы и методы и использовать рисунки в документации.

Экспериментальные пакеты

Предыдущие дистрибутивы Freeform имеют еще два экспериментальных пакета. Один из них является попыткой создания собственной библиотеки абстракции баз данных, а другой - мощное средство организации объектных баз данных.

Особенный интерес представляет именно это средство, поскольку, в отличии от других похожих средств, оно не использует какие-либо дополнительные механизмы как генератор SQL кода или классов по схеме, "улучшителей" классов и т.д.

Для построения собственной объектной базы не надо знать ничего другого, кроме собственно языка PHP. Инструменты данного пакета сами создадут реляционную модель базы по определению каждого класса и, в случае изменения этих определений, перестроят базу.

Экземпляры классов сами отвечают за сохранение всех изменений своих полей, так что разработчик работает с этими экземплярами как с любыми другими классами.

Тем временем...

Если PHP установлен на платформе windows, то можт возникнуть необходимость работать с почтовым клиентом Outlook.

Как это делается, можно посмотреть здесь: http://www.severnsolutions.co.uk/twblog/archive/2004/11/19/phpoutlookdcom

Кроме того, пакет позволяет выбирать экземпляры нескольких классов одной иерархии в одном запросе, позволяет создавать персистентные массивы и коллекции объектов, принадлежащих одному владельцу.

С помощью Freeform Framework было создано приложение ProjectsHosted, онлайн-каталог проектов с собственной системой отслеживания ошибок, менеджером релизов, новостей, подписок, форумами и т.д. Приложение работает по адресу http://dev6.php5.nedlinux.com, оттуда можно получить последнюю версию Freeform и других пакетов.

В этой вводной статье мы представили Freeform Framework в общих чертах. Далее мы детально опишем технические аспекты реализации некоторых компонентов с целью лучшего ознакомления со средствами PHP5 и примерами их использования в разработке веб-приложений.

Последнюю версию Freeform Framework можно скачать с сайта http://dev6.php5.nedlinux.com/

Freeform Framework – первые шаги

Автор: Денис Попель

Для демонстрации возможностей MVC архитектуры, средств обработки форм и стандартной шаблонной системы, предлагаемых Freeform Framework, рассмотрим простое предложение, которое отображает форму ввода имени пользователя и, в случае его правильного заполнения, выводит приветствие.

Это простейшее веб-приложение будет состоять из одного класса и использовать один шаблон.

Начнем с класса, который будет отвечать за отображение страницы – мы назовем его HWHome. Все классы, которые обрабатывают действия пользователя, расширяют абстрактный класс Action из пакета ‘freeform’. Поскольку мы имеем дело с формой, определяем в классе переменную, которая будет содержать ссылку на объект формы:

```
class HWHome extends Action {
  private $form = null;
```

При построении экземпляра нашего класса, который будет отображать форму и приветствие, фронт-контроллер вызовет метод onInit(), в котором мы создадим экземпляр класса формы. Для создания формы нужно определить объект Request, который содержит данные, полученные от пользователя. Каждый объект Action получает ссылку на такой объект при своем создании. Также форме нужно сообщить, кто обрабатывает отправку формы. В данном случае это объект класса HWHome (т.е. мы сами обрабатываем данные формы). Последний параметр указывает метод отправки - здесь используется HTTP GET:

```
  function onInit() {
    $this->form = new Form($this->getRequest(), new Location($this),
Request::METHOD_GET);
```

Здесь мы также добавляем в форму поле ввода. В данном случае форма содержит единственное поле для ввода строки текста с именем 'name'. Это поле - объект класса TextInputField, который используется для представления полей ввода строки текста. При построении поля ввода мы задаем значение по умолчанию, которое отобразится при первом обращении к странице с формой, а также фильтр - объект класса RegexValidator (реализующего интерфейс Validator), который проверит, содержит ли введенное пользователем имя хотя бы один символ:

```
$this->form->addField(
      'name',
      new TextInputField('(Ваше имя)', new RegexValidator('.+')));
  }
```

Теперь определяем метод process(), который отвечает за собственно обработку данных пользователя и подготовку выходного документа. Здесь мы проверим, отправил ли пользователь свое имя, и, если это так, то в результате он увидит персональное приветствие.

Для этого мы сперва получаем объект Responce, связанный с вызванным Action. Каждый Action получает ссылку на такой объект при создании. Через объект Responce мы возвращаем выходной документ (экземпляр класса, который реализует интерфейс Document), тело которого будет возвращено пользователю:

```
function process() {
  $r = $this->getResponce();
```

Дальше создаем выходной документ. В данном случае - это объект класса HTMLDocument, который содержится в пакете 'html' и позволяет создавать сложные документы на основе XML шаблонов. Для его построения мы должны указать объект Responce, который будет возвращать его тело:

```
$d = new HTMLDocument($r);
```

Теперь документу нужно указать путь к файлу шаблона. Поскольку мы будем держать файл шаблона как файл ресурса, то используем встроенные средства поиска этого файла. С помощью объектов Package мы можем получить доступ к ресурсам пакета:

```
$p = $this->getPackage();
  $d->setTemplate($p->getResourcePath('HWHome.html'));
```

Дальше мы проверяем, была ли форма отправлена, и, если да, то верно была ли она заполнена. Если все проверки успешны, получаем значение поля ввода 'name' из нашей формы; если же нет – то форма или не была отправлена, или была заполнена с ошибками. В нашем примере шаблон сам определит, имела ли место ошибка, и автоматически отобразит напоминание (код шаблона см. ниже):

```
if($this->form->isValidSubmission()) {
     $n = $this->form->getField('name')->getValue();
  } else {
     $n = 'Гость';
  }
```

Теперь мы готовы отправить выходной документ пользователю. Для этого устанавливаем две шаблонные переменные, первая из которых будет содержать имя, введенное пользователем, а вторая – ссылку на объект формы, чтобы документ смог автоматически отобразить поля и их значения, возможные предупреждения об ошибках и т.д. В последней строчке мы передаем объекту Responce выходной документ. После этой строчки Freeform отправит тело документа пользователю

```
   $d->setVariable('name', $n);
    $d->setVariable('form', $this->form);
    $r->setDocument($d);
  }
}
```

Теперь рассмотрим шаблон, который будет использоваться для создания выходного документа классом HWHome, только что созданным нами. Каждый шаблон для документов класса HTMLDocument - это XML-файл, очень похожий на обычный HTML документ, но, возможно, содержащий специальные теги:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
```

Эта строчка выводит приветствие. Обратите внимание, как мы обращаемся к переменной шаблона:

```
 Привет, {%name}!<hr/>
```

Следующий тег отображает форму. Обратите внимание что для отображения формы средствами Freeform Framework мы указываем только имя шаблонной переменной, содержащей форму ввода. Остальные атрибуты (метод отправки, адрес обработчика и т.д.) будут добавлены автоматически:

```
<HTMLShowForm key="form">
      Ваше имя:
```

Этот тег отображает поле ввода. Тип поля проставляется, исходя из класса объекта-поля ввода, значение выводится также автоматически. Заметьте, что в этом теге можно использовать атрибуты для стиля или скриптов:

```
<HTMLInput name="name" size="25"/>

      <input type="submit" value=" Вперед! "/><br/>
```

Следующий тег отобразит свое содержимое только в том случае, если поле ввода "name" было заполнено с ошибками. Чтобы увидеть это предупреждение, отправьте форму с пустым полем:

```
 <HTMLIfInputFieldError key="name">
       <b>Пожалуйста, введите ваше имя!</b>
     </HTMLIfInputFieldError>
```

Завершаем шаблон, следя за тем, чтобы все открывающие теги были правильно закрыты:

```
</HTMLShowForm>
  </body>
</html>
```

Эти файлы Вы можете получить в приложении к номеру (там они сохранены в кодировке UTF-8). Для испытания этого примера Вам понадобится Freeform Framework, последнюю версию которого можно скачать с сайта http://dev6.php5.nedlinux.com/, а также PHP версии 5.0.3 (предыдущие версии имеют ошибку в обработчике XML документов). В дистрибутиве Freeform Framework содержатся инструкции по его инсталляции, а в комментариях файлов HWHome.php5 и HWHome.html – подсказки, куда их записать. Если вы проделали все правильно, пример можно запустить, набрав в броузере что-то типа http://localhost/index.php5?action=HWHome.

Идеи

Языки описания пользовательских интерфейсов

Данная статья обозревает перспективные концепции декларирования пользовательских интерфейсов для веб-приложений. В ней рассматриваются технологии UIML, XUL, XAML, MXML и Web Applications

Автор: Дмитрий Шейко
Ведущий программист Red Graphic Systems
www.cmsdevelopment.com

Пользовательские интерфейсы и веб

Что такое пользовательский интерфейс? По логике вещей, это то с чем мы сталкиваемся каждый день в повседневной жизни: столовые приборы, дверные ручки, пульты управления телевизором и т.д. Сложилось так, что в мире информационных технологий пользовательский интерфейс, прежде всего, ассоциируется с GUI операционной системы. Это неудивительно, ведь ныне элементы управления компьютерными программами нам столь же привычны, как и тумблеры бытовых электроприборов. Пользовательские интерфейсы - неотъемлемая часть любого веб-приложения. Любой сайт, к которому мы можем обратиться посредством Интернет, является пользовательским интерфейсом для доступа к информации. Однако в настоящее время задача программирование пользовательских интерфейсов для веб несколько иное, нежели в случае настольных программ.

Современная операционная система базируется на устоявшемся стандарте пользовательского интерфейса для всех, взаимодействующих с нею программ. Соответственно имеется определенная модель, которой подчиняются любые интерфейсные решения в программах для данной операционной системы. А раз так, то создание пользовательского интерфейса сводиться к использованию функций стандартных библиотек. Например, это могут быть функции Win32 API или объекты MFC для программ MS Windows.

Подобный подход имеет одно замечательное свойство. Если пользователь научился пользоваться хотя бы одной программой, он быстро освоится с любой другой для той же операционной системы. Но такое положение вещей не может быть отнесено к веб-приложениям. Каждый новый сайт – это новый информационный и графический дизайн, а также новый пользовательский интерфейс. В данном случае едва ли применимы программные библиотеки для какого-то одного определенного стандарта пользовательского интерфейса.

И до сих пор чаще всего при разработке сайтов используется «ручное» программирование, что подразумевает внесистемное задание оформления и программных реакций, для каждого элемента пользовательского интерфейса и для каждого его состояния. В случае крупного полнофункционального решения подобный подход означает «начало конца». Но давайте обратимся к существующим и перспективным стандартам в области описания пользовательских интерфейсов.

Ссылки по теме:

Win32 API (http://msdn.microsoft.com/library/en-us/dnanchor/html/anch_win32com.asp)
MFC (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcmfc98/html/mfchm.asp)

Язык пользовательских интерфейсов UIML

В 90х годах HTML обрел огромную популярность и, прежде всего за счет свой простоты. Для того, что бы создать небольшой сайт не требовалось особых навыков в программировании и специальных инструментальных средств. Любой желающий мог это сделать и почти каждый попробовал. Однако прародитель HTML язык SGML подразумевал структуризацию документов, а это значительно более глубокая модель нежели простое оформление внешнего вида данных. Изначальная идея упорядоченной структуры распределенных данных вернулась вместе с XML и породила эпоху мета описания абстрактных составляющих веб-ресурсов.

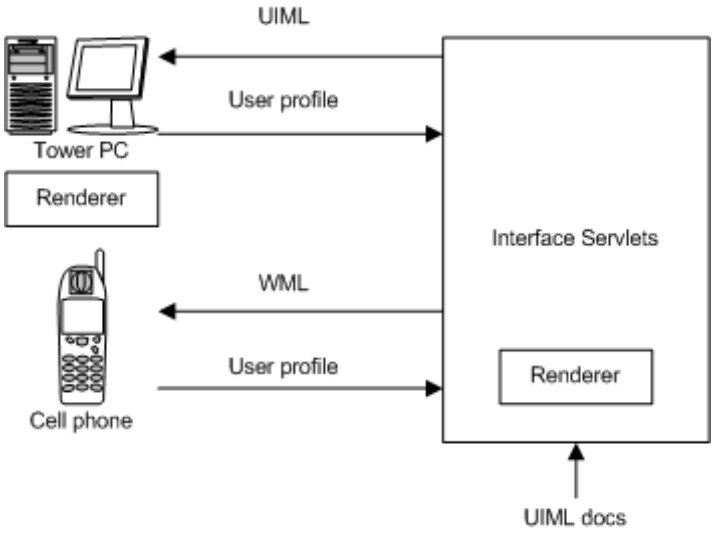
На этом фоне четко вырисовывалась задача вынесения разметки UI из программного кода приложений. Кроме того, появилась технология каскадных таблиц стилей (CSS), что открыло путь к созданию настраиваемого под конкретное устройство оформления интерфейсов. Эти обстоятельства явились предпосылками созданию языка UIML (User Interface Markup Language). Первая спецификация UIML была представлена компанией Harmonia в январе 1998 года. Ныне доступна спецификация 3.0 на сайте проекта www.uiml.org.

Что представляет собой UIML? В общих чертах это концепция, в которой путь данных от приложения до физического устройства отображения данных пролегает через абстрактные области логики, интерфейса и представления. Область интерфейса, включает описание структуры, стилей, содержания и поведения элементов. Задача языка UIML эффективно реализовать область интерфейса.

Если взглянуть глубже, то выясниться, что UIML определяет следующее:

- составные элементы пользовательского интерфейса;
- каким образом будут представлены элементы пользовательского интерфейса (визуально/вербально/тактильно);
- какого рода содержание будет использовано в пользовательском интерфейсе (текст, изображения, звуки и т.д.);
- какова будет реакция элементов пользовательского интерфейса на действия пользователя;
- каким образом будет производиться контроль событий пользовательского интерфейса (Java Swing classes или теги HTML);
- с каким внешним API будет взаимодействовать пользовательский интерфейс.

Рис. 1. Модель кросс- платформенных пользовательских интерфейсов



Но довольно теории, давайте взглянем на то, как это работает. Ниже приведен пример UIML документа. Мы определяем область приложения с помощью элемента APP и далее назначаем группы для его элементов. Внутри групп мы определяем элементы пользовательского интерфейса. Мы также можем определить свойства элементов посредством контейнера DEFINE.

```
<UIML>
 <HEAD>
  <TITLE>CMS loosely remind of BCWB</TITLE>
  <AUTHOR>Dmitry Sheiko, Red Graphic Systems</AUTHOR>
  <DATE>17/02/05</DATE>
  <VERSION>0.1</VERSION>
 </HEAD>
 <APP NAME="CMS" CLASS="App">
  <GROUP NAME= "MainFrame" CLASS="frame">

  <GROUP NAME="Common" CLASS="menu">
   <ELEM NAME="DocEdit" CLASS="menuitem"/>
   <ELEM NAME="DocView" CLASS="menuitem"/>
   <ELEM NAME="Quit" CLASS="menuitem"/>
  </GROUP>
  <GROUP NAME="Structure" CLASS="menuortoolbar">
   <ELEM NAME="AddItem" CLASS="menuitem"/>
   <ELEM NAME="ChangeItem" CLASS="menuitem"/>
   <ELEM NAME="DeleteItem" CLASS="menuitem"/>
  </GROUP>

  <GROUP NAME="Templates" CLASS="menuortoolbar">
   <ELEM NAME="AddTemplate" CLASS="menuitem"/>
   <ELEM NAME="ChangeTemplate" CLASS="menuitem"/>
   <ELEM NAME="DeleteTemplate" CLASS="menuitem"/>
  </GROUP>

Листинг 1. Документ UIML (продолжение на следующей странице)
```

```
<ELEM NAME="Desktop" CLASS=" DesktopArea"/>

  </GROUP>
 </APP>

 <DEFINE NAME="Quit">
  <PROPERTIES>
   <CLASS VALUE="menuitem"/>
    <ACTION
     VALUE="MainFrame.VISIBLE=false"
     TRIGGER= "select"
    />
   </PROPERTIES>
 </DEFINE>
</UIML>
```

Листинг 1 (продолжение). Документ UIML

Надо также заметить, благодаря активному использованию CSS, документ не кажется перегруженным излишней информацией. Стилевой файл для UIML может содержать как CSS, так и Java AWT (Abstract Windowing Toolkit).

```
/*
<AUTHOR>Dmitry Sheiko, Red Graphic Systems</AUTHOR>
<DATE>17/02/05</DATE>
<VERSION>0.1</VERSION>
*/

APP.App{
 +TOOLKIT: jfc;
 +RENDERING-PREFIX: java.awt;
}
GROUP.frame {
 RENDERING: "java.awt.Frame";
 LAYOUT: BorderLayout;
 SIZE: "400,400";
 FONT-FACE: Serif;
 FONT-SIZE: 10;
 FONT-STYLE:Plain;
CONTENT: "Error:No Content";
}
GROUP.menu {
 RENDERING: "java.awt.Menu";
}
ELEM.menuitem {
 RENDERING: "java.awt.MenuItem";
}
GROUP.menuortoolbar {
 RENDERING: "java.awt.Menu";
}
ELEM. DesktopArea {
 RENDERING: "java.awt.Panel";
 ALIGNMENT: Center;
}
```

Листинг 2. Стилевой файл для UIML

Теперь структура и внешний вид интерфейса описаны. Опытный разработчик к этому моменту наверняка задумается о формате содержания элементов пользовательского интерфейса. В UIML используется база данных контента (Content Database), где и хранится содержание элементов интерфейса.

