

Апрель-Май '05 № 12

php {INSIDE}

электронный журнал для веб-разработчиков



Мультиязычные веб-приложения

GetText и PHP

Создание мультиязычных
шаблонов в Smarty

Содержание

В фокусе	
Многоязычные приложения с использованием GetText и PHP.....	3
Создание мультязычных шаблонов в Smarty.....	15
Идеи	
Реализация ядра фреймворка, на примере Freeform Framework.....	18
Установка Apache2, SSL и PHP5 с акселераторами кода.....	24
Работа с CVS в Zend Studio 4.x (Windows).....	30
Отрисовка связанного дерева с помощью XSLT.....	33

От редактора

Приветствую! Выпуск который находится перед Вами уже двенадцатый по номеру, но тринадцатый по счету. Для программиста ситуация вполне привычная – ведь в нашей работе (а для кого-то и хобби) многое начинается с нуля.

К моменту выпуска, некоторые члены редколлегии, в лице вашего покорного слуги, а так же Антона Чаплыгина и Александра Косарева (привет Болгарии!) поработали над запуском новой версии сайта редакции, который теперь призван быть еще и источником новостей о мире PHP. В качестве движка мы выбрали Drupal – по рекомендации все тех же Антона (который сейчас активно занимается продвижением gentoo.ru и ведет проект “Учимся программировать вместе с Питоном” - и все это на Drupal) и Александра. Так что заходите на <http://phpinside.ru> за свежими новостями и небольшими обзорами.

Важно! Просим не забывать, что журнал “PHP Inside” является открытым проектом разработчиков для разработчиков. Мы призываем присоединиться к нашей команде и делать журнал вместе. Для этого не обязательно быть писателем. Наверняка вам есть что сказать по многим вопросам, ведь вы сталкиваетесь с проблемами и решаете их – расскажите всем о вашем варианте решения! А если вы в состоянии читать и переводить англоязычные статьи или новости – опять же связывайтесь с нами, у нас есть много интересных текстов.

По вопросам помощи выпускам журнала PHP Inside просьба писать по адресу: nw@phpinside.ru.

Команда номера

Авторы и переводчики

*Кузьма Феськов
Андрей Олищук
Денис Попель
Михаил Корнеев
Денис Юрашку*

Редакционная коллегия

*Александр Смирнов
Александр Войцеховский
Андрей Олищук [nw]
Антон Чаплыгин
Елена Тесля*

Выпуск номера

*Андрей Олищук [nw]
Антон Чаплыгин
Денис Зенькович*

Контактные данные

<http://phpinside.ru>
nw@phpinside.net

Многоязычные приложения с использованием GetText и PHP

Все чаще и чаще современный разработчик сталкивается с необходимостью поддержки разных языков в рамках одного проекта. Говоря о разных языках мы здесь имеем ввиду любой язык, в алфавите которого присутствуют символы, отличные от латинских. Латинский язык мы не считаем за отдельный язык, ибо символы этого алфавита, как правило, присутствуют всегда и нормально поддерживаются на уровне баз данных. В этом материале мы будем говорить о по-настоящему многоязычных приложениях, в которых, наряду с латинскими символами, могут присутствовать, скажем, русские, китайские и так далее буквы.

Организация подобных приложений сложна, потому что она влечет за собой множество проблем и вопросов, которые разработчику приходится решать. Неправильное проектирование такого приложения ведет к тому, что добавление нового языка в проект сопровождается изменением таблиц, редактированием скриптов и другими всевозможными подводными камнями. В результате таких переработок проект может начать работать нестабильно или вообще перестать работать, такие переделки могут потребовать значительных временных и рабочих ресурсов. В общем, в этом материале мы поговорим с вами о том, как же упростить себе жизнь на этапе проектирования и по максимуму облегчить процессы, описанные выше.

GetText – описание возможностей

В преодолении вышеуказанных препятствий, система GetText предлагает нам помощь в следующем:

- освобождает наши скрипты от языкозависимых данных, точнее нам не надо будет заботиться о том, с каким языком будут работать наши скрипты;
- предлагает набор утилит для работы с языковыми данными;
- а также, разработки сторонних авторов позволяют в значительной степени облегчить труд переводчиков и стандартизировать данные;
- очень важная особенность, если вы по каким-то причинам не перевели те или иные строки, скажем, на китайский, то GetText вернет вам их на языке по умолчанию (обычно английский), что очень удобно, так как освобождает нас от полного перевода всех текстов.

Автор: Кузьма Феськов
<http://php.russofile.ru>
kusma@russofile.ru

Новости phpinside.ru

PHP 5.1 быстрее предыдущих версий в несколько раз!

Себастьян Бергманн опубликовал (<http://www.sebastian-bergmann.de/blog/archives/504-PHP-5.1-Performance.html>) результаты тестирования производительности PHP версий 4.3.11, 5.0.4 и 5.1.0.

Как показало тестирование, прирост производительности в PHP 5.1 оказался очень существенным даже по сравнению с PHP 5.0.4 - почти 400%!

Подробности можно узнать на сайте Себастьяна Бергмана по приведенной выше ссылке.

Все эти возможности в свое время привлекли мое внимание и теперь разработка моих скриптов в значительной степени стала более эффективной. Возможности этой утилиты при командной разработке или при значительном разделении труда при производстве приложений вообще трудно переоценить.

От описательной части, перейдем к делу.

Настройка GetText

Прежде чем перейти к процессу освоения новой технологии, давайте произведем установку нужного софта, или проверим наличие оного. Поскольку в основном (по моему опыту) разработка ведется на основе Windows систем, то в рамках данной статьи я буду говорить именно о работе в Windows. В Linux системах как правило указанных действий вообще производить не надо, так как все нужные утилиты там устанавливаются по умолчанию.

Для получения нужных нам файлов идем сюда: <http://sourceforge.net/projects/gettext>. На этой странице вы найдете два пакета программ: gettext-win32 и libiconv-win32. Первый – это собственно файлы GetText, второй – это утилита для работы с текстом в разных кодировках, которая тоже требуется для нормальной работы GetText. Вам необходимо скачать файлы: gettext-runtime-0.13.1.bin.woe32.zip (это уже скомпилированные под Windows файлы GetText) и gettext-tools-0.13.1.bin.woe32.zip (это скомпилированные вспомогательные утилиты), а также libiconv-1.9.1.bin.woe32.zip (это скомпилированные файлы iconv).

Для дальнейшей установки нам понадобятся файлы: gettext-runtime-0.13.1.bin.woe32.zip и libiconv-1.9.1.bin.woe32.zip.

Скопируйте файлы intl.dll, gettext.exe, asprintf.dll, envsubst.exe, ngettext.exe из первого пакета в папку SYSTEM32 вашей Windows, затем тоже самое сделайте с файлами charset.dll, iconv.dll, iconv.exe из второго пакета.

Теперь перейдите в папку dlls вашей инсталляции PHP и скопируйте из нее файл libintl-1.dll в ту же папку, что и предыдущие файлы.

Далее, найдите php.ini (как правило он находится в папке установки PHP или в папке WINDOWS вашей системы). Раскомментируйте в нем 2 строки: extension=php_gettext.dll (чтобы включить поддержку GetText) и extension=php_iconv.dll (чтобы включить поддержку iconv). Теперь перезапустите ваш сервер (например, Apache) и посмотрите phpinfo(); PHP сообщит вам, что GetText и iconv расширения подключены и включены (enabled).

У нас остался еще один пакет: gettext-tools-0.13.1.bin.woe32.zip. Создайте на своем диске отдельную папку и распакуйте эти утилиты туда.

Поскольку мы с вами говорим о профессиональной работе, то нам понадобится бесплатный редактор poedit (<http://www.poedit.org/>). Установите его, он нам понадобится в дальнейшем.

Новости phpinside.ru

Первая трансляция PHP-podcast. Интервью с Узом Фурлонгом

(<http://pro-php.com/index.php?id=3>)

В одном из сообщений на <http://phpinside.ru> мы рассказывали об открытии проекта онлайн-радио (podcast) ориентированного на PHP. И, наконец, первый podcast от создателей проекта - интервью с Узом Фурлонгом (Wez Furlong).

Маркус Уитни (Markus Whitney) провел почти часовую беседу с Узом, в течении которой Уз Фурлонг рассказал об истории и текущем состоянии PECL, важности PECL::Event и причинах возникновения PDO в PHP 5.1.

Отзыв самого Узза о состоявшемся интервью можно найти на его сайте.

<http://www.netevil.org/node.php?uuid=42a3d0b5-c2bd-4906-6073-2a3d0b51f927>

И так, когда все действующие лица собраны воедино, давайте посмотрим, как же это работает.

GetText – взгляд изнутри

Как правило, наши с вами скрипты в той или иной форме выводят данные. Данные эти можно разделить на несколько типов. Нам в данном случае интересуют статические тексты, которые “зашиты” внутри наших скриптов. Это, например, сообщения об ошибках, какие-то другие вспомогательные данные. Наша задача с вами избавиться от необходимости редактирования скриптов (здесь не имеется ввиду доработка скриптов или исправление ошибок).

Для начала, нам необходимо проделать с нашими скриптами несложные изменения, эти изменения вносятся 1 раз и более никогда. В дальнейшем нам просто надо будет сразу придерживаться определенных правил.

В качестве отступления, я приведу некоторые рассуждения. Часто начинающие, и даже не очень, программисты в своих скриптах пишут сообщения на русском языке. Впредь, я советую вам избавляться от такого подхода и писать сообщения латиницей.

Причин здесь несколько: во-первых, при использовании не латиницы возникает вопрос о кодировке, в которой сообщения содержатся в ваших скриптах, что вызывает дополнительные неудобства для пользователей, которые не говорят по-русски или не совсем разбираются в обилии кодировок, во-вторых, это привносит сложность, когда вам внезапно придется изменить кодировку своего приложения – здесь могут полезть разные нестыковки (один скрипт в одной кодировке, другой – в другой), такой подход вызовет много дополнительных действий. Вы плохо знаете английский? Ничего страшного – пишите мета-тэги, переводчик в последующем сможет подправить любую вашу неправильную фразу, не правя для этого скрипты.

И так, вернемся к тому, что же необходимо проделать с нашими скриптами.

Как правило, в скриптах строки используются следующим образом:

```
echo 'string';
print ('string');
$label = 'string';
return 'string';
```

Установленное выше расширение PHP добавляет в наш арсенал несколько команд, при помощи которых мы можем преобразовать примеры выше следующим образом:

```
echo _('string');
print (_('string'));
$label = _('string');
return _('string');
```

или

```
echo gettext('string');  
print (gettext('string'));  
$label = gettext('string');  
return gettext('string');
```

Эти примеры полностью идентичны. Советую использовать первый способ, так как он более экономичный в плане записи и, на мой взгляд, более наглядный, так как не вносит в структуру скрипта значительного “мусора”.

