Спикльмайр С., Фридли К., Спикльмайр Д., Брэнд К.

ZOPE

Разработка Web-приложений и управление контентом

УДК 004.738.5
ББК 32.973.26-018.1
С72

Спикльмайр С. и др.
С72 Zope. Разработка Web-приложений и управление контентом: Пер. с англ. – М.: ДМК Пресс. – 464 с.: ил. (Серия «Для программистов»).

ISBN 978-5-94074-189-3

Обьектно-ориентированный сервер приложений Zope – стремительно развивающееся средство, основная задача которого заключается в быстрой разработке и поддержке корпоративных сайтов. Zope позволяет легко оснащать сайты всеми необходимыми функциональными средствами за счет повторного использования решений, оформляемых в виде тиражируемых продуктов. На сегодняшний день создано множество продуктов, предоставляющих как типовые решения – форумы, опросы, электронные магазины, так и инструменты для программирования и интеграции с другими средами – коннекторы к базам данных, языки описания шаблонов и скриптов. Встроенные в Zope средства управления доступом могут применяться при решении сложных проблем Web-разработки.

Книга ориентирована на специалистов, столкнувшихся с необходимостью быстрого развертывания сайта. Здесь рассмотрены свыше 50 продуктов Zope, позволяющих решать типичные проблемы Web-разработки; приведены рекомендации по их использованию и интеграции с другими системами, а также по написанию и отладке собственных Zope-продуктов.

ISBN 0-7357-1110-0 (англ.)
ISBN 978-5-94074-189-3 (рус.)

Содержание

Глава 1. Цель Web – оптимизация Web-разработки

В этой главе приведен обзор концепций, связанных с публикацией объектов, и даны примеры создания динамических Web-страниц с помощью Zope.

Zope – среда публикации объектов

Даже если вы знаете, как расшифровывается Zope (Z Object Publishing Environment – Среда публикации объектов Z), то все равно нужно выяснить, что же стоит за этим набором слов. Все в Zope так или иначе связано с объектами, но это вовсе не означает, что для работы с этой средой необходимо быть опытным объектно-ориентированным программистом. На самом деле в разработке Web-приложений на основе Zope можно зайти довольно далеко, вообще не думая об объектах. Даже многие поставщики контента не знают, что сайт, куда они ежедневно направляют материалы, работает под управлением Zope. Тем не менее для вас было бы лучше понимать, как объектно-ориентированные механизмы помогают добиться такой мощности и гибкости, которая с трудом достижима иными способами. В части IV мы познакомим вас с тонкостями создания сложных Web-приложений, а пока начнем с трех постулатов, которые можно считать аксиомами Zope:

- каждая страница является результатом работы процесса, именуемого *публикацией объектов*;
- каждая страница состоит из *отдельных элементов*;
- создание страницы *автоматизировано*.

Чтобы стало понятнее, сравним типичный «заход» на страницу традиционного Web-сайта с тем, что происходит в *процессе публикации объектов*. Вы знаете, что простая Web-страница – это не более чем файл, то есть обычный текстовый документ, хранящийся в каталоге на диске.

Например, если вы зайдете на страницу по адресу http://www. PlainOldWebSite.com/home.html, то браузер пошлет запрос в домен PlainOldWebSite, а находящийся по этому адресу Web-сервер найдет файл home.html, откроет его и считает содержимое. Затем вашему браузеру, возможно, находящемуся на другом краю света, будет отправлен поток битов, который на принимающей стороне будет восстановлен в исходном виде и выведен на экран монитора. Впрочем, осознание того факта, что Web начинался как разновидность копирования файлов, не умаляет важности мгновенных глобальных коммуникаций.

Публикация – это процесс

При сопоставлении традиционного Web-сайта с процессом публикации объектов выявляются фундаментальные различия. Рассмотрим простейшую ситуацию, типичную для Zope. Когда браузер запрашивает страницу http://www.ObjectOfTheWeb.com/MyHome.html, публикатор объектов Zope (Zope Object Publisher, или просто ZPublisher), находящийся на сайте ObjectOfTheWeb, начинает выполнять *метод* render объекта MyHome.html. В объектно-ориентированной терминологии метод – это способ *что-то сделать*. В данном случае результат представляет собой страницу, скомпонованную в соответствии с «определением» *контента* страницы, а в процессе компоновки, скорее всего, будут созданы дополнительные объекты и вызваны их методы. По завершении процесса страница отсылается назад браузеру, и этот шаг ничем не отличается от традиционной схемы.

Браузер получит самую обычную Web-страницу, содержащую разметку на языке HTML. Это и неудивительно, поскольку только такие страницы браузер и способен отображать.