```
# Record format: Key, Name (from .uiml file), Attribute, Value
#
EnglishQuittingDialog   CONTENT     Are you sure?!
EnglishQuittingFinishedMsg    CONTENT     Good-bye!
EnglishDocEdit        CONTENT  Edit document
EnglishOKButton         CONTENT     OK
```

Листинг 3. База данных контента

UIML на мой взгляд представляет собой наиболее удачное решение именно по части описания логики пользовательских интерфейсов из ныне существующих. Что вполне закономерно, учитывая тот факт, что его инициатор компания Harmonia специализируется на пользовательских интерфейсах. Однако, в отличие от прочих языков, рассматриваемых в данном обзоре, UIML не поддерживается какими-либо браузерами. Для выполнения UIML-трансформации следует воспользоваться одним из сторонних UIML-процессоров на стороне сервера. Впрочем, по адресу (http://www.uiml.org/tools/index.htm) представлен внушительный список open source процессоров UIML.

Ссылки по теме:

UIML (http://www.uiml.org)
Java AWT (http://www.oreilly.com/catalog/javawt/book/)

«Нет больше данных, есть только XUL»

В настоящее время весьма популярен язык описания пользовательских интерфейсов XUL (XML User-interface Language). XUL является частью среды разработки кросс- платформенных интерфейсов, известной как XPFE. Это полнофункциональный язык разметки, на объекты приложений, такие как окна, метки и кнопки. Язык соответствует стандарту W3C XML 1.0. Приложения, написанные на XUL, также могут использовать HTML, CSS, DOM, Java-script. И главное, XUL пытается разделить представление данных и логику приложений. Делается это посредством следующих абстрактных слоев:

- Содержание (content): объявление окон и элементов пользовательского интерфейса ассоциированных с ними
- Оформление (skin): включение CSS и изображений, определение вида приложения
- Локализация (locale): текст, отображаемый в пределах приложения, распределен по специальным локальным файлам, что обеспечивает переносимость языка.

Теоретически XUL обеспечивает кросс- платформенные интерфейсы (по крайней мере на данный момент он доступен в операционных системах Windows, Unix, Mac). Впрочем, первое яркое впечатление от технологии сразу же омрачает ее жесткая привязка к ядру Mozilla (Gecko).

Кстати говоря, с названием технологии связан один курьезный случай. Так сложилось, что аббревиатура XUL происходит от имени персонажа Зуул из фильма «Охотники за приведениями». Ключевой оказалась фраза из фильма «Нет больше Даны, есть только Зуул» трансформировавшаяся в слоган «Нет больше данных, есть только XUL». Может быть, именно по этому сообщество XUL так усердно следит за корректностью произношения названия языка.

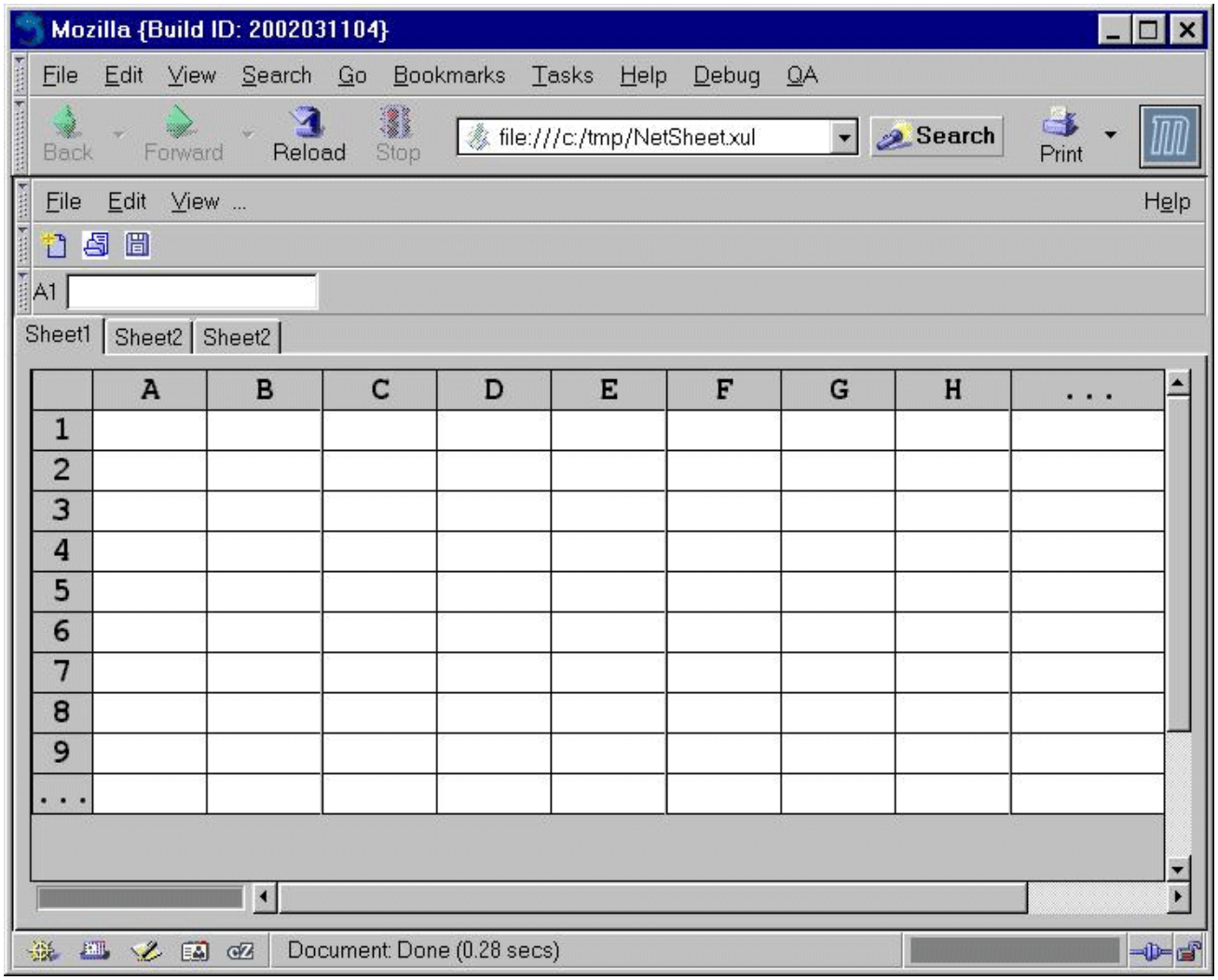
Пример простого интерфейса Drag & Drop на языке XUL

Источник: http://www.xulplanet.com/tutorials/xultu/dragex.html

```
<window title="Widget Dragger" id="test-window"
  orient="horizontal"
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
 <script src="chrome://global/content/nsDragAndDrop.js"/>
 <script src="chrome://global/content/nsTransferable.js"/>
 <script src="dragboard.js"/>

 <stack id="board"
   style="width:300px; height: 300px; max-width: 300px; max-height: 300px"
   ondragover="nsDragAndDrop.dragOver(event,boardObserver)"
   ondragdrop="nsDragAndDrop.drop(event,boardObserver)">
 </stack>

 <vbox>

  <button label="Button"
          elem="button" ondraggesture="nsDragAndDrop.startDrag
(event,listObserver)"/>
  <button label="Check Box"
          elem="checkbox" ondraggesture="nsDragAndDrop.startDrag
(event,listObserver)"/>
  <button label="Text Box"
          elem="textbox" ondraggesture="nsDragAndDrop.startDrag
(event,listObserver)"/>
 </vbox>

</window>
```

Пример интерфейса таблиц посредством XUL

http://www.devx.com/webdev/Article/9312/0/page/3



Между XUL и Flex

В то время как XUL располагает такими преимуществами как взаимодействие с различными популярными объектно-ориентированными языками, полноценная поддержка XPath и CSS, Flex небезосновательно предвещает эпоху насыщенных веб-приложений (RIA). На этом фоне возникают гибридные технологии, такие как ZULU. Это еще один язык разметки кросс-платформенных пользовательских интерфейсов, совмещающий стандарт XUL и технологический возможности Flash MXC

Сылки по теме:

http://zulu.netspedition.com/zulu/main/overview.shtml

http://www.mozilla.org/docs/web-developer/
http://www.mozilla.org/xpfe/
http://www.xulplanet.com
http://www.sitepoint.com/print/1140
http://xpoint.ru/forums/programming/XUL/faq.xhtml

«XAML – язык авалонский»

Было бы странно, если бы столь перспективную нишу рынка проигнорировала компания Microsoft. Ныне в активном развитии XAML (eXtemsible Application Markup Language) – язык интерфейсов платформы Windows Longhorn.

Модель приложений Longhorn включает объект Application. Его набор свойств, методов и событий позволяет объединить веб-документы в связанное приложение. Объект Application контролирует выполнение программы и генерирует события для пользовательского кода. Документы приложения пишутся на XAML. Впрочем, с помощью XAML описывается, прежде всего, пользовательский интерфейс. Логика приложения по-прежнему управляется процедурным кодом (C#, VB и т.д.). XAML может использоваться как для браузер-базированных приложений, так и для локальных настольных приложений.

XAML включает основные четыре категории элементов: панели, элементы управления, элементы, связанные с документом и графические фигуры. Заявлено 7 классов панелей, которые задают принципы отображения вложенных в них элементов. Для задания положения элементов относительно границ родительской панели используются атрибуты на манер свойств в объектно-ориентированных языках. Подобный синтаксис не очень вяжется с рекомендациями CSS, но будет привычен программистам настольных приложений.

Пример задания атрибутов элементам в XAML

```
<Border Background="green"
       Canvas.Top="100px" Canvas.Left="100px"
       Height="100px" Width="100px" />
```

Приложения, объявленные в XAML, могут включать множество страниц. Элемент управления PageViewer позволяет разбивать содержание на страницы и обеспечивает навигацию по ним. Элемент ContextMenu помогает в создании навигационных меню приложения. Код процедурного языка может быть размещен непосредственно в файле XAML или же назначен при сборке проекта.

В настоящее время стабильной версии Longhorn нет, но Microsoft выпустила в ноябре 2004 Avalon CTP, позволяющий использовать XAML на платформах Windows XP и Windows Server 2003. Последнюю версию этого пакета можно найти по этому адресу:
http://www.microsoft.com/downloads/details.aspx?familyid=C8F904E1-B4CA-402B-ACCF-AAA2BD60DA74&displaylang=en

Пример простого интерфейса на XAML с 3-мя кнопками, визуализация одной из которых зависит от наличия ее содержания.

Источник:
http://www.joemarini.com/tutorials/tutorialpages/xamlpropertytriggers.php

```
<DockPanel xmlns="http://schemas.microsoft.com/winfx/avalon/2005"
Background="#ffffffff"
xmlns:x="http://schemas.microsoft.com/winfx/xaml/2005">
  <DockPanel.Resources>
    <Style>
      <Button Margin="5"/>
      <Style.VisualTriggers>
        <PropertyTrigger Property="Button.HasContent" Value="False" >
          <Set PropertyPath="Visibility" Value="Collapsed" />
          <Set PropertyPath="Margin" Value="0" />
        </PropertyTrigger>
      </Style.VisualTriggers>
    </Style>
  </DockPanel.Resources>
  <Button Height="30" Width="100">Button One</Button>
  <Button Height="30" Width="100">Button Two</Button>
  <Button Height="30" Width="100">Button Three</Button>
  <Button Height="30" Width="100" IsEnabled="False">Button Four</Button>
</DockPanel>
```

Пример пользовательского интерфейса XAML, с динамически изменяемым оформлением

Источник: http://www.optim.ru/cs/2004/3/Avalon/Avalon.asp

Ссылка по теме:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnintlong/html/longhornch03.asp

Язык насыщенных веб-приложений MXML

Macromedia традиционно выделяется на рынке поставщиков веб-технологий нетипичным подходом. Так скажем, ныне повсеместно используемый Flash столь разительно отличается от прочих технологий доставки информации, что его было бы как-то не ловко даже рассматривать в параллели с теми же языками разметки. Однако в эпоху XML и Macromedia не осталась в стороне от моды на декларативные языки. Ответ компании обозначился технологией Flex, содержащей XML-базированный язык MXML (Macromedia Flex Markup Language).

Как и рабочая группа Mozilla, и компания Microsoft, разработчики Flex стремились создать язык, эффективно сочетающий две популярных парадигмы: язык разметки и объектно-ориентированный программный язык. MXML позволяет наглядно описать структуру пользовательского интерфейса, по которой он будет воссоздан клиентским приложением. ActionScript выполняет задачи контроллера (программная реакция на события в среде) и обеспечивает уровень модели приложения.

Что стоило бы выделить среди преимуществ Flex

Flex помимо стандартных элементов форм ввода данных содержит столь актуальные компоненты пользовательского интерфейса как Tree component (структуризация данных), DataGrid component (управление большими массивами данных), различные навигационные компоненты (TabNavigator, ViewStack, Accordion, и прочее).

Как помнится, одно из основоположных свойств XML - возможно назначения собственных тегов. Flex эффективно наследует эту идею. Мы можем создать приложение, поместить его в отдельный файл с именем MyInnerApp.mxml после чего в приложениях Flex станет доступным тег <MyInnerApp />, ссылающийся на исходный код.

Flex располагает средствами для интеграции приложений. Мы можем воспользоваться протоколом SOAP и передать их Flex-приложения инструкции удаленному сервису, а затем принять от него данные. Это позволяет использовать при разработке приложений FLEX сервис ориентированную архитектуру (SOA).

Специфика интерфейсов от Macromedia в их интерактивности, мультимедийной насыщенности. Очевидно, что имеется богатая библиотека спецэффектов (библиотека событий), доступных элементам приложений Flex. Надо отметить, что и документация к технологии Flex выполнена в лучших традициях Macromedia. Особенно впечатляет интерактивный тур в технологию http://www.macromedia.com/software/flex/productinfo/brz_overview/.

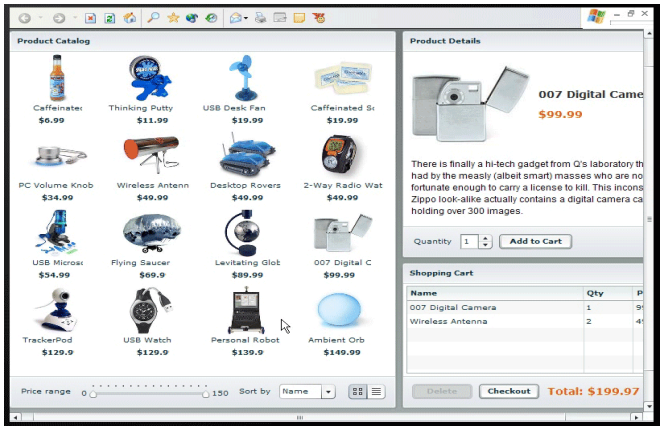
На клиентской стороне приложения Flex устанавливаются на браузерах, располагающих расширением Flash Player 7. Данное обстоятельство обеспечивает приложениям Flex самую широкую поддержку на клиентских устройствах. С другой стороны необходимая серверная поддержка реализуется компонентом Flex Presentation Server, устанавливаемым на сервер приложений J2EE (Macromedia JRun, IBM Websphere, BEA WebLogic, Apache Tomcat и т.д.). Это тот самый механизм, который строит новое поколение насыщенных приложений (RIAs - Rich Internet Applications). Начальная цена Flex Presentation Server составляет 12 тыс. долл.

Пример одного из подходов к разделению слоев структуры интерфейса и программного контроллера в MXML приложении.

Источник: http://www.rewindlife.com/archives/000121.cfm