И так, наши скрипты переделаны указанным способом и мы переходим к следующему этапу.

GetText – утилита XGetText

Изменить скрипты – это только половина работы. Что же делать дальше? А дальше нам необходимо создать 2 файла, которые будут содержать в себе все, “обернутые” указанной выше конструкцией, строки. В первом файле они будут содержаться в виде текста, а второй файл – это скомпилированная версия первого.

Для начала нам необходимо извлечь все строки из наших скриптов. Поможет нам в этом утилита `xgettext.exe` из пакета утилит.

В упрощенном виде, вам необходимо запустить ее следующим образом:

```
xgettext my_script.php
```

В результате вы получите первый файл: `messages.po`.

Давайте рассмотрим другие команды этой утилиты:

--files-from=FILE – вы можете составить перечень всех своих скриптов создав отдельный файл, где каждая строка – это название файла. Тогда за один проход утилита может обработать сразу все ваши скрипты. В результате работы вы получите **ЕДИНЫЙ** .po файл.

--directory=DIRECTORY – добавить директорию в список для поиска скриптов.

--default-domain=NAME – NAME определяет название .po файла (NAME.po).

--output=FILE – перенаправляет вывод данных в указанный (FILE) файл.

--output-dir=DIR – выходящие файл будут созданы в каталоге DIR.

--language=NAME – NAME указывает на каком языке программирования вы написали свои скрипты, в данном случае PHP.

--join-existing – если у вас уже есть .po файл, то вы можете добавлять к нему строковые данные при помощи этой команды не создавая нового файла.

--exclude-file=FILE.po – если в новых скриптах есть строки, которые уже есть в вашем .po файле, вы можете пропустить такие строки при помощи этой команды.

--extract-all – эта команда заставит утилиту извлекать все строки подряд без определения языка программирования. Использование этой опции не рекомендуется – вы получите слишком “мусорный” .po файл.

--keyword[=WORD] – вы можете задать список так называемых стоп слов, которые будут пропущены утилитой.

Остальные опции вспомогательные и вы можете разобраться с ними сами, запустив `xgettext --help` для вывода англоязычной подсказки.

Итак, файл создан, что дальше?

GetText – Описание .PO файла

Полученный в результате предыдущих действий файл – это текстовый файл, содержащий в себе все строки, нуждающиеся в переводе.

Далее я приведу пример такого файла:

```
# 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 ""
```

По сути, это и есть файл, который вы должны отдать переводчику. Переводчик должен задать значение следующим строкам:

```
#: prueba.php:13
msgid "This is a test" // это оригинальная фраза
msgstr "" // здесь будет переведенная фраза
```

Тут многие из вас, наверное, почешут голову и скажут, что все это полный отстой. Советую дочитать материал до конца и вы поймете, что все вовсе не так плохо, потому что далее мы также рассмотрим заявленную нами в названии статьи “профессиональную” работу.

Далее, давайте рассмотрим структуру заголовка файла (только самые важные):

```
# Copyright (C) YEAR Free Software Foundation, Inc.  
# FIRST AUTHOR , YEAR.
```

Укажите здесь свои авторские копирайты.

```
"Project-Id-Version: PACKAGE VERSION\n"
```

Версия проекта и его название.

```
"Last-Translator: FULL NAME \n"
```

Укажите здесь имя и email адрес переводчика.

```
"Language-Team: LANGUAGE \n"
```

Название языка (например, Russian).

```
"Content-Type: text/plain; charset=CHARSET\n"
```

Кодировка, в которой сделан перевод (например, UTF-8).

Рекомендую вам всегда делать переводы в Unicode кодировке UTF-8. Это избавит вас от многих проблем, а данные от GetText вы всегда сможете получить в той кодировке, в которой вам необходимо.

Ну вот, все заголовки заполнены, а сообщения переведены – двигаемся дальше.

GetText – утилита MsgFmt

GetText не может работать с .po файлом. Чтобы информация из него стала доступной, его надо скомпилировать. Этим занимается заявленная в названии утилита. В результате ее работы у вас появится второй файл с расширением .mo.

```
Msgfmt file.po file.mo
```

Вот, собственно, и все.

GetText – структура каталогов

Для удобства, создайте в корне своего приложения папку locale. Внутри этой директории необходимо создать папки для каждого из поддерживаемых языков: ru/, en/ и так далее. Внутри них необходимо создать папку LC_MESSAGES, а в эту папку, в свою очередь, необходимо поместить созданные нами .mo файлы для каждого языка (.po файлы можете поместить туда же, это не мешает).

Давайте посмотрим, как будет выглядеть дерево папок в финальном виде:

```
/src
  /locale/en/LC_MESSAGES/messages.mo
  /locale/ru/LC_MESSAGES/messages.mo
  /locale/messages.po
  /locale/messages.mo
```

Для правильного нахождения двухбуквенного сочетания для каждого языка воспользуйтесь этой таблицей: <http://www.loc.gov/standards/iso639-2/langcodes.html>, или документацией по GetText.

В предыдущих нескольких главах мы с вами познакомились с технологией работы GetText. Настало время рассмотреть варианты “Профессиональной работы”.

GetText – профессиональная работа

Все описанные выше шаги необходимы для понимания как все это работает и что за чем следует. Но, как вы понимаете, это все очень сложно и не всегда доступно. Например, врядли вы толково сможете объяснить все это переводчику, а уж тем более просить его переводить какой-то файл непонятного формата – это задача не из легких.

На этом этапе мы с вами воспользуемся замечательным бесплатным редактором poedit, который вы уже скачали и установили себе в одной из первых глав нашего повествования. С запуском этого редактора мы переходим к профессиональной части.

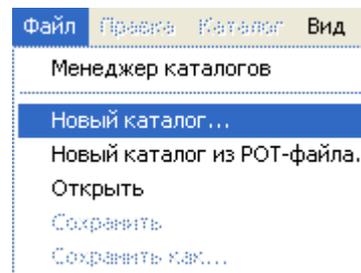
Poedit поддерживает множество языков интерфейса, в том числе и русский.

При первом запуске выберите язык интерфейса (поддерживается русский). А также редактор попросит ввести ваше имя и email. Другие установки оставьте по умолчанию. Во вкладке “Парсеры” удалите все, кроме PHP.

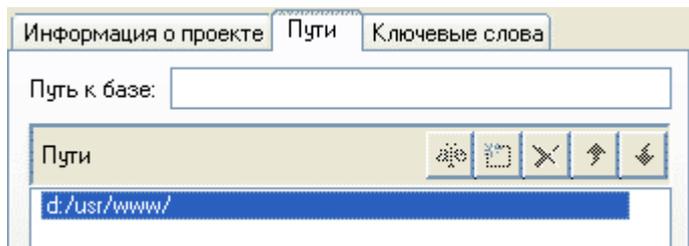
И так – начинаем работать. Во вкладке “Файлы” выбираем “Новый каталог” (рис. справа).

Далее заполняем появившуюся форму:

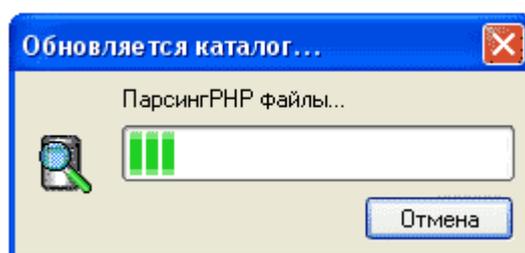
Информация о проекте	Пути	Ключевые слова
Название проекта и версия:	<input type="text" value="my_site"/>	
Команда:	<input type="text" value="My command"/>	
E-mail команды:	<input type="text" value="my_email@my_site.ru"/>	
Язык:	<input type="text" value="Russian"/>	
Страна:	<input type="text" value="RUSSIAN FEDERATION"/>	
Кодировка:	<input type="text" value="utf-8"/>	
Кодировка исходного кода:	<input type="text" value="utf-8"/>	
Формы множественного числа:	<input type="text"/>	



В следующей вкладке задайте путь (пути), где находятся скрипты вашего проекта.

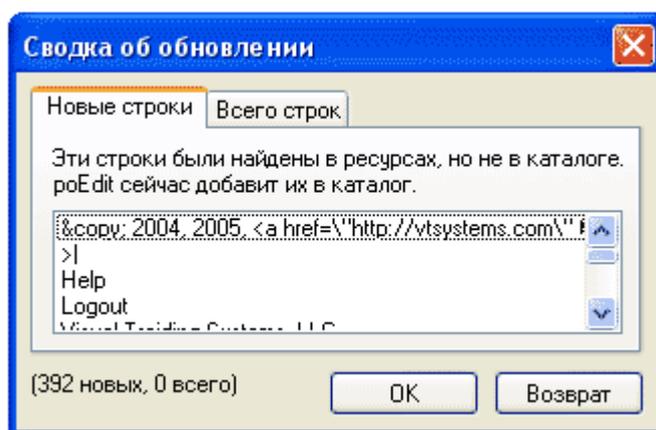


Теперь жмем “ОК”. Редактор начнет рекурсивно обрабатывать все найденные в указанной папке и ее подпапках скрипты на предмет наличия в них “обернутых” строк.



По окончании работы, редактор создаст .po файл и предложит сохранить его на диск. Это файл-проект, который содержит в себе все сообщения, которые нуждаются в переводе для всего вашего проекта (или тех скриптов, которые редактор нашел по указанному вами пути).

Перед вами появится меню, в котором содержится перечень всех новых строк, которые нуждаются в переводе, а во второй вкладке будут показаны те строки, которые исчезли (удалены за ненадобностью) из ваших скриптов. После нажатия “ОК”, .po файл будет обновлен – новые строки добавятся, старые удалятся.