Но при этом ваша персональная страница портального сайта может разительно отличаться от того, что увидят по тому же адресу другие посетители. Отличия обусловлены настройками, которые вы задавали в момент регистрации. Страница может также включать часто обновляемую информацию, например последние новости или заголовки присланных вам почтовых сообщений. Сегодня это уже не вызывает удивления – для многих популярных порталов такая практика считается стандартной. Если вы собираетесь включить в свой сайт механизм динамического создания страниц, то данная книга – как раз то, что вам надо. Если же вы пока опасаетесь, что это слишком сложно, сообщим вам приятную новость: превращение сложных вещей в простые – одна из самых сильных сторон Zope.

Динамическая генерация страниц

Важно сразу понять, что при заходе на сайт http://www.ObjectOfTheWeb.com вы получаете не просто копию хранящегося на диске файла, а результат работы некоего серверного процесса. Запрос страницы MyHome.html приводит к вызову метода render объекта MyHome.html, который генерирует свое представление на языке HTML. Публикация объекта – это процесс *компоновки и выдачи* Web-страницы. Вот почему запрос файла /MyHome.html может возвращать совершенно разные страницы разным посетителям сайта.

Компоновка страниц по запросу

Сравнивать статическую страницу с процессом публикации объектов – примерно то же самое, что сравнивать готовое платье с пошитым на заказ костюмом. Поскольку каждая страница создается заново в момент обработки запроса, то система может позволить себе скомпоновать страницу специально для вас. Осознав это, вы поймете, почему в среде публикации объектов так естественно выглядят динамические, интерактивные функции. Когда страница *компонуется по запросу*, добавление «кусочка» персонализированной или часто обновляемой информации

«в общий котел» – это не более чем включение еще одного элемента в страницу, которая и так целиком составлена из подобных элементов. *Элементами* мы и будем называть в этой книге любой обособленный публикуемый фрагмент.

Как построить Web-сайт

Следующая аксиома, еще более важная на практике, гласит, что в процессе публикации объектов генерируются страницы, составленные из дискретных элементов. Это означает, что каждый элемент может независимо создавать и обновлять человек (или люди), отвечающий за *представление* и *контент* сайта. Например, часто на всех страницах сайта можно встретить стандартный баннер и навигационное меню. В среде Zope элементы страницы формируются по отдельности и вставляются на свое место в процессе публикации объектов.

Встроенный графический интерфейс для управления контентом

Обычно пользовательский интерфейс для поставщиков контента в Zope строится в виде Web-приложения на основе форм. Сам браузер является средством предоставления графического интерфейса не только для посетителей Zope-сайта, но и для тех, кто создает и обновляет контент.

Это подводит нас еще к одной уникальной особенности Zope – встроенным инструментам администрирования и создания контента, известным под общим названием *интерфейс управления Zope* (Zope Management Interface – ZMI), которые сами работают в среде Web и являются примером использования пронизывающей всю Zope парадигмы *через Web* (Through the Web – TTW). Эта парадигма иллюстрирует, какой мощности и гибкости можно достичь, встраивая Web непосредственно в процессы сопровождения и управления сайтом.

Конструируя Web-сайт таким образом, мы даем поставщикам контента возможность вносить изменения в любой элемент в удобное для них время, не пользуясь никакими программами, кроме браузера. Этот подход называется *интерактивным управлением контентом*.

Отправляя очередной запрос, посетитель сайта увидит последнюю версию элемента на всех страницах, где он присутствует, тогда как остальные элементы могли не измениться. Это *динамическая публикация объекта*.

Какой бы удобной ни была парадигма TTW для посетителей и поставщиков контента, это лишь один из интерфейсов, доступных разработчикам на платформе Zope. В главе 2 вы еще узнаете об использовании протокола FTP для взаимодействия с Zope и познакомитесь с недавно появившимся стандартом WebDAV.

Элементы аналогичны SSI, только лучше

Поскольку каждый элемент страницы можно обновлять и публиковать независимо, то на ум приходит аналогия между Zope и другой распространенной технологией разработки Web-приложений – *включением на стороне сервера* (Server Side Includes – SSI). Технология SSI упрощает вставку стандартных элементов во все страницы сайта. Однако среда Zope не подвержена некоторым ограничениям,

затрудняющим использование SSI. Например, было бы идеально иметь в точности одну активную копию каждого элемента, входящего в состав сайта, и при этом гарантировать, что всегда публикуется текущий элемент. SSI решает эту задачу, но требует, чтобы каждая страница явно ссылалась на включаемый элемент. Если включаемый элемент часто изменяется, то приходится обновлять все ссылки на него. Zope позволяет избавиться от этого утомительного занятия, поскольку содержит механизм динамического выбора элемента. Возможность ссылаться на последнюю версию элемента – это один из принципов правильной организации в системах управления базами данных и в компьютерном программировании. Процесс публикации объектов пользуется динамическими возможностями Web для расширения идей SSI далеко за пределы типичной практики применения. В результате самые передовые принципы становятся достоянием всех разработчиков Web-приложений.