```
<?xml version="1.0" ?>
<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml"
initialize="appController.initialize();">
    <TempConverterController view="{this}" id="appController" />
    <mx:Label text="Temperature in Farenheit:" />
    <mx:TextInput id="farenheit" width="120" />
    <mx:Button id="myButton" label="Convert" />
    <mx:Label text="Temperature in Celsius:" />
    <mx:Label id="celsius" width="120" fontSize="18" />
</mx:Application>
...
class TempConverterController {

    [Inspectable]
    public var view;

    function initialize() {
        view.myButton.addEventListener("click", this);
     }

    function click(event) {
        view.celsius.text=(view.farenheit.text-32)/1.8;
     }

}
```

Пример электронного магазина. Источник: http://www.macromedia.com/software/flex/productinfo/brz_overview/



Оперная партия веб-приложений

Если бы разработчики браузеров в полной мере придерживались одних и тех же стандартов, это очень облегчило жизнь веб-разработчикам. Написав приложения под один из браузеров можно было не беспокоиться о том, что оно иначе поведет себя в другом. Казалось бы о поддержке стандартов W3C заявлено всеми крупными игроками на этом рынке, но не для кого не секрет, что один и тот же HTML-код может по-разному транслироваться браузерами IE, Netscape и Opera, не всегда одинаково воспринимается ими CSS. А про Java-script и DOM, полагаю, не стоит даже упоминать. Разработчики браузеров в стремлении привлечь большее внимание к своим продуктам, технологически опережают независимые организации по стандартизации. В результате на рынке множество сопоставимых, но разных технологий, что кроме всего прочего создает жесткую конкуренцию. Когда группа Mozilla шествует по миру под флагами XUL, Microsoft предвещает эру Longhorn/XAML, Opera Sofware просто вынуждена сделать какой-либо ответный шаг. По крайней мере, мне именно в таком свете видится работа компании над спецификацией Web Applications 1.0 (http://www.whatwg.org/specs/web-apps/current-work/). Данная спецификация не несет в себе каких-либо революционных новаций, но обрисовывает многие актуальные (по крайней мере для браузера Opera) задачи:

- управление pop-up меню и контекстными меню;
- обслуживание событий сервера в приложении без перегрузки страницы. Удаленный вызов процедур на стороне сервера и манипуляции фрагментами XML-документов;
- обслуживание устройство независимых событий DOM ;
- формы пользовательского интерфейса для отображения динамических деревьев и списков;
- предустановленный редактор HTML;
- API для Drag&Drop;
- API для манипуляций с выделениями в содержании;
- API для буфера обмена;

Данная спецификация еще совсем сырая. Ее редакция от 1 марта 2005 буквально пестрит красными пометками «возможно, это будет так, но может быть и иначе…». Завершение ее разработки вероятно обогатит браузер Opera новыми возможностями, но вряд ли это будет заметно на фоне более значимых инноваций конкурентов.

Пример: Задание модели событий для формы календарь

Источник: http://www.whatwg.org/specs/web-apps/current-work/

```
<calendar>
 <div class="vcalendar">
  <span class="prodid">-//hCalendar//EN</span>
  <span class="version">2.0</span>
  <p class="vevent">
   <a href="http://www.web2con.com/">
    <span class="dtstart">20041005</span>-
    <span class="dtend">20041007</span>
    <span class="summary">Web 2.0 Conference</span>
   </a>
  </p>
 </div>
</calendar>
```

Итоги

Общая картина вырисовывается следующим образом. Microsoft сконцентрировала усилия на то, что бы стереть грань между браузером и настольными приложениями. Для этого создается платформа Longhorn, включающая язык разметки интерфейсов XAML. В основе Longhorn предусмотрен системный слой CLR, гарантирующий переносимость приложений для различных устройств.

В тоже время рабочая группа Mozilla не покушается на стереотипы веб-разработчиков, но, тем не менее, предоставила инструмент XUL, позволяющий писать веб-приложения, по функциональности близкие к настольным. Очевидное преимущество сообщества Mozilla на данный момент в наличии целого ряда уже реализованных приложений, представленных на сайте MozDev.org. В стороне от схватки сообщества open source и софтверной империи Microsoft компания Macromedia успешно продвигает технологию Flex.

Решения от Macromedia традиционно отличаются впечатляющим мультимедиа и Flex не исключение из этого правила. Популярность Flash в проектах презентационного характера трудно переоценить, а технологический дуэт MXML и ActionScript 2 открывает Macromedia нишу полнофункциональных бизнес-порталов. Впрочем, эффект от качественных Flex-приложений по-прежнему требует дополнительных трудозатрат, так что, смею предполагать, эта технология не станет основным инструментом большинства веб-разработчиков.

Но спустимся на грешную землю. Типовое техническое задание на веб-проект включает требования поддержки наиболее популярных браузеров. Т.е. автоматически отпадают технологии XAML и XUL. Можно было выбрать Flex, ведь плагин Flash Player, легко интегрируется в различные браузеры. Однако, стоимость Flex Presentation Server ставит под сомнение рентабельность для большинства компаний, занятых в веб-разработке. Кроме того, каждое из заявленных решений влечет язык разметки пользовательского интерфейса.

Однако, в каждом из случаев это скорее язык браузера/платформы, нежели концепция описания интерфейса. Именно по этому я включил в обзор язык UIML, не привязанный к какой-либо платформе, но четко разделяющий абстрактные приложения. Таким образом, UIML задает структуру интерфейса, его оформление, его содержание и его поведение. Это модель описания интерфейсов, которая просто-напросто наиболее интуитивно понятна. А так UIML аполитичен на мировом ИТ-рынке, он с одинаковой легкостью может использоваться для трансляции кода в любые браузеры, на любые устройства. UIML не разрушает привычный технологический процесс веб-разработки, но дополняет его.

Разработчик может по-прежнему опираться на выбранные ранее технологии HTML, XHTML, CSS, XSL, WML и т.д. А что касается доступности, то нам остается лишь выбрать на свой вкус один из open source процессоров UIML и установить его на свой сервер.

Сравнительная таблица языков разметки пользовательского интерфейса

Язык разметки	**MXML**	**XAML**	**XUL**	**UIML**
Готовность	В настоящее время	Официально ожидается в 2006 году	В настоящее время	В настоящее время
Доступность приложений				
Платформы	Выполняется на любой платформе через Flash Player 7	Выполняется только на Longhorn	Выполняется на платформах Windows/Unix/Mac через Gecko-базированный бразуер	Преобразованный документ доступен на любых платформах
Устройства	-	Независимость в рамках CLR	-	Устройство-независимый язык
Серверная часть	Flex Presentation Server	Longhorn	Нет специальных требований	Любой из UIML-процессоров на стороне сервера

Объектно-ориентированный программный язык	ActionScript 2.0	Языки семейства .NET	Javascript, Python, C++	Java, Javascript
Расширенные элементы пользовательского интерфейса для навигации и контроля данных	Отлично	Хорошо	Средне	Нет поддержки
Векторная графика	Отлично	Отлично	Средне	Нет поддержки
Спецэффекты элементов пользовательского интерфейса	Отлично	Хорошо	Средне	Использует Java AWT
Модель событий (диапазон контролируемых событий, эффективность модели)	Хорошо	Хорошо	Хорошо	Отлично
Ассоциативный уровень восприятия модели	Хорошо	Средне	Средне	Отлично
Документация	Отлично	Средне	Средне	Хорошо

Где моя CVS, чувак?

Автор: Андрей Олищук

Такой немного отвязный заголовок был взят для этой статьи неспроста. Цель данного материала – рассказать о базовых возможностях системы контроля версий CVS (Concurrent Versions System). Работать с этой технологией довольно просто, но для работы с ней необходимо прежде всего установить и настроить серверную часть системы. Установка и настройка CVS-сервера для новичка - это вопрос довольно обширный и автору не хотелось бы останавливаться на нем подробно, так как это немного отвлекает от ответа на главные вопросы – что же умеет CVS и как без особых временных затрат поработать с ней на практике. Поэтому для начала попробуем поэксперементировать на бесплатном CVS-хостинге http://cvsdude.org (Dude (англ) – чувак).

Для чего нужны системы контроля версий?

Подобные системы позволяют вести полную историю изменений ваших файлов с исходными кодами. Всегда можно посмотреть предыдущие редакции (в терминологии CVS это называется ревизиями) файлов с исходниками, ознакомиться с комментариями для каждой ревизии и выстроить общую картину внесенных изменении, а если это необходимо, то и совершить откат к предыдущим версиям. Помимо выполнения своей архивной функции, CVS, фактически, является инструментом групповой разработки и позволяет нескольким разработчикам вести одновременную работу над теми или иными файлами.

Представьте простую ситуацию, когда вы взяли из общего хранилища PHP-файл и начали дописывать в него некоторый код. В это же время ваш коллега обнаружил в этом файле очень опасную ошибку, также скачал его к себе, поправил ошибку и снова положил файл в общее хранилище. Вы, не зная об этом, дописали свой код и положили обновленный файл туда же. Что произойдет? Тем самым вы затрёте все изменения, сделанные вашим коллегой в то время как он будет думать, что ошибка в скрипте им закрыта. Конечно, можно рассылать друг другу уведомления или разделять файлы на “сферы влияния”, однако CVS гораздо эффективнее решает подобные проблемы.

CVS помогает в работе даже в том случае, если над проектом работает один разработчик. В практике веб-разработчиков нередки случаи, когда PHP-система, написанная для одного заказчика, начинает развиваться и к другому заказчику попадает уже в более продвинутом виде, а на пути к третьему клиенту ее совсем не узнать, так как в ней могут быть переписаны ключевые моменты. При всем при этом, необходимо сопровождать все три версии.

Для решения этого вопроса также можно воспользоваться различными уловками, однако CVS послужит в данной ситуации не только централизованным архивом, но и даст разработчику в руки инструменты для отслеживания изменений и параллельного развития нескольких версий одной системы.

К плюсам CVS можно отнести то, что она работает на различных платформах (в том числе и Win32) и распространяется бесплатно на условиях GPL. Помимо нескольких вариантов реализации серверной части, существует множество клиентских программ, которые также работают под различными платформами и распространяются бесплатно.

Инструментарий

Для работы с CVS необходимы две вещи: CVS-сервер и CVS-клиент. Как было сказано в самом начале статьи, в качестве сервера мы будем использовать бесплатный CVS-хостинг на сервере cvsdude.org, чтобы сразу не погружаться в установку и настройку своего собственного, так как это лучше сделать после некоторого знакомства с системой.

Вы можете самостоятельно посетить cvsdude.org и зарегистрировать отдельный аккаунт, чтобы далее работать с ним (рекомендуется), а можете также использовать тот тестовый аккаунт, который был заведен специально для этой статьи (подробнее о настройках далее). Бесплатный CVS-хостинг на cvsdude.org имеет ограничения на объемы закачиваемой информации в 2 мегабайта (что вполне достаточно для веб-проектов) и ограничение на количество пользователей одного проекта. Разрешено всего два пользователя: собственно сам администратор и еще один дополнительный пользователь. При внесении изменений в репозиторий (хранилище), всем пользователям на e-mail рассылаются уведомления с описанием внесенных изменений.

Теперь определимся с программой-клиентом. Здесь очень многое зависит от операционной системы вашего компьютера, с которого вы планируете работать с CVS. Не смотря на то, что среди веб-разработчиков существует огромное количество почитателей *nix, многие из них в качестве “домашней” операционной системы (ОС) используют Windows. PHP, к счастью, в большинстве случаев позволяет писать и тестировать код под одной ОС, а использовать его в работе на другой ОС.

Я для себя выбрал клиентскую программу под Windows - tortoiseCVS (http://www.tortoisecvs.org/). Вам не обязательно устанавливать эту программу, если вы хотите попробовать какую-то другую или по тем или иным причинам вам доступно только иное ПО. Если же сейчас для вас нет никакой разницы в выборе CVS-клиента, то tortoiseCVS будет очень хорошим стартом. В любом случае, все примеры в данной статье будут приводиться именно на tortoiseCVS, но это не исключает возможности использования любого другого CVS-клиента.

Итак, для начала посетите сайт http://www.tortoisecvs.org/ , скачайте последнюю версию программы и установите ее на своем компьютере. Установка не отличается какими либо особенностями, поэтому ее рассмотрение опустим. Теперь вам нужен только доступ в интернет и подготовку инструментария можно считать выполненной!

Тем временем...

Если вы хотите, чтобы HTML-код, генерируемый вашим PHP-скриптом был более читабельным в исходном виде (например, при использовании команды браузера View Source или View HTML), то в конце каждой выводимой на экран строки ставьте специальные символы \r\n. Пример:

```
<?php
echo “<h1>Заголовок</h1>\r\n
<p>Абзац</p>“;
?>
```

Для получения размера файла в килобайтах, можно использовать следующий код:

```
$KB_size = number_format(filesize
($file_name)/1024)." Кб";
```

Подробнее об использовании функции number_format() можно узнать по адресу: http://ru2.php.net/manual/ru/function.number-format.php

Самые основы CVS

Перед тем как приступить к работе с CVS, необходимо познакомиться с некоторыми терминами и принципами ее работы.

Схема работы с системой довольно проста. На сервере (в нашем случае это cvsdude.org) существует репозиторий (хранилилще) в котором хранятся все файлы и служебная информация к ним. Для начала работы, в репозитории создается модуль – прообраз самостоятельного проекта.

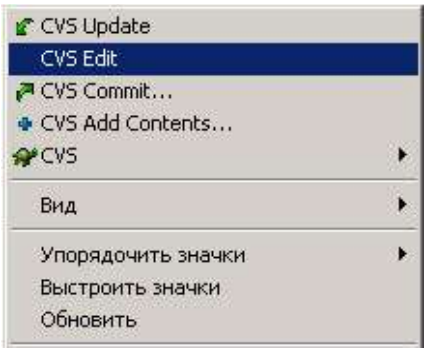
Начиная разработку, программист скачивает на свою локальную машину копию репозитория (процедура скачивания называется check out) и в последствии работает уже с этой копией на своей машине. После окончания работы, отредактированные локальные файлы заливаются обратно в репозиторий и хранятся там (операция “заливки” называется check in или commit).

Когда будет необходимость приступить к редактированию в следующий раз, то вам уже не понадобится делать check out всего модуля. Достаточно будет выполнить команду update и к вам на машину зальются только те файлы, которые были обновлены со времени вашего последнего чекаута.

Конечно же CVS предоставляет разработчикам множество различных возможностей, но в рамках данной статьи будут упомянуты только некоторые из них.

Начинаем работу

Как было сказано выше, в рамках статьи мы рассматриваем работу CVS-клиента tortoiseCVS на операционной системе Windows. Этот клиент отличается тем, что встраивается в контекстное меню и не имеет единого графического интерфейса (кроме диалоговых окон для конкретных команд и окна конфигурационных настроек), поэтому не ищите эту программу в меню программ Windows. Там вы найдете только окно конфигурации и справочные материалы. Вот так может выглядеть контекстное меню после установки клиента.



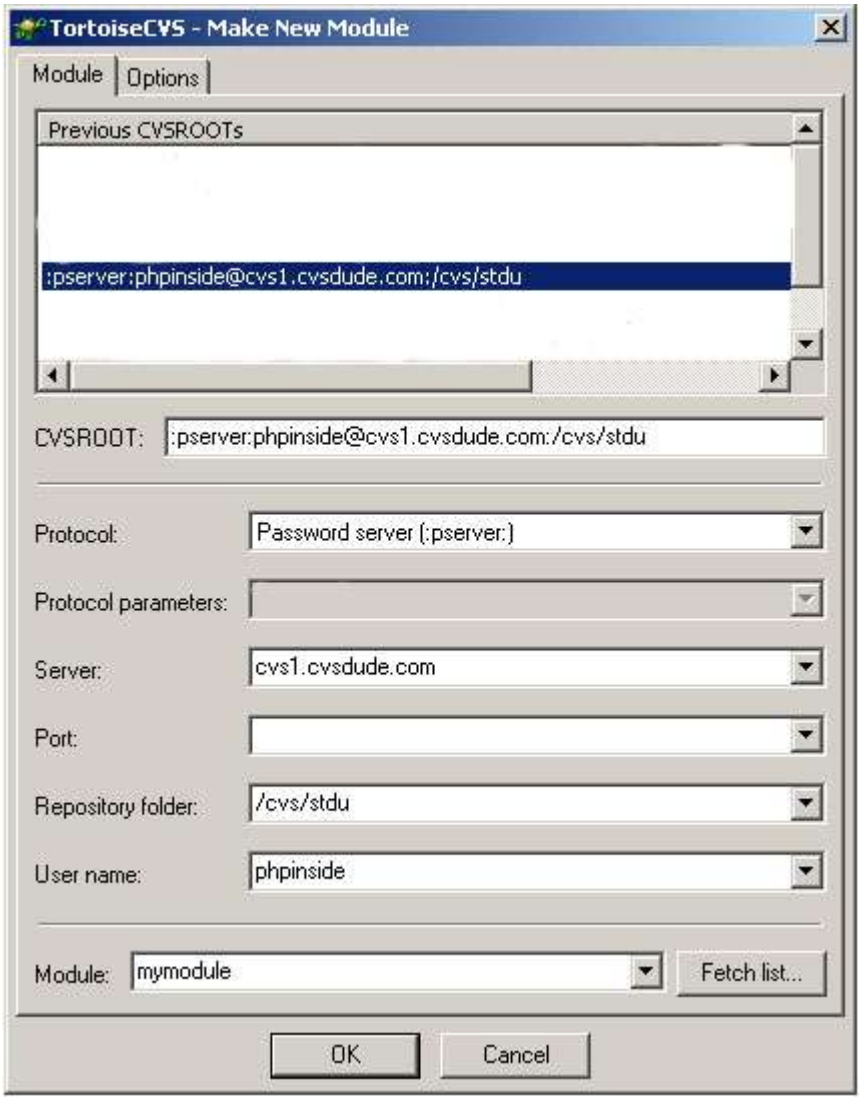
Если вы регестрировали свой аккаунт на cvsdude.org (что рекомендуется), то для начала работы вам необходимо создать модуль на их CVS-сервере. Для этого создайте директорию на локальном диске (к примеру, с именем mymodule), зайдите в эту директорию, создайте какой нибудь простой PHP-файл, откройте контекстное меню и выберите команду CVS -> Make new module.

Тем временем...

Если из строки, содержащей путь к файлу нужно извлечь имя самого файла, то можно воспользоваться функцией basename(). Например, название текущего файла можно получить следующим образом: ;
$curr_file = basename($PHP_SELF);

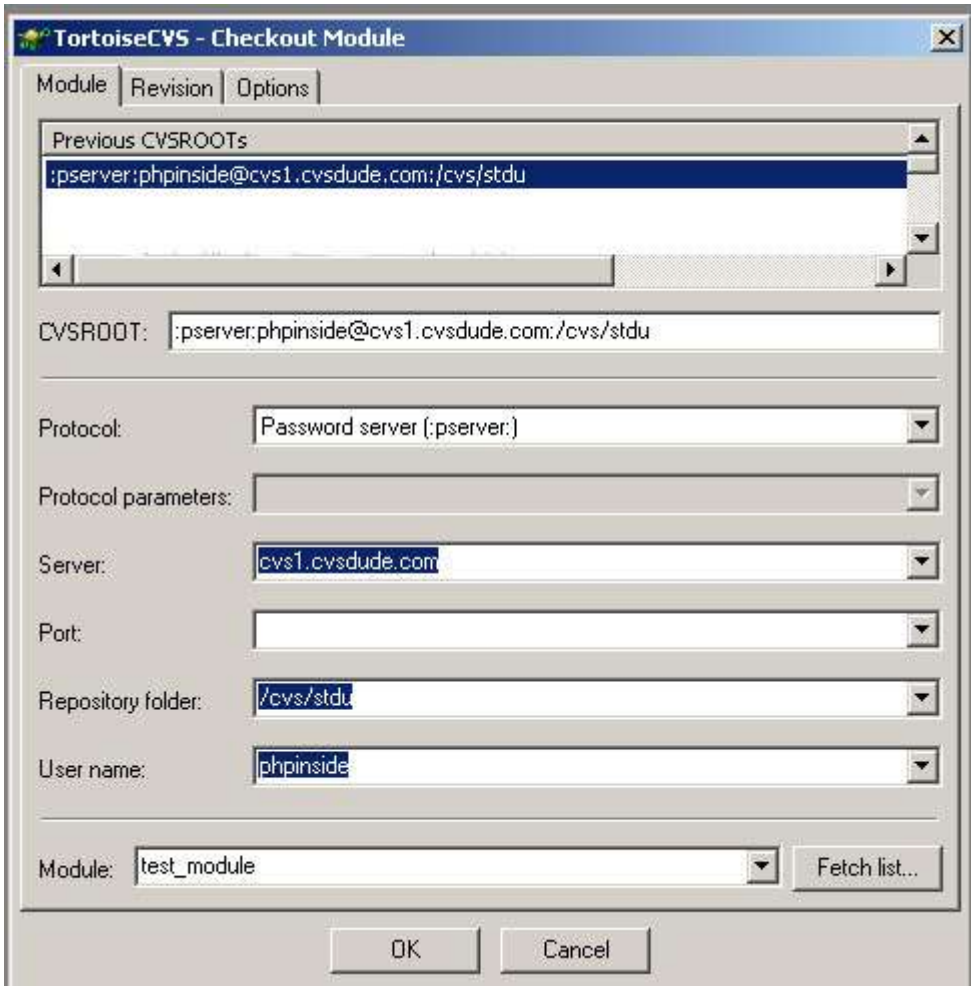
Более подробную информацию по использованию функции basename() можно получить по адресу: http://ru.php.net/manual/ru/function.basename.php

Далее введите ваши данные по образцу:



Нажмите кнопку OK. Теперь модуль будет создан.

Для начала работы над уже заведенным мной тестовом модулем, зайдите, используя проводник Windows (explorer) в ту директорию, в которую хотите скачать рабочую версию проекта. К примеру, C:\CVS. Далее, в контекстном меню (открывается нажатием правой кнопки мыши) выберите пункт CVS Checkout и введите данные, которые показаны на рисунке ниже. Если вы регистрировали свой аккаунт на cvsdude, то введите данные по аналогии.



Когда данные введены, жмите кнопку OK. Для подключения к CVS-серверу cvsdude.com у вас, конечно же, должно быть установлено соединение с интернетом и должен быть открыт порт 2401 (обычно его закрывают только в корпоративных локальных сетях).