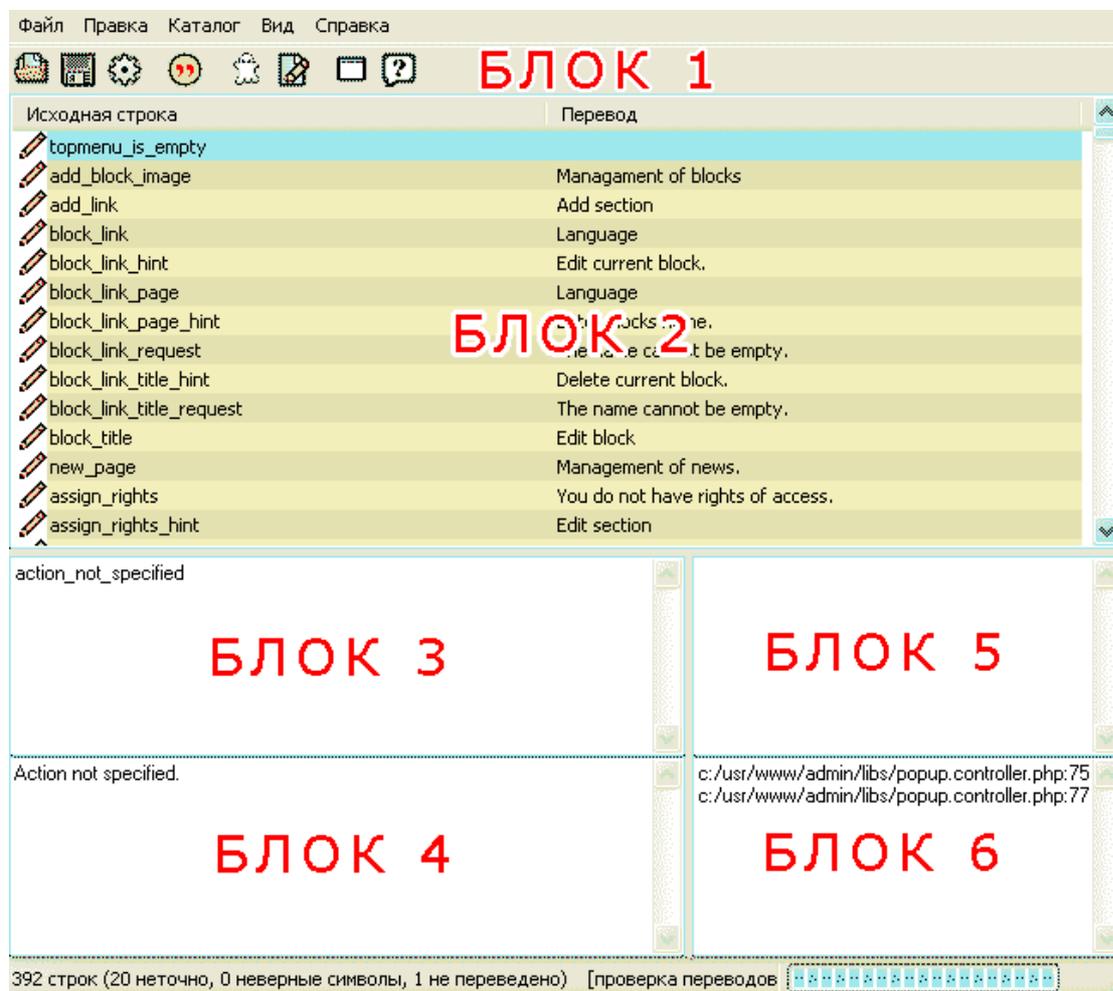


Редактор сам попытается подобрать перевод к новым строкам из уже имеющихся переведенных строк. Строки, к которым перевод удалось подобрать (обращаю ваше внимание, что редактор делает это по странному алгоритму и вам не стоит на него надеяться), будут выделены коричневым цветом.

Строки, к которым перевод не подобран и которые нуждаются в переводе будут сине-зелеными, белые строки – это нормально переведенные строки.

Исходная строка	Перевод
topmenu_is_empty	
add_block_image	Managment of blocks
add_link	Add section
topmenu_manage	Management of comment
comments_archive	Public date
© 2004, 2005, <a href='\http://www.ems, LLC	© 2004, 2005, <a href='\http://www.ems, LLC
>	>

Итак, теперь, если передать редактор вместе с .po файлом переводчику, то ему будет очень легко и понятно переводить указанный файл. Редактор поддерживает все языки (в том числе и китайские иероглифы). Давайте рассмотрим все элементы редактора.



БЛОК 1 – это блок вспомогательного меню, куда вынесены некоторые нужные функции. Первые 2 иконки понятны, о третьей я расскажу ниже. Четвертая иконка включает показ кавычек в окнах оригинала (БЛОК 3) и перевода (БЛОК 4).

Пятая иконка - если нажата, то это означает что для строки был подобран перевод самим редактором и этот перевод не точен. Следующая иконка вызывает окно редактирования комментария (БЛОК 6). В поле комментария вы можете дать пояснение переводчику к любой, требующей перевода, строке.

Следующая иконка убирает заголовок окна, растягивая область экрана на весь десктоп. Последняя иконка – вызов справки. БЛОК 5 – блок автоматических комментариев. БЛОК 2 – это перечень всех переведенных и непереведенных строк. Нажмите на любую строку, чтобы произвести перевод или отредактировать его.

Как видите, процесс перевода очень прост и вот уже скоро (буквально после нажатия кнопки сохранить) редактор создаст для вас автоматически файл .mo.

У редактора есть еще одна крайне полезная функция. Которая прячется за неопианной ранее иконкой (см. справа). Она нам понадобится, когда в наш проект внесены изменения: какие-то скрипты удалены или значительно дополнены или просто внесены в них изменения.



После нажатия этой иконки редактор полностью пересканирует указанный при создании проекта каталог и выявит все новые не переведенные строки в ваших скриптах, а также найдет все удаленные из скриптов строки. Обращаю ваше внимание, что если вы используете в своем проекте шаблонизатор Smarty, то вам необходимо очистить папки со скомпилированными шаблонами, так как редактор не может корректно обработать имена файлов этих шаблонов.

Таким образом, вы получаете мощный инструмент для автоматизированного перевода ваших приложений на разные языки, а также удобный инструмент для переводчика.

Теперь мы с вами переходим к финальной части нашего рассказа – а именно к тому, как же все таки .mo файл заставляет наши скрипты говорить на разных языках.

Do you speak по-русски?

Наконец все нужные файлы созданы и переведены, и вы с чистым сердцем можете заставить ваше приложение бодро откликнуться на нужном вам языке. Давайте посмотрим, как это работает.

В конфигурационный файл или в самом начале вашего индексного файла необходимо добавить следующие строки.

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

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

// 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');
Пройдемся по порядку.

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

Эта строка включает нужную пользователю кодировку. В данном случае русскую. А вот таким образом включаем английскую:

```
// Задать английский язык текущим
putenv ("LC_ALL=en_US");
```

Теперь нам необходимо настроить локаль, в которой работаем. Для русской локали это выглядит так:

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

Для английской – так:

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

Все сообщения теперь будут выводиться в кодировке, соответствующей выбранной вами локали. Если это вас по каким-то причинам не устраивает, можно указать кодировку принудительно. Вот такая команда включит принудительно Unicode (UTF-8).

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

Переменная `$domain` в данном случае указывает домен (или имя файла `.mo`) нашего проекта. Все файлы `.mo` должны называться так: `$domain.mo`.

Далее указываем корневую папку, где у нас содержатся все переводы (`.mo` файлы для всех языков).

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

Как вы понимаете, доменов может быть несколько. Это необходимо, например, если ваше приложение состоит из нескольких самостоятельных частей. У каждой части может быть свой языковой файл.

Теперь необходимо подключить (выбрать) заданный выше домен:

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

Все – наше приложение “выучило” язык, заданный вами в настройках!

И вот уже привычное `echo _('Hellow world!');` будет сиять на вашем экране как “Привет мир” или на любом другом языке мира!

Итог

В рамках данного материала мы с вами подробно познакомились с замечательной технологией GNU GetText. Рассмотрели принципы функционирования и организации многоязычных приложений. Рассмотрели возможности оптимизации работы программистов и переводчиков, а также заложили основы профессионального использования в своих приложениях предлагаемых технологий.

Вы всегда можете скачать полностью рабочий пример и протестировать его на своем компьютере (http://php.russofile.ru/gettext_sample.rar).

Новости phpinside.ru

Вышел PHP 4.4.0 RC1

<http://qa.php.net/~derick/>

Вышел релиз-кандидат PHP версии 4.4.0. Как утверждают разработчики, данный релиз содержит в основном исправления ошибок. Средняя цифра в версии была изменена по причине внесения новшеств во внутреннее API, что может повлиять на совместимость с некоторыми расширениями, разрабатываемыми третьими сторонами.

В данном релизе было исправлено около пятидесяти ошибок и тестирование все еще продолжается.

Создание мультиязычных шаблонов в Smarty

В данной статье будет рассмотрен способ реализации мультиязычных шаблонов с помощью шаблонизатора Smarty. Сразу замечу, что здесь мы коснемся создания именно мультиязычных **шаблонов**, а не приложений. Для того, чтобы вывести текст из БД на том или ином языке, нужно поработать на уровне PHP-скрипта. Мультиязычные шаблоны в свою очередь неплохо подходят для интернационализации HTML-форм и других элементов интерфейса веб-приложения.

Автор: Андрей Олищук

Реализовывать мультиязычность в данной статье мы будем с помощью конфигурационных файлов Smarty. Предполагается, что читатель уже знаком со Smarty, и шаблонизатор уже установлен в его системе.

План прост – для каждого языка нашей системы создадим отдельный локализационный файл с названием <язык>.conf.

Допустим, нас интересует локализация русскоязычного приложения под украинский и английский языки. Для этого в директории \$smarty->config_dir (обычно это 'configs/') создадим три файла: ru.conf, en.conf и ua.conf следующего содержания (на примере авторизационной формы):

```
[lang]
login_form_name = Вход в систему
login_input     = Логин
password_input  = Пароль
submit_button   = Вход
```

Листинг 1. Файл ru.conf

```
[lang]
login_form_name = System login
login_input     = Login
password_input  = Password
submit_button   = Submit
```

Листинг 2. Файл en.conf

```
[lang]
login_form_name = Вхід до системи
login_input     = Логін
password_input  = Пароль
submit_button   = Вхід
```

Листинг 3. Файл ua.conf

Теперь посмотрим на PHP-файл, в задачи которого входит определение, на каком из языков “заговорит” веб-приложение. Сначала рассмотрим псевдокод:

1. Из GET-переменной \$lang получаем язык, на котором будет “говорить” система. Он определяется, например, двумя буквами (ru, en, ua).
2. Если GET-переменная пуста (язык не определен), то в качестве языкового файла используем установленный по умолчанию (ru.conf). Если GET-переменная не пуста, то методом конкатенации получаем имя языкового файла (например, ua.conf).

3. Проверяем, существует ли такой файл в конфигурационной директории. Если нет, то опять же возвращаемся к ru.conf. Если существует, то устанавливаем его в качестве основного языкового файла.