Чтобы понять, как можно применить публикацию объектов в типичной ситуации, представьте себе некий коммерческий сайт. Подумайте, как упростилась бы жизнь, если бы художник мог заменить старый баннер новой рекламой путем простой подстановки одного элемента вместо другого, не задумываясь о том, на каких страницах он встречается. Точно так же было бы несложно заменить навигационное меню на всех страницах, не прекращая работу сайта. Такой динамический, интерактивный способ работы играет важнейшую роль в управлении контентом, а весь его потенциал раскрывается именно в объектно-ориентированной среде.

Правила заимствования

Приглядимся внимательнее к стандартным элементам, обсуждаемым в этой главе. Например, баннер сайта часто дополняется подбаннерами, обозначающими различные разделы сайта. Методика публикации объектов позволяет строить целые разделы сайта, в которых всегда отображается правильный подбаннер, не указывая при этом, какой подбаннер должен появляться на каждой странице. Достаточно просто поместить ссылку на объект подбаннера в определение страницы, и нужная картинка будет выбрана в процессе публикации объектов. Возможны и более точные ссылки вплоть до баннера, отображаемого на конкретной странице, причем глубина размещения элемента в иерархии страниц сайта не имеет ни малейшего значения. Тот же принцип применим к контенту любого типа, включая графику, сценарии, звуковое сопровождение, видео и т.д.

Интеллектуальный выбор элементов

Наверняка вы уже догадались, что «правильное» решение задачи – это на самом деле результат применения третьей аксиомы, описывающей *автоматическую компоновку страниц*. Процесс публикации объектов следует строгим правилам выбора элементов, размещаемых на конкретной странице. Эта процедура, называемая *заимствованием* (acquisition), возможна только в контексте процесса и неприменима к статическим страницам. Публикатор ZPublisher представляет собой постоянно находящееся в памяти ядро; он отыскивает каждую страницу, а также

все элементы, от которых она зависит, и подготавливает их для компоновки. Это можно сравнить с работой хорошего официанта, который вовремя замечает, что вам надо долить свежего кофе, а также следит за многими другими деталями и печется обо всем тихо и незаметно. Развивая эту аналогию, можно сказать, что необходимость наполнить пустую чашку – это подразумеваемое или неявное действие, не требующее специального явного приглашения. Отношения между элементами и страницами, в состав которых они входят, так же очевидны для ZPublisher, как пустая чашка для официанта.

Нетрудно предположить, что за всем этим стоит весьма сложный механизм, но на практике поведение выглядит очень простым и предсказуемым. Когда ZPublisher видит ссылку на объект в определении страницы, он пытается найти этот объект. Сначала он просматривает ту папку Zope, в которой находится само определение страницы. Папка Zope напоминает каталог файловой системы в том смысле, что является контейнером для группирования взаимосвязанных объектов. Если объекта в этой папке не оказалось, ZPublisher продолжает поиск. На следующем шаге просматривается «родительская» папка, то есть та, что содержит папку, с которой начался поиск. Затем ZPublisher поднимается вверх по иерархическому дереву, пока не найдет объект. Наличием такого простого, но в то же время мощного механизма поиска объясняется простота конструирования сайтов с минимальным дублированием ресурсов. При компоновке страницы на ней будет опубликован первый найденный ZPublisher объект с именем, упомянутым в определении страницы. Если объект изменится, то новая версия будет выведена при следующей публикации любой ссылающейся на него страницы.

Заимствование можно прояснить, представив, что страницы Zope заимствуют элементы из среды, в которой они появились, причем эта среда динамически собирается ZPublisher: страницу можно сравнить с хамелеоном, который меняет свою окраску с зависимости от окружения. Публикатор ZPublisher просто отвечает на запросы, автоматически размещая на публикуемых страницах правильные элементы. По мере обретения опыта вы привыкнете воспринимать это как должное.

Заимствование аналогично наследованию

Бросается в глаза сходство между заимствованием и идеей «наследования» в объектно-ориентированном программировании. Однако механизм заимствования встроен в Zope, и для его применения не нужно вникать в детали определения классов и усваивать прочие сложности. Мы уже видели, как заимствование применяется для выбора нужного баннера. Точно так же с помощью этого механизма можно включать в навигационное меню дополнительные пункты в зависимости от текущего раздела сайта. Такой уровень сложности редко бывает необходим, но трудно удержаться от искушения, раз уж вы поняли, насколько это просто!

Подробнее о работе механизма заимствования мы поговорим в главе 9. Если вас заинтересовала общая идея и другие возможные применения этого механизма, обратитесь к статье Джозефа Джила (Joseph Gil) и Дэвида Г. Лоренца (David H. Lorenz), с которой все и началось (http://www.ccs.neu.edu/home/lorenz/papers/oopsla96).

