2003 - Том 6 - Выпуск 6

О реализации службы управления содержанием

Сысоев Т.М., Нестеренко А.К.,
Бездушный А.Н., Серебряков В.А.
ВЦ РАН

Бездушный А.А.
МФТИ

Важная часть любой информационной Web-системы будь-то сайт или портал - это представление информации, данных – содержания информационной системы. Соответственно важнейшей функцией Web-системы является управлять этим содержанием. Служба управления содержанием призвана обеспечить сквозное управление единицами содержания, их взаимосвязями, образуемыми ими структурами и существенно снизить трудоемкость сопровождения информационной Web-системы. Как правило, службы управления содержанием осуществляют манипулирование слабоструктурированной информацией, отличающейся нерегулярностью взаимосвязи ее элементов, редкостью изменений, такой как информационно-публицистические материалы сайта, пресс-релизы. Вследствие этого они обычно обеспечивают иерархическую каталогизацию данных одного типа – разделы/документы. В системе ИСИР [4-7] мы стремимся в рамках службы управления содержанием обеспечить разностороннюю поддержку информационного содержания порталов и сайтов. Мы полагаем, что служба должна включать не только средства управления неструктурированной информацией, но интегрироваться со средствами управления структурированной информацией, предоставлять средства сопряжения слабоструктурированных и структурированных данных, в частности, включения вторых в первые, механизмы атрибутно-полнотекстового индексирования данных обоих видов, поддержки технической и семантической интероперабельности с распределенными источниками содержания Web-системы. В работе рассматривается решения и некоторые технические моменты текущей реализации такой службы системы ИСИР.

Управление содержанием

В рамках Web-системы существует потребность в наборе служб, обеспечивающих высокоуровневую обработку хранящихся в нем материалов. На данные службы возлагаются следующие задачи:

1. **Отображение данных пользователям портала**. Имеется в виду доступ к данным через Интернет по протоколу HTTP. Данные портала можно разделить на две категории: статическую и динамическую. К первой относятся статьи, новости, и другая текстовая информация несложной структуры. Во вторую входит сложноструктурированная информация, хранящаяся, как правило, в базе данных, например информация о проектах, публикациях.

2. **Предоставление заданному кругу лиц возможности по управлению**

материалами, то есть совокупности средств, позволяющих модифицировать и добавлять данные портала. С рассмотренным в данной работе программным обеспечением редакторы так же взаимодействует с помощью протокола HTTP.

3. Пользователям должна предоставляться **возможность поиска информации**. При этом должны поддерживаться различные виды поиска: полнотекстовый, атрибутивный. Результаты поиска должны сортироваться по релевантности.

4. Поддержка **распределенности** предоставления данных - обеспечение технической и семантической интероперабельности с распределенными источниками. Это требование порождает качественно новые задачи, связанные с взаимодействием нескольких серверов. В данной работе эта задача не рассматривается, ей будет посвящена одна из следующих статей.

Служба управление содержанием (CMS) предоставляет пользователям интерфейс для управления содержанием материалов сайта. В первую очередь это относится к статической (относительно редко меняющейся) его части – такой как структура разделов, статьи.

С точки зрения посетителя сайт представляется в виде набора страниц, связанных между собой ссылками. Каждая такая страница описывается на языке HTML (Hypertextmarkuplanguage). Изначально сайты представляли собой коллекцию HTML документов, размещенную в файловой системе. При этом изменять такой сайт мог лишь квалифицированный персонал, знакомый с этим языком разметки.

В процессе роста популярности Интернета, появились специальные программы, предназначенные для визуального редактирования HTML документов (MicrosoftFrontPage, MacromediaDreamweaver, AllaireHomeSite). Работать с такими программами мог значительно более широкий класс пользователей. Однако в настоящее время такой подход используется редко.

Обычно страницы одного сайта имеют похожую структуру и общий дизайн. Это приводит к наличию блоков HTML кода, повторяющегося в каждой странице. В итоге, при хранении сайта как коллекции HTML документов появляется большая избыточность информации. Как следствие такой избыточности, смена оформления или добавление нового раздела могут привести к необходимости одновременных изменений во всех имеющихся страницах. Добавление новой страницы так же должно производиться на основе существующих шаблонов, и может повлечь модификацию других страниц. Это явление характерно для HTML, поскольку в нем описываются как сами данные, так и их внешний вид.

Следующим недостатком является ограниченность взаимодействия такого сайта с пользователями. Не поддерживаются такие функциональности как подбор индивидуальных новостей на основе выбранных зарегистрированным посетителем тем, общение между пользователями - в общем, все что так или иначе связано с автоматической генерацией страниц. Стоит заметить, что подобные операции вообще стали возможны только лишь с повышением вычислительной мощности машин.

Еще одной причиной является неудобная организация процесса редактирования. Все изменения осуществляются одним лицом (или ограниченной группой лиц). В случае с часто обновляющимся сайтом, таким как представительство СМИ, это неприемлемо – у организации может быть штат авторов, но все новости проходят через одного человека, что может привести к задержкам.