Теперь собственно сам пример кода:

```
<?php
//Подключение Smarty
include "smarty/Smarty.class.php";
$smarty = new Smarty;
$smarty->template_dir = 'templates/';
$smarty->compile_dir = 'templates_c/';

//Определяем путь, по которому будут храниться
//наши языковые файлы
$smarty->config_dir = 'configs/';
$smarty->cache_dir = 'cache/';
//Указываем язык "по умолчанию"
$default_language = "ru.conf";

if ($_GET['lang'] != "") {
    $tmp = $_GET['lang'] . ".conf";
    if (file_exists($smarty->config_dir . "/" . $tmp)) {
        $LANGUAGE = $tmp;
    }
    else {
        $LANGUAGE = $default_language;
    }
}
else {
    $LANGUAGE = $default_language;
}
//Вывод lang.tpl
$smarty->assign ("LANGUAGE", $LANGUAGE);
$smarty->display("lang.tpl");
?>
```

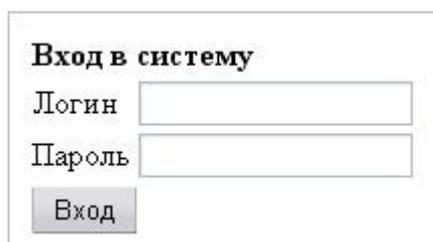
Листинг 4. Файл lang.php

Ну и наконец, шаблон lang.tpl:

```
<{config_load file="$LANGUAGE" section="lang"}>
<form action="">
  <table>
    <tr><td colspan=2><b>{#login_form_name}</b></td></tr>
    <tr><td>{#login_input#}</td>
      <td><input type=text name="login"></td></tr>
    <tr><td>{#password_input#}</td>
      <td><input type=password name="password"></td></tr>
    <tr><td colspan=2><input type=submit value="{#submit_button#}">
  </td></tr>
  </table>
</form>
```

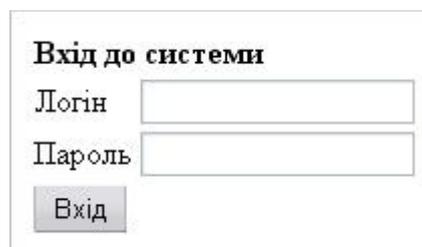
Листинг 5. Файл lang.tpl

Теперь, авторизационная форма будет выглядеть так:



Вход в систему
Логин
Пароль

Результат lang.php



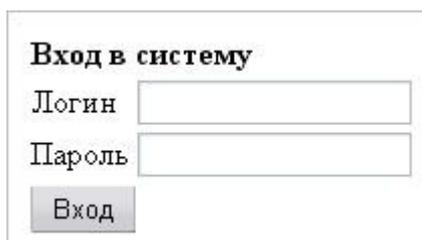
Вхід до системи
Логін
Пароль

Результат lang.php?lang=ua



System login
Login
Password

Результат lang.php?lang=en



Вход в систему
Логин
Пароль

Результат lang.php?lang=yzxfa

Реализация ядра фреймворка, на примере Freeform Framework

В последнем номере PHPInside мы познакомились с основными чертами нового MVC фреймворка, созданного на платформе PHP5. В этом номере мы продолжим обозрение Freeform Framework и детально рассмотрим его ядро. Как уже отмечалось, ядро этой системы достаточно абстрактное, что позволяет разработчикам интегрировать и портировать уже существующее программное обеспечение. Именно абстрактность ядра делает любую систему гибкой, расширяемой и масштабируемой.

Ядро Freeform Framework состоит из интерфейсов и классов, которые описаны в пакетах freeform и util. В первом пакете находятся классы, которые собственно и являются основой архитектуры фреймворка, в то время как второй пакет определяет вспомогательные классы и интерфейсы, которые могут быть использованы и в других системах или приложениях, не созданных на основе Freeform Framework.

Основные классы ядра

Поскольку Freeform Framework четко определяет, как происходит обработка запроса (это, фактически, единственное жесткое определение, которое невозможно как-либо изменить, и которое, по-сути, описывает архитектуру системы), сперва рассмотрим классы, которые используются при обработке любого запроса.

Request, Responce, Action, Location

Эти четыре класса являются, пожалуй, самыми часто используемыми при разработке веб-приложений с помощью Freeform Framework.

Класс **Request** инкапсулирует данные запроса. Его основное предназначение - дать приложению единый интерфейс доступа к данным запроса, независимо от его типа, кодировки и конфигурационных настроек PHP. Freeform Framework гарантирует, что все данные, поступающие от удаленного клиента, будут иметь кодировку UTF-8 и не будут иметь экранированных кавычек и слешей, даже если директива PHP `magic_quotes_gpc` включена. Более того, с помощью Freeform Framework можно создавать приложения, которые могут воспринимать данные не только от браузеров, но и от других типов клиентов.

При этом, уже созданные Вами классы Action не должны каким либо образом переписываться.

Автор: Денис Попель

Новости phpinside.ru

Развитие phpDocumentor продолжается после года простоя

<http://greg.chiaraquartet.net/archives/66-development-restarts-after-a-year.html>

Последний релиз проекта phpDocumentor состоялся более года назад - 11 апреля 2004.

В воскресенье, 12 июня Грег Бивер заявил, что он возобновляет работу над проектом и готовит новый релиз - 1.3.0 RC4. Напомним, что в апреле предыдущего года был выпущен 1.3.0 RC3.

Как говорит Грег, работа над новым релизом началась в аэропорту Шарля де Голля в Париже и продолжается в Италии. Пока ожидается, что новый релиз будет содержать исправления ошибок предыдущего релиза.

Для взаимодействия с такими клиентами достаточно создать потомок класса RequestAdaptor и определить в нем всего два метода - isSupportedContentType() и getParameters(), с помощью которых Freeform Framework сможет определить, является ли класс обработчиком для входящего типа данных и, если да, то вернуть массив всех входящих параметров. Далее, класс Request через свои методы позволяет получить доступ к данным запроса, таким как заголовки, параметры и cookies.

Класс **Response** отвечает за управление ответом на запрос. Его основными функциями является возврат результирующего документа клиенту, перенаправление его на другой адрес, добавление заголовков ответа и контроль за кешированием ответа на стороне как клиента, так и сервера (поддержка кеширования была добавлена в версии 1.2.0a, которая вышла 3 мая).

Именно благодаря использованию этого класса Freeform Framework дает возможность абстрагироваться от типа документа, который будет возвращен пользователю. Как уже отмечалось в предыдущей статье, классы Response возвращают клиентам экземпляры интерфейса Document, который, в свою очередь, определяет всего два метода - getHeaders() и getBody(). Первый из них предназначен для возвращения HTTP-заголовков документа (таких, как MIME-тип, кодировка, метод сжатия и т.д.), а второй - собственно тела документа.

Кроме того, поскольку управление кешированием происходит именно в классе Response, каким бы не был класс возвращаемого документа, ему не надо реализовать собственную систему управления кешированием.

Класс **Action** и его потомки являются самыми распространенными в каждом приложении, созданном на основе Freeform Framework. Именно они и реализуют функциональность приложений, поскольку они ответственны за реакцию на действия пользователя (переход по ссылке или отправка форм). Следуя шаблону программирования MVC, для каждой отдельной функциональности приложения (фактически, для каждого типа страницы) создается отдельный класс-потомок Action. Основными его методами являются:

- onInit() - вызывается всегда после создания экземпляра класса. В нем, как правило, проходит инициализация внутреннего состояния объекта
- getAccessController() - вызывается всегда после onInit(). Этот метод должен вернуть экземпляр интерфейса AccessController, который отвечает за политику безопасности, сопряженную с данным классом Action
- process() - вызывается, когда AccessController, возвращенный предыдущим методом, позволил исполнение данного Action, или когда этот Action не защищен (доступен всегда)
- onAccessDenied() - вызывается, когда AccessController, возвращенный методом getAccessController(), запретил исполнение этого Action.

Новости phpinside.ru

Вышла версия PHP 5.1 beta 1

<http://www.php.net/downloads.php>

Бета-версия PHP 5.1 стала доступна на сайте php.net.

В числе основных нововведений называют: PDO (новый интерфейс для работы с базами данных), улучшение производительности, апгрейд PCRE до 5.0, улучшение функциональности и работоспособности некоторых направлений, таких как SPL и SOAP.

Энди Гутманс выделяет среди основных достижений PDO. "Это то, чего я так долго ждал", пишет он в своем блоге. Среди направлений для ближайшей будущей работы, он выделил улучшение совместимости PHP с юникодом.

Впрочем, уже на следующий день Узз Фурлонг заявил об ошибке в ядре PDO, которую нужно было поправить до выхода первой беты PHP 5.1.

Архитектурной особенностью, которая отличает Freeform Framework от других подобных решений, является то, что класс Action и его потомки не отвечают за обработку данных форм. Это позволяет строить иерархии классов, в которых родительские классы реализуют некую базовую функциональность, которая присуща всем классам (например, подготовка выходного документа, получение имени пользователя, настройка локали и т.д.), а наследующие их классы-потомки только занимаются обработкой данных и подготовкой отображения результатов.

Кроме того, Freeform Framework является одной из немногих систем, в которой возможно размещение нескольких различных форм ввода на одной странице, а также использование одной и той же формы на нескольких страницах.

Класс **Location** предназначен для построения ссылок на другие страницы и формирования адреса обработчика форм. Основная цель его создания - освободить дизайнеров от включения URL в ссылки и формах в шаблоны документов. Вместо этого, в пакете html, например, существуют специальные теги для встраивания ссылок в выходные документы, которые, изучив данные объекта Location, создадут правильный URL другой страницы или формы.

Обработка форм: классы Form и поля ввода

Freeform Framework имеет встроенную мощную систему обработки данных из форм. Форма - это экземпляр класса Form, которому заданы список полей формы, метод отправки (GET/POST) и объект Location, который указывает на класс Action, который будет собственно заниматься обработкой отправки. Форма может автоматически определить, была ли она отправлена, и, если да, то проверить, правильно ли заполнены все поля.

Каждое поле наследует абстрактный класс InputField. Каждое поле имеет значение и объект класса Validator, который определяет, правильное ли это значение. В пакете util определен класс RegexValidator, который определяет, соответствует ли значение заданному регулярному выражению. Также в пакете freeform определены классы TextField, CheckBoxInputField, RadioInputField и SelectInputField, которые позволяют встраивать соответствующие поля ввода в форму.

При этом эти поля вместе со средствами пакета html позволяют автоматически заполнять списки выбора и проставлять значения таких атрибутов, как value и type автоматически, без участия дизайнера в этом процессе.