Поведение и свойства объектов

Поведение в этом контексте – это почти то же самое, что компьютерное приложение, чего и следовало ожидать, поскольку Zope – это сервер Web-приложений, то есть готовый каркас для разработки систем. Приятный побочный эффект применения Zope для построения сайтов, обладающих функциональностью приложения, состоит в том, что законченный продукт уже располагает встроенной системой управления контентом, снабженной графическим интерфейсом пользователя. Универсальность Web привела многих опытных разработчиков к выводу, что современные настольные (desktop) приложения будут эволюционировать в сторону сетевых (webtop). Каждый, кому приходилось администрировать большое число настольных компьютеров, объединенных в сеть, сразу поймет все преимущества запуска приложений в окне стандартного браузера, доступного в любой точке Земного шара!

Группировка взаимосвязанных элементов

Прежде чем двигаться дальше, вернемся на минутку к технологии SSI. Вы можете определить страничные элементы, например баннер или меню, так, что они будут состоять из других элементов. Более того, поскольку на каждой странице, вероятно, присутствует какой-то объект баннера наряду с другими элементами, было бы удобно агрегировать несколько стандартных элементов в некий объемлющий «макроэлемент». Такого рода составные объекты и механизм группирования нескольких объектов в единое целое мы будем называть соответственно *пакетами* (wrappers) и *упаковкой* (wrapping). Позже вы увидите, как можно применить пакеты, заимствование, объекты и методы для того, чтобы легко и быстро создать сложный Web-сайт или законченное Web-приложение.

Упаковка – это технология SSI, дополненная объектно-ориентированной инкапсуляцией

Идея упаковки восходит к известным принципам объектно-ориентированного программирования, в частности к практике *инкапсуляции* базовой информации в *суперклассе*. Хотя сама идея проста, в типичном сценарии применения SSI все сколько-нибудь сложное оказывается либо нереализуемым, либо с трудом поддающимся сопровождению. На наше счастье, техника заимствования и публикации объектов намного упрощает дело, и скоро вы в этом убедитесь.

Уникальные комбинации

В среде, обеспечивающей естественную поддержку повторного использования элементов, можно создать произвольное число страниц, содержащих уникальные комбинации конечного числа элементов. На самом деле это довольно типичная ситуация, особенно в порталах, позволяющих пользователю самому определять, какие элементы должны отображаться на его персонализированных страницах.

Алхимия заимствования

Трактовка Web-сайта как набора элементов, на первый взгляд, не слишком отличается от восприятия пользователем страницы на экране браузера. Однако в действительности это очень далеко от того, как HTML-редактор отображает «исходный текст» статической страницы. Вообразите, насколько проще было бы обновлять отдельные элементы и управлять их совокупностью, если бы была гарантия, что каждый элемент сам найдет свое место на странице. Возможно, вы раньше не задумывались о декомпозиции сайта на элементы и такой взгляд для вас непривычен, но на объектно-ориентированном жаргоне для подобного подхода есть специальный термин – *факторизация*. Смысл его в том, что, начав рассматривать элементы, составляющие сайт, как дискретные сущности, вы должны более тщательно анализировать место каждого элемента в общей картине.

Сайт крупным планом

Перестав рассматривать отдельные картинки и ссылки на странице глазами посетителя, подумайте о том, из каких *факторов* состоит каждый раздел, а затем представьте целостную картину всего сайта. Обозревая весь сайт, вы обнаружите, к примеру, элемент-баннер, который можно считать отдельным фактором. Поскольку баннер есть почти на каждой странице, то его следует включить в пакет, определяющий элементы на самом верхнем уровне.

Факторы уровня раздела должны находиться в пакете раздела и т.д. Продолжая в том же духе, вы постепенно начнете понимать, какому уровню иерархии сайта должен принадлежать каждый элемент, чтобы определение страницы можно было выразить максимально просто. Удачная факторизация сайта позволит существенно упростить и удешевить его сопровождение. Среди экспертов по Zope ходят легенды о том, как удавалось создавать гигантские сайты в рекордно короткие сроки. Эффективный дизайн сайта упрощает работу с ним как посетителям, так и персоналу группы сопровождения. Доказательством ценности постоянного интерактивного обновления информации могут служить общее число заходов и количество повторных визитов одних и тех же посетителей. Чем проще управлять сайтом, тем более актуальную, полезную и ценную информацию можно на нем предложить.

Время подвести итоги

Мы уже сформулировали несколько ключевых идей, которые, возможно, оказались для вас внове. Коротко суммируем изложенный материал.

Вы познакомились с основными аксиомами Zope:

- каждая страница является результатом работы процесса, именуемого *публикацией объектов*;
- каждая страница состоит из *дискретных элементов*, которые *обновляются интерактивно*;
- страница компонуется *динамически*, причем *элементы включаются неявно* благодаря механизму *заимствования*.

Теперь у вас есть некоторое представление о взаимозависимостях между этими аксиомами. Разумеется, на статических страницах невозможно было бы в любой момент отображать самую последнюю информацию, например заголовки новостей или почтовых сообщений. Необходимо, чтобы некоторый *процесс* автоматически *заимствовал* нужные элементы на страницу, обеспечивая ее актуальность. Только *интерактивный* подход к управлению контентом способен обеспечить работу такой *динамической* системы. Итак, мы подготовили фундамент для знакомства с простыми примерами, на которых изложенные идеи будут продемонстрированы в действии.