Для преодоления этих трудностей создается специальное программное обеспечение, способное автоматически генерировать HTML страницы и предоставляющее возможности по удобному редактированию данных. Как правило, предоставляется разделение между данными и оформлением: редактор работает с данными, дизайнер с шаблонами, и все это преобразуется в HTML документы в процессе генерации сайта или по мере поступления запросов от посетителей.

Такого рода программы (CMS) могут быть как универсальными, при наличии некоторых ограничений, заложенных в их архитектуре, так и создаваться для определенного сайта. В последнем случае они пригодны только для его поддержки.

Особенности универсальных CMS

В настоящий момент существует большое количество (порядка сотни) таких систем, сильно отличающихся по своим характеристикам. В целом в данной области пока не сложилось какого-нибудь общепризнанного стандарта. Существует несколько систем, созданных крупными компаниями (MicrosoftCMS, IBMWebSpherePortal, BEAWebLogicPortal, OraclePortal).

Для большинства подобных продуктов характерно следующее:

- **Удобный интерфейс редактирования данных. Облегчает доступ к системе пользователям без навыков HTML. Как минимум, это осуществляется путем предоставления работы с формами. В то же время интерфейс может повторять знакомые пользователям офисные приложения, либо быть интегрированным с ними.**
- **Разделение данных и дизайна. Требование следует из предыдущего. Часто встречающийся, и наиболее эффективный по скорости обработки подход – использование шаблонов страниц. Например, можно хранить шаблоны, в котором “помечены” места для вставки изменяемых частей (набранного редактором текста). Стоит отметить, что формат таких шаблонов меняется от системы к системе. Набирает популярность альтернативный подход, использующий язык преобразования XML документов (XSLT). Как правило, он менее производителен, но обладает рядом достоинств, из которых следует отметить его стандартизацию.**

- **Система прав. Доступ к системе управления ограничивается, как правило, с использованием процедуры авторизации. Вводятся понятия пользователя, группы пользователей. Пользователям и группам назначаются права доступа к материалам сайта.**
  Система может предоставить посетителям сайта возможность зарегистрироваться, в результате чего им становятся доступны дополнительные возможности. Права доступа в этом случае могут применяться и к материалам сайта, в этом случае часть материалов становится доступной посетителям только после регистрации.

- **Аудит. Система может регистрировать все производимые в ней изменения. Это позволяет, например, собирать статистические данные по активности редакторов или узнать, кто последним модифицировал какой-либо документ. Кроме того, информация, доступная при наличии такого сервиса, может быть полезна другим частям системы. Например, на основе списка добавленных и измененных за заданный период ресурсов может осуществляться их индексирование и обмен.**

- **Организация совместного доступа. При одновременной работе нескольких редакторов может возникнуть ситуация, когда два человека пытаются одновременно изменить один и тот же документ. Для корректной обработки такой ситуации существует несколько моделей разрешения конфликтов. Чаще всего используется модели блокировки (в момент редактирования материала он недоступен другим редакторам), и уведомлений. В случае отсутствия обработки конфликтов, изменения, сделанные раньше, теряются. Иногда это приемлемо.**
  Хорошая реализация совместного доступа может зависеть от решения достаточно сложных проблем – таких, как версионность материалов.

- **Открытость архитектуры / расширяемость. В универсальной системе трудно учесть все требования, которые могут возникнуть у пользователей, поэтому полезно предоставить средства для написания собственных расширений. Часто предоставляется документация по системе, и специальный язык программирования, предназначенный для создания динамических страниц. Некоторые системы предоставляют возможности по созданию собственной модели данных (онтологии), с которой работает пользователь.**

- **Стандартные приложения. Помимо публикации материалов, существует много стандартных для WEB задач. Как правило, они**

выполняются подключаемыми к системе модулями, использующими ее программный интерфейс (API). Наиболее распространенными задачами является поддержка форумов для общения посетителей, предоставление посетителям комментировать опубликованные материалы, рассылки публикуемых новостей по электронной почте, опросы посетителей. В качестве примера более сложного приложения можно привести систему торговли через интернет.
Хорошо продуманный интерфейс к системе и ее популярность стимулирует разработчиков к созданию под нее приложений. Так, например, в настоящий момент для IBM WebSpherePortal существует более 200 подобных расширений, созданных более чем 70 компаниями.

- **Способ хранения данных. Как правило, используется только база данных или база данных совместно с файловой системой.**

Различия в архитектуре.

Для нас важным является анализ различий между этими системами, поскольку он позволяют оценить несколько альтернативных подходов к решению части задач.

Способ взаимодействия с посетителями

Есть два крайних подхода: сайт в виде коллекции HTML документов предварительно генерируется и затем выкладывается на сервер, либо система самостоятельно обрабатывает запросы от посетителей [2, 3]. Возможен так же смешанный подход, когда часть документов генерируется заранее, а часть – по мере поступления запросов от пользователя.