Расширяя класс Form, можно создавать формы с определенным набором полей и использовать их на разных страницах. По сути, Freeform Framework - единственная система, которая позволяет переиспользовать формы. Что же касается упомянутой выше возможности взаимодействовать не только с браузерами, то классы Action, которые используют формы, могут без каких-либо изменений получать данные и от других типов клиентов.

То есть, форма - это сложный фильтр, который позволяет определить, является ли набор параметров верным, но ей абсолютно безразлично, пришли ли эти данные из браузера, или от сетевого приложения - результат будет одним и тем же.

Рассмотрим пример создания формы для входа в систему, которую можно размещать на разных страницах:

```
<?
class MyLoginForm extends Form {
    // Конструктор, который принимает только объект Request. Поскольку нам
известна
    // архитектура нашего приложения, мы вписываем остальные
    // параметры в теле конструктора
    function __construct(Request $rq) {

        // Поскольку в нашем приложении только один Action обрабатывает форму входа
        // в систему, мы прописываем его Location здесь
        parent::__construct($rq, new Location('LoginAction'), Request::METHOD_POST);

        // Создаем поле для ввода имени пользователя
        $this->addField('userName', new TextField('', new RegexValidator
(RegexValidator::USER_NAME)));
        // Добавляем поле пароля
        $this->addField('password', new TextField('', new RegexValidator('.+'),
TextField::PASSWORD));
    }
}
?>
```

Теперь в каждом классе Action, который создает страницу, на которой должна присутствовать форма входа в систему, достаточно добавить строчку

```
<?
...
$document->setVariable('loginForm', new MyLoginForm($this->getRequest()));
...
?>
```

или же создать промежуточный абстрактный класс, в котором это будет происходить автоматически. Конечно, чтобы это работало, нужно в шаблонах добавить соответствующие тэги для отображения самой формы. Напомним, что средствами пакета html можно создавать шаблоны или их части, которые можно повторно использовать, встраивая их в другие шаблоны, так что достаточно только раз "нарисовать" эту форму, и потом использовать ее в других шаблонах.

Система безопасности

При создании Freeform Framework была предпринята попытка создать расширяемую архитектуру безопасности, которая не завязана на встроенную систему регистрации пользователей (т.е., была бы зависима от модели данных). В результате проверка безопасности происходит только тогда, когда Action явно этого требует, но, в таком случае, эту проверку невозможно обойти. Единственным строительным элементом, который отвечает за проверку возможности исполнить тот или другой Action, является интерфейс AccessController.

Реализуя этот интерфейс, создается отдельная иерархия классов, которая может быть использована и в других приложениях, у другими Action. Именно такое разделение Action и инфраструктуры применения политики безопасности позволяет быстро создавать корпоративные приложения с очень высокой степенью защиты от несанкционированного доступа.

Например, мы можем создать AccessController, который будет проверять, вошел ли пользователь в систему:

```
<?
class UserAccessController implements AccessController {
    function isAccessible() {
        return !is_null(Session::getUser());
    }
}
?>
```

Конечно, здесь подразумевается, что форма входа и ее обработчик уже созданы. Теперь в разных классах Action достаточно добавить метод

```
<?
class MyOtherAction extends Action {
    ...
    function getAccessController() {
        return new UserAccessController();
    }
    ...
}
?>
```

чтобы автоматически ограничить доступ пользователей, которые не зарегистрировались. Опять же, можно создать промежуточный абстрактный класс SecureAction, в котором определен этот метод. Также в нем целесообразно определить метод onAccessDenied():

```
<?
...
function onAccessDenied() {
    $this->getResponse()->relocate(new Location('AccessDeniedAction'));
}
...
?>
```

В этой статье мы коротко познакомились с архитектурными особенностями ядра Freeform Framework, делая акцент на расширяемости приложений, созданных его на основе. В следующих публикациях мы остановимся на практических моментах, а именно - на создании форм и знакомству с пакетом html.

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

Примечание: в прошлом номере PHP Inside во вложении к материалам о Freeform Framework вкралась ошибка. В файле шаблона необходимо удалить строку с комментарием, идущем в самом начале файла.

Установка Apache2, SSL и PHP5 с акселераторами кода

В этом руководстве рассматривается установка веб-сервера Apache (2.0.x) с SSL и PHP5 (с Zend Optimizer и ionCube Loader). Руководство следует рассматривать как практическое – оно не покрывает теоретическую сторону вопроса. Для этого вы сможете найти массу документов в сети. И, конечно же, данный документ идет без каких либо гарантий!

Автор: Фалко Тимме (Falko Timme)

Перевод: Андрей Олищук

Оригинал:

http://www.falkotimme.com/howtos/apache2_ssl_php5_zendoptimizer_ioncubeloader/

Получаем исходники

Для работы нам понадобятся следующие программные продукты: openssl, Apache 2.0.x и PHP 5. Установка будет производиться из директории /tmp.

```
cd /tmp
wget http://www.openssl.org/source/openssl-0.9.7g.tar.gz
wget http://ftp.plusline.de/ftp.apache.org/httpd/httpd-2.0.53.tar.gz
```

Далее, посетите <http://www.php.net> и скачайте последнюю версию PHP. Скачанный архив поместите в директорию /tmp.

Установка Openssl

```
tar xvfz openssl-0.9.7g.tar.gz
cd openssl-0.9.7g
./config
make
make install
```

Установка и конфигурирование Apache2

```
cd /tmp
tar xvfz httpd-2.0.53.tar.gz
cd httpd-2.0.53/
./configure --enable-ssl --with-ssl=/usr/local/ssl/ --enable-suexec --with-suexec-docroot=/usr/local --enable-cgi --enable-rewrite --enable-so --enable-logio --prefix=/usr/local/apache --enable-module=most --enable-shared=max --bindir=/usr/bin --sbindir=/usr/sbin --sysconfdir=/etc/httpd
```

Внимание! Параметры ./configure должны идти одной строкой! Вы можете указать и другие команды конфигурации, которые соответствуют вашим нуждам. Для получения справки по параметрам конфигурации наберите:

```
./configure --help
```

После того как все опции указаны и применены, наберите:

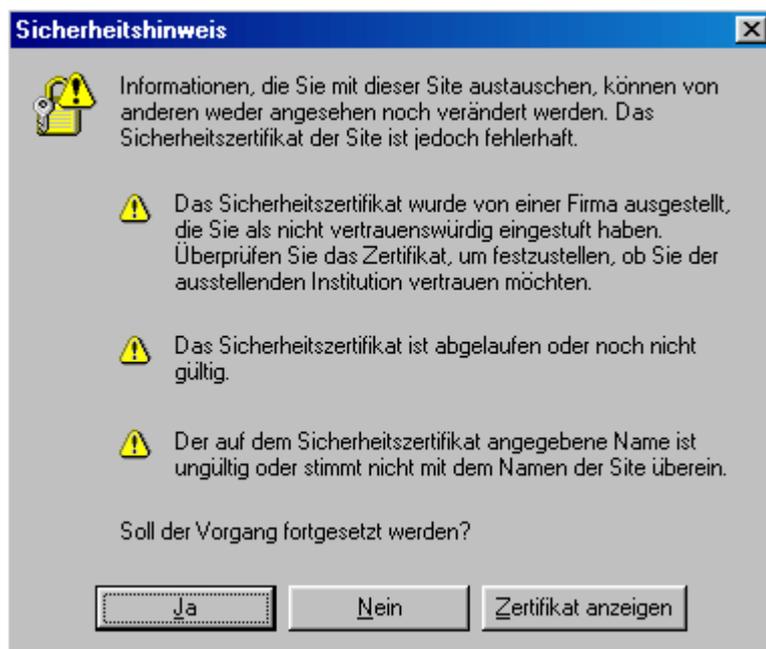
```
make
make install
```

Эти команды установят Apache2 в директорию /usr/local/apache. Корневой папкой для веб-документов будет /usr/local/apache/htdocs, а папкой для логов: /usr/local/apache/logs.

Если существует необходимость запуска Apache2 с поддержкой SSL, то необходимо сгенерировать файл /etc/httpd/ssl.crt/server.crt, иначе при старте Apache может быть получена ошибка.

```
mkdir /etc/httpd/ssl.crt
openssl genrsa -des3 -passout pass:asecretpassword -out /
etc/httpd/ssl.crt/server.key.org 1024
openssl req -new -passin pass:asecretpassword -passout pass:asecretpassword -key
/etc/httpd/ssl.crt/server.key.org -out /etc/httpd/ssl.crt/server.csr -days 3650
openssl req -x509 -passin pass:asecretpassword -passout pass:asecretpassword
-key /etc/httpd/ssl.crt/server.key.org -in /etc/httpd/ssl.crt/server.csr -out /
etc/httpd/ssl.crt/server.crt -days 3650
openssl rsa -passin pass:asecretpassword -in /etc/httpd/ssl.crt/server.key.org
-out /etc/httpd/ssl.crt/server.key
mkdir /etc/httpd/ssl.key
mv /etc/httpd/ssl.crt/server.key /etc/httpd/ssl.key/server.key
chmod 400 /etc/httpd/ssl.key/server.key
```

Рекомендуется принимать все значения по умолчанию при создании файла /etc/httpd/ssl.crt/server.crt, иначе вы будете получать вот такое окно при запуске браузера (а данном случае IE) для доступа к SSL-серверу:



Если вы не хотите получать это сообщение, вам придется получить “настоящий” сертификат (но это стоит денег!) или установить свой сертификат в разряд доверенных вручную и только для своей машины. Для получения дополнительной информации смотрите:

<http://www.instantssl.com/> (очень рекомендуется!)

<http://www.verisign.com/>

<http://www.thawte.com/>

<http://www.baltimore.com/>

<http://www.ipsca.com/>

<http://www.entrust.com/>

<http://www.geotrust.com/>

Установка PHP5

```
cd /tmp
tar xvfz php-5.0.4.tar.gz
./configure --with-apxs2=/usr/sbin/apxs --with-mysql=/var/lib/mysql --enable-
track-vars --enable-sockets --with-config-file-path=/etc --enable-ftp --with-
zlib --with-openssl=/usr/local/ssl --enable-force-cgi-redirect --enable-exif --
with-gd --enable-memory-limit --disable-debug --disable-rpath --disable-static
--with-pic --with-layout=GNU --enable-calendar --enable-sysvsem --enable-sysvshm
--enable-sysvmsg --enable-trans-sid --enable-bcmath --with-bz2 --enable-ctype --
with-db4 --with-iconv --enable-filepro --with-gettext --enable-mbstring --
enable-shmop --enable-wddx --disable-xml --with-xmlrpc --enable-yp --with-zlib
--without-pgsql --enable-dbx --enable-experimental-zts --without-mm --enable-gd-
native-ttf --with-imap-ssl --enable-soap --enable-dbase
```

Внимание! Параметры `./configure` должны идти одной строкой! При конфигурации можно указать и нужные вам параметры. Для справки наберите:

```
./configure -help
```

В PHP5 вы должны указать параметр `--with-mysql[=DIR]` если хотите работать с MySQL, так как по умолчанию PHP5 не поддерживает MySQL и требует настройки до сборки. Ну и конечно, MySQL уже должна быть установлена в системе. Если вы устанавливали MySQL из rpm-пакета, то проверьте что было установлено также соответствующее ПО `mysql-devel`.