Пакеты – взгляд изнутри

В этой книге мы не станем углубляться в детали языка HTML, хотя он используется во многих примерах. Для начала приведем пример очень простой HTML-страницы, а затем покажем, как добиться того же результата с помощью публикации объектов в Zope. Если предположить, что на каждой странице сайта PlainOldWebSite должен быть стандартный текстовый баннер, то при заходе на этот сайт посетитель увидит нечто подобное тому, что изображено на рис. 1.1.

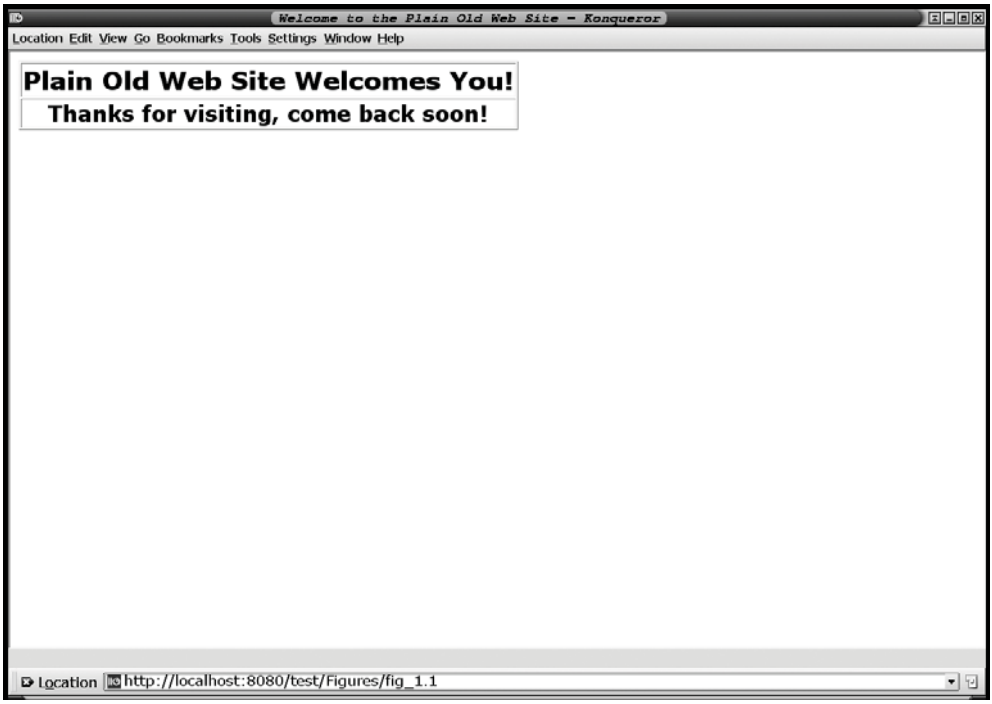

Рис. 1.1. Пример простой Web-страницы

HTML-код такой страницы может выглядеть так:

```
<HTML>
<HEAD>
<TITLE>Welcome to the Plain Old Web Site</TITLE>
</HEAD>
<BODY>
<table border='1'><tr><td align='center'><H1>Plain Old Web Site
➥Welcomes
You!</H1></td></tr>
<tr><td align='center'><H2>Thanks for visiting, come back
soon!</H2></td></tr></table>
</BODY>
</HTML>
```

Этот текст – точная копия файла home.html, хранящегося на сервере Plain OldWebSite.com. Если открыть в браузере исходный текст страницы MyHome.html с сайта ObjectOfTheWeb.com, то ничего нового по сравнению с приведенным выше вы не увидите. Но давайте посмотрим на элементы (объекты Zope), из которых состоит страница. Почти все страницы сайта ObjectOfTheWeb упакованы в объект с именем `standard_dtml_wrapper`. В листинге 1.1 показана начальная версия, которую мы будем совершенствовать в главе 2.

Листинг 1.1. Пример метода на языке DTML – `standard_dtml_wrapper`

```
01 <HTML>
02 <HEAD>
03   <TITLE><dtml-var title></TITLE>
04  </HEAD>
05  <BODY>
06   <dtml-var banner>
07   <dtml-var expr="_[page_body]" fmt=structured-text>
08  </BODY>
09 </HTML>
```

Конечно, это сильно упрощенный пример, но даже он позволяет уяснить некоторые основные положения. Первое, на что следует обратить внимание, – это наличие нескольких элементов, которые выглядят как HTML-теги, но начинаются с префикса `<dtml-`. Они стоят в местах, где вы ожидаете увидеть обычную HTML-разметку, и сопровождаются текстом, который должен быть выведен браузером. Это и есть ссылки на страничные элементы. Далее мы подробно объясним, что означает каждая ссылка.