В первом случае достигается весьма высокая скорость работы и легкость внедрения, поскольку не требуется специального программного обеспечения помимо стандартного WEB сервера. При этом, однако, теряется множество дополнительных сервисов. Фактически, все, что говорилось о недостатках статических сайтов, применимо к этому случаю, за исключением сложности процесса редактирования.

Второй подход лишен этих недостатков, однако более сложен в реализации. Система включает в себя WEB сервер (или интегрируется с существующим сервером). Как только система получает запрос от пользователя, выполняется программный код, который в результате генерирует HTML страницу. Страница может быть сформирована, в том числе, и с учетом предпочтений пользователя.

Кроме дополнительной сложности, генерация HTML документов “на лету” повышает нагрузку на сервер, и, обычно, такие системы выдерживают меньшее число одновременных посетителей. Как решение этой проблемы часто используют кэширование – запоминание результатов сложных вычислений. Это оправдано, поскольку данные в среднем меняются гораздо реже, чем отображаются, и в

случае применения кэша пересчет требуется один раз после их изменения.

Метод разделения данных и оформления

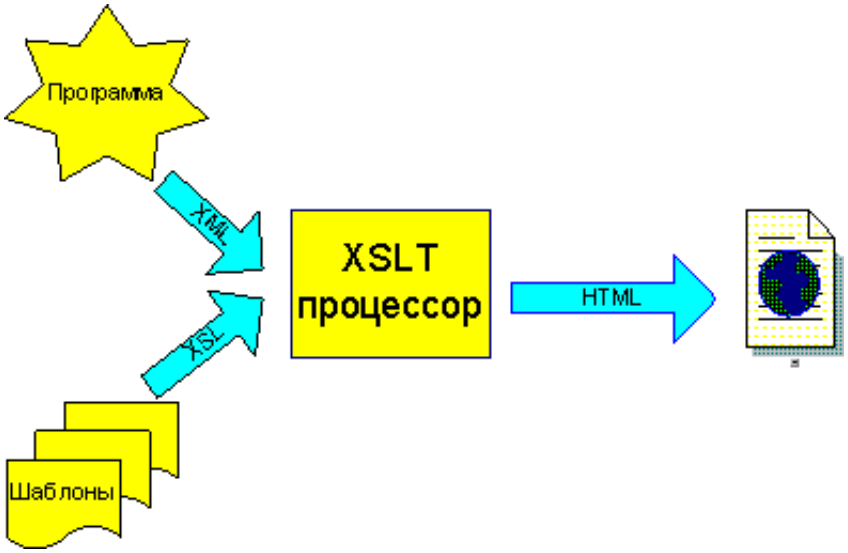
Как уже упоминалось, в настоящее время существуют два основных подхода к генерации страниц: использование шаблонов и XSLT преобразования. Спецификация на XSLT появилась во второй половине 1999 года. Подход с применением шаблонов исторически возник намного раньше, и поэтому используется во многих системах, основанных несколько лет назад.

К достоинствам этого метода можно отнести простоту его реализации. В результате возникло множество несовместимых между собой форматов. В частности, если шаблон является чем-то большим, чем HTML страница с автозаполняемыми полями, то это приводит к необходимости введения каких-то управляющих конструкций. То есть, в шаблоне есть вставки на каком-либо языке программирования.

Использование шаблонов привело к множеству различных языков, функциональности которых во многом пересекаются. Разделение дизайна и данных привело к разделению лиц, работающих над сайтом, на редакторов и оформителей. Последние вынуждены работать с языком шаблонов, предоставляемым системой. Изучение новой системы приводит к необходимости изучения нового (хотя нередко и похожего) языка. Так же возникают сложности при попытке использования существующих шаблонов в других системах. Использование XSLT устраняет часть этих недостатков, и при этом предъявляет дополнительные требования.

XSLT являетсяаббревиатуройдля XSL Transformations, где XSL - Extensible Style Language. Этот язык определяет правила, по которым из одного XML документа получается другой документ (чаще всего в формате XHTML/HTML или произвольный XML). Сами правила описываются так же на языке XML.

Процесс применения XSLT преобразования схематично изображен на следующем рисунке:



Как видно, для использования XSLT необходимо, чтобы программа генерировала данные в формате XML. При этом в созданном документе должны быть все данные, нужные для создания страницы. В то же время, в него не помещается оформление документов – все это заложено в шаблоне. XSLT процессор принимает исходный XML документ, шаблон, и генерирует HTML по инструкциям, заданным в шаблоне. Один и тот же шаблон применяется для генерации различных страниц.

Возможности языка позволяют достаточно сложные манипуляции с исходными данными, благодаря чему можно осуществлять практически любые осмысленные преобразования. Пожалуй, единственным недостатком является более низкая производительность – XSLT преобразование требует больше ресурсов, чем генерация страниц на основе шаблонов. Последнее отчасти компенсируется тем, что современные браузеры способны осуществлять XSLT преобразование на стороне клиента, при этом разгружая сервер.