Если вы используете `-r-with-gd` и получаете сообщение об отсутствующей библиотеке `libpng`, то установите ее и попробуйте выполнить конфигурационную команду снова. Для установки `libpng` на Debian выполните:

```
apt-get install libpng-dev libpng2 libpng2-dev libpng3
```

Или используйте дистрибутивы rpm, которые можно найти здесь: <http://www.rpmfind.net> или <http://www.libpng.org/pub/png/libpng.html>.

Для продолжения установки PHP выполните:

```
make
make install
```

PHP установится на вашу машину (обычно в `/usr/local/bin/php`) и может быть использован из командной строки, как и в качестве модуля Apache. Теперь необходимо создать конфигурационный файл для `php.ini`.

Наиболее простым путем будет использование поставляемого вместе с PHP файла `php.ini-dist`:

```
cp /tmp/php-5.0.4/php.ini-dist /etc/php.ini
```

Теперь можно отредактировать `php.ini`, если вам нужны какие-либо особые настройки.

Конфигурация Apache

Теперь, добавим в `/etc/httpd/httpd.conf` несколько записей в соответствующие его секции (ищите по аналогии с уже существующими записями).

```
AddHandler cgi-script .cgi
AddHandler cgi-script .pl
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
AddType application/x-httpd-php .php .php5 .php4 .php3
```

Далее, создадим `/etc/init.d/httpd` с кодом:

```
#!/bin/sh

case "$1" in
start)
    /usr/sbin/apachectl startssl
    ;;
stop)
    /usr/sbin/apachectl stop
    ;;
restart)
    $0 stop && sleep 3
    $0 start
    ;;
reload)
    $0 stop
    $0 start
    ;;
*)
echo "Usage: $0 {start|stop|restart|reload}"
exit 1
esac
```

Выполним команду:

```
chmod 755 /etc/init.d/httpd
```

Теперь, чтобы Apache стартовал при загрузке системы, выполним следующее:

```
ln -s /etc/init.d/httpd /etc/rc2.d/S20httpd
ln -s /etc/init.d/httpd /etc/rc3.d/S20httpd
ln -s /etc/init.d/httpd /etc/rc4.d/S20httpd
ln -s /etc/init.d/httpd /etc/rc5.d/S20httpd
ln -s /etc/init.d/httpd /etc/rc0.d/K20httpd
ln -s /etc/init.d/httpd /etc/rc1.d/K20httpd
ln -s /etc/init.d/httpd /etc/rc6.d/K20httpd
```



```
tar xvfz ioncube_loaders_lin_x86.tar.gz
cd ioncube/
mkdir /usr/local/lib/ioncube
mv ioncube_loader_lin_5.0.so /usr/local/lib/ioncube/
```

Теперь отредактируйте `/etc/php.ini` и добавьте строчку:
`zend_extension=/usr/local/lib/ioncube/ioncube_loader_lin_5.0.so` в самом начале:

```
[PHP]
zend_extension=/usr/local/lib/ioncube/ioncube_loader_lin_5.0.so
```

Zend Optimizer

Скачайте последнюю версию Zend Optimizer с сайта http://www.zend.com/store/free_download.php и сохраните ее в `/tmp/` директории.

```
cd /tmp/
tar xvfz ZendOptimizer-2.5.7-linux-glibc21-i386.tar.gz
cd ZendOptimizer-2.5.7-linux-glibc21-i386/data/5_0_x_comp/
mkdir /usr/local/lib/Zend
mv ZendOptimizer.so /usr/local/lib/Zend/
```

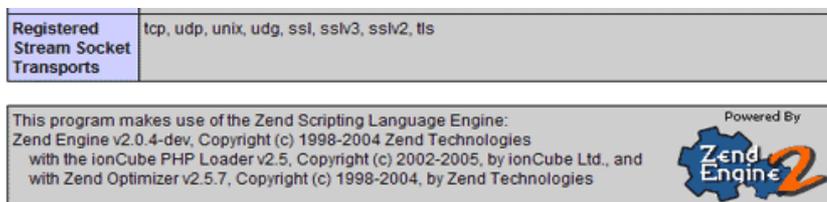
Отредактируйте файл `/etc/php.ini` и добавьте две или более строчки в секцию `[PHP]`. Теперь начало `php.ini` может выглядеть вот так:

```
[PHP]
zend_extension=/usr/local/lib/ioncube/ioncube_loader_lin_5.0.so
zend_extension=/usr/local/lib/Zend/ZendOptimizer.so
zend_optimizer.optimization_level=15
```

Для того, чтобы изменения вступили в силу, перезагрузите Apache2:

```
/etc/init.d/httpd restart
```

Если на данном этапе вы запустите созданный ранее PHP-скрипт `info.php`, то сможете увидеть в браузере примерно следующее:



PHP Credits

Apache: <http://www.apache.org/>
OpenSSL: <http://www.openssl.org/>
PHP: <http://www.php.net/>
Zend: <http://www.zend.com/>
ionCube: <http://www.ioncube.com/>

Работа с CVS в Zend Studio 4.x (Windows)

Вы много раз пытались заставить свою Zend Studio работать с CVS на платформе Windows? После прочтения этой статьи данная проблема не вызовет у вас затруднений. Для примера, мы настроим вашу Zend Studio для работы с www.sourceforge.net.

Сначала перечислим несколько пакетов, которые понадобятся нам в дальнейшем:

- Аккаунт на sourceforge.net – если у вас такового нет – зайдите на сайт и создайте. Также убедитесь, что вы зарегистрировали свой проект, после того как создадите аккаунт.
- Zend Studio (далее – ZDE) – у автора статьи версия 3.5.1, у меня – 4.0 (www.zend.com).
- Putty и несколько утилит из этого пакета: `putty.exe`, `pagent.exe`, `plink.exe`, `puttygen.exe` (<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>).

Теперь у нас есть все для настройки и мы пошагово пройдем с вами весь путь.

ШАГ 1.

Запустите `putty.exe`, чтобы связаться с сервером:

```
Host: projectname.sourceforge.net
Port: 22
Protocol: SSH
```

Соединитесь с сервером нажав кнопку “Open”. В диалоге введите свой логин и пароль. Как только вы активировали ваш SSH аккаунт, была создана домашняя директория.

ШАГ 2.

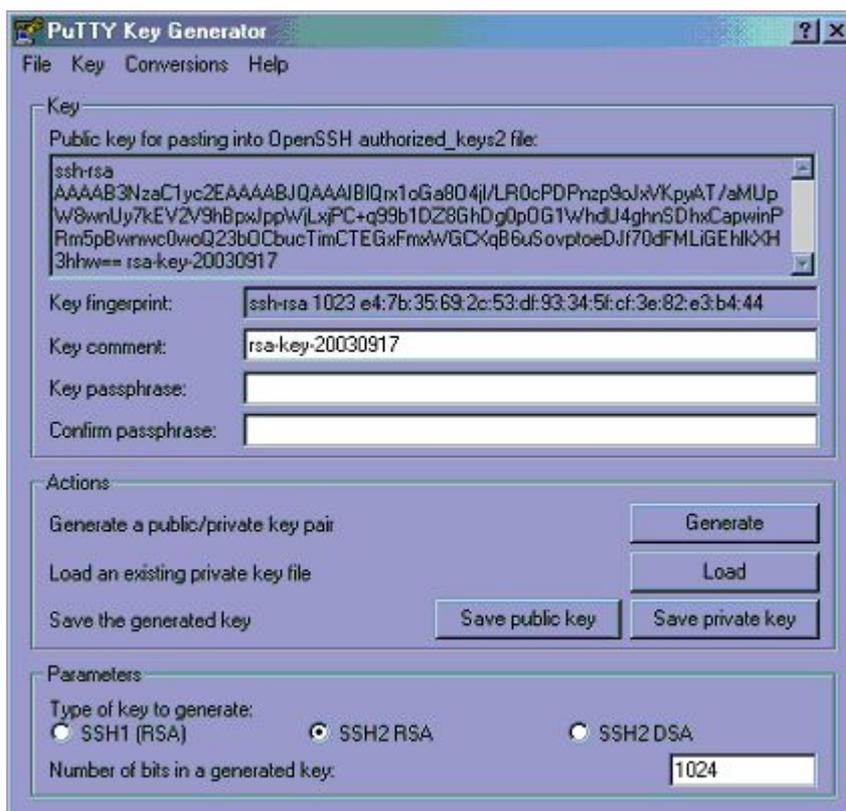
Теперь мы должны создать публичный/приватный (Public/Private) ключевую пару, она нужна, чтобы более не вводить постоянно свой пароль при запуске CVS команд.

Запустить: `puttygen.exe`

Выберите: SSH2 RSA (мне пришлось использовать SSH2 DSA, что не внесло никаких изменений в ход настройки).

Теперь нажмите кнопку “Generate” и поводите беспорядочно мышкой на пустой части окна для генерации случайных данных. Как только данный шаг закончен – вы увидите следующую картинку. (От переводчика: у меня несколько иная ситуация – ключи уже были созданы ранее. Поэтому я воспользовался функция экспорта ключей: “Conversion” -> “Import key”. После импорта генерировать ключ уже не надо. Далее процедура идет также как и у автора).

Автор: Joseph Crawford Jr
Перевод с комментариями:
Кузьма Феськов
<http://php.russofile.ru>
kusma@russofile.ru
Оригинал:
<http://www.weberdev.com/ViewArticle/415>



Теперь сохраните сгенерированный ключ “Save private key” и “Save public key”. Если вы защитите ваш приватный ключ паролем – вам придется каждый раз вводить его в последующих операциях. Вы можете оставить поле пароля пустым.

PuTTYgen.exe нам более не нужен – закрываем.

Теперь вам необходимо связаться со своим пользовательским (не проекта) аккаунтом на сервере и загрузить туда свой публичный ключ.

