Январь / Февраль '05 № 10

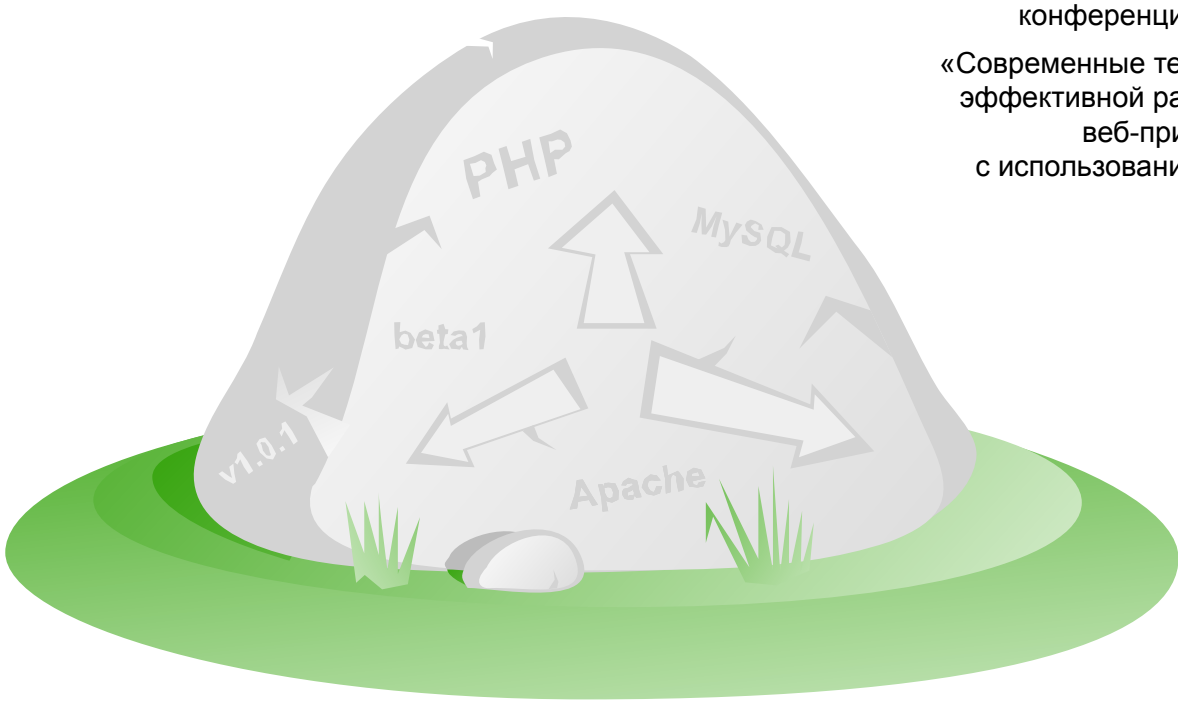
PHPInside

электронный журнал для веб-разработчиков

phpCONF 2005

Четвертая международная конференция, г. Киев

«Современные технологии эффективной разработки веб-приложений с использованием PHP»



Кто на свете всех быстрее?

Сравнительные тесты производительности серверного програмного обеспечения

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.

Команда номера

Авторы и переводчики
Игорь Федоров
Артур Гатин
Андрей Олищук
Кузьма Феськов

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.net

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.

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! Действует система скидок "early-bird" - для тех, кто оплачивает до 1апреля 2005 года, регистрационный взнос составляет 500 грн. - 2600 руб.

Представление докладов

Рабочий язык конференции - русский и английский, представление докладов возможно на любом из них. Тезисы докладов представляются в программный комитет на русском или английском языке электронной почтой по адресу: 2005@phpconf.ru - в формате RTF. Мы надеемся, что тезисы будут давать достаточно полное представление о содержании предлагаемого доклада. Мы не ограничиваем вас перечисленными выше темами и готовы рассмотреть любые интересные доклады, так или иначе связанные с разработкой веб-приложений.

Важные даты:

- 14 февраля - окончание приема тезисов докладов
- 18 февраля - утверждение программы конференции, извещение о статусе доклада (принят/не принят)
- 25 апреля - окончательный срок сдачи материалов для включения в информационный справочник конференции

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

Программа мероприятия

12 Мая (Четверг)

8:45-10:00	**Регистрация участников. Вручение сопроводительных материалов**
10:00-10:15	**Вступительное слово, представление докладчиков и тем**. Андрей Зинченко, компания Миротел г.Киев, Украина
10:15-11:00	**Свой проект свободно распространяемого программного обеспечения: создание, продвижение, проблемы существования** Козак Андрей г.Кривой Рог, Украина 1.1 История OpenSource ПО. 1.2. Зачем нужен свой проект в OpenSource? 2. Создание своего проекта. 2.1. Хостинг. 2.2. Документация. 2.3. Выбор лицензии (GPL, LGPL, BSD). 2.4. Контроль версий. 2.5. Стиль кода. 3. Продвижение своего проекта. 3.1. Привлечение новых разработчиков. 3.2. Реклама. 3.3. Использование в других проектах. 4. Проблемы существования и их решения. 4.1. Отсутствие единомышленников. 4.2. Финансирование проекта.
11:00-11:15	Кофе-брейк
11:15-12:00	**XML Sapiens как универсальная концепция сайтостроения в разрезе XML/PHP** Дмитрий Шейко ведущий программист www.redgraphic.ru, автор спецификации XML Sapiens www.xmlsapiens.org г.Минск, Беларусь 1. Почему в веб-разработке неприменимы модели Windows Forms, Win32 API, MFC. 1.1 Расчищаем путь от ТЗ к готовому проекту; 1.2 Подходы к описанию пользовательских интерфейсов UIML и XUL; 1.3 Инструментарий разработчика PHP GTK и SMARTY; 1.4 Разделение данных и представления в W3C XSL; 1.5 Комплексное решение для динамических сайтов XML Sapiens.

	2. Динамический сайт в объектах. 2.1 Управление пользовательскими интерфейсами сайтов в популярных open source проектах CMS; 2.2 Задачи веб-разработчика по управлению объектами сайта; 2.3 Экспресс экскурс в теорию динамического сайта. 3. Разделяем функциональность, данные и представление по "рецепту" XML Sapiens. 3.1 "Три кита" в структуре документа динамического сайта; 3.2 Конструирование пользовательских интерфейсов с помощью контейнеров динамических данных; 3.3 Синтаксис языка описания контейнеров динамических данных. 4. Переносимость динамических сайтов. 4.1 Платформы доставки контента и семантический веб; 4.2 Прореха модели XSL; 4.3 Разделение сценариев пользовательских интерфейсов и кода представления. 5. Как XML Sapiens практически реализуется в PHP. 5.1 Процессор XML Sapiens собственными руками; 5.2 Алгоритм процессора XML Sapiens.
12:00-12:15	**Кофе-брейк**
12:15-13:40	**Разработка современной CMS. Преимущества, которые дает PHP5 при разработке таких систем** Константин Погорелов, веб-программист, компания "AWWSOFT" Дмитрий Магунов, руководитель проекта компания "AWWSOFT" г.Одесса, Украина 1. Требования к современной CMS с примерами существующих решений. Обзор реализации нижеперечисленных аспектов в популярных CMS. 1.1. Модульность (расширяемость, возможность кастомизации под потребности заказчика). 1.2. Многоязычность. 1.3. Масштабируемость. 1.4. Удобство в использовании. 2. Обработчик шаблонов. 2.1. Общие требования к шаблонизатору. 2.2. Язык шаблонов. Особенности языка для описания шаблона. Краткий обзор существующих языков. 2.3. Виртуализация данных и функций их представления. 3. Предлагаемая модель CMS. 3.1. Составляющие CMS.

	3.2. Иерархия классов. 3.3. Быстродействие системы. Оптимизация и кеширование. 3.4. Преимущества использования PHP5 при реализации данной модели. 4. Краткий пример создания, администрирования и функционирования сайта на AWWCMS.
14:00-15:00	**Обед**
15:00-16:15	**Оптимизация PostgreSQL** Борзов Алексей [SadSpirit], независимый разработчик, ведущий раздела PostgreSQL на PHPClub.ru г.Москва, Россия 1. Настройка сервера 1.1 Объём используемой памяти: общий буфер сервера, сортировка результатов запроса, информация о свободномпространстве 1.2 Журнал транзакций и контрольные точки 1.3 Сбор статистики по выполняемым запросам 1.4 Оптимальное использование дисковой подсистемы сервера. Новые возможности версии 8.0 2. Поддержание базы данных в порядке 2.1 Сборка мусора при помощи команды VACUUM 2.2 Обновление статистики оптимизатора при помощи команды ANALYZE 2.3 Автоматизация процесса при помощи утилиты pg_autovacuum, преимущества перед cron 2.4 Когда надо (и надо ли) использовать REINDEX и VACUUM FULL 3. Оптимизация конкретных запросов 3.1 Чтение планов запроса: способы просмотра таблицы, разные типы объединений 3.2 Возможности индексов в PostgreSQL: частичные индексы, индексы по выражениям 3.3 Использование статистики для локализации проблемных запросов 3.4 Хорошо известные грабли, на которые не стоит наступать
16:30-17:50	**MySQL - просто о сложном** Войцеховский Александр Николаевич [young], независимый разработчик, ведущий раздела PHP в деталях http://detail.phpclub.ru. г.Киев, Украина 1. Вступление: сага об индексах. 2. В погоне за скоростью. 2.1 Оценка SQL-запроса. 2.2 Внутренняя оптимизация запросов mySQL и ее подавление. 2.3 Тюнинг сервера mySQL. 2.4 mySQL и кеширование.

	3. Кластеризация. 3.1 Кластер How-To. 3.2 Настройка и запуск кластера своими руками. 4. Локализация mySQL - что нового? 5. Триггеры и пользовательские функции - это просто.
17:50-18:30	**Построение поисковых систем. Принципы и Реализация**. Юрий Ефимович Логинов, технический директор ЗАО "МЕТА" http://meta.ua Харьков, Украина Основные моменты: 1. Взаимодействие внешних и внутренних поисковых систем с сайтом. 2. Использование метаданных и структурной информации. 3. Проблемы эффективности поисковых систем по сайту. 4. Установка и настройка поиска по сайту (на примере siteMETA http://sitemeta.com/rus/) 5. Подходы по интеграции siteMETA с сайтами построенными на PHP 6. Использование XML в кросс-платформенных решениях взаимодействия споисковой системой.

13 Мая (Пятница)

09:00-10:00	**Регистрация участников. Вручение сопроводительных материалов.**
10:00-10:15	**Обзор мастер-классов, анонс тем круглого стола**
10:15-10:35	**Целесообразность модульного тестирования в php** Юдин Сергей Юрьевич, программист ООО "БИТ", разработчик LIMB г.Пенза, Россия 1. Плюсы и минусы тестирования. 2. Запахи тестового кода и связь этих запахов с недостатками в архитектуре тестируемого кода. Некоторые примеры. 3. TDD как способ изучения шаблонов проектирования.
10:35-11:45	**Две стороны документирования: PHPdocumentor и DocBook** Борис Безруков [lovchy], технический директор DeltaInform, OSS-энтузиаст, член группы документирования PHP. г.Москва, Россия 1. Документирование исходного кода: стандарт PHPdocumentor. 1.1 История возникновения, общая информация- Вкратце о JavaDoc- PEAR::PHPdoc- Суть подхода 1.2 Обзор возможностей- Документирование процедурного кода- Константы и включения (include, require)- Документирование ОО-кода; PHPdocumentor и PHP5. 1.3 Технология и сообщество- PHPdocumentor и GPL, PEAR 1.4 Элементы разработки пользовательской документации: разбор полётов

	2. Пользовательская документация: DocBook 2.1 Описание стандарта- Суть технологии- Использование XML- Стандарт де-юре 2.2 Краткий обзор DTD- Пример создания документа- XML vs SGML- Entities: ввод терминов и разбиение на документы- Формирование документов: DSSSL и XSLT 2.3 Соавторство; связка DocBook и CVS на примере документации PHP- Краткое введение в CVS- Использование GNU-AutoTools, организация мультиязычности- Общая организация документов- LiveDocs
11:45-12:00	**Кофе-брейк**
12:00-12:45	**XML в PHP5** Жолудов Денис [DAN], программист, ООО "Астелнет". Ведущий раздела "PHP & XML-технологии" на PHPClub.ru г.Москва, Россия 1.1. История внедрения XML-технологий в PHP. 1.2. Развитие расширений PHP для обработки XML документов. 2. XML в PHP5 2.1. Стандарты W3C и библиотека libxml. 2.2. Расширения PHP5 для обработки XML-документов. 2.3. Приемы использования расширений DOM, SimpleXML, XMLReader. 2.4. XSLT как язык описания шаблонов. 2.4.1. Callback-функции в XSL. 2.4.2. Мультиязычные шаблоны. 2.5. PEAR и PECL пакеты для работы с XML. 2.6. XML-RPC и SOAP. 3. XML и СУБД. 4. Краткий обзор PHP-Opensource проектов, использующих XML-технологии. 5. Заключение. Чего ожидать в будущем?
12:45-13:30	**Внедрение современных открытых стандартов в ОАО "АВТОВАЗ" на примере связки PHP5::SOAP и Xforms** Анохин Александр Владимирович [chameleon], начальник бюро экспертизы Интернет-проектов ОАО "АВТОВАЗ". Соавторы: • Булов Владимир Геннадьевич, начальник центра развития информационных технологий ОАО "АВТОВАЗ"; • Литвинов Кирилл Анатольевич, инженер-программист ОАО "АВТОВАЗ".г.Тольятти, Россия 1. Введение в веб-сервисы 1.1. Предпосылки перехода на веб-сервисы и SOA 1.2. Достоинства и недостатки веб-сервисов 1.3. Технологии, применяемые при организации веб-служб

	2. Стандарт Xforms 2.1. Ввод данных через веб на данный момент 2.2. Обзор преимуществ Xforms 2.3. Технически подробности 2.3.1. Введение 2.3.2. Отделение содержания от представления 2.3.3. Модель 2.3.4. Экземпляры данных 2.3.5. Элементы управления 2.3.6. Состав документа 2.3.7. Хост-разметка 2.3.8. Встраивание модуля Xforms 2.3.9. Определение пользовательского интерфейса 2.4. Существующие имплементации стандарта 2.4.1. Серверные процессоры 2.4.2. Клиентские плагины 2.4.3. Средства разработки 3. Пример использования Xforms как клиента веб-службы, реализованной на базе PHP5::SOAP. 4. Заключение: перспективы развития и ресурсы.
13:30-14:30	**Обед**
14:30-15:45	**Поддержка нескольких СУБД в проекте. Существующие подходы, их анализ. Методика перевода существующего проекта на поддержку нескольких СУБД**. Соловьёв Денис [ForJest], вольнонаёмный разработчик программного обеспечения. г. Днепропетровск, Украина 1. Исходные условия, мотивация для поддержки нескольких СУБД. 2. Шаг первый - использование абстрактного доступа. Достоинства, преимущества. 2.1. Общие идеи, предпосылки для возникновения. 2.2. Преимущества, которые мы получаем, чем хороша абстракция доступа к СУБД. 2.3. Обзор популярных библиотек. Сводная таблица, краткий анализ. 3. Шаг второй. Первые трудности 3.1. Общность против особенностей. 3.2. Более пристальный взгляд в абстракцию. Причины конфликта. 3.3. Решения, преподносимые библиотеками абстрактного доступа. 3.4. Популярные решения в проектах, пришедшие на ум разработчикам.