Права доступа и процесс редактирования

Для разграничения прав доступа и назначения пользователям ролей применяются различные технологии, часть которых описана в этом разделе.

Самая простая из рассматриваемых моделей – список управления доступом (AccessControlList, ACL). В ее рамках с каждым объектом ассоциирована таблица прав, определяющая, какие операции с этим объектом может производить пользователь или группа пользователей. Среди операций обычно встречаются возможность чтения и возможность изменения. Такой подход достаточно эффективен, если объекты образуют иерархию – в этом случае права могут наследоваться, и нет необходимости определять их для каждого объекта.

Данный подход все еще не позволяет распространить на процесс работы над сайтом идеи из более традиционных областей деятельности – например, публикация периодических изданий. В крупных проектах зачастую оправдано назначать пользователям роли – такие как автор, редактор, администратор. Утверждается, что автор может только создать материал, а редактор имеет возможность выложить созданный автором материал в публичный доступ.

Одно из решений заключается в расширении системы прав. Каждому материалу приписывается одно из возможных “состояний”. Права доступа в таком случае определяются ролью пользователя и состоянием материала. Например, если материал имеет два состояния (неопубликован, опубликован), то “автор” имеет право изменять его в первом состоянии, “редактор” – во втором и переводить из первого состояния во второе. В рассмотренном примере достаточно наличия флага “опубликован” у материала, однако добавление состояний (таких как “предложен к публикации”) позволяет моделировать некоторые более сложные процессы, например процесс возврата материала редактором на доработку автору. Данная модель достаточно удобно распространяется и на пользователей сайта – достаточно “гостевому” пользователю задать право на просмотр в состоянии “опубликован”.

Однако наиболее полно такая модель описывается концепцией “рабочих процессов” – workflow. По сравнению с вышеописанным подходом она позволяет ставить дополнительные ограничения и условия, в частности, временные рамки, и в целом более компактно описывать сложные вещи. Существуют стандарты для задания таких процессов, среди которых следует отметить XPDL – XML ProcessDefinitionLanguage.

Центральным понятием концепции является процесс, состоящий из множества “деятельностей” (activity). Если процессу сопоставить ориентированный граф, то деятельности будут соответствовать его вершинам. Для каждой деятельности заданы критерии, определяющие возможность ее начала и завершения. Кроме того, известна информация об участниках деятельности (персоны или приложения). В свою очередь деятельности соединены “переходами” (transitions), которые можно изобразить как дуги графа, определяющего процесс. Каждому переходу сопоставлены условия, при которых он срабатывает. Деятельность может иметь несколько входящих и исходящих переходов, часть их которых может срабатывать одновременно (то есть возможно распараллеливание и слияние процессов). Так же, переходы могут задавать циклы, которые соответствуют повторяющимся видам деятельности.

Видно, что интеграция подобной системы в CMS позволит наиболее гибко, по сравнению с другими описанными методами, контролировать процесс обновления. Лицо или организация, ответственное за наполнение сайта, получает возможность:

- **Задавать сроки на выполнение каждой деятельности и процесса в целом.**
- **Получать уведомления о ходе выполнения процессов**
- **Вовлекать в процесс произвольное количество различных видов деятельности и исполнителей**

Требования и характеристики CMS

От системы управления материалами в основном требуется предоставить визуальный интерфейс к функциям, предоставляемым моделью данных. Сама модель, применяемая для статических данных, фиксирована, и подробно описана в следующем разделе. Для визуализации данных применяется технология XSLT, что нашло свое отражение в модели.

Система должна поддерживать авторизацию и ограничение доступа, основанную изначально на задании прав доступа к объекту.

Существует так же ряд менее важных в настоящий момент требований, реализация которых будет осуществлена в дальнейшем. Прежде всего, это использование службы рабочих процессов (workflow) для организации наполнения и редактирования материала. Кроме этого, полезна поддержка взаимодействия с пользователями (форумы, комментарии, предложение новостей) на уровне CMS.