Объект `standard_dtml_wrapper` – пример так называемого *DTML-метода*. Напомним, что метод – это *способ что-то сделать*, поэтому сейчас мы начнем изучать, «как это делается» в Zope.

Такой незнакомый «почти HTML» не очень-то похож на процесс, правда? А все потому, что собственно процессом публикации объектов занимается ядро ZPublisher, которое интерпретирует содержимое каждого объекта и выполняет

указанное *действие*. Вы можете твердо рассчитывать на то, что Zope поработает для вас интерпретатором, но для начала надо освоиться с терминологией. Аббревиатура DTML расшифровывается как Document Template Markup Language (Язык шаблонов для разметки документа). Этот язык напоминает HTML; по крайней мере, теги начинаются с символа < и заканчиваются символом >. Когда Zope интерпретирует тег `standard_dtml_wrapper` и выполняет содержащиеся в нем инструкции, как раз и осуществляется процесс публикации объектов.

Элементы программирования – присваивание значений переменным

Взгляните на ссылку `<dtml-var` внутри тега `title` в листинге 1.1. Вы ожидаете увидеть в этом месте заголовок страницы в виде строки текста, например «Взгляните на объект Web!». Но вместо этого вашему взору предстает конструкция

```
<dtml-var title>
```

Это ссылка на DTML-переменную с именем `title`. *Переменные* в Zope ничем не отличаются от переменных в языке программирования или в математическом уравнении. В частности, их значения могут изменяться в зависимости от некоторых условий. В данном случае заголовок страницы – это значение переменной `<dtml-var title>`. Основная задача DTML как раз и состоит во включении таких ссылок на страницу, то есть в размещении имен объектов (страничных элементов) в тех местах, где должно появиться содержимое элемента. Это в точности аналогично размещению гиперссылок (`href`) или графики (`img`) на обычной HTML-странице.

Возможно, у вас возник вопрос, откуда берется значение переменной `title`, – ведь у каждой страницы должен быть свой заголовок. Чтобы дать ответ, придется познакомиться с другим типом объектов – DTML-документом. Напомним, что изначально вы запросили страницы DTML-документа по адресу MyHome.html. У DTML-документа с таким именем есть свойство `title`, которое подставляется вместо ссылки `<dtml-var title>` в процессе публикации объектов. Так происходит потому, что DTML-документ *вызывает* DTML-метод. Иными словами, свойства DTML-документа *заимствуются* DTML-методом. Совсем простой DTML-документ может выглядеть, как показано в листинге 1.2.

Листинг 1.2. DTML-документ MyHome.html

```
01 <dtml-let page_body="'MyHome.stx'">
02  <dtml-var standard_dtml_wrapper>
03 </dtml-let>
```

Не впечатляет, верно? Но сейчас вы уже понимаете, что во второй строке находится ссылка на пакет (DTML-метод) `standard_dtml_wrapper`, о котором только что шла речь. В первой строке еще одной переменной (`page_body`) присваивается значение, равное имени другого DTML-документа (MyHome.stx),

который содержит элементы, вынесенные со страницы (пример факторизации). Вы вполне могли бы поинтересоваться, зачем для такой простой страницы нужно три элемента, и спросить, не проще ли обойтись обычным HTML. Что ж, если бы вы всегда создавали только такие тривиальные страницы, то ваши сомнения были бы оправданы.

Но примите во внимание, что пакет используется на каждой странице, поэтому на самом деле появилось только два новых элемента. Полезно также подумать о том, насколько легко можно «включить» существующие элементы в новые страницы, просто вставив обернутые в пакет объекты. Некоторые из таких элементов, например стандартный баннер и меню, упоминались выше. Хотя приведенный пример элементарен, возможности механизма заимствования мы уже реализовали.

Давайте еще немного задержимся на данном примере, поскольку вы только начинаете знакомиться с процессом заимствования. Подчеркнем, что для добавления *новой* страницы к сайту понадобилось выполнить следующие шаги:

1. Создать новый DTML-документ.
2. Включить в него два DTML-тега.
3. Поместить уникальный контент в отдельный DTML-документ (в данном случае MyHome.stx).

Все общие страничные элементы, например баннер, предоставлены готовым пакетом. А теперь спросите себя, удавалось ли вам раньше так легко добавить новую страницу? Если ответ окажется отрицательным, добро пожаловать в мир Zope.

Подготовка к созданию страницы – выявление элементов

Может показаться, что MyHome.html – тоже пакет, но это не совсем так. На самом деле смысл существования MyHome.html – в *инициализации* некоторых полезных переменных, такие как заголовок страницы и переменная page_body, чтобы впоследствии настоящий пакет *позаимствовал* их. Заметим, что сам пакет не изменяется и с помощью такой же техники в него можно поместить произвольную страницу. После того как MyHome.html установил значения переменных и тем самым сообщил пакету, какой заголовок использовать в документе MyHome.stx, все остальные элементы тоже будут найдены и включены в страницу на этапе публикации объектов.