Чтобы проделать это вы должны пройти на my.sf.net area, нажать на кнопку “Account options” и найти раздел “Host Access Information”. В этом разделе вы можете добавить и загрузить свой ключ. После чего вы должны подождать. Ваши изменения вступят в силу примерно через 6 часов.

ШАГ 3.

Настраиваем ваш приватный ключ. Запустите `pageant.exe` – она появится в трее. Нажмите на иконке правой кнопкой и выберите вкладку “Add key”. Укажите программе ваш приватный ключ и нажмите “Open”. Если вы ввели пароль – у вас его запросят. Каждый раз при запуске Windows вы должны будете повторять этот шаг.

ШАГ 4.

Далее мы настраиваем переменные, которые понадобятся ZDE для работы с CVS.

Запускаем “My Computer” (Мой компьютер) -> “View System Information” (Информация о системе) -> “Advanced” (Дополнительно) -> “Environment Variables” (Переменные среды) (За правильный русский перевод в скобках не ручаюсь – у меня отсутствует русская Windows).

Вам необходимо создать переменную:

Переменная: CVS_RSH

Значение: путь до plink.exe (например, c:\putty\plink.exe)

После создания переменной перезапустите все открытые приложения, включая ZDE.

ШАГ 5.

Настраиваем ZDE. Запускаем “Tools” -> “CVS” -> “Checkout”.

В появившейся форме заполните диалоговые поля:

```
CVS Root: :ext:username@cvs.sourceforge.net:/cvsroot/projectname/  
Module name: Module name (у моего проекта, например, webdev)  
Working directory: папка на вашем диске, где будут храниться файлы.
```

Далее жмем “OK” и все должно заработать!

Отрисовка связанного дерева с помощью XSLT

Одна из вещей, которая мне нравится в XML и связанных с ним технологиях - это то, что там всегда можно научиться чему-то новому. Я использую XML уже более трех лет, и чуть ли не каждую неделю я с удовольствием обнаруживаю что-нибудь новенькое, что можно делать с его помощью, или, даже лучше, более лёгкий способ сделать что-то, в чем, по моему мнению, я уже разбираюсь.

Одно из таких открытий произошло недавно в связи с отрисовкой иерархий типа "связанное дерево". Связанное дерево - это дерево, в котором узлы соединены со своими детьми и родителями линиями, как это часто изображают в файловых менеджерах или оглавлениях (см. пример на рисунке 1). Приложение, над которым я сейчас работаю, использует такой тип отображения для различных целей, что затрудняет реализацию его для различных типов данных. Однако, все типы иерархий доступны в виде DOM-деревьев. Это привело меня к мысли, что если я смогу реализовать отрисовку связанных деревьев только с помощью XSLT, то смогу избавиться от большого количества кода для отрисовки деревьев, заменив его одним XSLT-шаблоном, тем самым сильно упростив наш код.

В результате несколько сот строк подверженного ошибкам Java-кода (разбросанного по нескольким классам) были заменены на несколько дюжин строк XSL в единственном шаблоне, что повысило надежность, простоту поддержки и внесения изменений. В этой статье я расскажу, как мы это сделали.

Я уверен, что большинству читателей уже известно, что XSL и XSLT - это два связанных стандарта описывающих преобразование одних и тех же данных из XML в какой-либо другой требуемый формат. XSL-шаблон является своего рода инструкцией для поиска данных в исходном XML, их преобразования и вывода преобразованной версии. У этой технологии существует множество применений, но, вероятно, самым распространенным является преобразование XML в HTML, подходящий для просмотра человеком.

Еще одна спецификация, называемая XPath, широко используется в XSLT. XPath - это функциональный язык для поиска информации в дереве XML. В XSLT он играет роль фильтра по образцу для поиска в документе данных, которые надо преобразовать.

Лично для меня, XPath стал наиболее сложной для понимания частью XSLT-инструментария. Одна из сторон XPath, доставившая мне особенно много неприятностей - это использование "осей" для того, чтобы "гонять" XSLT-процессор по документу в разных "направлениях". Описанная здесь техника основана на использовании трех различных осей и я надеюсь, что она послужит полезным примером мощи осей тем кому они, как и мне, оказались не по зубам.

Статья предоставлена:

<http://detail.phpclub.ru>

Перевод: Михаил Корнеев,
Денис Юрашкун

Новости phpinside.ru

Gubed PHP Debugger для отладки PHP-скриптов

<http://gubed.mccabe.nu/>

Gubed PHP Debugger - это, как понятно из названия, программный продукт для отладки PHP-скриптов. Gubed распространяется в исходных кодах (тарбол для Linux) и в бинарном виде (инсталлятор для Windows) под лицензией GPL.

Для работы с отладчиком не требуется вносить какие либо изменения в серверное ПО, и, что не менее важно, вообще не нужно никаких компонентов PHP и веб-сервера.

Скачать Gubed PHP Debugger можно с его сайта на портале sourceforge.net.

Дерево из четырех палочек

Если вы внимательно рассмотрите Рисунок 1, вы заметите, что для того, чтобы нарисовать дерево нам нужно только четыре фигуры. Во-первых, каждый элемент присоединяется к дереву с помощью горизонтальной линии, идущей от центра вправо. Если элемент не является последним ребёнком своего родителя, он соединяется с деревом с помощью фигуры, похожей на повернутую на 90 градусов влево букву "Т". Последний ребёнок присоединяется к дереву при помощи фигуры, похожей на "L".

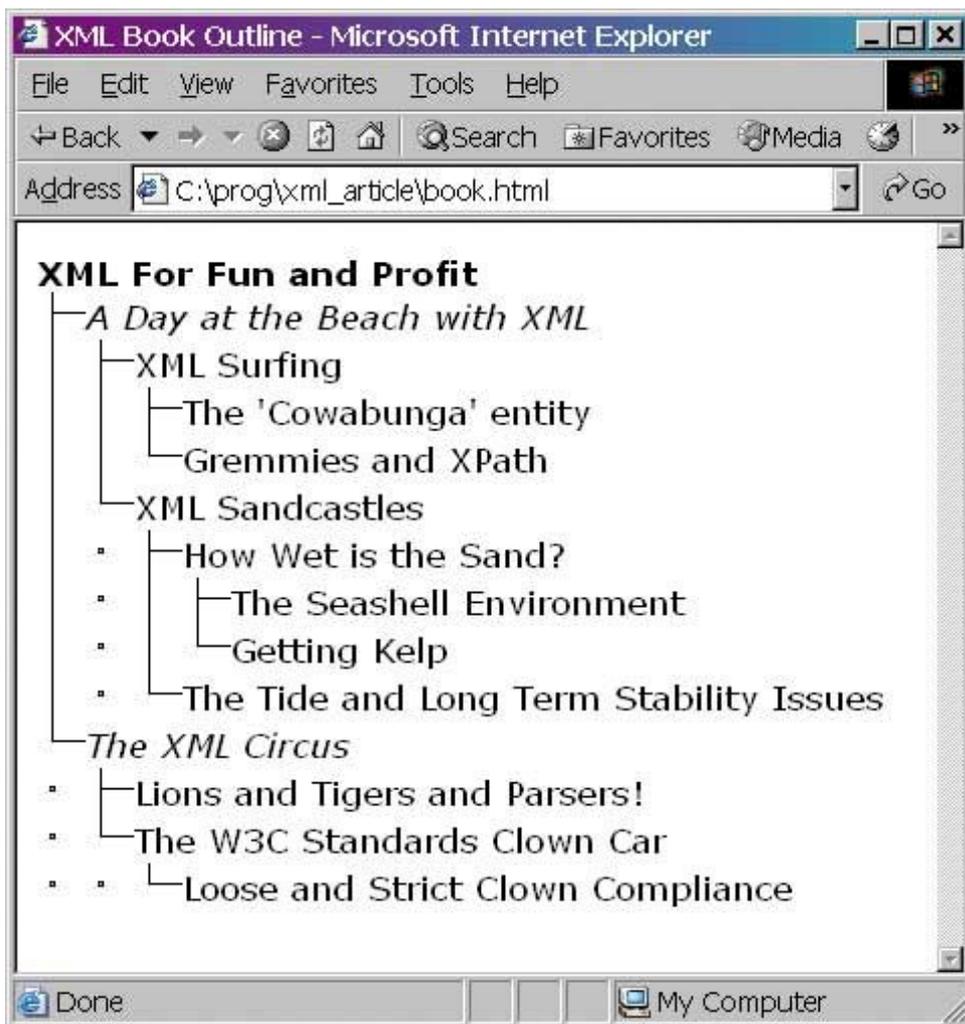


Рисунок 1.

Если элемент расположен в дереве глубже первого уровня иерархии, то слева от описанных выше соединителей появится один из двух других, по одному на каждого предка элемента. Если предок не является последним ребёнком своего родителя, то рисуется вертикальная линия, которая вместе с другими такими же линиями соединяет элемент со следующим элементом на этом уровне. Если элемент является последним у своего родителя, отображается пустая картинка-"распорка".

Для этой статьи я нарисовал четыре картинки: `tree_tee`, `tree_corner`, `tree_bar`, и `tree_spacer` (см. Рисунок 2, их также можно скачать по адресу www.sys-con.com/xml/sourcec.cfm). Все они соответствуют этим описаниям, я только добавил точку в середину пустого разделителя, чтобы было видно, когда он используется.

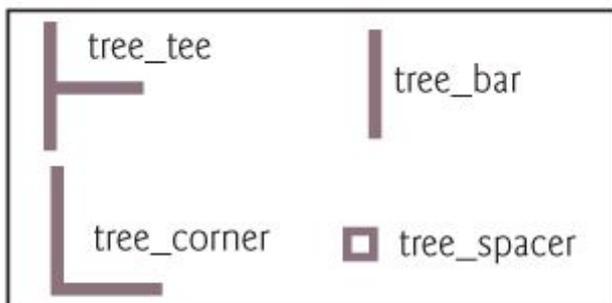


Figure 2 • Tree components

Рисунок 2.

Теперь, проанализировав проблему, мы можем описать примерный алгоритм отрисовки дерева:

- Для каждого уровня от начала дерева до родителя текущего элемента рисуем пустой разделитель, если это последний элемент у своего родителя, и вертикальную линию если нет.
- Рисуем "L"-соединитель если текущий элемент является последним у своего родителя, и "T"-коннектор если нет.
- Отображаем название текущего элемента.