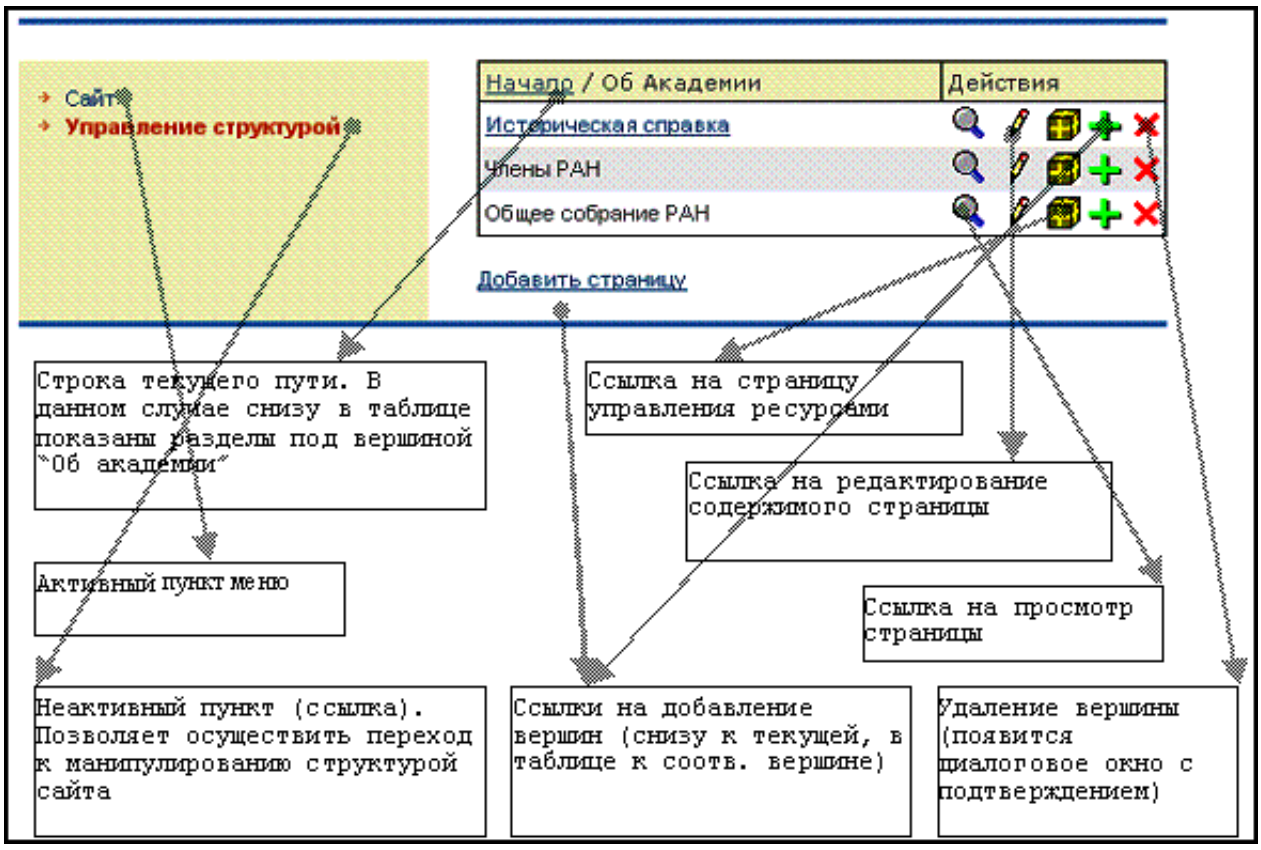
CMS является частью программного обеспечения системы ИСИР. Поэтому она использует ту же платформу и принципы, что и основная часть портала. В качестве платформы выбран язык Java. Выбор был сделан на основе достаточно хорошего знания разработчиками этой платформы и ее приспособленности для подобного рода проектов – крупных решений на базе WEB. В пользу данного выбора говорит так же очень высокая переносимость полученного кода (работа системы проверялась на платформах Windows и Solaris), и доступность большого количества кода, так или иначе связанного с проектом. Последнее позволило не начинать разработку с нуля, а использовать несколько готовых решений для стандартных задач.

Систему в целом можно описать как набор взаимодействующих сервисов, или служб. Для поддержки работы сервисов и их взаимодействия используется JakartaTurbine. Turbine является каркасом (framework) – сама по себе не является законченным продуктом, но достаточно удобна в качестве фундамента. Она определяет, что такое сервис, правила, по которым он должен быть реализован, этапы жизни сервиса (например, инициализация, конфигурация или удаление), содержит управляющий сервисами код и реализацию сервисов общего назначения, полезных в большинстве программ, работающих в WEB.

Интерфейс доступа к модели данных описан в соответствующем разделе. Для отображения форм редактирования форм пользователю используется стандартные для портала программные продукты – в данном случае это ApacheCocoon, которая, хотя и является в первую очередь системой для публикации XML документов, оказалась достаточно удобной для написания под нее управляющего программного кода. В частности, это продиктовано использованием формата XML в самой модели.