При первом подключении будет запрошен пароль. В окошке ввода пароля наберите phpinside и нажмите кнопку OK. При последующих подключениях пароль обычно не запрашивается. Если все прошло удачно, то на вашем локальном диске появится новая папка test_module, в которой и хранятся все файлы проекта. Далее вам предстоит работать с ней, поэтому не рекомендуется ее удалять, даже не смотря на то, что в любые другие места вы можете скачать неограниченное количество копий (ограничения накладывает только объем вашего жесткого диска).

Работа над проектом

Когда модуль скачан на локальный компьютер, приходит время начать работу непосредственно с его файлами. Благодаря CVS, с ними можно выполнять целый ряд операций.

Создание и фиксация файлов в проекте

В только что полученной директории модуля создайте файл test.php и поместите в него какой либо PHP-код. Вы можете также вернуться к редактированию спустя некоторое время. После того, как вы решили, что файл готов для фиксации в CVS, выделите его кликом мыши, откройте контекстное меню и выберите команду Add. Эта команда регистрирует файл на сервере CVS, но пока еще не добавляет сам файл в репозиторий.

Когда команда Add будет выполнена, вы увидите, что пиктограмма файла сменилась с вопросительного знака на плюсик. Это свидетельствует о том, что изменился и статус файла в вашей локальной системе. Вопросительный знак означал, что CVS не знает такого файла. Появившийся плюсик говорит о том, что данная версия файла еще не фиксировалась на CVS-сервере с этого компьютера. Другими словами, плюсик означает статус “файл известен CVS, но не сохранен в репозитории”.

Для фиксации файла, через контекстное меню выберите команду CVS commit. Перед вами откроется окно управления фиксацией, где к данной версии (ревизии) можно написать какой либо комментарий. Не поленитесь и впишите пару строк текста. Комментарий будет относится не просто к файлу, а к этой конкретной версии файла. Если файл в дальнейшем будет обновляться, то к каждой его версии можно будет создавать комментарий и, таким образом, выстраивать целую историю изменений. Подтвердите фиксацию нажатием кнопки OK. Пиктограмма зафиксированного файла изменится на другую.

Получение обновлений

Параллельно с вами, над проектом может работать и другой пользователь. Для имитации действий второго пользователя, попробуйте скопировать к себе модуль в другую локальную директорию, указав в параметрах чекаута имя пользователя phpinside2 и пароль phpinside. Если вы заводили свой аккаунт, то зайдите в его настройки на cvsdude.com и заведите дополнительного пользователя (меню сайта project management -> authorized accounts).

После проведенного чекаута у вас на машине будет две локальные копии текущего модуля (проекта). В старой копии (сделанной от имени первого пользователя) измените PHP-файл и зафиксируйте его в репозитории командой контекстного меню CVS commit.

Таким образом, возникнет такая ситуация, когда во второй вашей локальной копии, файлы окажутся устаревшими. Поэтому, перед тем как начать работу со второй копией, выделите ее директорию кликом мыши и выполните команду контекстного меню CVS update. CVS-система сама сравнит версии всех файлов локальной копии и версии, хранящиеся в репозитории и положит вам в локальную копию все обновления. Те файлы, которые были не тронуты со времени последнего чекаута, так и останутся в репозитории и не создадут лишнего интернет-траффика.

Тем временем...

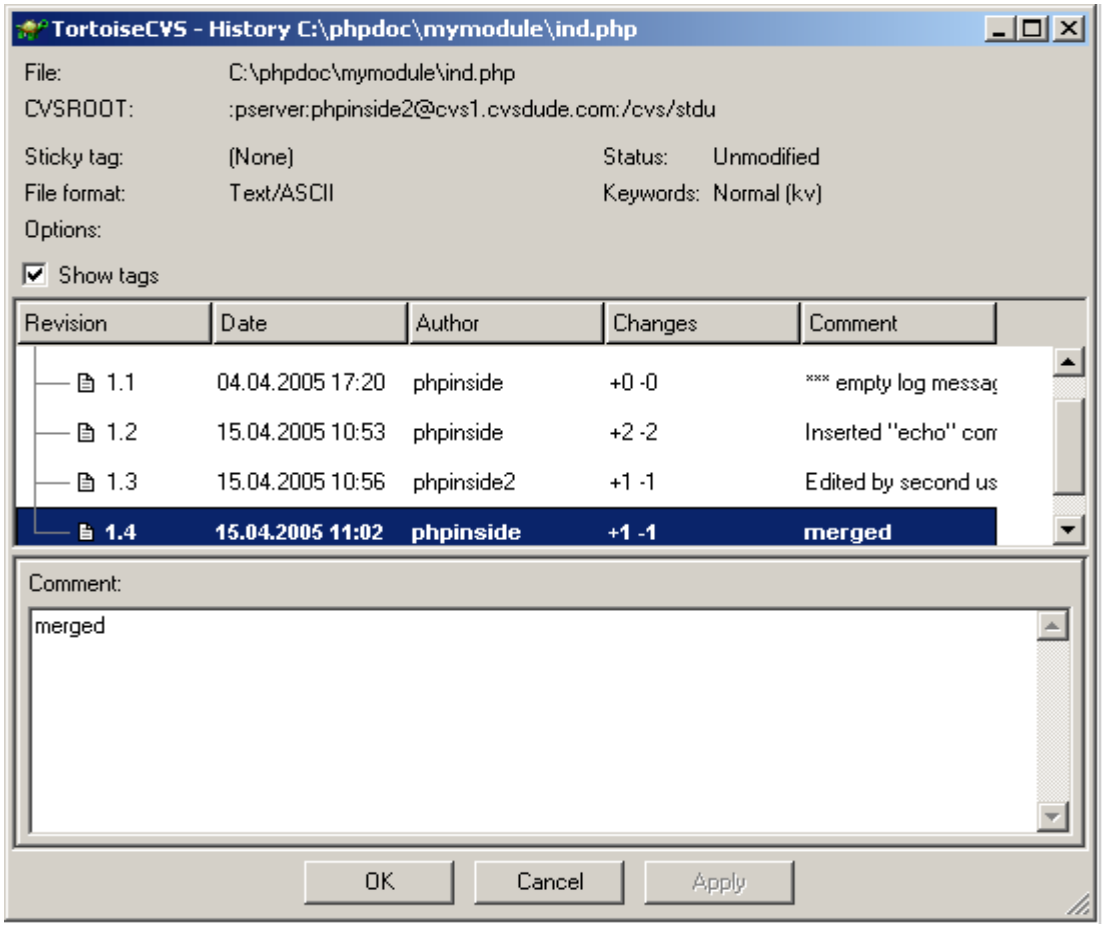
Для того, чтобы получить вчерашнюю дату, можно воспользоваться возможностями функции date(), к примеру, следующим образом:

```
$yesterday = date('d/m/y', mktime(0, 0, 0, date("m") , date("d") - 1, date("Y")));
```

История версий

Каждая команда CVS commit не просто фиксирует ревизию файла на CVS-сервере, но и обозначает его как новую версию. Обычно, самой первой ревизии присваивается номер 1.1, а следующей, соответственно – 1.2. Для всех последующих ревизий приращение идет на 0.1. Конечно существуют возможности повлиять на эту нумерацию, но с этим вы сможете разобраться позже самостоятельно.

Если при каждой фиксации вы оставляли комментарии, то теперь будет возможность посмотреть не только дерево ревизий, но и историю всех изменений конкретного файла. Для этого просто выделите нужный файл и через контекстное меню выполните команду CVS -> History. Появится примерно следующее окно:



В данном окне можно увидеть какая ревизия когда была выложена, кто автор, сколько строк изменено и какие оставлены комментарии. Однако это еще не все.

Сравнение двух ревизий

Не закрывая окна с историей ревизий, выделите две любые ревизии (например, удерживая клавишу Ctrl) и вызовите контекстное меню. В контекстном меню выберите команду Diff (selected revisions). С помощью данной команды можно открыть специальный редактор и увидеть текст обеих ревизий с подсвеченными различиями между ними. Вот как это может выглядеть:

Используя эту опцию, можно посмотреть все различия между этими версиями.

Итоги

Здесь мы рассмотрели самые основы работы с CVS, используя CVS-хостинг http://cvsdude.com и CVS-клиент tortoiseCVS. Конечно, самые “вкусности” CVS остались за бортом (например, ветвление проекта, разрешение конфликтов между ревизиями, слияние веток и ревизий и многое другое), но возможно вам будет полезен и достаточен уже описанный выше функционал. Главное, чтобы немного разобравшись с основами вы смогли разобраться с деталями самостоятельно и в короткие сроки.

Zend Platform: Подробности

Комплекс Zend Platform "расположен" между непосредственно PHP (Zend Engine) и PHP-скриптами организации, то есть предоставляет платформу, на которой базируется комплекс Web-сервисов, бизнес-приложений, систем управления контентом, интранет- и b2b-приложений. Такая тесная интеграция позволяет Zend Platform глубоко проникать в ваши PHP-приложения, улучшать производительность PHP и повышать контролируемость сред разработки, тестирования и производственных сред.

Оригинал статьи: http://www.zend.com/store/products/zend-platform/in-depth.php
Перевод: Данил Миронов [patrungel]

Рис. 1. Взаиморасположение Zend Platform и комплекса приложений PHP

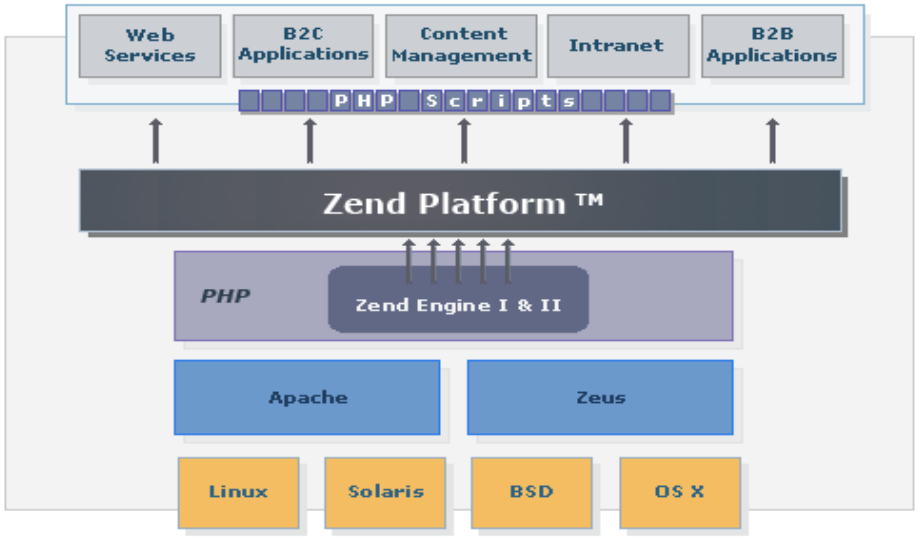
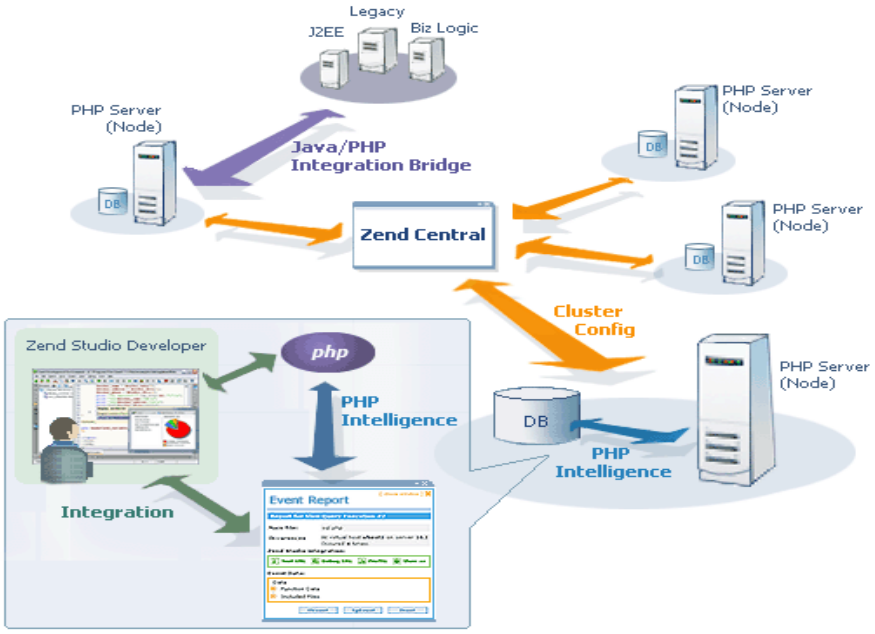


Рис. 2. Архитектура Zend Platform



PHP Intelligence

Одним из рабочих механизмов Zend Platform является PHP Intelligence - это событийно-управляемая система, которая в режиме реального времени проводит анализ ваших PHP-приложений. Полученная с её помощью оперативная информация позволяет быстрее и легче разрешать возникшие проблемы.

С помощью PHP Intelligence, разработчики и IT-специалисты наконец смогут проникнуть внутрь своих PHP-приложений и получить любые жизненно важные сведения, которые им только могут понадобиться, включая спады производительности, ошибки скрипта/функций, проблемы с базами данных. При этом разработчик получает полный контекст проблемы.

Здесь Zend Platform:

- Отслеживает поведение скрипта/функции и подсчитывает нормальные средние значения, предоставляя тем самым наилучшее указание на проблемные участки кода.
- До минимума снижает время восстановления рабочей системы, предоставляя чёткое и тщательное описание проблемы с учётом полного контекста проблемы и правил, определённых пользователем.
- Позволяет менеджеру сайта незамедлительно сигнализировать о критических ошибках при помощи стандартных XML-based модулей отчётов (SMS, email, и т. п.).
- Благодаря тесной интеграции с Zend Studio, предоставляет отладочную информацию и данные по рабочему профилю программы для целей устранения неполадок.
- Объединяет события, и тем самым снижает "уровень шумов", и даёт возможность игнорировать уже известные или незначительные проблемы.

Контроль над конфигурацией PHP

В основе Zend Platform лежит мощная система управления PHP. Данная система обеспечивает полный контроль над комплексом PHP-приложений, включая настройки производительности, настройки оповещений о событиях, и т. д.

К тому же Zend Platform предоставляет среду, в которой application-менеджеры могут управлять конфигурацией самого PHP – всё через тот же самый единый web-интерфейс. Вся система работает через SSL и шифрование с открытым ключом, таким образом обеспечивается безопасность уровня предприятия.

Zend Platform экономит драгоценное время системных администраторов и снижает влияние человеческого фактора при настройке нескольких одинаковых серверов, предоставляя возможность:

- Конфигурирования удалённых серверов.
- Клонирования конфигурации PHP с одного сервера на другой или с одного сервера на целую серверную группу.

Система управления производительностью PHP

Zend Platform оснащён четырьмя модулями управления производительностью, они позволяют проводить аудит скоростей работы и отклика Ваших web-приложений, а также улучшать эти показатели. Эти модули унаследовали все лучшие черты Zend Performance Suite, которая уже стала стандартным инструментом оптимизации производительности PHP web-сайтов.

Акселератор Кода. Как правило, Акселератор Кода не требует каких-либо изменений в конфигурации или в самих приложениях. Применяются как различные методы кэширования, так и методы оптимизации скомпилированного PHP-кода, а в результате имеет место в среднем трёхкратное улучшение производительности.

Динамическое кэширование контента. Принцип работы Динамического Кэширования Контента основан на том наблюдении, что многие динамически сгенерированные web-страницы нередко бывают абсолютно одинаковыми. Следовательно, можно сохранить результаты исполнения от первого запроса и на последующие запросы страница поставляется "как есть", приложение, собственно, не исполняется. В наиболее распространённых конфигурациях Динамическое Кэширование Контента увеличивает производительность от 20 до 150 раз.

Сжатие Файлов. Система сжатия файлов Zend Platform позволяет владельцам сайтов выдавать динамически сгенерированный контент в сжатом виде, при этом пропускная способность канала будет использована лишь на 10% от исходной величины.

Zend Platform поддерживает прозрачную для конечного пользователя декомпрессию доставленных файлов, данная поддержка предоставлена для всех распространённых web-браузеров. Благодаря интеграции модулей в самой Zend Platform, Сжатие Файлов осуществляется в связке с Динамическим Кэшированием Контента, позволяя серверу сжимать каждую динамическую страницу лишь единожды, таким образом, непроизводительные затраты в работе ЦП снижаются до уровня незначительных.

Менеджер Загрузок Zend. У Apache даже такая тривиальная задача, как отправка большого файла, требует отдельного полноценного процесса. Для сайтов, на которых выложены большие мультимедийные или просто бинарные файлы (видео, изображения, аудио, trial-версии ПО и т. п.), это означает, что Apache быстро исчерпает все свои ресурсы и перестанет откликаться на запросы.

Менеджер Загрузок Zend от Zend Platform способен подключиться непосредственно к любой существующей конфигурации Apache/PHP и разгрузить процесс отправки больших файлов через Apache, тем самым освобождая последнего для обработки более сложных запросов, связанных с PHP. Как правило, при поддержке Менеджера Загрузок сервер способен на 10-кратное увеличение числа одновременных файловых загрузок.

Мост интеграции PHP/Java

Мост интеграции PHP/Java от Zend Platform позволяет компаниям, вложившимся в серверы приложений J2EE, использовать и PHP - язык номер один в мире для web-приложений. Существует и обратная связь: при помощи Моста интеграции PHP/Java различные PHP-направленные компании могут применять в своей работе J2EE сервисы, недоступные в скриптовых языках. При этом Мост интеграции PHP/Java от Zend Platform не ограничивается взаимодействием с одними лишь J2EE- и унаследованными системами, он предоставляет также возможность работать с простыми Java-объектами.