	3.5. Краткие выводы. Ресурсоёмкость поддержки и доработки для новых СУБД. 4. Шаг третий. Новые идеи и методика. 4.1. Применение современных технологий. Вывод оптимального решения с помощью рефакторинга. 4.2. Шаг первый. Вынесение набора операций и написание тестов. 4.2.1. Кратко о тестах. Зачем и как писать. Преимущества против трудозатрат. 4.2.2. Разбиение на операции. Основные концепции методики. 4.2.3. Почему концепции лишь основные и почему вопросы позже. 4.3. Шаг второй. Пожинаем плоды тяжёлого труда и получаем желаемое. Окупленные трудозатраты. 4.4. Небольшое отступление в теорию. Что можно делать дальше. Взгляд назад. 4.5. Оценка ресурсоёмкости поддержки и включения в проект новых СУБД, после переработки с учётом методики.
15:45-16:00	**Кофе-брейк**
16:00-17:00	**Платежные системы - это не страшно** Бондарев Евгений, Project-manager в компании RoyalClub, модератор форума по PHP на http://www.xpoint.ru г.Киев, Украина 1. Введение. Основные понятия. Немного истории. 2. Что внутри. Как оно работает. 3. Механизмы проведения платежа. 3.1. Общие вопросы:- безопасность обмена данными с ПС;- целостность данных: контрольные суммы и цифровые подписи. 3.2. Прямое обращение к gateway платежной системы. 3.3. Merchant-panel на сервере платежной системы. 4. Примеры. 4.1. WebMoney. 4.2. Итернет.деньги (PayCash). 4.3. Int-Commerce. 4.4. E-Gold, PayPal. 4.5. Обзор Skannet.dk. 4.6. Обзор Менатеп. 4.7. Обзор RuPay.
14:30-17:00	**Мастер-класс: Влияние тестирования на дизайн кода (TDD в PHP)** Щеваев Павел Александрович, программист ООО "БИТ", http://bit-creative.com, разработчик LIMB, http://limb-project.com. г.Пенза, Россия 1. Test Driven Development и PHP. 2. Преимущества TDD.

	3. Запахи тестового кода или же признаки хороших тестов. 4. Использование статических методов. 5. Использование паттерна Singleton (одиночка) в проекте. 6. Тестирование иерархий классов. 7. Тестирование закрытых методов. 8. Тестирование DAL (Drivers, DAOs etc.) и всего, что связано с получением данных из persistence layer. 9. TDD как способ изучения настоящего ООП, а не процедурного программирования с использованием синтаксиса объектов.
17:00-17:15	**Кофе-брейк**
17:15-18:30	**Круглый стол и лотерея (темы будут уточнены в первый день конференции путем анкетирования участников)**

В фокусе

Сравнение производительности Apache 1 и Apache 2

Автор: Илья Альшанецкий (Ilia Alshanetsky)
Перевод: Андрей Олищук

В ответ на возрастающий флейм по поводу стабильности и юзабильности веб-сервера Apache 2 в сравнении с Apache 1, я решил провести серию тестов для того, чтобы определить, насколько две версии веб-серверов отличаются друг от друга. Целью данной серии тестов было определение наиболее быстрого сервера для обслуживания статических HTML-страниц, сравнение качества компресии и, конечно же, выявление версии наиболее подходящей для работы с PHP-приложениями.

Более подробные результаты сравнительного тестирования будут приведены ниже. Сейчас приведу только краткое резюме выявленных результатов.

- Apache 2 примерно на 4% быстрее работает со статическими страницами;
- Модуль Apache 2 mod_deflate быстрее модуля Apache 1 mod_gzip примерно на 60% при компрессии статических страниц;
- Работа с PHP на Apache 2 медленнее Apache 1 DSO на 27% и на 31% медленнее чем Apache 1 static.

Как проводилось тестирование

Для тестирования была использована “чистая” версия PHP 4.3.10, скомпилированная с обычными флагами “- -disable-cgi –-disable-cli –without-pear”. Настройки sapi варьировались между: –-with-apxs (Apache 1 DSO), – -with-apxs2 (Apache 2 DSO apache2handler), – -with-apache (Apache 1 static).

Работа с PHP на Apache2 медленнее чем на Apache1

Apache 1 был скомпилирован с флагами “- -enable-module=so” и “- -enable-module=rewrite”. Для статической сборки (static build) было также добавлено “- -activate-module=src/modules/php4/libphp4.a”. Apache 2 был скомпилирован с опциями: “- -with-mpm=prefork – -enable-rewrite – -enable-so –enable-deflate”. При компиляции всего программного обеспечения использовалась опция оптимизации -O2. Apache специально не переконфигурировался – была использована стандартная конфигурация. Изменения конфигурации веб-сервера коснулись только подключения PHP и mod_gzip (для Apache 1). Apache 2 был сконфигурирован на “слушание” порта 8080.

Сервер-”жертва” имел следующую конфигурацию: Celeron 1 Ghz, 768 Mb RAM, ОС: Linux 2.6.8-rc2. Для каждого теста было выполнено 10 запусков, каждый из которых выполнял примерно 10000 запросов. Concurrency level варьировался от 10 до 100 запросов. Для выполнения тестов использовалось программное обеспечение ab (apache bench) ревизии 1.73.

Тексты тестовых файлов можно найти в приложении к журналу.

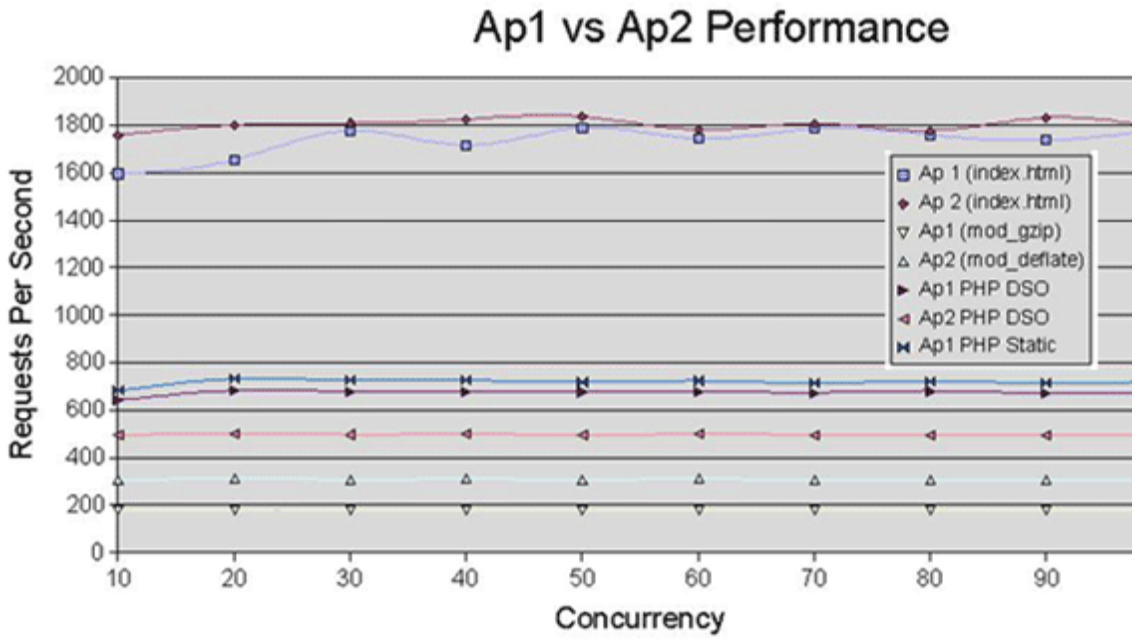
Результаты тестирования

В таблице 1 приведены результаты тестирования.

Таблица 1. Результаты тестирования

Concurrency Level	Ap 1 (index.html)	Ap 2 (index.html)	Ap1 (mod_gzip)	Ap2 (mod_deflate)	Ap1 PHP DSO	Ap2 PHP DSO	Ap1 PHP Static
10	1595,41	1758,4	177,51	307,03	642,18	494,8	683,39
20	1655,36	1800,83	179,71	310,95	682,73	502,79	731,85
30	1777,78	1808,65	179,97	308,88	679,07	497,96	724,8
40	1716,44	1825,48	178,8	309,82	680,78	499,28	725,32
50	1789,55	1837,22	179,89	308,49	675,49	496,4	717,67
60	1744,29	1780,63	180,98	309,92	678,29	499,48	723,8
70	1785,71	1807,66	180,41	307,59	674,22	496,99	714,64
80	1759,01	1777,46	179,3	307,87	681,25	495,52	722,91
90	1737,02	1832,51	178,59	307,09	670,65	493,34	712,96
100	1783,48	1788,91	178,36	307,27	671,28	498,65	721,71

В графическом представлении они выглядят примерно следующим образом:



Оригинал статьи находится по адресу: http://ilia.ws/archives/32-Apache-1-vs-Apache-2-Performance.html

CVS против VSS

Авторы:
http://www.pushok.com/
Перевод: Андрей Олищук

В этой статье вашему вниманию будет предложена таблица со сравнительными характеристиками систем контроля версий CVS и MS Visual SourceSafe. Вся приведенная ниже информация является исключительно нашей точкой зрения. Возможно мы будем относиться к MS Visual SourceSafe с предубеждением, так как очень любим CVS, однако, мы перешли на CVS не так давно, а перед этим долгое время использовали именно MS Visual SourceSafe.

Специализация нашего бизнеса – оффшорная разработка программного обеспечения и при сравнении двух программных продуктов многое определялось нашей спецификой. Мы выбрали CVS для работы и довольны своим выбором, так как CVS полностью отвечает нашим требованиям (удаленный доступ к репозиторию, ветвление, система уведомлений, надежность и цена). Мы представляем собой небольшую группу разработчиков и поэтому вопрос цены был одним из самых важных для нас, а цена CVS равна нулю. Однако прелесть CVS не только в цене. Сейчас мы полностью доверяем CVS как системе контроля исходных кодов. Итак, нужно ли платить больше?

Сейчас мы полностью доверяем CVS как системе контроля исходных кодов

№	Критерий	CVS	VSS	CVS\VSS
1	Доступ к репозиторию	CVS построен на базе клиент-серверной платформы. Доступ к репозиторию может осуществляться через VPN, LAN или интернет. Существует несколько различных протоколов обычный/безопасный, с компрессией/без. Серверы работают как на *nix так и на Win32.	VSS в действительности не является клиент-серверным приложением. Это клиент к файловой системе. Внутри LAN (JIBC) используется механизм совместного доступа, который не очень подходит для сетей с низкой пропускной способностью. Для этого существуют дополнительные решения, которые стоят отдельную сумму.	10\4
2	Формат репозитория	Структура самого репозитория аналогична структуре проекта, хранящегося в нем. Все файлы сохраняют за собой те же имена, но к ним добавляется расширение (.v). Каждый файл содержит полную версию содержимого с добавлением информации RCS. В случае возникновения проблем их можно легко обнаружить. Формат репозитория никак не зависит от метода доступа к нему (локально или удаленно), поэтому локальный репозиторий можно в любой момент превратить в CVS-сервер и предоставлять к нему удаленный доступ.	Формат репозитория абсолютно неудобочитаем для человека. Файлы хранятся под непонятными именами, такими как *abaaaaa.aaab* и т.п. Небольшое повреждение (например вторжение вируса) может полностью уничтожить данные.	10\2

№	Критерий	CVS	VSS	CVS\VSS
3	Размер репозитория	CVS хранит разницу между строками кода для каждой ревизии. Это означает то, что добавление одной строки в код приведет к добавлению всего лишь нескольких строк в репозиторий (включая служебную информацию).	VSS хранит файлы для каждой версии целиком. Добавление одной строчки кода, приведет к добавлению новой версии, а значит и файла. Это приводит к разрастанию размеров репозитория в геометрической прогрессии.	10\2
4	Ветвления	Поддерживается реальная работа с ветвлением, в том числе возможность слияния различных веток. Это очень важно для поддержки веток кода рабочей версии и релиза.	Реальной поддержки ветвления нет. Исключение составляет только тот случай, когда один файл может существовать в нескольких проектах.	10\1
5	Параллельная работа	Поддерживается	Поддерживается	10\10
6	Монопольная работа	Монопольная работа возможна только на клиентской машине. (В последних версиях CVS существует команда unedit – прим. редакции)	Встроенная поддержка	5\10
7	Сравнения версий/история	Встроенная поддержка	Встроенная поддержка	10\10
8	Уведомления	CVS можно настроить для посылки уведомлений по e-mail при изменении кода в репозитории.	Не поддерживается	10\0
9	Доступ через Веб (веб-интерфейс)	Существуют инструменты для доступа к CVS-серверу через веб-интерфейс	Не поддерживается	10\0
10	Интеграция с системами контроля ошибок	Возможна	Не поддерживается	10\0
11	IDE	CVS де-факто является стандартом для некоторых интегрированных сред разработки	Поддерживается интеграция только через SCC API	10\7
12	Графический интерфейс	Может работать как из командной строки, так и с графическим интерфейсом, который реализован в большом количестве программных продуктов, многие из которых бесплатные и при этом доволно мощные.	В VSS есть собственный VSS Explorer, посредством которого осуществляются все операции (для административных функций существует дополнительное приложение). Никаких иных программных продуктов нет.	10\10
13	Развитие	CVS имеет давние корни и много разработчиков. Эта система работает превосходно. Необходимо иметь ввиду, что разработка постоянно продолжается и лучше не использовать альфа- или бета-версий. Используйте только стабильные релизы.	Это продукт Microsoft	8\10

Сравнение производительности MySQL на различных серверных ОС

Автор: Тони Бурк
(Tony Bourke)
Перевод: Андрей Олищук

С выходом 2.6 Linux kernel, FreeBSD-5-STABLE, Solaris 10 и теперь вот NetBSD 2.0, перед вами может встать вопрос: какая же из этих операционных систем обеспечивает наибольшую производительность для базы данных? В моей предыдущей статье (http://software.newsforge.com/article.pl?sid=04/12/27/1238216&tid=152) я рассказывал об инструментах для тестирования этих почтенных ОС и о методологии тестирования. В результате реализации описанной методологии и родилась данная статья о производительности MySQL на следующих платформах: OpenBSD 3.6; NetBSD 2.0; FreeBSD 5.3 and 4.10; Solaris Express (build 69); and Linux 2.4 and 2.6 (Gentoo-based).