Один из моих сослуживцев красноречиво назвал последовательность картинок, составляемую на шагах 1 и 2 "заборчиком", что достаточно хорошо отражает действительность - вы как бы строите заборчик, в котором штакетинами служат наши картинки, по одной штакетине на каждый уровень, начиная от корня. Самая правая картинка всегда будет либо "T" либо "L" соединителем, а расположенные левее - либо вертикальными палочками, либо пустыми разделителями. Также мы используем термин "отображение элемента" для обозначение того, что рисуется справа от "заборчика" - в нашем случае это будет просто название элемента, отформатированное в соответствии с типом элемента. Поэтому, мы можем описать алгоритм так: "Для каждого элемента дерева рисуем "заборчик", потом отображаем элемент".

Но, как обычно, вся сложность заключается в мелочах. Поэтому давайте посмотрим, как нам реализовать XSLT.

Детали

Исходный документ, показанный на Рисунке 1, называется "book.xml" вы найдете его в Листинге 1. Это очень простой документ, содержащий в себе оглавление книги. В корневой элемент книги вложены элементы - главы, а в них, в свою очередь, вложены секции, в которые также могут быть вложены секции. У каждого элемента есть атрибут `title`, который используется для отображения названий в дереве.

Как видите, этот документ очень прост, но принципы преобразования будут работать и для гораздо более сложных документов.

Стиль, который мы применим к XML-документу, называется "connected_tree.xsl", и он показан в Листинге 2. И снова, этот шаблон искусственно сделан простым, но с помощью идей, проиллюстрированных в нем, вы сможете делать гораздо более сложные преобразования.

Строчки 1-13 XSL-шаблона содержат стандартные заголовки, предшествующие любой трансформации в HTML.

Более интересные вещи начинаются со строки 17, в которой создается шаблон для обработки корневого элемента XML-документа. В этом шаблоне мы создаем статический HTML. Один из этих компонентов это встроенный CSS, в котором мы задаем отступы и границы для картинок, чтобы они соединялись между собой без зазоров, а также выравниваем текст по вертикали, чтобы текст казался естественным продолжением крайней правой картинки.

В 28-й строке мы выводим название книги (значение атрибута title элемента book), заключенное в теги `...`, чтобы выводилось жирным шрифтом. После этого мы применяем подходящие шаблоны к его дочерним элементам, используя режим "line". В этом и всех остальных случаях применения шаблонов к дочерним элементам, они будут обрабатываться в порядке следования в документе. Сначала будет происходить обработка элемента, потом его дочерних элементов, и так далее, и таким образом мы получим нужное нам дерево.

Режимы (modes) позволят нам обрабатывать один и тот же элемент по-разному в зависимости от обстоятельств, что, как вы увидите, и является ключом к решению этой задачи. Режим "line" работает с отрисовкой всей линии, включая "заборчик" и отображение элемента. Другие режимы - "item" и "tree" - будут рассмотрены дальше.

Все элементы обрабатываются в режиме "line" одинаково, поэтому у нас есть только один шаблон с режимом "line", который обрабатывает для все элементы, он начинается в строке 36. Внутри этого шаблона мы полностью отрисовываем одну горизонтальную "линейку" нашего дерева. Мы заключаем вывод в `<div>` соответствующего класса, чтобы добиться отсутствия границ и выравнивания текста по вертикали.

Внутри `<div>`-а находятся два блока. Первый, в строке 38, называется "graft", именно он и занимается построением "заборчика", ниже мы обсудим, как это происходит. Второй, в строке 39, снова вызывает `apply-templates` для текущего узла, но на этот раз в режиме "item", а не "line", как раньше. Вне `<div>`, конце шаблона, мы вызываем его для дочерних элементов, продолжая делать полный обход дерева.

Режим "item" используется для упоминавшегося ранее отображения объекта. Это происходит по-разному для каждого типа элементов. В этом примере для каждого из двух типов элементов, упомянутых в этой книге (chapter и section), определен свой шаблон для режима "item" (строки 46-52). Как я уже говорил - мы попросту выводим title объектов с различным форматированием для каждого из типа.

Итак, мы наконец подошли к самому интересному. Я уже упоминал выше, что шаблон "graft" (начинающийся со строки 57) строит "заборчик" для вызвавшего его элемента. Как он это делает? Чтобы понять это, надо начать думать в обратном направлении - справа налево. Поэтому сейчас перейдите сразу к строке 62, на которой мы рисуем коннектор, соединяющий элемент со всем остальным деревом. Как мы отмечали выше, это будет либо "L"-соединитель, если это последний дочерний элемент у своего родителя, либо "T"-соединитель. Именно так и работает этот код, и самое главное здесь это понять, как определяется, что данный элемент является последним ребенком своего родителя.

Для того, чтобы сделать это, мы должны использовать другую XPath-ось. Обычно при работе используется ось child, которая выбирает дочерние узлы текущего узла. Однако, помимо нее существует множество других осей, позволяющих перемещаться по документу в разных "направлениях". Первая ось, которую мы обсудим - following-sibling, которая позволяет с помощью которой можно выбрать все дочерние узлы того же родителя, что и текущий элемент, располагающиеся в документе после него.

Эта ось используется в условии в строке 63. Если "following-sibling::*" возвращает непустую выборку, то условие возвратит значение true и мы используем "T"-соединитель, иначе, если выборка пустая, мы используем "L"-соединитель. Ось following-sibling (и парная ей preceding-sibling) очень удобны, если вам нужно сделать что-то со всеми оставшимися элементами своего уровня.

Итак, мы нарисовали самый правый элемент нашего "заборчика". А как насчет всего остального? Если помните, мы должны отрисовать по одному графическому элементу для каждой главы или раздела, являющихся родительскими для данного, причем это будут либо вертикальная линия, либо пустая картинка в зависимости от того, следуют ли за этим родительским элементом другие элементы на том же уровне. Половина решения должна быть очевидна из самой формулировки, но как нам посетить наших родителей, чтобы принять решение?

Ответ прост: с помощью другой оси - ancestor. В строке 59 вы увидите, что мы применяем шаблоны в режиме "tree" к узлам выбранным по оси "ancestor::*". Это значит "выбрать всех предков текущего узла, от корня и ниже, за исключением самого узла" А это именно то, что нам нужно, чтобы сгенерировать оставшуюся часть дерева, что мы и делаем в двух последних шаблонах, помеченных для использования в режиме "tree".

Первый шаблон, в строке 74, просто запрещает подстановку картинки для корневого узла (в нашем случае book). Мы уже имели дело с этим элементом в шаблоне "/book" ранее, поэтому нет смысла трогать его еще раз.

Второй шаблон, использует для выбора между картинкой с вертикальной линией и пустой картинкой логику, основанную на использовании оси following-sibling, которую мы уже обсуждали раньше.

Путешествие в мир XSLT продолжается

Вот и все! Теперь, применив XSLT-шаблон к исходному XML, мы получим HTML-дерево, показанное на Рисунке 1. Есть много способов сделать это, я для таких экспериментов обычно использую Apache Xalan из командной строки. Работает это следующим образом (конечно, если вы его уже скачали, установили и внесли в CLASSPATH):

Не забудьте, что здесь описан только один из способов отрисовки связанного дерева, и существует еще множество способов подстроить это под себя, расширить, улучшить и как-то ещё "повернуть в руках". По моему собственному опыту, лучший способ разобраться в XSLT - больше экспериментировать, поэтому призываю вас "поиграть" с кодом connected_tree.xsl и book.xml и посмотреть, что получится. Многое в XSLT и XPath может поначалу показаться чересчур сложным и непонятным, но позже вы с удивлением обнаружите, что на самом деле это очень даже разумные подходы. А после этого сможете и сами поделиться интересными находками с другими.

Об авторе

Craig Berry работает главным архитектором в PortBlue Corporation (www.portblue.com), софтверной компании из Лос-Анджелеса, разрабатывающей продвинутую экспертную систему, основанную на технологии J2EE. Он работает в области проектирования и разработки ПО уже 20 лет, в самых различных областях, начиная от искусственного интеллекта и заканчивая распределенным потоковым видео.

Исходники

Листинг 1

```
<?xml version="1.0"?>
<book title="XML For Fun and Profit">
<chapter title="A Day at the Beach with XML">
<section title="XML Surfing">
<section title="The 'Cowabunga' entity"/>
<section title="Gremmies and XPath"/>
</section>
<section title="XML Sandcastles">
<section title="How Wet is the Sand?">
<section title="The Seashell Environment"/>
<section title="Getting Kelp"/>
</section>
```

```
<section title="The Tide and Long Term Stability Issues"/>
</section>
</chapter>
<chapter title="The XML Circus">
<section title="Lions and Tigers and Parsers!"/>
<section title="The W3C Standards Clown Car">
<section title="Loose and Strict Clown Compliance"/>
</section>
</chapter>
</book>
```

Листинг 2

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output
method="html"
encoding="UTF-8"
indent="yes"
version="1.0"
omit-xml-declaration="yes"
media-type="text/html"
standalone="yes"
/>

<!-- Шаблоны для генерации текстового контента -->

<xsl:template match="/book">
<html>
<head>
<title>XML Book Outline</title>
<style type="text/css">
body { font-size: smaller }
div, img { border: 0px; margin: 0px; padding: 0px }
div.Node * { vertical-align: middle }
</style>
</head>
<body>
<b><xsl:value-of select="@title"/></b>
<xsl:apply-templates mode="line"/>
</body>
</html>
</xsl:template>

<!-- Рисуем каждую строку дерева -->

<xsl:template match="*" mode="line">
<div class="Node">
<xsl:call-template name="graft"/>
<xsl:apply-templates select="." mode="item"/>
</div>
<xsl:apply-templates mode="line"/>
</xsl:template>

<!-- Отображаем различные типы элементов -->

<xsl:template match="chapter" mode="item">
<i><xsl:value-of select="@title"/></i>
</xsl:template>

<xsl:template match="section" mode="item">
<xsl:value-of select="@title"/>
</xsl:template>
```

```
<!-- Шаблоны, используемые для генерации "заборчика" из различных соединителей -->

<xsl:template name="graft">
<!-- Отрисовываем картинку-соединители для всех предков -->
<xsl:apply-templates select="ancestor::*" mode="tree"/>

<!-- Рисуем коннектор для текущего узла -->
<xsl:choose>
<xsl:when test="following-sibling::*">

</xsl:when>
<xsl:otherwise>

</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Запрещаем отрисовку соединителя для корневого узла -->

<xsl:template match="book" mode="tree"/>

<!-- Рисуем соединители для всех остальных узлов -->

<xsl:template match="*" mode="tree">
<xsl:choose>
<xsl:when test="following-sibling::*">

</xsl:when>
<xsl:otherwise>

</xsl:otherwise>
</xsl:choose>
</xsl:template>

</xsl:stylesheet>
```