Здесь нет ровным счетом никакого дублирования информации. Имеются общие и уникальные страничные элементы, но каждый в одном экземпляре. Если некоторый элемент будет изменен, то при следующем запросе страницы, включающей этот элемент, будет возвращена последняя версия.

Чтобы уяснить, как это происходит в более сложных примерах, приближенных к реальности, вам понадобится установить Zope и приступить к экспериментам.

Установка Zope

Лучше начинать с последней версии Zope, поэтому отправьтесь на страницу http://www/zope.org и щелкните по ссылке **Download** (Загрузить) на верхней панели. Ближе к началу следующей страницы имеется еще одна ссылка, которая ведет на список заранее подготовленных дистрибутивов Zope. Имейте в виду, что все наши инструкции относятся к той структуре сайта zope.org, которая имела место на момент написания этой книги. Но все течет и меняется, поэтому не исключено, что ссылки окажутся не на том месте и вам придется поискать их самостоятельно.

Если у вас медленное соединение с сетью, вряд ли вам захочется повторять эту процедуру слишком часто. Правда, дистрибутивы Zope «весят» меньше 6 Мб, так что эпизодические обновления вы можете себе позволить. Zope – функционально богатый и гибкий продукт, но объем дистрибутива остается на удивление небольшим по сравнению с объемами многих других серверов приложений. Щелкнув по кнопке загрузки, займитесь другими делами, но не пропустите момент завершения – это произойдет раньше, чем вы думаете. Обратите также внимание, что Zope может работать под управлением разных операционных систем. Выберите именно ту, какая вам нужна, причем полную версию, поскольку на сайте размещаются еще и обновления или «заплатки», которые пригодятся тем, кто хочет всего лишь модернизировать уже имеющуюся версию, а не начинать все заново.

Где найти дополнительную информацию

Как правило, установка и запуск Zope не вызывают затруднений, но, если вдруг возникнет заминка, есть сайты, где можно быстро получить помощь. Главный из них – все тот же zope.org, где есть раздел **How-To** (Как сделать), в котором освещаются начальная установка и множество других вопросов, интересных как начинающим, так и опытным пользователям. Раздел снабжен мощной поисковой системой. Другой популярный ресурс – это архив почтовых сообщений, который поддерживает компания New Information Paradigm (NIP). Он расположен по адресу http://zope.nipltd.com/public/lists.html и тоже допускает поиск.

Поиск можно производить по ключевым словам, а результаты сортировать по дате, автору и теме. Если вы не найдете ответа на свой вопрос, можете отправить сообщение с просьбой о консультации по адресу: zope@zope.org.

Все сказанное относится к любому вопросу, а не только к начальной установке, но лучше сначала поискать ответ самостоятельно. Скорее всего, на ваш вопрос уже не раз отвечали, поэтому просмотрите имеющуюся информацию. Сообщество Zope прекрасно осуществляет техническую поддержку, и скоро вы сами сможете отвечать на вопросы новичков.

Есть несколько вещей, о которых стоит сообщить заранее, чтобы вы не тратили время на поиски ответов. При установке Zope вас либо попросят ввести пароль администратора, либо предложат пароль по умолчанию (в зависимости от платформы и типа инсталляции).

Запишите введенный пароль, чтобы потом не забыть его![1]

Как всегда при загрузке дистрибутива рекомендуется внимательно прочитать файл `read_me`. После этого запускайте команду `start`. В зависимости от быстродействия вашей машины при первом запуске Zope, возможно, придется немного подождать.

Порт по умолчанию – 8080

Возможно, вы ожидали, что вместе с Zope поставляется собственный Web-сервер, а может быть, это будет для вас сюрпризом. Как бы то ни было, порт Zope по умолчанию выглядит несколько необычно с точки зрения большинства пользователей. После IP-адреса необходимо добавить `:8080`. Эта строка сообщает браузеру, что надо соединиться с портом 8080, а не со стандартным для протокола HTTP портом 80. Zope предполагает, что на порту 80 уже может работать другой Web-сервер, поэтому для предотвращения конфликтов выбирает другой порт.

В большинстве систем имеется внутренний возвратный (loopback) адрес, который позволяет приложениям обмениваться данными с сервером, работающим на той же машине, даже если в ней нет сетевой карты. Это означает, что браузер может «видеть» Web-сервер Zope на возвратном адресе, несмотря на то что ваш компьютер не включен в локальную сеть. По умолчанию возвратный IP-адрес равен `127.0.0.1`, а его синонимом является `localhost`. Чтобы соединиться с Zope по этому адресу, наберите один из двух URL: http://localhost:8080 или http://127.0.0.1:8080. Чтобы перейти на административный интерфейс, добавьте к URL строку `manage`, например: http://127.0.0.1:8080/manage.