Перед тем как перейти непосредственно к результатам, хочу предупредить вас, что главный вопрос на который я отвечаю в этой статье таков: “Как будет работать MySQL под этой операционной системой?” И здесь вы не найдете ответа на вопрос “Какая из операционных систем лучше всех?” В любом тесте производительности бывают победители и побежденные, однако лучшее качество в одной категории, доказанное на ограниченном числе тестов не делает победителя “Лучшей Операционной Системой”. Любая операционная система имеет свои преимущества и недостатки и один тест показывает только один недостаток или преимущество на узком участке.

Любая операционная система имеет свои преимущества и недостатки и один тест показывает только одно преимущество или недостаток на узком участке

Результаты сгруппированы по средствам тестирования.

Super Smack 1.2

Здесь тесты показали небольшую разницу. Тип таблиц MySQL – MyISAM. Super Smack вызывался со следующими параметрами:

- super-smack /usr/share/smacks/select-key.smack 10 10000
- super-smack /usr/share/smacks/update-select.smack 10 10000

На тестах select-key, NetBSD 2.0 была лучше всех. Linux 2.4 и 2.6 так же отработали хорошо, а OpenBSD 3.6 показала результаты превосходящие обе версии FreeBSD.

При переходе на два процессора результаты изменились коренным образом. Linux оказался единственной операционной сиситемой, которая серьезно улучшила свои результаты. Превосходство OpenBSD и NetBSD исчезло и были замечания к FreeBSD 4.11 с libc_r.

При тестировании update-select Linux 2.4 и 2.6 вместе с NetBSD 2.0 снова лидировали. Результаты FreeBSD 5.3 заметно снизились, а OpenBSD 3.6 и FreeBSD 4.11 отработали со средними показателями.

Как и на тестах select-key, только Linux 2.4 и 2.6 сохранили свое лидерство при переходе на два процессора. NetBSD 2.0 снова снизила свои показатели, но не очень сильно и заняла второе место после Linux.

SysBench 0.3.1 1M Rows

The SysBench 1M rows обеспечивает хороший набор тестов. С самого первого теста практически все данные были закешированы сервером MySQL, таким образом не произведя практически никаких операций с диском.

Для тестов SysBench я использовал таблицы InnoDB. SysBench вызывался со следующими параметрами:

1M Rows:

```
Установка:
sysbench --num-threads=10 --test=oltp --mysql-host=172.16.3.7 --mysql-
user=root --mysql-password=mysql --oltp-table-size=1000000 prepare
Запуск:
sysbench --num-threads=10 --test=oltp --mysql-host=172.16.3.7 --mysql-
user=root --mysql-password=mysql --oltp-table-size=1000000 run
Очистка памяти:
sysbench --num-threads=10 --test=oltp --mysql-host=172.16.3.7 --mysql-
user=root --mysql-password=mysql --oltp-table-size=1000000 cleanup
```

10M Rows:

```
Установка:
sysbench --num-threads=10 --test=oltp --mysql-host=172.16.3.7 --mysql-
user=root --mysql-password=mysql --oltp-table-size=10000000 prepare
Запуск:
sysbench --num-threads=10 --test=oltp --mysql-host=172.16.3.7 --mysql-
user=root --mysql-password=mysql --oltp-table-size=10000000 run
Очистка памяти:
sysbench --num-threads=10 --test=oltp --mysql-host=172.16.3.7 --mysql-
user=root --mysql-password=mysql --oltp-table-size=10000000 cleanup
```

В этих тестах на высоте оказались 2.4 и 2.6 Linux kernels, оставив позади Solaris 10 и FreeBSD 4.11 с linuxthreads. NetBSD 2.0 и FreeBSD 5.3 с KSE и linuxthreads показали средний результат, а FreeBSD 4.11 с libc_r и OpenBSD 3.6 были намного хуже остальных.

При добавлении второго процессора для тестов 1M rows общая картина несколько меняется.

Solaris 10 поднимается почти до уровня производительности Linux 2.4 и 2.6, а NetBSD резко ухудшает свои показатели. FreeBSD 4.11 с linuxthreads показывает относительно хорошие результаты.

SysBench 0.3.1 10M Rows

Тест SysBench OLTP с 10M rows создает InnoDB-файл в 2.6 GB, который не влезает в кеш (MySQL или операционной системы) с 512 MB памяти.

В тестах на однопроцессорных системах, Linux снова опережает всех. Solaris, FreeBSD 5.3 (linuxthreads и KSE), и FreeBSD 4.11 (linuxthreads) повторяют результаты. FreeBSD 4.11 с libc_r показывает себя хуже, так же как и OpenBSD 3.6. Настоящим сюрпризом стали плохие результаты NetBSD 2.0, хотя в остальных тестах эта операционная система показала себя вполне неплохо. На этот раз она была наихудшей в группе. Я пытался комбинировать системные настройки, но не смог добиться от ОС лучших результатов.

Масштабируемость: обзор

Так как можно зафиксировать результаты для одно- и двупроцессорных режимов, мы имеем возможность посмотреть, насколько хорошо рассматриваемые операционные системы позволяют масштабировать MySQL. Для этого можно взять дельту между результатами двух запусков, с одним процессором и с двумя, выраженную в процентах. Некоторые операции более зависят от количества процессоров нежели другие. К примеру, операции чтения конечно же получают наибольший прирост производительности, так как для них не производится блокирования таблиц (table-locking), однако это преимущество заметно лишь в том случае, если данные были занесены в кэш. Операции записи в свою очередь ограничены блокировками таблиц и скоростью записи логов, которая зависит уже от характеристик диска. Поэтому для таких операций процессорная масштабируемость в любом случае несколько ограничена.

Масштабируемость: Super Smack

При прохождении select-key тестов очень хорошо себя зарекомендовали Linux 2.4 и 2.6, которые практически удваивали свою производительность при добавлении второго процессора. FreeBSD 5.3 с KSE сработала тоже довольно неплохо. При этом FreeBSD 5.3 с linuxthreads и FreeBSD 4.11 с linuxthreads показали себя средне. FreeBSD 4.11 с libc_r had не увеличили своей производительности при добавлении второго процессора. OpenBSD 3.6, NetBSD 2.0 и NetBSD 2.0 показали даже некоторое допустимое ухудшение производительности.

Тесты update-select показали более худшие результаты для всех систем. Это прежде всего обосновано их природой операций записи. Не смотря ни на что, Linux снова показал лучший прирост. FreeBSD 5.3 и с linuxthreads и с KSE были на уровне, а вот FreeBSD 4.11 с linuxthreads удивила практическим отсутствием улучшений. NetBSD 2.0 снова показала снижение производительности, как и OpenBSD.

Масштабируемость: SysBench 1M Rows

SysBench OLTP тесты 10M rows создают 2.6 GB InnoDB файл, который не был закеширован (MySQL или самой ОС) на 512 MB физической памяти. В результате возникли большие проблемы ввода/вывода и количество выполненых транзакций за период времени оказалось не очень хорошим.

Для однопроцессорных тестов Linux снова опередил всех. Solaris, FreeBSD 5.3 (linuxthreads и KSE), и FreeBSD 4.11 (linuxthreads) показали примерно те же результаты. FreeBSD 4.11 с libc_r отработала хуже, как и OpenBSD 3.6. Настоящим негативным сюрпризом стала NetBSD 2.0. Она показала неплохие результаты на других тестах, но на тесте с проблемами ввода/вывода, результаты были худшими в группе. Я испробовал различные тонкие настройки файловой системы, но количество транзакций за период времени так существенно и не возросло.

Отдельно о Solaris

Я натолкнулся на странную тенденцию Solaris 10 при 10M row тестах SysBench. Эта ОС показывала хорошие результаты на 1M тестах, но на 10M тестах очень резко снизила свои показатели приблизительно до 3.6 транзакций в секунду, что хуже даже чем у NetBSD 2.0. Это фактически седьмая часть от результатов Linux, что абсолютно не впечатляет. Я связался с Питером Зайцевым (Peter Zaitsev) и он организовал нам переписку с Дженни Чен (Jenny Chen) из SUN с которой мы попытались выявить причину такого резкого снижения производительности.

Чен рекомендовала монтировать файловую систему без записи логов и устанавливать sticky bit (chmod +t) на файлы с данными InnoDB и логи. Ни один из этих шагов не помог и я продолжил свои изыскания.

Linux 2.4 и 2.6 показали себя лучше всех на протяжении полного набора тестов

Окончательный ответ я нашел в легендарной книге Адриана Коккрофта (Adrian Cockcroft) и Ричарда Петтита (Richard Pettit) “Sun Performance Tuning: Java and the Internet”. Решением стало монтирование раздела (data partition) с опцией forcedirectio. Эта опция предотвращала кеширование данных самой ОС и когда я запустил тест, swap-диск оказался без работы и результаты впечатлили: 21 транзакция в секунду против прежних 3.6.

Как это ни странно, но установка sticky bit снижала производительность примерно на одну транзакцию в секунду. У меня получилось примерно 20 с +t и 21 без +t. Отключение логов при монтировании файловой системы тоже никак не повлияло на производительность. Опция noatime также не имела эффекта.

Хотя forcedirectio не упоминается в документации MySQL, я обнаружил, что опция directio специально рекомендуется на странице 161 книги “ Sun Performance Tuning: Java and the Internet” для операций с большими файлами (файл данных InnoDB “весил” 2.6 GB при тестах 10M rows). Эта книга была написана в 1998 году и в ней рассматривался Solaris 2.6. На другие операционные системы, рассматриваемые в этой статье, данный сценарий никак не повлиял. NetBSD 2.0 (так же показавшая плохие результаты) не показывала никакой активности swap-диска при тестах 10M rows.

Выводы

Оба Linux 2.4 и 2.6 показали себя лучше всех на протяжении полного набора тестов, доминируя на всех тестах вне зависимости от нагрузок. Их масштабируемость так же была превосходной, что подтвердилось при добавлении дополнительного процессора.

Я был удивлен работой Linux 2.4, так как ожидал заметного превосходства Linux 2.6, однако эти два ядра от теста к тесту попеременно опережали друг друга.

Solaris 10 показал себя с лучшей стороны и по скорости и по масштабируемости. По моему мнению, можно утверждать, что Solaris 10 является мощной платформой для MySQL. Правда я не смог найти порта Super Smack на эту платформу и эти тесты не проводились, так что имейте это в виду.

NetBSD 2.0 так же была вполне сильной, хотя на двух направлениях и подкачала. Во-первых, MySQL на NetBSD 2.0 не увеличила производительность при добавлении дополнительного процессора. Возможно, во всех случаях стоит использовать однопроцессорное ядро этой ОС, даже не смотря на то, что в системе могут присутствовать два процессора. Во-вторых, плохая производительность ввода/вывода на тестах SysBench 10M row. Если проблемы с несколькими процессорами вполне понятны – это первый релиз этой ОС поддерживающий мультипроцессорность, то неполадки с вводом/выводом остались для меня загадкой.

FreeBSD 5.3 отработала относительно хорошо в обоих режимах KSE и linuxthreads, хотя я немного разочаровался ее результатами. Все же, тесты показали что “родная” поточная модель (threading model) FreeBSD-5 готова к полноценному использованию и, по крайней мере для работы с MySQL, может заменить долгоиграющую FreeBSD с linuxthreads.

Однако FreeBSD 4.11 модель linuxthreads определенно помогла с производительностью (иногда даже обгоняя FreeBSD 5.3). С libc_r производительность отставала по многим тестам и показала не очень хорошую масштабируемость, поэтому я настоятельно рекомендовал бы использовать FreeBSD 4.11 с linuxthreads для работы MySQL.

Я считаю, что время потраченное на тесты стоило того. Я больше узнал о производительности MySQL в целом и хотел бы поблагодарить Питера Зайцева за его советы по методологии и также Дженни Чен из SUN за ее вклад.

Идеи

Многоязычные приложения с использованием PHP и GetText

Автор: Луис Аргерич (Luis Argerich)
Перевод: Кузьма Феськов
http://www.russofile.ru/kuzma/

Если вы веб-разработчик, который создает как небольшие ресурсы, так и большие порталы с сотнями страниц, то в один прекрасный момент вы можете столкнуться с необходимостью перевода вашего проекта на другой язык, или обеспечить поддержку нескольких языков в рамках одного проекта.

Перевод подразумевает под собой только перевод строк в коде. Вы можете подойти к этому с умом, а можете просто просматривать весь код и переводить в нем все строки. Последний метод достаточно болезнен и порождает много ошибок, к тому же, такой код – это всего лишь заплатка, а не код для многократного использования. Интернационализация помогает построить стратегию производства сайтов так, чтобы вы всегда легко могли перевести ваши файлы на другой язык. Именно описание такого подхода – цель этой статьи.

Добро пожаловать в GNU

GNU имеет в своем арсенале очень хороший набор инструментов для производства многоязычных приложений. Этот набор называется GetText. Он был разработан для языков C и C++, но так же прекрасно подходит для других языков, типа PHP.

Возможно, GetText уже установлен на вашей UNIX системе. Чтобы проверить это, наберите в командной строке getetxt и нажмите Ввод (Enter). Если указанного набора компонентов нет, возьмите его, например, здесь: ftp.gnu.org. Произведите установку.

Если gettext у вас уже установлен, то мы переходим к следующей части нашей статьи.

От переводчика

Если вы используете в своей работе не только Unix / Linux -системы, но так же и Windows, вам не стоит отчаиваться, так как далее я опишу шаги, которые потребуется проделать для установки GetText на вашей Windows-системе.

Вам необходимо скачать файлы с этой страницы: http://sourceforge.net/projects/gettext. Для работы вам понадобится gettext-win32 и пакет libconv-win32.

После того, как вы скачали указанные выше файлы, проделайте следующие манипуляции:

1) скопируйте файлы intl.dll, gettext.exe, asprintf.dll, envsubst.exe, ngettext.exe из пакета gettext-win32 в папку SYSTEM32 вашей Windows. Затем, проделайте тоже самое с файлами charset.dll, iconv.dll, iconv.exe из пакета libconv-win32.

2) Далее найдите файл libintl-1.dll (он находится в папке dlls вашей инсталляции PHP) и поместите его так же в папку SYSTEM32.

3) И, наконец, откройте для редактирования ваш php.ini и раскомментируйте в нем строку extension=php_gettext.dll.

Перезапустите веб-сервер Apache и выполните команду phpinfo(). Если вы все сделали правильно, то PHP сообщит вам о том, что модуль GetText установлен и enabled (включен).