Мост интеграции PHP/Java от Zend Platform – это уникальная разработка, в числе достоинств которой – невиданные ранее производительность и масштабируемость. В отличие от прочих решений по интеграции PHP и Java, огромные требования к оперативной памяти которых очень затрудняют их внедрение, Мост интеграции PHP/Java от Zend Platform потребляет ограниченное количество памяти, что практически диспропорционально количеству выполняемой им работы.

Люди

Наши. Денис Колисниченко – автор книг по PHP и Linux

Интервью: Андрей Олищук

В продолжение серии материалов "Наши", мы решили взять интервью у автора нескольких книг по PHP (в том числе уже и по PHP5) и Linux – Дениса Колисниченко. Денис пошел нам на встречу и любезно согласился ответить на все наши вопросы.

nw: Добрый день! Наши читатели знают вас прежде всего как автора самоучителя по PHP 5 и ряда других книг по PHP и Linux, однако написание книг это, наверное, ваша не единственная работа? Расскажите пожалуйста о том, кем вы работаете?

Денис Колисниченко (ДК): Второй год занимаюсь разработкой программного обеспечения под заказ и настройкой Linux-серверов. Разрабатываю проекты любой сложности - от простого PHP-сценария до сетевой БД.

До этого работал системным администратором и программистом – по совместительству :)

Я самый обыкновенный человек – работа, дом, хобби...

nw: Расскажите немного о себе не как о писателе и IT-специалисте, а как об обычном человеке. Есть ли у вас хобби? Как вы проводите свое свободное время? Какую музыку любите?

ДК: Свободного времени очень мало - все занимает работа. Но хобби все-таки есть - автомобили, в частности BMW - у меня 520 E34 – хороший, надежный автомобиль (когда не ломается :)).

Свободное время делится примерно пополам между любимой девушкой и бумером (иногда больше в пользу последнего, что вызывает "бесконечные и громкие" возмущения со стороны девушки :))) А в остальном я самый обыкновенный человек – работа, дом, хобби.

Музыка - самая разнообразная, кроме откровенной "попсятины". Раньше слушал то, что "потяжелее", но от такой музыки со временем устаешь (сейчас слушаю ее под настроение) и начинаешь слушать что-то вроде шансона (современная поп-музыка ничем не привлекает).

nw: По некоторым нашим данным, ваша книга "Самоучитель PHP 5" пользуется довольно хорошим спросом. У вас есть какие либо тайны написания успешных книг? Расскажите чуть подробнее о процессе работы над книгами. Насколько это трудоемкий процесс? Как много уходит времени на написание одной книги?

ДК: Каких-либо особых тайн или секретов написания книг нет. Есть два момента, на которые нужно обратить внимание при создании книги, но об этом - чуть позже.

Процесс создания книг в "двух словах": создание рукописи, вычитка рукописи редактором, верстка, вычитка макета и, наконец, печать макета.

Наиболее трудоемкий процесс для автора - это написание рукописи. Сами понимаете, хорошую книгу за месяц не напишите, даже если это будет ваше основное занятие. Самое главное - описать ту или иную технологию понятным читателю языком. Ведь можно просто перевести на русский официальную документацию, HOWTO и издать. Но кто такую книгу купит? Зачем? Ведь можно просто скачать документацию (а может она уже установлена на компьютере).

Самое главное – описать ту или иную технологию понятным читателю языком

Бывает даже и такое, что приходится переписывать целые главы или добавлять новые перед самой верстской - просто с момента начала написания книги до ее вычитки редактором проходит определенное время, а компьютерная литература - это не роман, который всегда остается неизменным, за это время много что может измениться, например, выйти новая версия дистрибутива или ядра (как это было при написании второго издания Linux-сервера - вышло ядро версии 2.6) со всеми вытекающими последствиями. Придется или переписать определенную часть книги или просто добавить описание новых возможностей.

Смотрите сами: печать занимает от двух недель до месяца. Еще с неделю (а иногда и больше - все зависит от региона) пройдет, пока книга окажется на прилавках. Конечно, в фирменных магазинах издательства книга появляется почти сразу - не проходит и недели, а вот пока книга "дойдет" до других городов или стран...

И вот, прошел месяц с момента выхода кнги из типографии (а может даже два), в магазине эта книга сразу появляется как "Новинка". Понимающий читатель берет ее в руки и читает "версия ядра 2.4". Да какая же это новинка? Если версия 2.6 вышла, например, три месяца назад. Вот поэтому и приходится добавлять новый материал - чтобы информация была актуальной. Это и есть первый "секрет".

Не секрет - я сам часто читаю и форумы и книги других писателей - учиться всем нужно

Второй - это хороший редактор. Ведь редактор - это первый читатель книги. Он ее вычитывает, убирает шероховатости, мелкие ошибки и т.д. Огромное спасибо Марку Финкову - главному редактору "Науки и Техники" - он и есть первый читатель всех моих книг.

nw: Что вам нравиться делать больше - использовать ту или иную технологию или писать о ней? Почему?

ДК: Когда как. Иногда хочется создать что-то своими руками – например, написать то или иное приложение. А иногда описать, как использовать тот инструмент, который используешь в повседневной работе. Просто хочется поделиться своим опытом работы с другими читателями. Не секрет - я сам часто читаю и форумы и книги других писателей - учиться всем нужно.

nw: Планируете ли вы написание какой либо новой книги? Если да, то о чем она будет?

ДК: Это секрет :) Скоро (наверное, ближе к лету) выйдут две моих новых книги, но о чем они, я не скажу.

nw: Как вы пришли к PHP? Почему именно PHP?

ДК: С PHP работаю относительно недавно - с конца 2000 года. Тогда "домашний" провайдер переделал свой сайт - он был как раз написан на PHP. Вот я и решил попробовать - разобраться, что это такое.

PHP-проекты - это половина сдаваемых мною проектов

nw: Используете ли вы сами PHP 5 в рабочих проектах? Какие нововведения вам кажутся наиболее полезными и как вы их применяете?

ДК: Конечно использую! PHP-проекты - это половина сдаваемых мною проектов. Вторая половина - это Windows-приложения, как правило, базы данных. В последнее время было несколько "гибридных" проектов - движок на PHP, а модуль управления - Windows-приложение (совмещающее в себе Web- и FTP-клиент). Если раньше такие проекты не заказывались вообще, то сейчас пользуются спросом.

Суть заключается в следующем: движок выводит информацию о ресурсах проекта, содержающуюся в базе данных MySQL. Обновление ресурсов и самой базы данных выполняет Windows-приложение. В этом случае пользователю не нужно знать ни основы SQL, ни что такое FTP и как с ним нужно работать - он просто указывает нужную информацию и нажимает обновить - за него все делает программа.

Кроме комфорта, пользователь получает "в нагрузку" привычный ему Windows-интерфейс.

nw: С какими другими технологиями вам приходится работать(версии ОС, СУБД, веб-серверы...)? Какие их связки вам больше всего нравятся? Почему?

ДК: Если "в двух словах", то наиболее часто использую следующи продукты:

- Delphi - не могу сказать, что это лучшая среда для разработки Windows-приложений, поскольку 100% ошибусь, но к Delphi я привык (использую ее с 1999 года), да и по роду занятий она мне подходит (базы даных и сетевые приложения). На Delphi можно написать абсолютно любое приложение пользовательского уровня.
- InterBase - используется как сервер БД в паре с Delphi, когда создается сетевая БД
- PHP + MySQL + Apache - идеальная комбинация, которая присутствует у большинства хостинг-провайдеров. Возможности MySQL не безграничны, но для создания небольшой базы данных для Web-проекта их вполне хватает. SQLite практически не использую (пока) - причина указана выше – у большинства провайдеров установлен именно MySQL.
- Linux и Windows - несмотря на то, что я написал несколько книг, посвященных Linux, в своей работе я использую обе эти ОС "равномерно".

Сейчас, например, работаю в Windows XP, а ответы на это интервью пишу в The Bat'e.

nw: И наконец, чего бы вы пожелали нашим читателям?

ДК: Прежде всего, я хочу поблагодарить всех читателей - тех, кто уже купил мою книгу, тех кто собирается это сделать, и тех, кто покупать ее вообще не будет - просто за то, что он читает это интервью и журнал "PHP Inside".

Пожелать могу постоянного совершенствования, достижения поставленных целей и успеха в работе - это по работе. А по жизни - можно пожелать только одного - здоровья, а все остальное приложится :)

nw: Спасибо за ответы!

План врезок