В некоторых настольных системах для успешного запуска Zope следует активизировать сетевые службы. Например, попробуйте соединиться с провайдером по модему и запустите Zope. После того как Zope стартует, можете отключиться от сети и далее работать автономно.

При первом запуске Zope вам нужно будет создать свою учетную запись. Для этого щелкните по кнопке **Add** (Добавить), а в следующем окне введите свои имя и пароль дважды, во избежание случайной ошибки. Затем выберите пункт **Manager** (Управление) и нажмите **Add**, чтобы сохранить изменения. Вы увидите иконку папки с именем `acl_users`. Выберите эту папку для создания учетной записи. Затем закройте браузер, снова запустите его и войдите в систему под собственным именем, а не как администратор. Администратору Zope запрещено создавать объекты, но пользователь, наделенный соответствующими полномочиями, может это делать. Войдя в систему под своим именем, вы отдали дань системе безопасности Zope.

Если все прошло гладко, то вы разделили опыт многих тысяч любопытствующих новичков и теперь готовы к работе с Zope.

---

[1] Пожалуйста, из соображений безопасности забудьте этот совет! Запомните пароль и немедленно уничтожьте листочек, на котором вы его записали. – *Прим. науч. ред.*

Интерфейс управления Zope: все под контролем

Человеку, который впервые видит интерфейс управления Zope (ZMI), сразу бросается в глаза навигационная панель слева, вкладки в верхней части страницы и кнопки управления (рис. 1.2).

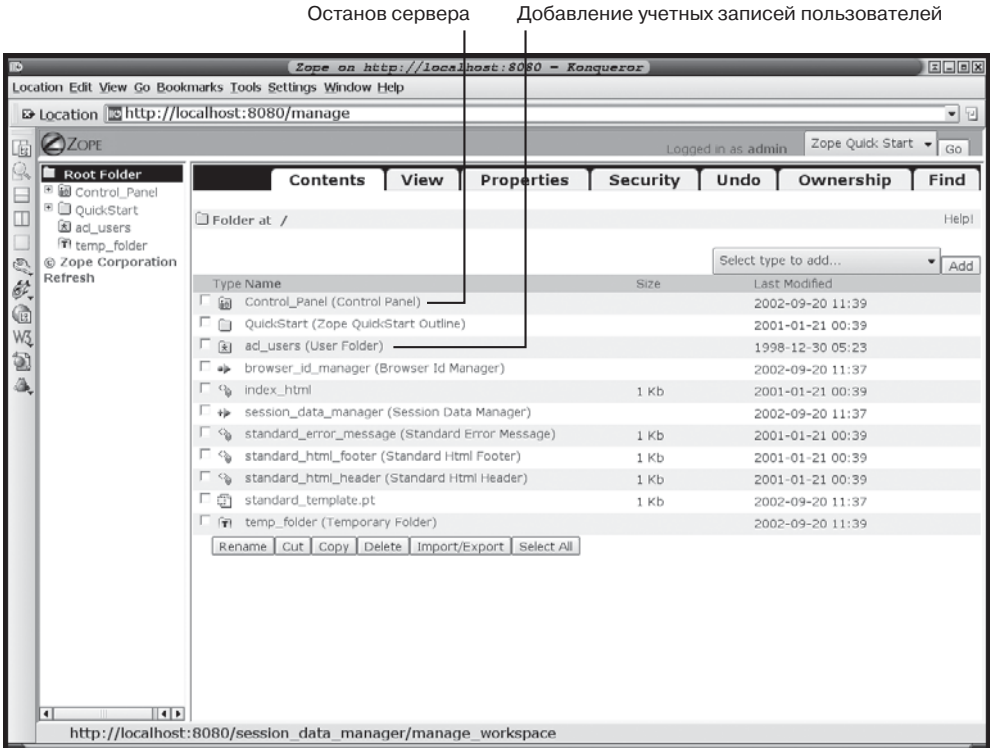

Рис. 1.2. Интерфейс управления Zope (TTW – Through the Web)

Обратите внимание на кнопку **Delete** (Удалить). Вы, наверное, подумали: «Что будет, если я случайно удалю не тот объект?». А если и не подумали, то еще подумаете. Но у Zope на это есть ответ! Щелкните по вкладке **Undo** (Отменить) в верхней части страницы. Если вы выполнили какое-либо «отменяемое» действие, будет показан список, который начинается с самого последнего изменения. Чтобы отменить его, поставьте флажок рядом с верхним элементом списка и нажмите кнопку **Undo**. При необходимости восстановить более старую версию объекта вам придется последовательно отменить все изменения, происходившие позже. Операция эта не из приятных, так что лучше тщательно тестировать каждую версию, чтобы не возвращаться более чем на один шаг.

Вероятно, вы хотели бы также знать, как остановить систему. Щелкните по узлу **Control_Panel** (Панель управления) сразу под корневым узлом **RootFolder**