Интернационализация PHP скриптов

Чтобы использовать GetText в ваших приложениях, написанных на PHP, вы должны будете изменить все ваши скрипты, однако, это необходимо будет сделать всего один раз. Модификации не очень сложны и вы, возможно, даже захотите написать какую-нибудь утилиту, которая будет сканировать код и задавать вопрос типа: “хотите изменить это?” и так далее.

Итак, что же вы должны сделать со своим кодом? Все очень просто, вы всего лишь должны будете заменить все строки (string) на подобные:

```
<?php
print (_("Привет мир"));
echo(_(“Привет мир”);
return _(“Привет мир”);
?>
```

и так далее (примеры расширены переводчиком).

Обратите внимание, что строки (string) следует заключать в двойные кавычки, иначе вспомогательные утилиты не смогут вычленить текстовые строки в ваших исходных кодах (прим. переводчика).

Да, вы используете неизвестную функцию PHP “подчеркивание” (function “underscore”), которая является псевдонимом к “длинной-чтобы-писать” функции GetText (function gettext). Каждая строка, которую вы хотите перевести, должна быть переведена. Как только вы привыкните к такому способу, вы начнете видеть его преимущества. Всегда пишите ваши строки (string) обернутыми в конструкцию _(“string”); (как говорилось выше, можно писать и так: gettext(“string”); (прим. переводчика).

Извлечение строк из кода

GetText имеет в своем арсенале утилиту xgettext, которая предназначена для извлечения всех “обернутых” строк из ваших исходных кодов. В результате своей работы она создает файл с расширением .po.

Чтобы создать .po файл из своих исходников, введите:

```
$xgettext -a src/*.php
```

Чтобы узнать о возможностях этой утилиты подробнее, обратитесь к документации.

В результате работы утилиты вы получили .po файл, который должен быть похож на этот:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR Free Software Foundation, Inc.
# FIRST AUTHOR, YEAR.
#
#, fuzzy
msgid
msgstr
«Project-Id-Version: PACKAGE VERSION\n»
«POT-Creation-Date: 2000–12–08 19:15-0300\n»
«PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n»
«Last-Translator: FULL NAME \n»
«Language-Team: LANGUAGE \n»
«MIME-Version: 1.0\n»
«Content-Type: text/plain; charset=CHARSET\n»
«Content-Transfer-Encoding: ENCODING\n»

#: prueba.php:12
msgid “Hello world”
msgstr

#: prueba.php:12 prueba.php:13
msgid "
"
msgstr

#: prueba.php:13
msgid “This is a test”
msgstr
```

Это и есть тот файл, который вы дожны отдать переводчикам. Переводчик должен будет дать значение всем msgstr для конкретного языка. Вы можете сделать в файле комментарии, чтобы дать переводчику контекст или какие-либо пояснения. Для этого используйте знак #.

Например:

```
#: prueba.php:12
```

Тем временем...

Для того, чтобы Smarty нормально функционировал на сервере с PHP, работающим в безопасном режиме (SAFE MODE), необходимо в файле Smarty.class.php установить значение переменной $use_sub_dirs в FALSE.

Если этого не сделать, то возможна ошибка “Safe mode restriction in effect”, так как создание директории с правами доступа 0777 будет запрещено, но Smarty будт все равно пытаться создавать.

Если есть необходимость, чтобы файлы с расширением .htm (как и любые другие) обрабатывались как PHP-файлы, необходимо в httpd.conf прописать:
AddType application/x-httpd-php .htm

Если доступа к httpd.conf нет, то остается вариант с использованием файла .htaccess с теми же настройками.

This is displayed at the beginning of the script (это выводится в начале скрипта)

msgid "Hello world"

msgstr

И так далее. Теперь вы имеете “мастер” .po файл для всех “обернутых” строк в ваших скриптах. Если вы будете использовать gettext для непереведенных строк, то в качестве результата будет выдаваться строка из msgid, что является очень хорошим свойством системы.

ОТ ПЕРЕВОДЧИКА.

Для Windows-систем существует специализированный бесплатный редактор .po файлов, который позволяет облегчить работу переводчиков и упростить получение переведенных файлов. Для детальной информации обратитесь на сайт http://poedit.sourceforge.net/. Редактор поддерживает множество языков интерфейса, в том числе и русский язык.

Для полноценной работы с .po файлами, необходимо заполнить поля в его заголовке:

«Project-Id-Version: PACKAGE VERSION\n» – Замените PACKAGE VERSION названием своего продукта.

«Last-Translator: FULL NAME \n» – Замените FULL NAME на имя, фамилию переводчика и его e-mail-адрес.

«Language-Team: LANGUAGE \n» – Замените LANGUAGE на название языка, на который переводится приложение (например, Russian).

«Content-Type: text/plain; charset=CHARSET\n» – Замените CHARSET на точное название кодировки, в которой будут переводимые строки (например, UTF-8, Windows-1251).

Создание .mo файлов

.mo файл – это компилированный для использования с GetText .po файл. Вы должны создать .mo файл для каждого, поддерживаемого вашей системой, языка. Делается он таким способом:

```
$msgfmt messages.po -o messages.mo
```

Редактор poedit при сохранении переведенного .po файла автоматически генерирует .mo файл, что создает дополнительные удобства от его использования (прим. переводчика).

Создание каталогов

Лучший способ для использования gettext – это создание в корне директорий вашего продукта папки locale. Внутри этой директории необходимо создать поддиректории для всех поддерживаемых системой языков. Внутри них необходимо создать директорию LC_MESSAGES, куда, собственно, вы и положите имеющиеся у вас к данному моменту .mo файлы.

Приведу пример дерева папок:

/src

/locale/

en/LC_MESSAGES/messages.mo

messages.po

ru/LC_MESSAGES/messages.po

messages.mo

Чтобы найти буквенные сочетания для всех возможных языков, воспользуйтесь этой ссылкой: http://www.loc.gov/standards/iso639-2/langcodes.html (оригинальная ссылка автора статьи была перемещена на другое место. Здесь указана новая ссылка, прим. переводчика), либо документацией на GetText. Обращаю ваше внимание, что в официальной документации на GetText есть ошибки в буквенных сочетаниях (прим. переводчика).

Итак, вы создали дерево директорий и поместили в них необходимые файлы с языковыми ресурсами (.mo файлы). Что же необходимо сделать в ваших PHP скриптах, чтобы все это начало работать? Примеры несколько расширены и изменены, поскольку в оригинальной статье есть некоторые неточности. Эта часть статьи очень сильно отличается от оригинала (прим. переводчика).

В соответствии с пожеланиями пользователя, включаем выбранный им язык приложения:

```
// Задать русский язык текущим
putenv («LC_ALL=ru_RU»);

//или
// Задать английский язык текущим
putenv («LC_ALL=en_US»);
```

Далее указываем текущие настройки локали:

```
// Задать русскую кодировку Windows-1251 (если у вас Windows)
//или KOI-8 (если у вас Linux)
setlocale (LC_ALL, "Russian");

//или
// Задать анлийскую кодировку Windows-1252 (если у вас Windows)
setlocale (LC_ALL, "English");
```

Все сообщения вашего приложения будут выводиться в указанной здесь кодировке. Если вы, по каким-либо причинам, хотите выводить сообщения в другой кодировке, скажем в UTF-8, то в PHP есть команда, которая принудительно указывает gettext в какой кодировке вы хотите получать сообщения:

```
bind_textdomain_codeset($domain, 'UTF-8');
```

Значение переменной $domain я объясню ниже.

Теперь следует указать домен (корневую папку), в котором у нас содержатся все языки проекта.

```
$domain = 'my_site';
bindtextdomain ($domain, «./locale»);
```

“my_site”, в данном случае, название нашего приложения. Замечу, что все файлы с переводом должны иметь название вида my_site.mo, иначе gettext их не увидит.

Как вы могли заметить, у нас появилась та самая переменная $domain, о которой я говорил выше. Она содержит название текущего домена. Доменов может быть несколько. Это нужно в том случае, если ваше приложение состоит из нескольких независимых частей.

Далее мы выбираем текущий домен, тем самым указывая gettext откуда брать переведенные сообщения:

```
textdomain ($domain);
```

Итак, на этом настройки gettext в вашем скрипте закончены. Приведу листинг полностью, в порядке следования действий:

```
// Задаем текущий язык проекта
putenv(«LANG=ru_RU»);

// Задаем текущую локаль (кодировку)
setlocale (LC_ALL,"English");

// Указываем имя домена
$domain = 'my_site';

// Задаем каталог домена, где содержатся переводы
bindtextdomain ($domain, «./locale»);

// Выбираем домен для работы
textdomain ($domain);

/* Если необходимо, принудительно указываем кодировку (эта строка не
обязательна, она нужна, если вы хотите выводить текст в отличной от
текущей локали кодировке).*/
bind_textdomain_codeset($domain, 'UTF-8');
```

Ну что ж, остается только проверить, все ли правильно мы сделали. Для этого установите нужный вам язык и запустите свое приложение. Все сообщения должны выводить на заданном вами языке. Если этого не произошло, то где-то вкралась ошибка: либо вы не перевели все сообщения, либо неправильно создали дерево каталогов с переводами, возможно, неправильно подобрали буквенное сочетание для нужного вам языка. В общем, вариантов много. Надеюсь все же, что все у вас заработало!

Какая полезная вещь эти свойства...

Автор: Артур Гатин

Данная статья научит вас обращаться со свойствами и правильно их применять. Если вы имеете представление об объектно-ориентированном программировании, то это хорошо. Если же нет, то не расстраивайтесь и попробуйте понять. Если сразу у вас что-то не получиться, в этом нет ничего страшного.

У классов в ООП (объектно-ориентированном программировании) имеются поля и методы. Очень часто методы класса являются обработчиками его полей. Так вот, конструкция, осуществляющая доступ к полю класса через методы чтения и записи, называется свойством. Преимущества использования свойств в ООП будут очевидны после следующих примеров.

В PHP 5.0.0 есть специальные возможности для реализации свойств. Это функции для их чтения и записи: set() и get(). Данные функции описываются внутри класса и должны иметь именно эти имена, т.к. они являются для них зарезервированными. Вызываются эти функции, если происходит доступ к несуществующему полю объекта. Данная статья поможет правильно описывать и реализовывать эти функции в PHP 5.0.0. Рассмотрение способов реализации будем осуществлять на примерах.

Когда-то надо начать...

Пример 1

Начало описания класса:

Опишем класс TextMessage, который у нас будет выдавать некоторое текстовое сообщение. В нем опишем поле $Message, которое у нас будет хранить строку (текст сообщения), а также опишем поле $Font, которое будет являться массивом и будет хранить характеристики шрифта (цвет и размер). В PHP 5.0.0 мы можем использовать очень важное преимущество ООП – области видимости. Области видимости используются тогда, когда нам необходимо ограничивать доступ к описанному элементу класса. В данном примере поле $Message не имеет ограничений на доступ, т.е. к нему мы можем получить прямой доступ из любого места программы, где создан экземпляр описанного нами класса TextMessage. К полю же $Font, которое является массивом и хранит у нас характеристики шрифта, мы будем обращаться лишь внутри класса для вывода сообщения. С учетом этого мы должны поместить поле $Font в область видимости private:

```
<?php
class TextMessage
{
  var $Message = 'Ура, ТОВАРИЩИ!';
  private $Font = array('Size' => 10, 'Color' => 'blue');
}
?>
```

Мы видим, что в переменной $Message будет первоначально храниться строка 'Ура, ТОВАРИЩИ!'. У массива $Font есть элементы с индексами 'Size' и 'Color', содержащие соответственно размер 10 и цвет 'blue' для шрифта по умолчанию. При этом внутри программы мы будем работать с элементами массива $Font не напрямую, а через специальные функции чтения и записи. Следовательно, нам необходимо свойство.

Доступ к полю $Font будем осуществлять именно при помощи функций set() и get(). Преимущество такого подхода будет наглядно продемонстрировано на примере. Функция set() является функцией для записи свойства, т.е. у нас она будет менять характеристики шрифта сообщения в массиве $Font, а функция get() является функцией для чтения свойства, т.е. она будет считывать характеристики шрифта сообщения из массива $Font. Начнем с функции __get().

Функция __get()

Функция __get() по синтаксису должна иметь один параметр: имя считываемого свойства. Идентификатор параметра может иметь отличные от примера названия. Здесь ограничений нет (например, вместо $prop_name можно использовать $name и др.):

```
function __get($prop_name)
{
}
```

Сама функция должна вернуть считанное значение свойства с именем $prop_name. В нашем примере именем свойства должен быть один из индексов массива $Font, поэтому перед считыванием проверяем, есть ли у нас в массиве $Font элемент с индексом $prop_name. Если 'да', то в возвращаемом значении элемента с индексом $prop_name из массива $Font:

```
function __get($prop_name)
{
if (isset($this->Font[$prop_name]))
{
return $this -> Font[$prop_name];

}

}
```

В теле функции мы можем выполнять некоторые действия или инициализацию (например, увеличение какого-то глобального счетчика, изменение значения какого-то поля и др.). Вот вам налицо одно из преимуществ, получаемых при использовании свойств. Если бы доступ к полю производился напрямую, то инициализацию пришлось бы оформлять отдельно. Для примера будем выводить сообщение о считывании свойства:

```
echo 'Мы считали свойство <b>'.$prop_name.'!</b> <br>';
```

Таким образом, общий вид нашей функции будет следующим:

```
<?php
  function __get($prop_name)
  {
    if (isset($this->Font[$prop_name]))
    {
      echo 'Мы считали свойство <b>'.$prop_name.'!</b> <br>';
      return $this -> Font[$prop_name];
    }
  }
?>
```

Мы реализовали функцию считывания свойства. Теперь нам необходимо реализовать функцию, устанавливающую свойство. Для этого мы должны использовать функцию __set().

Функция __set()

В отличие от функции __get(), она будет иметь два параметра: имя свойства и выводимое значение. Это будет выглядеть примерно так:

```
function __set($prop_name, $prop_value)
{
}
```

Сама функция должна вернуть результат выполненной операции: если установка нового значения $prop_value вместо старого значения свойства с именем $prop_name произошла удачно, то возвращаем истину, иначе – ложь. В случае, если возвращается ложь, то значение свойства не изменяется. В нашем примере имя свойства $prop_name должен быть один из индексов массива $Font. Проверяем аналогично, как и для функции __get():

```
if (isset($this->Font[$prop_name]))
```

Хотелось бы подчеркнуть, что в элементе с индексом Color в массиве $Font должна лежать строка, а в элементе с индексом Size в массиве $Font должно лежать целое число. Поэтому перед установкой свойства мы должны проверить: если у нас значение переменной $prop_name (индекс элемента в массиве $Font) равно 'Color', то значение переменной $prop_value должно быть строкой. Эта проверка будет выглядеть так:

```
($prop_name == 'Color')and(is_string($prop_value)
```

Аналогично: если у нас значение переменной $prop_name (индекс элемента в массиве $Font) равно 'Size', то значение переменной $prop_value должно быть целым числом. Тогда получим:

```
($prop_name == 'Size')and(is_int($prop_value))
```

Если одно из вышеописанных условий соблюдается, то в элемент с индексом $prop_name из массива $Font мы помещаем значение переменной $prop_value:

```
function __set($prop_name, $prop_value)
{
if (isset($this->Font[$prop_name]))
{
if 'Color')and(is_string($prop_value
or($prop_name == 'Size')and(is_int($prop_value)))

{

$this -> Font[$prop_name] = $prop_value;

}

}

}
```

В теле функции мы также можем выполнять некоторые действия или инициализацию (например, увеличение какого-то глобального счетчика, изменение значения какого-то поля и др.). Для примера будем выводить сообщение об удачной установке свойства:

```
echo 'Мы установили свойство <b>'.$prop_name.'!</b> <br>';
```

Далее если все произошло удачно, то как значение функции мы возвращаем 'истину', иначе возвращаем 'ложь'. Таким образом, общий вид нашей функции будет следующим:

```
function __set($prop_name, $prop_value)
  {
    if (isset($this->Font[$prop_name]))
    {
      if (($prop_name == 'Color')and(is_string($prop_value))
        or($prop_name == 'Size')and(is_int($prop_value)))
      {
        $this -> Font[$prop_name] = $prop_value;
        echo 'Мы установили свойство <b>'.$prop_name.'!</b> <br>';
        return true;
      }
    } else
      {
        return false;
      }
  }
```

Функция вывода сообщения

Теперь нам необходимо вывести наше сообщение. Опишем функцию Show TextMessage:

```
function Show TextMessage()
{
echo '<font color="'.$this -> Color.'" size="'.$this -> Size.'">'.$this ->
Message.'</font><br>';

}
```

Проанализируем что она выполняет. Функция выводит строку $Message с отформатированным шрифтом цвета $Font['Color'] и размером $Font['Size']. Видно, что считывание этих значений будет осуществляться с помощью функции get(), т.к. в строчке происходит обращение к несуществующим полям класса: Color и Size. Названия этих полей будут подставляться вместо параметра $prop_name в функцию get().

А что у нас получилось?

Таким образом, общее описание класса будет выглядеть так:

```
<?php

class TextMessage
{
  //Поле, отвечающее за сообщение
  var $Message = 'Ура, ТОВАРИЩИ!';
  //Поле, отвечающее за свойства шрифта
  private $Font = array('Size' => 10, 'Color' => 'cyan');

  //Функция считывания свойства
  function __get($prop_name)
  {
    //Проверка существования элемента с индексом $prop_name в массиве $Font
    if (isset($this->Font[$prop_name]))
    {
      //Вывод сообщения о совершении операции
      echo 'Мы считали свойство <b>'.$prop_name.'!</b> <br>';
      //Вовращение значения элемента с индексом $prop_name из массива $Font
      //как значение функции
      return $this -> Font[$prop_name];
    }
  }
//Функция установки свойства
  function __set($prop_name, $prop_value)
  {
    //Проверка существования элемента с индексом $prop_name в массиве $Font
    if (isset($this->Font[$prop_name]))
    {
      //Проверка переменных:
      //для $prop_name, равной Color, $prop_value - строка
      if (($prop_name == 'Color')and(is_string($prop_value))
        //для $prop_name, равной Size, $prop_value - целое число
        or($prop_name == 'Size')and(is_int($prop_value)))

Листинг 1. Продолжение на следующей странице
```

```
{
        //Присваивание элементу с индексом $prop_name в массиве $Font
        //значение переменной $prop_value
        $this -> Font[$prop_name] = $prop_value;
        //Вывод сообщения о совершенной операции
        echo 'Мы установили свойство <b>'.$prop_name.'!</b> <br>';
        //Вовврат значения функции об удачной установке свойства
        return true;
      }
  } else
      {
        //Вовврат значения функции о неудачной установке свойства
        return false;
      }
  }

  //Функция вывода сообщения $Message
  //с цветом шрифта $Font['Color'] и размером $Font['Size']
  function ShowTextMessage()
  {
    echo '<font color="'.$this -> Color.'" size="'.$this -> Size.
         '">'.$this -> Message.'</font><br>';
  }
}
?>
```

Запуск программы.

После описания класса мы можем уже создать экземпляр класса и работать с ним. Рассмотрим как это будет происходить. Для начала создадим объект класса TextMessage:

```
$Message = new TextMessage;
```

Далее установим размер шрифта следующим образом:

```
$Message -> Size = 5;
```

Проанализируем эту запись. У объекта $Message так же, как и у класса TextMessage, нет поля Size. Значит для обработки этой команды будет вызвана функция __set(). Первый параметр $prop_name у этой функции будет равен 'Size', а второй параметр $prop_value примет целое значение 5. В теле функции мы проверяе условие: если в массиве $Font существует элемент с индексом $prop_name, то для $prop_name со значением 'Size' значение параметра $prop_value должно являться целым числом. Все эти условия соблюдаются. Значит элемент массива $Font с индексом 'Size' примет значение 5, т.е. произойдет следующее присваивание: $Font['Size'] = 5.

Аналогичные рассуждения будут при анализе следующей строчки:

```
$Message -> Color = '115599';
```

Из вышеприведенных рассуждений легко понять, почему будут игнорироваться следующие строчки:

```
$Message -> Size = '10';
$Message -> Color = 665599;
```

Иными словами, результатом работы следующего фрагмента программы

```
$Message = new TextMessage;
$Message -> Size = 5;
$Message -> Color = '115599';
$Message -> Show TextMessage();
$Message -> Size = '10';
$Message -> Color = 665599;
$Message -> Show TextMessage();
```

является следующее:

Мы установили свойство **Size**!

Мы установили свойство **Color**!

Мы считали свойство **Color**!

Мы считали свойство **Size**!

Ура, ТОВАРИЩИ!

Мы считали свойство **Color**!

Мы считали свойство **Size**!

Ура, ТОВАРИЩИ!

Пример 2

Можно попробовать переделать наш класс немного по-другому. Перепишем функцию __set(). Функция также будет иметь два параметра: $prop_name и $prop_value. Функция будет работать так: если значением параметра $prop_value является строка, то положим это значение в элемент массива $Font с индексом 'Color'. Если его значение будет целым числом, то положим это значение в элемент массива $Font с индексом 'Size'. Тогда функция примет вид:

```
<?php
  function __set($prop_name, $prop_value)
  {
    //Проверка значения $prop_value как целого числа
    if (is_int($prop_value))
    {
      //Присваивание элементу с индексом Size в массиве $Font
      //значение целой переменной $prop_value
      $this -> Font['Size'] = $prop_value;
//Вывод сообщения о совершенной операции
      echo 'Мы установили свойство <b>Size!</b> <br>';
      //Возврат значения функции об удачной установке свойства
      return true;
    }

Листинг 2. Продолжение листинга на следующей странице
```

```
else
      {
        //Проверка значения $prop_value как строки
        if (is_string($prop_value))
        {
          //Присваивание элементу с индексом Color в массиве $Font
          //значения переменной $prop_value как строки
          $this -> Font['Color'] = $prop_value;
          //Вывод сообщения о совершенной операции
          echo 'Мы установили свойство <b>Color!</b> <br>';
          //Вовврат значения функции об удачной установке свойства
          return true;
        } else
          {
            //Вовврат значения функции о неудачной установке свойства
            return false;
          }
      }
  }
```

Из данной реализации функции __get() можно понять, почему результат выполнения данного фрагмента программы

```
$Message -> Size = 5;
$Message -> Color = '115599';
$Message -> Show TextMessage();
$Message -> Razmer = 6;
$Message -> Tzvet = 'red';
$Message -> Show TextMessage();
```

будет таким:

Мы установили свойство **Size**!

Мы установили свойство **Color**!

Мы считали свойство **Color**!

Мы считали свойство **Size**!

Ура, ТОВАРИЩИ!

Мы установили свойство **Size**!

Мы установили свойство **Color**!

Мы считали свойство **Color**!

Мы считали свойство **Size**!

Ура, ТОВАРИЩИ!

Немного о реализации свойств в предыдущих версиях PHP

Начиная с версии 4.3.4 в PHP осваивается новый способ реализации свойств. Функции имели те же названия: set() и get(), но синтаксис у функции get() был другой. Она имела два параметра, как и функция set(). Первый являлся именем свойства, второй – его значением. Функция возвращала истину или ложь.

Выводимое функцией значение (т.е. значение запрашиваемого свойства) необходимо было вставить во второй параметр. Реализация функции __get() из второго примера выглядела бы так:

```
<?php
  function __get($prop_name, $prop_value)
  {
    //Проверка существования элемента с индексом $prop_name в массиве
$Font
    if (isset($this->Font[$prop_name]))
    {
      //Присваивание переменной $prop_value значения
      //элемента с индексом $prop_name из массива $Font
      $prop_value = $this -> Font[$prop_name];
      //Вывод сообщения о совершенной операции
      echo 'Мы считали свойство <b>'.$prop_name.'!</b> <br>';
      //Вовврат значения функции об удачном считывании свойства
      return true;
    } else
      {
        //Вовврат значения функции о неудачном считывании свойства
        return false;
      }
  }
```

Еще одно очень важное замечание. Чтобы функции set() и get() работали, необходимо для класса вызвать функцию overload(). При этом результат работы следующего фрагмента программы

```
overload('TextMessage');
Message = new TextMessage;
$Message -> Size = 5;
$Message -> Color = '115599';
$Message -> Show TextMessage();
$Message -> Size = '10';
$Message -> Color = 559900;
$Message -> Show TextMessage();
```

был бы таким:

Мы установили свойство **Size**!

Мы установили свойство **Color**!

Мы считали свойство **Color**!

Мы считали свойство **Size**!

Ура, ТОВАРИЩИ!

Warning: Unable to set property: Size in C:\Documents and Settings\Art\Рабочий стол\For PHP 434.php? on line 81

Warning: Unable to set property: Color in C:\Documents and Settings\Art\Рабочий стол\For PHP 434.php? on line 82

Мы считали свойство **Color**!

Мы считали свойство **Size**!

Ура, ТОВАРИЩИ!

Вывод

Преимущества:

Во-первых, использование свойств очень удобно при обращении к скрытым внутри класса полям. Таким образом ограничивается доступ к ним, что позволяет избежать многих ошибок.

Во-вторых, внутри функций get() и set() можно проводить дополнительную проверку типов возвращаемых в функцию параметров.

В-третьих, внутри функций get() и set() помимо выполнения считывания и записи свойств, можно выполнять скрытые действия.

Недостатки:

Во-первых, реализация свойств с помощью функций get() и set () является проблемной, т.к. функции вызываются лишь при обращении к несуществующим полям объекта.

Во-вторых, если функции get() и set() должны являться обработчиками нескольких полей, то реализация этих функций требует внесения в них дополнительных проверок и конструкций.

В целом...

Использование свойств во многих случаях может повысить удобочитаемость программы и оградит от многих ошибок.

Генерация URL. Создание минимально зависимых модулей и компонент

Автор: Игорь Федоров

В данной статье мы обсудим проблему повторного использования модулей и компонент (в дальнейшем будем называть их просто юниты) в Веб-индустрии, где юниты, отвечающие за визуализацию оказываются жестко привязаны к их родителю, что делает невозможным использование их другими юнитами.

Основная причина кроется в URL. Разработчикам юнитов постоянно приходится иметь дело с данными и логикой построения URL. При дублировании данных или логики появляются проблемы повторного использования юнитов и изменения значений, так как приходится изменять в n-ом количестве мест.

Рассмотрим простой URL:

```
engine.php?lang=rus#anchor
```

Здесь данными являются:

- Путь – engine.php.
- GET параметры – lang=rus.
- Якорь – anchor.

логика:

- какой из разделителей использовать ? или &.
- кодирование GET параметров urlencode().
- проверка данных, зависящих от каких либо условий.

Два наиболее распространенных способа разработки:

1. Как правило, разработчики сайтов задают полный URL в юните, учитывая все дополнительные данные для родителей.

2. Также приходится сталкиваться с проектами, которые частично решают эту проблему из известных PostNuke, где используется функция для создания URL.

Самый главный недостаток этой функции состоит в том, что родитель может быть только один (ядро системы), а иначе происходит дублирование.

Еще один способ решения – через класс generate_url, где данные и логика построения URL находятся в одном месте. Generate_url создан на основе паттерна Value Object (Объект-значение).

«Каким образом следует проектировать объект, который будет широко использоваться, но для которого идентификация не имеет особого значения? Настройте состояние объекта в момент его создания и никогда не меняйте его. В результате выполнения любых операций в отношении данного объекта должен получится новый объект.

Объекты являются отличным способом организовать логику для последующего понимания и роста. Однако существует одна маленькая проблема (хорошо, хорошо, вообще-то проблем больше, однако сейчас мы коснемся только одной из них).

Представьте что я – объект, и у меня есть прямоугольник (Rectangle). Я вычисляю некоторое значение, от этого прямоугольника, например его площадь. Чуть позже некто (например, другой объект) вежливо просит меня предоставить ему мой прямоугольник для выполнения некоторой операции. Чтобы не оказаться невежливым, я предоставляю мой прямоугольник. А через пару мгновений прямоугольник был модифицирован у меня за спиной! Значение площади, которое я вычислял раньше, теперь не соответствует действительности и не существует способа известить меня об этом.

Это классический пример проблемы наложения имен (aliasing). Если два объекта ссылаются на один и тот же третий объект и если один из первых двух тем или иным образом изменяет третий, общий для них объект, второму объекту лучше не полагаться на текущее состояние общего объекта.

Существует несколько способов решения проблемы наложения имен.

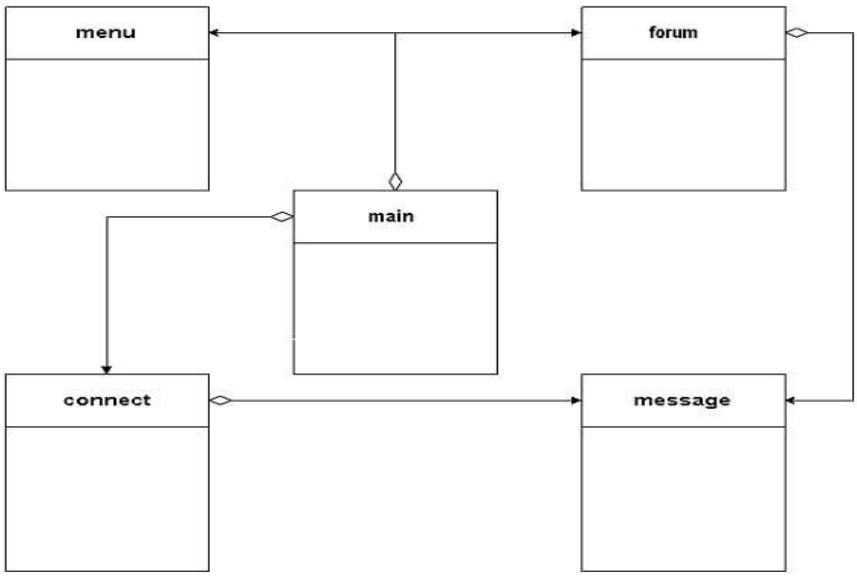
.....

.....

В рамках Value Object каждая операция должна возвращать новый объект, изначальный объект должен оставаться неизменным. Пользователи должны знать, что они используют объект-значение. В этом случае полученный объект следует сохранить»

(Кент Бек «Экстремальное программирование: разработка через тестирование»).

Все плюсы и минусы использования каждой из технологий рассмотрим на примере простого сайта, в котором есть четыре модуля: Форум, Знакомства, Сообщения, Главное меню.



Тем временем...

Для преобразования даты, записанной в обычном “человекочитаемом” виде в Unix timestamp, может пригодиться функция strtotime(), которая, по возможности, преобразует строку с любой датой написанной английским языком в формат Unix timestamp. Вот примеры:

```
echo strtotime("next Thursday");
//выведет что-то вроде 1111611600
```

Подробнее читайте здесь: http://ru3.php.net/strtotime

Для того, чтобы считать содержимое текстового файла в строку, можно воспользоваться функцией file_get_contets(). Она аналогична работе file(), только возвращает не массив строк, а одну строку.

Подробнее читайте здесь: http://ru.php.net/file_get_contents

На схеме видно, что из основной программы вызывается menu, а так же может вызываться forum или connect. В свою очередь, каждый из этих юнитов вызывает модуль message. В примерах мы будем выносить создание URL в переменные. Создание Html опустим.

Попробуем решить эту задачу, прописывая полный URL в юнитах. Описания классов:

```
<?php
Class Main {

$lang = get_language();
$menu = new menu($lang);
$menu_tpl = $menu -> show(); //Возвращает HTML код.
$module = new $module($lang);
$module_tpl = $module -> show(); //возвращает HTML код.
/*
Подставляем переменные $*_tpl в шаблон и выводим на экран.
*/
}
Class menu {
Var lang = '';
function menu($lang) {
   $this -> lang = $lang;
}

function show() {
  $menu = array();
//Возвращает массив модулей доступных в main меню.
  $data = $this -> get_all_modules();
  $count = count($data);
  for ($i = 0; $i < $count; $i++) {
     $menu[$i] = 'engine.php?module=' . $data[$i] . ($this -> lang ? '
&lang=' . $this -> lang : '');
  }
  /*Подставляем "'&lang=' . $this -> lang"  только в том случае $this -> lang
не пуста.
  Подставляем массив меню $menu шаблон.
  */
return $result;
}
function get_all_modules() { .... }
}

Class forum {
Var lang = '';
function forum($lang) {
   $this -> lang = $lang;
}
function show() {
  $menu_tpl = $this -> show_menu(); //формирует меню форума.
  $send_message  = new message($this -> lang);
  $message_tpl = $send_message -> show();
  /*
   Подставляет переменные $*_tpl в шаблон.
  */
  return $result;
}
```

Листинг 3. Продолжение на следующей странице

```
function show_menu() {
$menu[$i] = 'engine.php?module=forum&action=' . $data[$i] . ($this -> lang ? '
&lang=' . $this -> lang : '') . '#submodule';
}
}

Class message{
Var lang = '';

function message($lang) {
   $this -> lang = $lang;
}

function show() {
   $form_action = 'engine.php?module=forum&action=message' . ($this -> lang ? '
&lang=' . $this -> lang : '') . '#message_list';
  /*
   Подставляет $form_action в шаблон.
  */
  return $result;
}
}
?>
```

Как видно, в коде URL прописываются в каждом юните, при этом их количество сведено к минимуму (на практике их намного больше), но достаточно, чтобы увидеть проблему.

Первая проблема состоит в том, что каждый модуль должен знать полный URL, при этом если engine.php перемещается в папку, например /main/engine.php, то придется править во всех юнитах.

Второе, что бросается в глаза – условный оператор ($this -> lang ? ' &lang=' . $this -> lang : ''), который конкатенируется в каждом юните, что тоже не очень красиво. К тому же, если добавится еще один параметр в системе, то придется вновь побывать во всех модулях.

Эти две проблемы возможно решить, генерируя базовый URL в main{} и передавая $main_url в класс через параметр:

```
$main_url = 'engine.php' . ($this -> lang ? ' ?lang=' . $this -> lang :
'');
```

Но у нас появляется другая проблема – мы не можем определить: дочерними юнитами ставить '?' или '&', так как lang может быть, а может отсутствовать (по умолчанию в системах он, как правило, отсутствует, а вот когда выбирается другой язык, появляется).

Здесь, конечно, можно определить переменную $separator, которая будет содержать необходимый символ. Но, вспомнив про якорь, мы понимаем, что нам необходима еще и третья переменная.

К тому же всегда необходимо помнить о том, что эти переменные нельзя модифицировать, так как эту переменную используют и другие модули.

Третья проблема – “?module=forum” – этот параметр в классе message делает невозможным использование класса другими классами (contact, main). Это решается переносом параметра форума в форум и передача трех параметров из форума. Но таким образом мы нагружаем форум дополнительной логикой, которой должен заниматься другой объект:

Форум должен проверить $separator и если он содержит ‘?’, то после добавления своего параметра он должен изменить содержимое переменной на ‘&’.

Итак, мы с горем пополам решили три проблемы, но появилось несколько недостатков:

1. три обязательных параметра в каждом объекте (и это только начало!);

2. не удалось перенести всю логику в один объект;

3. всегда необходимо помнить о том, что переменные нельзя перезаписывать.

Первое, что приходит в голову – использовать функцию. Таким образом, мы избавимся от переменных, и формирование URL будет происходить в одном месте.

Так и сделаем, возьмем функцию Post Nuke? и попробуем использовать в нашем примере. Функция Post Nuke:

```
<?php
/**
* generate a module function URL
* @param modname - registered name of module
* @param type - type of function
* @param func - module function
* @param args - array of arguments to put on the URL
* @returns string
* @return absolute URL for call
*/
function pnModURL($modname, $type='user', $func='main', $args=array())
{
    if (empty($modname)) {
        return false;
    }

    global $HTTP_SERVER_VARS;



    // Hostname
    $host = $HTTP_SERVER_VARS['HTTP_HOST'];
    if (empty($host)) {
        $host = getenv('HTTP_HOST');
        if (empty($host)) {
            return false;
        }
    }
```

Листинг 4. Продолжение на следующей странице

```
 // The arguments
    $urlargs[] = "module=$modname";
    if ((!empty($type)) && ($type != 'user')) {
        $urlargs[] = "type=$type";
    }
    if ((!empty($func)) && ($func != 'main')) {
        $urlargs[] = "func=$func";
    }
    $urlargs = join('&', $urlargs);
    $url = "index.php?$urlargs";

    // <rabbitt> added array check on args
    // April 11, 2003
    if (!is_array($args)) {
        return false;
    } else {
        foreach ($args as $k=>$v) {
            if (is_array($v)) {
                foreach($v as $l=>$w) {
                    $url .= "&$k" . "[$l]=$w";
                }
            } else {
                $url .= "&$k=$v";
            }
        }
    }

    // The URL
    return pnGetBaseURL() . $url;
}
?>
```

Будем считать, что адаптировали функцию к нашему примеру, где три системных параметра $modname, $type, $func необходимых ей убрали и сделали чистку, убрав все лишнее.

Изменим строчку в функции:

```
$url = "engine.php? "(get_language() ? '?lang=' . get_language() : '')
```

Теперь ее интерфейс будет иметь такой вид:

```
function pnModURL($args=array(), $anchor = '');
```

Попробуем применить ее в самом проблемном месте:

```
<?php
Class message{
Var lang = '';
function message($lang) {
   $this -> lang = $lang;
}
function show() {
   $form_action = pnModURL(array('module' => 'forum', 'action' =>
'message'), 'message_list');
  //Получим: engine.php?lang=rus&module=forum&action=message#message_list
  //Подставляет $form_action в шаблон.
  return $result;
} }
?>
```

Итак, функция решает проблемы, но только в том случае, если она вызывается в юните одного уровня вложенности, например, в форуме. Однако в message параметр ‘module’ => ‘forum’ опять все портит. У нас есть еще один способ решения проблем:

Class generate_url:

- add(array() [, string]); – добавить параметры(Get, #anchor).
- add_anchor (string); – добавить якорь.
- get(); – получить URL.

В ядре системы мы инициализируем этот объект и задаем ему параметры по умолчанию:

```
$main_url = new generate_url('/engine.php', array('lang' => 'rus'));
```

У каждого модуля есть конструктор, в который передается объект $main_url, в котором уже заданы параметры вышестоящего юнита.

Пример взаимодействия:

```
<?php
Main {

$data = get_language(); //Возвращает массив array('lang' => 'rus') или array().
$main_url = new generate_url('/engine.php', $data);
$forum = new forum($main_url);
}

Class forum {

var $forum_url;
function forum(&$main_url) {

     $this -> forum_url =& $main_url -> add(array('module' => 'forum'),
'submodule');
     //Если сейчас запросить URL: engine.php?lang=rus&module=forum#submodule
}

function show() {
  $menu_tpl = $this -> show_menu(); //формирует меню форума.
  $send_message  = new message($this -> forum_url);
  $message_tpl = $send_message -> show();

  /*
   Подставляет переменные $*_tpl в шаблон.
  */

  return $result;
}

function show_menu() {
  ....
   $menu_object =& $this -> forum_url -> add(array('action' => $data[$i]));
   $menu[$i] = $menu_object -> get();
  1.//Получим: engine.php?lang=rus&module=forum&action=list#submodule
  2.//Получим: engine.php?lang=rus&module=forum&action=options#submodule
  ….
}
}
Листинг 5. Продолжение на следующей странице
```

```
Class message {

var $message_url;
function forum(&$main_url) {
    $this -> message_url =& $main_url -> add(array('action' => 'message'),
'message_list');
}

function show() {

  $form_action = $this -> message_url -> get();
  //Получим: engine.php?lang=rus&module=forum&action=message# message_list