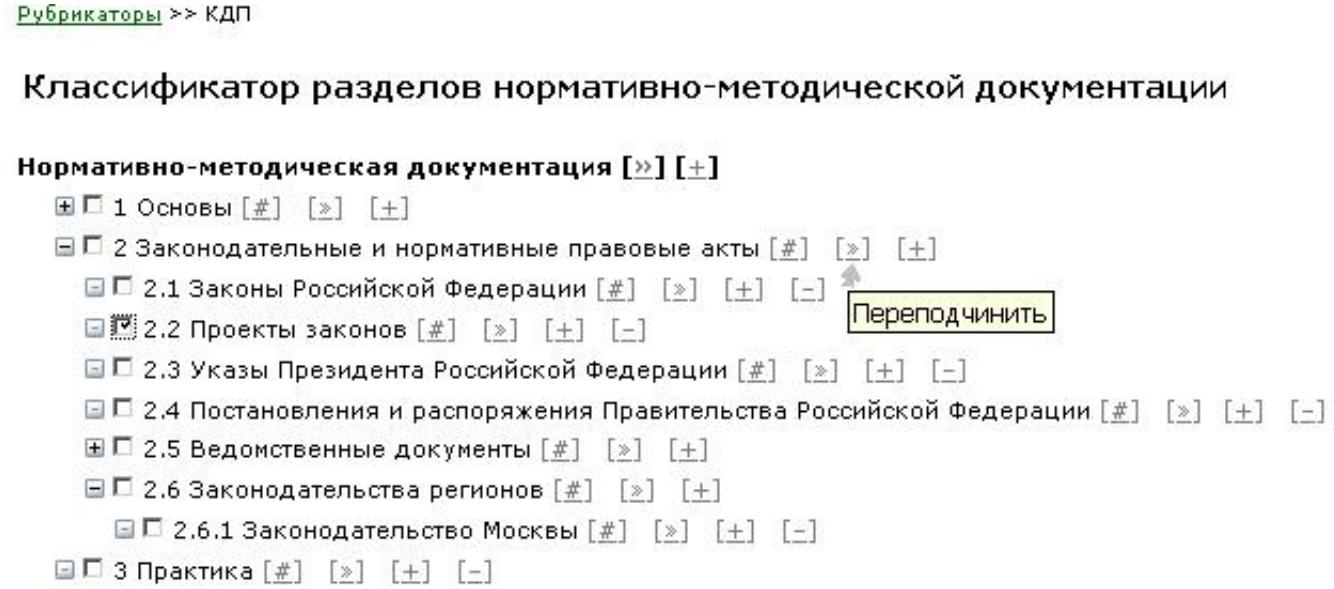
На следующих рисунках приведен внешний вид основных страниц управления разделами и их документами для двух видов визуальных интерфейсов.

Страница управления разделом и его документами:

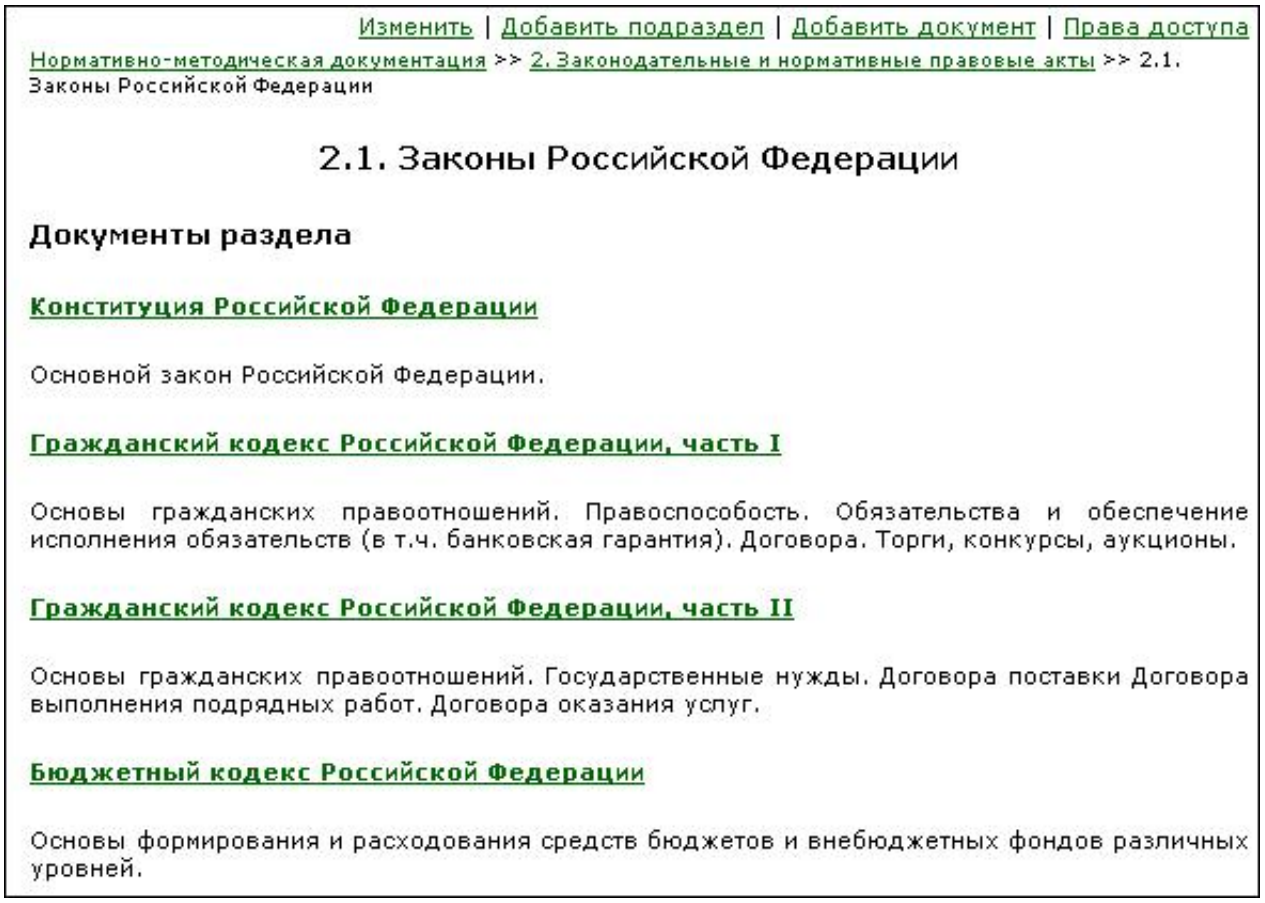


Страница управления структурой разделов:

Страница управления структурой разделов альтернативного интерфейса:



Страница управления разделом и его документами альтернативного интерфейса:

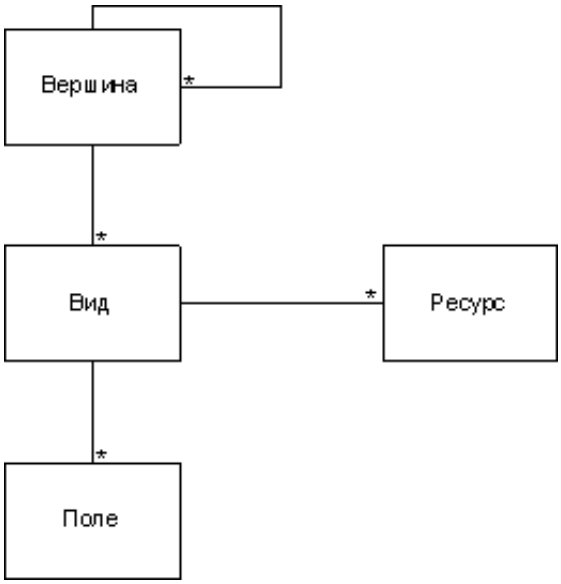


Модель данных

Основные понятия и их связь.

Модель данных является основным сервисом нижнего уровня. При

проектировании учитывались требования поддержки произвольной иерархической структуры материалов и их многоязычности. Основные сущности модели и их взаимоотношения приведены на следующей схеме.



Вершина сайта – основной структурный элемент иерархии. У каждой вершины может быть несколько дочерних вершин. На самом верхнем уровне расположена “корневая” вершина. Вершина непосредственно не содержит каких-либо данных. Все множество вершин соответствует дереву материалов сайта.

Каждой вершине сопоставлено число – ее порядковый номер. Это число используется при сортировке вершин. Если две дочерние вершины имеют в качестве порядкового номера одно и то же число, то порядок их следования не определен.

Кроме этого, у каждой вершины есть название – строка текста, которая используется для ее идентификации. При этом все дочерние вершины любой вершины должны иметь различные названия.

Вид. Понятие вида возникает из требования поддержки многоязычности. Вид содержит данные на определенном языке. Вершина может иметь несколько ассоциированных с ней видов, по одному для каждого языка, на котором представлены данные. Считается, что если два вида принадлежат одной вершине, то они содержат одни и те же данные на разных языках. Вид содержит несколько стандартных атрибутов, а так же набор полей, который задается схемой данного сайта.

Ресурс представляет собой дополнительные данные, ассоциированные с данным видом. Есть два типа ресурсов – произвольные данные, хранящиеся в виде файла на сервере, или ссылка. Например, картинки к статье хранятся как ее ресурсы. Стоит отметить, что ресурс привязан в виду, а не к вершине, благодаря чему есть возможность хранить различные ресурсы для разных языков.

Поле непосредственно хранит данные. Существуют два типа полей – один из них предназначен для хранения произвольной строчки без элементов форматирования, второй – фрагмент в формате HTML или XHTML. Поля первого типа применяются для простых свойств, таких как ключевые слова. Тип полей учитывается при их отображении и индексировании.

Разница между HTML и XHTML определяется автоматически. Основной причиной, по которой был оставлен формат HTML, является поддержка существующих данных, которые имеют преимущественно этот формат. С точки зрения обработки он гораздо менее удобен.

Идентификация вершин

Идентификация вершин необходимо, прежде всего, для организации перекрестных ссылок между ними. Существуют два способа сослаться на вершину: по пути от корня до вершины и по уникальному идентификатору.

Ссылка по имени имеет следующий вид:

<p1>/<p2>/../<pN>

то есть просто перечисление имен вершин, разделенных прямым слешем. Здесь <p1> - имя дочерней вершины корневой вершины, <pN+1> - имя дочерней вершины для вершины с номером N в цепочке.