  /*
   Подставляет $form_action в шаблон.
  */

}
}
?>
```

В этом примере видно, что форум в конструкторе принимает объект $main_url, в который задает свои параметры. В ответ на это он (форум) получает новый объект, с которым в дальнейшем работает или передает дочерним юнитам (message).

По этой же схеме может работать модуль Знакомства и использовать этот же unit (message). Таким образом, юнит может использоваться родителями на любом уровне вложенности без привязки к конкретному URL и родителю!

В последнем примере мы избавились от всех недостатков описанных выше:

1. данные и логика находятся в одном объекте, теперь клиенту остается лишь добавлять в объект параметры и получать результат;

2. неограниченная вложенность юнитов;

3. избавились от проблемы наложения имен, так как в основе класса лежит паттерн Value Object;

Недостатки:

1. класс должен всем клиентам передаваться через параметр;

2. расход памяти на создание объектов.

Как показывает практика, эти два недостатка незначительны в сравнении с достоинствами, которые он дает. К статье прилагается generate_url.class.php и unit тесты test_generate_url.class.php, которые вам помогут лучше понять, как использовать класс.

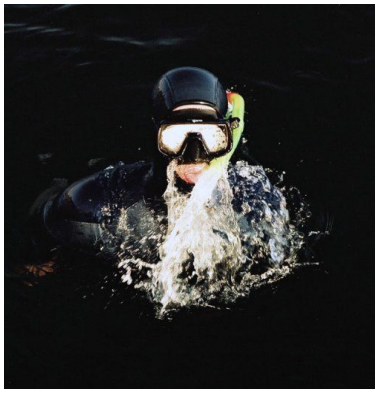
Люди

Наши. Актив PHP.COM.UA

Интервью:
Андрей Олищук [nw]

Это интервью открывает серию материалов "Наши", в которых мы познакомим читателей PHP Inside с людьми, так или иначе связанными с развитием и использованием PHP и смежных технологий в нашем отечестве. Здесь не будет никаких границ – ни возрастных, ни временных, ни политических. Мы расскажем о людях и их проектах, чтобы в итоге можно было составить представление о жизни и работе коллег, чтобы можно было увидеть и сказать – вот такие мы!

В отечественном сегменте интернета существует достаточно большое количество php-сообществ, однако не многим удается быть действительно популярными. В одном из прошлых номеров журнала мы рассказывали о некоторых из тех, кто стоял у истоков phpclub'a и теперь пришло время познакомиться с теми, кто принял активное участие в развитии еще одного популярного сообщества: http://php.com.ua. Сегодня мы беседовали с Дмитрием Сычевским (sych) и Андреем Королевым (BeGeMoT).



Дмитрий Сычевский: "В свободное время я превращаюсь в ползающе-лазающе-плавающе-ныряющее создание"

nw: Расскажите немного о себе. Где живете? Кем работаете/Где учитесь? Чем занимаетесь в свободное время?

Дмитрий Сычевский (далее ДС): Я живу в Киеве, в данное время учусь в магистратуре КПИ, по образованию – проффесиональный оптик :-) (линзы, очки, головки самонаведения, лазеры и тд), летом буду защищать диплом по теме Математическое моделирование процессов дифракции на компьютере – во как страшно звучит.

Так же в данное время работаю в Украинском Вычислительном Центре Гражданской Авиации, отдел АСУ и информационных технологий, специалистом широкого профиля :-). В свободное время я превращаюсь в ползающе-лазающе-плавающе-ныряющее создание которое обычно не сидит дома, а путешествует по пещерам, горам и морям, потому что знает цену свободному времени.

Андрей Королев: "Скоро я стану папкой самой красивой девочки в мире"

Андрей Королев (далее АК): А я вырос в не очень большом городе Житомире. Там ходил в детский садик, школу, лицей. Потом понесло меня в Киев, купили компьютер и началось! В общежитии у нас был свой сервер, на котором размещались страницы других ребят, вот и мне захотелось свою. Моя первая страница состояла из большого рисунка, разрезанного на четыре части со ссылками на простой HTML – моя маленькая гордость. А потом мне показали страницу с простой строчкой phpinfo() в исходнике. Это было года 3–4 назад, вот так и произошло знакомство со странным существом, называемым PHP.

Так как сеть КПИ отрезана от мира, было достаточно сложно найти новую документацию по языку. Но жил у нас такой человек как KVN, и он вёл проект php4you, что и стало отправной точкой в изучении PHP.

nw: Когда и каким образом php.com.ua вошел в вашу жизнь? Расскажите о том, кем, как и когда создавался этот сайт? Что было на нем с самого его рождения?

ДС: Как сайт вошел в жизнь? Просто как и остальные сайты подобной тематики, когда начал работать с php года четыре назад, так на него и вышел как на национальный ресурс. Можно сказать я «вырос» на нем, хотя в то время даже не думал что буду программировать на PHP и координировать действия сообщества – но жизнь все расставила по своим местам.

АК: Я, устроившись на работу, получил доступ в Интернет. Работа была связана с технической поддержкой, и в свободное время я разрабатывал NOC-систему для молодой компании. Мне требовался источник знаний и сначала в этом качестве выступали два ресурса: проект php.com.ua и phpclub.

ДС: php.com.ua был создан довольно давно, даже точно не скажу когда. У истоков тогда стояли lubes (Любомир Гайдамака) и Faeton (к сожалению имени не помню). С рождения на сайте было около десятка шпаргалок и статей, а так же форум. Но через некоторое время сайт начал умирать, так как человек, который его поддерживал непонятно куда исчез и на связь никто из администраторов не выходил. В таком состоянии сайт перебывал от полутора до двух лет – пока не появилась инициативная группа, которая решила заняться разработкой альтернативного ресурса, потому как владелец домена не сильно шел навстречу.

Сначала, конечно, трудно было избавится от плохой славы – застойного проекта но как известно нет ничего невозможного (по крайней мере для меня :-))

АК: Когда старый php.com.ua начал угасать, я стал уделять больше внимания phpclub'у. Но, к сожалению, я просто не успевал прочитывать все темы, и иногда пропускал многие топики, что мне не очень нравилось. Получалось читать только те топики, в названии которых для меня было что-то интересное. И тогда для меня одним из источников информации стал мануал и, особенно, примеры из него. Впрочем, иногда заглядывал и на php.com.ua, чтобы указать людям их грубые ошибки, или сказать, что найти и прочитать в мануале.

Через некоторое время появилось ещё несколько таких людей как я – которым не нравилась данная ситуация с php.com.ua. Был поставлен вопрос ребром: или нам отдают этот проект, или мы ищем подходящее имя сайта и начинаем свой. Lubes пошёл нам на встречу. Новая команда дополнила информацию накопившуюся к тому времени на сайте. Объявили конкурс на новый дизайн – получилось достаточно неплохо, как мне кажется.

ДС: Сначала, конечно, трудно было избавится от плохой славы – застойного проекта но как известно нет ничего невозможного (по крайней мере для меня :-)). Примерно год назад на сайте был проведен косметический редизайн, добавлено (и добавляется до сих пор) немного шпаргалок и довольно много статей, хотя и сейчас некоторые материалы на сайте находятся в плохо структурированном виде – их надо вычитывать, а что-то переписывать или выкидывать.

nw: Что php.com.ua представляет собой сейчас? Сколько в вашем сообществе людей/посетителей? Какие мероприятия вы проводите/устраиваете? Есть ли какие либо проблемы у вашего сообщества?

ДС: Прежде всего сообщество php.com.ua – это сообщество друзей, людей которые объединены тем, что связаны с ИТ и Веб-технологиями, хотя в сообществе есть люди которые с этим вообще и не связаны – им просто нравится сама существующая атмосфера.

Так же есть основное ядро Ukrainian PHP Group – это те люди которые активно занимаются развитием проекта или показали свой высокий профессионализм в нашей сфере деятельности – в принципе на группе все и держится.

Сейчас осещаемость сайта растет с каждым месяцем. Распределение по географии еще интереснее – php.com.ua позиционировался как национальный украинский проект, однако 51–53% посещений приходит из России, 3–7% из других стран, а остальное Украина. На этом фоне распределение по регионам показывет что аудитория покрывает почти 80–90% регионов Украины, хотя все эти данные примерные и на них точно завязываться нельзя :-))))

Крупных мероприятий мы пока не проводим потому как сейчас идет период «дозревания» идей: что делать дальше и как забыть нехорошее прошлое, хотя с удовольствием принимаем участие во разных мероприятиях других сообществ и всегда готовы к сотрудничеству.

За полтора года нами было проведено 2 небольших конкурса. один по редизайну, второй по программированию. Надеюсь, что конкурс по программированию станет ежегодным.

Что касается проблем, то конечно они есть и их довольно много, прежде всего – катастрофическая нехватка времени у основных «двигателей» сайта, – надеюсь скоро проблем станет меньше.

nw: Известно, что существует довольно большое количество крупных и мелких сообществ, связанных с PHP (например, phpclub, php-разделы на woВеб и sitepoint и так далее). Чем вы отличаетесь от этих сообществ? В чем, по вашему мнению, ваше главное преимущество перед всеми остальными? В чем ваши недостатки?

ДС: Как известно в перечисленных выше сообществах в основной массе общаются профессионалы, так сказать монстры своего дела, хотя лично я не люблю вешать на себя и других людей ярлыки типа ГУРУ, ЛАМЕР и тд. Наше направление деятельности больше развивается в сторону выращивания «молодняка», так сказать. У нас на сайте подобие детского сада и школы. Многие люди, которые уже выросли из детского возраста так и остаются здесь, а почему... это надо уже у них спрашивать. Главные наши принципы выражены в этих цитатах и изречениях: “Мы не Боги....., но кое что умеем” и «Тот, кто берется учить других, сам никогда не должен прекращать учиться.» (с) Джон Коттон Дана.

Тем временем...

С помощью fopen() и параметра “w” можно открыть текстовый файл, стерев все хранящиеся в нем данные. Но что если перед стиранием, нам необходимо выполнить над данными некоторые операции?

Для этих целей подходит функция ftruncate(), которая усекает файл до заданной длины. Алгоритм работы прост:
1. Открываем файл и считываем содержимое;
2. Выполняем нужные нам операции с содержимым;
3. Стираем данные в файле с помощью ftruncate();
4. Записываем новые данные и закрываем файл.

Подробнее читайте здесь: http://ru3.php.net/manual/en/function.ftruncate.php

АК: Главная идея проекта, это попытаться научить людей учится. Мы старались никогда не давать прямого и точного ответа, чаще советовали почитать там или там, посмотреть ту или иную статью. И каждый раз друг-другу напоминали, что мы сами такими был несколько лет назад. Но в целом..., в жизни каждого программиста настаёт момент, когда ему уже мало кто может помочь в решении той или иной проблемы кроме документации из первых рук, или такого же программиста, который может предложить что-то новое. И такой проект как phpclub больше подходит для более опытных программистов.

Я считаю, что нашему проекту не хватает журналистов, людей которые бы писали статьи для сайта, проводили обзоры.

А вот недостатки есть у всех. Даже у Zend :-). Я считаю, что нашему проекту не хватает журналистов, людей которые бы писали статьи для сайта, проводили обзоры. Были у нас лозунги и о создании своего журнала, но моя позиция достаточно проста: уже есть один русскоязычный журнал (у которого, кстати, были большие проблемы с наполнением ;) и этот журнал делали люди, которые занимаются проектами в Интернете уже не первый год, так что мы отказались от идеи создания своего журнала. Замечания по этому поводу были подняты и у нас на форуме, а уже следующий номер журнала меня приятно удивил, ребята убрали почти все недостатки, о которых мы говорили на форуме. Появилось больше информации по тематике, окружающей PHP и веб-разработки.

nw: Добавлю от себя, что топик на php.com.ua действительно послужил катализатором для нововведений в журнале. Но перейдем к другому вопросу. Он несколько связан с предыдущим. Краем уха я слышал о некотором неприятии части членов вашего сообщества к форумам phpclub'а? Действительно ли это так? Ведь многие разработчики участвуют и в ваших форумах и в форумах phpcluba. Как вы считаете, эти два сообщества дополняют друг друга или конкурируют между собой?

ДС: Не то что бы к форумам неприязнь (кстати, я очень часто читаю форум клуба), но к некоторым его участникам, а так же к той атмосфере, которую они там создают. Кидаться rtfmами, факами и ставить диагнозы – тут как говорится большого усилия не надо (я не ставлю под сомнения профессиональные знания и умения участников форума), а вот впечатление у людей это портит быстро. По атмосфере мне вобще больше импонирует форум на dklab.ru :-). По второй части вопроса я бы сказал, что скорее всего дополняют, хотя, кто-то возможно в этом видит и конкуренцию.

Кидаться rtfmами, факами и ставить диагнозы – тут как говорится большого усилия не надо

АК: Некоторое неприятие было с самого создания php.com.ua. Наша группа получилась где-то по серединке, если оценивать профессиональные знания и умения в веб разработках. У нас были все молодые ребята: старший курс института, или только-только его закончившие, а лидеры phpclub'а были уже матёрми профессионалами, которые или забыли, как они точно так же учились несколько лет назад, искали информацию по крупицам, задавали такие же <глупые> вопросы, или у них просто не было времени отвечать на вопросы новичков и их интересовали только темы своего уровня. И в принципе именно из-за этого мы начали свой проект.

Мы хотели сделать php.com.ua проектом, где каждый мог найти себе практическую помощь, а не теоретические рассуждения на темы связанные с php, хотя без этого конечно никак :-).

nw: Ваш форум является в некотором роде уникальным, так как общение, фактически, ведется на двух языках – украинском и русском. Как вы управляетесь с таким двуязычием? Нет ли проблем с взаимопониманием? Если есть, то как вы их обходите?

ДС: Мы родились, живем и работаем в Украине так что с восприятием родного языка проблем нет :-) Есть одно негласное правило – обсуждение темы, по возможности, ведется на том языке на котором она была начата. Думаю через несколько лет весь ресурс станет украиноязычным – в этом нет ничего сверхъестественного.

Сейчас взгляд направлен в сторону XML и всего того что с ним связано, не потому что это модно, а потому что это довольно удобный инструмент для реализации некоторых проектов.

nw: У вас наверное есть мысли о будущем php.com.ua? Поделитесь им пожалуйста.

ДС: Сейчас идет подготовка к переезду сайта на новое программное обеспечение, которое позволит сделать ресурс еще более открытым для пользователей и каждый желающий сможет вносить посильную лепту для его поддержки.

АК: К сожалению, я не могу ответить на этот вопрос. Я уже месяца 3–4 очень редко заглядываю на форум. Занимался написанием диплома, а сейчас я жду рождения доченьки и уже через 1,5 месяца я стану папкой самой красивой девочки в мире!

nw: Каковы ваши личные профессиональные интересы? Наверняка вы принимаете участие в других проектах, или ведете свои собственные? Расскажите о вашей профессиональной жизни и о тех технологиях, которыми приходится пользоваться чаще всего.

АК: На работе пришлось отойти от PHP – поменялись веянья и акценты. Сейчас я занимаюсь C# и разработкой веб приложений на платформе MS.

ДС: Конкретно ответить на вопрос сложно – так как по роду своей деятельности очень часто приходится пользоваться довольно широким спектором технологий и изучать что то новое. В основном занимаюсь написанием АРМов, шлюзов к промышленным системам. Самая экзотическая разработка – это веб-шлюз к автоматизированной системой заказа и бронирования авиабилетов.

Так же надеюсь, что несмотря на все жизненные трудности, удастся продолжить работу над SyBe.

- http://www.zend.com/php5/contest/contest.php?id=66&single=1
- http://www.php.com.ua/forum/viewtopic.php?t=3039&highlight=sybe

Сейчас взгляд направлен в сторону XML и всего того что с ним связано, не потому что это модно, а потому что это довольно удобный инструмент для реализации некоторых проектов.

nw: Наш журнал является информационным спонсором четвертой международной PHP-конференции, которая пройдет в мае сего года в Киеве. Вы планируете принять в ней участие? Почему?

ДС: 95% что буду на конференции, за остальные 5% сказать пока не могу. А почему? Просто хочется увидеть знакомые лица и пообщаться с коллегами в живую – это так редко бывает.

nw: Ну и наконец, что бы вы могли пожелать читателям PHP Inside и самому журналу?

ДС: Прежде всего здоровья, что бы работа всегда приносила радость и не обращайте внимание на мелкие неприятности, ведь Жизнь – это очень непредсказуемая вещь.

АК: Так держать. А читателям, быстрого пинга и надёжного коннекта.

nw: Спасибо за ответы!

Наши. PHP и Волжский автозавод

Автор: Владимир Булов
v.bulov@vaz.ru

В очередной статье из серии "Наши" речь пойдет о проблеме выбора архитектуры программного обеспечения и комплекса технических средств с которой столкнулись разработчики Волжского автомобильного завода при создании товаропроводящей сети с применением интернет-технологий. Будут продемонстрированы ИТ-решения проверенные на практике. Автор надеется на то, что исследования, рассуждения, примеры и артефакты изложенные в статье помогут читателю принять более обдуманное решение в выборе средств для реализации поставленных перед ним целей с наименьшими рисками при дальнейшей эксплуатации и развитии системы. Статья является частью серии докладов посвященных построению товаропроводящей сети, публикуемых на международной конференции «Интеллект&ИТ-Бизнес-Металл» и находящейся по адресу http://www.i-b-m.ru.

Реальная проблемная ситуация предшествующая началу работ данного направления

Результаты распада СССР, приватизации станций технического обслуживания и общая криминогенная обстановка вокруг завода создали ситуацию, когда посреднические фирмы практически заменили службу сбыта завода, тем самым блокировав прямую и обратную связь завода с потребителем.

В начале 1998 года для решения указанной проблемы руководством ВАЗа было принято решение о создании региональных складов для хранения и торговли автомобилями на территории России.

Дирекции по информационным системам было поручено автоматизировать отгрузку, учет и отпуск автомобилей на региональных складах. И, хотя в это время проекты с использованием интернет-решений широко применялись для решения подобных задач, было принято решение о реализации задачи на локальном комплексе технических средств с периодическим обменом данными с информационными системами завода через e-mail.

В короткие сроки был написан пакет прикладных программ с использованием клиент-серверной технологии (связка Delphi и MS SQL-сервер). Комплекс технических средств состоял из сервера с ОС NT и 3-х автоматизированных рабочих мест с ОС Windows. Запуск комплексов осуществлялся работниками завода. Всего было выставлено около 30 комплексов в различных городах России.

Эксплуатация комплексов в течение одного года показала следующие недостатки:

- при возникновении системных сбоев, а так же для установки новых систем, требовался выезд специалистов завода на место;

- часто меняющиеся требования к системе привели к многочисленным версиям ПО, которые приходилось сопровождать одновременно;
- неконтролируемость систем приводила к искажению отчетности, которая выявлялась только при инвентаризационных инспекциях;
- передача данных в информационную систему ВАЗа инициировалась на местах, что при низкой дисциплине исполнителей отрицательно влияло на качество и оперативность принятия управленческих решений на основе полученной информации;
- возникновение технических проблем по прокладке сетей в некоторых городах связанных с большой удаленностью складских площадок от центрального офиса, а значит и от сервера базы данных.

Язык PHP очень простой, рабочий код появляется почти сразу

Разрабатываемый подход

Проанализировав недостатки локальной реализации и учитывая мировые тенденции развития информационных технологий, группа специалистов предложила вариант с использованием интернет-среды как среды передачи информации и централизованной обработки данных, базирующейся на существующих данных в информационных системах завода.

Выбор программного обеспечения был проведен в рамках конкурса на территории г.Тольятти, в котором приняли участие как специалисты завода так и фирмы-производители ПО.

Для участников конкурса были выдвинуты некоторые ограничения в принятии архитектурных решений:

- в области операционных систем: программное обеспечение может быть разработано в различных операционных системах, но с последующей эксплуатацией в ОС Unix, так как из 150-серверов, эксплуатируемых в информационных системах завода, 130 работают в различных версиях UNIX и идет планомерная работа по созданию однородной операционной среды на серверах компании.
- в области систем управления базами данных: возможность работы с другими СУБД, хотя реализуемая задача решалась с использованием СУБД Oracle, но исторически на заводе на серверах с ОС Unix применялись СУБД Ingres, Informix, DB-2 и Oracle.
- в области использования среды Интернет: программное обеспечение должно нормально функционировать на сетях с низкой пропускной способностью, так как большинство удаленных точек использует для выхода в Интернет коммутируемые линии и коммерческая информация, передаваемая по открытым сетям Интернет, должна быть защищена от несанкционированного доступа.

Задание на проектирование состояло из следующих частей:

- доступ к предлагаемым таблицам;
- отображение на стороне клиента и корректировка;

- фиксация изменений.

В течении двух-трех недель экспертной комиссии было предложены реализации на Java, Oracle ВебDB, Perl, C++ и PHP. Все проекты были выполнены и продемонстрированы комиссии на технических средствах конкурсантов и работали согласно заданию.

Предоставленные проекты

Проект на Java

Установка JDK машины на сервере является достаточно сложным делом и требует высокой квалификации административного персонала, как во время установки, так и в дальнейшем при администрировании.

Существует целая гамма различных продуктов позволяющих быстро и эффективно разрабатывать приложения. Со слов разработчиков, отдача с момента обучения начинается через 3-6 месяцев.

Вариант проекта работал через JDBC-драйвер, что существенно увеличивало время доступа к данным. Так же следует отметить, что ограничение JDBC-драйвера на передаваемые данные не позволяет эффективно работать с некоторыми типами данных в СУБД Oracle. Время выполнения и нагрузка на процессор программ разработанных на Java было наибольшим из всех представленных проектов.

Проект на Oracle ВебDB

Отсутствие лицензированного Oracle ВебDB на требуемый процессор не позволило запустить проект на технических средствах корпорации, и оценка производительности выполнялась на средствах одного из конкурсантов. Следует отметить повышенные требования продуктов фирмы Oracle на совместимость к версиям операционной системы, системным библиотекам и используемым Вебсерверам.

Внешнее различие проектов было минимальным, хотя проект на Oracle ВебDB представлялся более законченным. Обучение производилось во многих центрах, и после 2-3 недельных курсов появлялись реальные результаты работы.

Продукт работает только с СУБД Oracle, - к другим СУБД прямого доступа нет. В результате испытаний было выявлено, что при автоматизированном проектировании форм создаваемый трафик существенно превышает трафик других проектов. Повлиять на это можно отказавшись от средств разработки, но это приводит к увеличению времени на разработку проекта.

Проект на C++

- Практически не требует конфигурирования на сервере.
- Существует целая гамма различных продуктов позволяющих быстро и эффективно разрабатывать приложения.
- Наивысшая производительность и минимальная нагрузка на систему.

- После изменения требуется пересборка проекта.
- Период разработки проекта уменьшается с увеличением библиотек разработанных специалистами.
- Длительный срок обучения специалистов.

Проект на Perl

- Поставляется с операционной системой и настраивается на конкретную СУБД пересборкой с сетевыми библиотеками базы.
- Достаточно специфический язык, трудный в освоение после классических языков. Своеобразное представление ООП требует некоторых навыков.
- В остальном эквивалентен PHP при использовании mod_perl.

Проект на PHP

- Поставляется с операционной системой и настраивается на конкретную СУБД пересборкой с сетевыми библиотеками базы.
- Сейчас существуют несколько сред для разработки программ на PHP, а в то время это был обычный текстовый редактор. С одной стороны язык интерпретатор подобен sh csh ksh. Синтаксис подобен C++.
- Язык PHP очень простой, рабочий код появляется почти сразу. Доступ к базам через библиотеки самих баз не накладывает никаких ограничений на доступ к данным.
- Используя ускоритель фирмы Zend производительность кода увеличивается на 40-60% .
- Легкая интеграция дополнительных модулей написанных на C/C++ через разделяемые библиотеки, при этом не требуется перенастройка APACHE и PHP.
- В последнее время появилась возможность выполнения кода на клиенте (plug-in).
- Минимальное время обучения программиста.

В результате проведенных испытаний был выбран язык программирования PHP. На нем и была реализована информационная система «Контроль сервисно-сбытовой сети» в состав которой входило 130 видеформ, 100 отчетов, 250 скриптов и 600 таблиц.

В результате проведенных испытаний был выбран язык программирования PHP. На нем и была реализована информационная система «Контроль сервисно-сбытовой сети» в состав которой входило 130 видеформ, 100 отчетов, 250 скриптов и 600 таблиц. На рабочих местах «Дилера», «Дистрибьютора» и «Администратора» работают 150 пользователей. Информационная система обеспечивает отгрузку и продажу 2600 автомобилей в день. Основная часть кода была разработана группой специалистов из семи человек за пять месяцев.

Концепция трех трехзвенных архитектур

Исторически существуют различные архитектуры программно-аппаратных средств для решения ИТ-задач. Вот далеко не полный перечень решений: централизованная обработка данных, клиент-серверное для X-windows, файловое хранилище данных, клиент-серверное для работы баз данных, Веб- решения и т.д.

Не вдаваясь в подробности реализации решений, это не входит в тематику статьи, мы опишем решения, использованные при проектировании товаропроводящей сети с точки зрения применения PHP.

Аппаратная трехзвенная архитектура. (Клиент-СерверПриложений-СервераБазДанных)

Состав аппаратно-программного комплекса состоит:

Клиент

- бездисковая станция БС или персональный компьютер ПК
- операционная система ОС Linux Fedora Core 2
- тонкий клиент Mozilla + X-forms + PHP-plugin
- PHP-shell скрипты
- операционная система Windows
- тонкий клиент Explorer + X-forms + PHP-plugin

СерверПриложений

- процессор Itanium2
- операционная система ОС Linux RedHad 3.0
- Apache + PHP + библиотеки сетевых компонентов баз данных
- PHP-shell скрипты

СервераБазДанных

- процессор Itanium2, HP, IBM
- операционные системы на соответствующие процессоры
- PHP-shell скрипты

Для многократного использования PHP сценариев достаточно развернуть дистрибутив PHP в используемых ОС и скрипты, библиотечные функции встроенные в PHP и разработанные можно использовать не только в Веб окружении, но и как команды операционной системы.

Сборка Apache, с различными библиотеками сетевых компонентов баз данных, позволяют открывать одновременно сессии с различными типами баз данных и обмениваться данными в рамках одной сессии.

Следует отметить привлекательность бездисковых станций для рабочих мест, где бизнес-функции четко прописаны и добавляются централизованно. Загрузка рабочего места бездисковой станции производиться из корпоративной сети либо с flash-носителя на котором располагается и закрытый ключ пользователя.

Модель-Представление-Управление=Model-View-Controller (MVC)

Проектирование Веб-приложений основанных на парадигме MVC позволяет хранить модель и ее описание в базе данных, отображение или визуализацию делать на клиенте, а бизнес-логику выполнять на сервере приложений.

Преимущества и недостатки такого подхода к проектированию информационных систем можно найти на сайте http://www.phpmvc.net/.

К одному из преимуществ можно отнести согласование с программно-аппаратным комплексом, описанным в предыдущей главе. Это позволяет балансировать нагрузку на систему в целом и, в случае необходимости, легко масштабировать решение.

Ssl-passwd-PKI

Сложность информационных систем, помимо прочего, порождает еще и проблему безопасности. Бремя контроля за комплексными средствами безопасности не должно ложиться на плечи конечных пользователей, особенно в условиях, когда системы становятся все более распределенными. Таким образом, управление идентификацией оказывается важнейшей функцией, особенно, когда это связано с материальными и финансовыми потоками и ответственностью за принимаемые решения.

По мере реализации разработанных принципов построения товаропроводящей сети, при включении в свою структуру новых подразделений, а в некоторых случаях и новых бизнес-партнеров, через сеть проходят все новые и новые важные данные подверженные всевозможным угрозам и рискам. Постоянно растущие объемы подобного трафика проходят через внешние сети, Интернет становится базовой средой для подключения пользователей к внутренним приложениям компаний.

Построение безопасной современной информационной системы - это управление возникающими вследствие определенных компромиссов рисками.

Для доступа к технопорталу используется ssl-протокол для шифрации информации от рабочего места пользователя до сервера технопортала - это первый уровень защиты информации проходящей через публичные сети Интернет. Для каждого участника товаропроводящей сети выдается технологический сертификат безопасности для использования его в ssl-соединении между рабочим местом и конкретной информационной системой. На основании сертификатов безопасности мы определяем пользователя как участника бизнес-процесса и даем доступ к следующему уровню.

На втором уровне используется имя и пароль доступа к учетным записям Oracle, и здесь же определяются роли пользователя в терминах системы управления базы данных. С полученным мандатом пользователь переходит в корпоративную информационную систему для выполнения предписанных ему функций.

При выполнении функций требующих юридического подтверждения факта совершенного действия применяется третий уровень аутентификации, основанный на использовании eToken-а как носителя учетных данных для пользователя и математического обеспечения фирмы ALADDIN встроенного в корпоративную информационную систему.

Высокое доверие при таком способе аутентификации и идентификации позволяет корпорации, используя электронно-цифровую подпись (ЭЦП) реализовывать бизнес-функции по продаже готовой продукции в регионах.

Заключение

В настоящей статье изложены принципы, которыми пользовались специалисты при построении товаропроводящей сети автомобилестроительной корпорации. На реализованных примерах показана верность концепции и предложена дальнейшая интеграция с существующими в корпорации информационными системами.

Для реализации концепции разработана архитектура моноплатформенного подхода к построению товаропроводящей сети и реализованы ее основные компоненты.

В качестве практического применения реализованы 5 задач, в которых работает одна тысяча пользователей.

Полученный опыт позволяет надеяться на успешную реализацию и других элементов товаропроводящей сети, которые могут быть легко интегрированы в рамках разрабатываемой системы.

Форум

Статистика сервера

Если вы когда либо пользовались разделяемым (shared) хостингом (я думаю среди читателей таких много) то вы вероятно знакомы с некоторыми инструментами для получения статистической информации о вашем сервере. Это конечно хорошо, но неужели вам никогда не хотелось создать один такой для себя? Если такое случалось, то это руководство для вас.

Материалы сайтов:
http://codewalkers.com
http://www.php.com.ua
http://php.net

Основная статистика

Начнем с рассмотрения обработки и вывода среднего количества загрузок сервера и его доступности (uptime) . Еденица измерения – сутки . Большинство этих задач решается функцией exec(). Все что мы будем делать – по большей части, передавать серверу команду uptime, которая в ответ будет возвращать информацию, нужную для нашего скрипта.

```
<?php

$uptime = @exec('uptime');
preg_match("/averages?: ([0-9\.]+),[\s]+([0-9\.]+),[\s]+([0-9\.]+)/
",$uptime,$avgs);

$uptime = explode(' up ', $uptime);
$uptime = explode(',', $uptime[1]);
$uptime = $uptime[0].', '.$uptime[1];

$start=mktime(0, 0, 0, 1, 1, date("Y"), 0);
$end=mktime(0, 0, 0, date("m"), date("j"), date("y"), 0);
$diff=$end-$start;

$days=$diff/86400;
$percentage=($uptime/$days) * 100;
$load=$avgs[1].",".$avgs[2].",".$avgs[3]."";
echo 'Average Load: '.$load;
echo 'Uptime: '.$uptime;

?>
```

Если вы запустите этот скрипт на вашем веб-сайте, то наверняка сможете получить статистику по загрузкам сервера и его времени доступности (uptime) в сутках. Это оказывается довольно просто.

Далее мы увидим, как показать время доступности сервера в течение года (в процентах). Основу мы уже заложили в вышеприведенном скрипте.

Uptime в процентах

Если вы посмотрите на следующий далее код, то увидите две избыточные и малополезные в данном случае функции date().

Вот весь блок кода, предназначенный для составления процентного соотношения аптайма сервера.

```
<?php
$start=mktime(0, 0, 0, 1, 1, date("Y"), 0);

/* Make date 1/1/(current year) */
$end=mktime(0, 0, 0, date("m"), date("j"), date("y"), 0);

/* Make todays date */
$diff=$end-$start;
$days=$diff/86400;
$percentage=($uptime/$days) * 100;

/* Work out percentage */
echo $percentage;
/* Print out percentage */
?>
```

Первым делом мы используем $start, $end и mktime(), чтобы получить количество секунд, прошедших с первого января текущего года по сегодняшний день. Для того чтобы узнать как долго сервер был онлайн мы вычтем $start из $end. Далее, чтобы работать с днями, разделим полученное значение на 86400 и получим количество дней, которое в свою очередь разделим на $uptime (вычисленный из первого примера кода) и умножим на сто для перевода в проценты. Снова просто, не так ли?

Как посчитать разницу в днях между двумя датами?

Средствами PHP можно посчитать разницу используя следующую функцию:

```
<?php

function date2Days($_timestamp) {
  $_year=(int) date('Y',$_timestamp);
  $_month=(int) date('m',$_timestamp);
  $_day=(int) date('j',$_timestamp);
  $_century = substr($_year,0,2);
  $_year = substr($_year,2,2);
  if($_month>2) $_month -= 3;
  else {
        $_month += 9;
        if($_year) $_year--;
        else {
         $_year=99;
         $_century --;
        }
  }
  return (floor((146097*$_century)/4)+floor((1461*$_year)/4)+floor
((153*$_month+2)/5)+$_day+1721119);
}

$daysDiff=date2Days($t2)-date2Days($t1);
?>
```

Существует способ решения той же задачи средствами SQL, однако он не работает с датами ниже 1970 года.

```
SELECT TO_DAYS(t2)-TO_DAYS(t1)
SELECT DATEDIFF(t1,t2)
```

Как сделать резервную копию файла?

Для создания резервной копии (или просто копии) файла можно использовать функцию copy(). Примерный скрипт копирования может выглядеть вот так:

```
<?php
$file = 'test.txt';
$newfile = 'test.txt.bak';

if (!copy($file, $newfile)) {
   echo "failed to copy $file...\n";
}
?>
```

План врезок