Такие ссылки достаточно удобны для пользователя, однако, они меняют свой вид при изменении порядка следования разделов или смене имен вершин. В случае, когда подобные операции осуществляются через интерфейс редактирования, замена ссылок осуществляется автоматически.

Ссылка второго типа – это просто значение уникального идентификатора для данной вершины. Идентификатор (GUID) генерируется при создании и остается неизменным при переносах и модификациях вершины, поэтому ссылки, которые его используют, так же остаются неизменными. В то же время с точки зрения пользователя они имеют нечитабельный вид.

Возможно существование вершин без идентификаторов второго типа. Такие вершины появляются, например, при создании XML файла вручную. В этом случае идентификатор будет присвоен при первой модификации файла, произведенной через управляющий интерфейс. Идентификатор хранится как атрибут корневого тэга.

Ссылки на вершины содержатся в полях, имеющих тип HTML или XHTML. Для ссылок используется стандартный синтаксис, за исключением собственно URL, то есть ссылки имеют вид:

<a href=”purl://<идентификатор>”> тест ссылки

где идентификатор задается одним из двух перечисленных выше способов. Такой

синтаксис позволяет достаточно просто обрабатывать ссылки (с использованием регулярных выражений для HTML и простого разбора атрибутов в случае XHTML).

Как видно, в ссылках не специфицируется язык. В то же время, вершина может быть отображена в одном из доступных языков. При переходе по ссылке выбор языка осуществляется в соответствии со следующими приоритетами: текущий язык пользователя, язык по умолчанию, и далее любой из языков данной вершины.

Кроме ссылок на вершины, необходимы так же ссылки на ресурсы. Ресурсы идентифицируются только с помощью GUID.

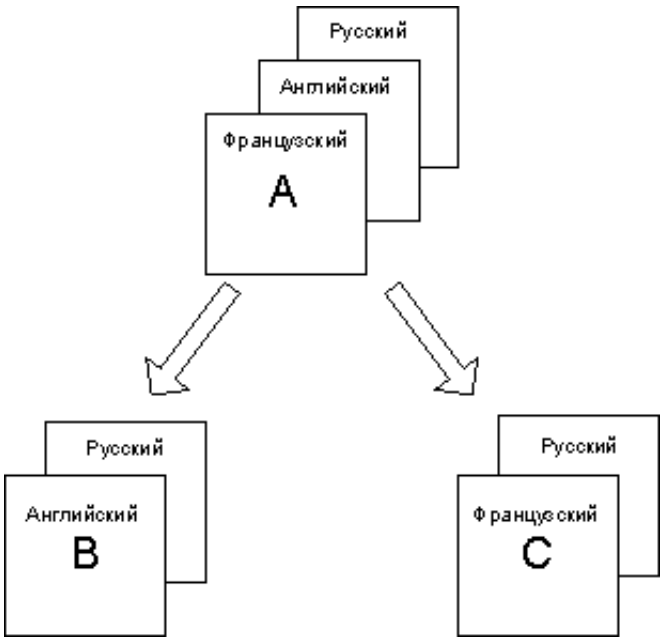
Поддержка многоязычности

Потенциально модель поддерживает произвольное количество языков для каждой из вершин, причем эти множества могут меняться произвольным образом при переходах от вершины к вершине. На практике устанавливаются некоторые разумные ограничения.

Выделяются два множества языков: языки интерфейса и языки данных. Второе множество определяет, на каких языках могут храниться данные. Первое множество содержит доступные языки интерфейса пользователя, при этом оно является подмножеством второго. Например, данные могут храниться на русском, английском и французском языках, а интерфейс пользователя может быть представлен только на русском и английском.

Для каждой вершины заданы языки интерфейса, в которых она может отображаться. При этом справедливо правило: если вершина может быть отображена в интерфейсе, соответствующем некоторому языку, то ее родительская вершина тоже может быть отображена в том же интерфейсе. Таким образом, множество языков интерфейсов, в которых данная вершина отображается, при продвижении вверх по иерархии может только увеличиваться.

В случае, если язык интерфейса не содержится во множестве языков, на которых представлены данные вершины, и в то же время вершина допускает такой язык интерфейса, название материала отображается в навигации с упоминанием языка, на котором будут отображены данные этого материала.



Пример допустимой иерархии изображен на рисунке. Помечены допустимые языки интерфейсов для данных вершин, причем набор языков, на которых представлены данные, может быть любой. Например, данные для вершины B могут храниться на французском языке.

Следует отметить, что карта сайта будет различной для разных языков интерфейса. В приведенном выше примере в карте сайта присутствуют все три вершины в случае интерфейса на русском языке, и по две вершины на английском и французском.

Такие ограничения на языки возникают из требования, чтобы для каждого из языков интерфейса иерархия допустимых вершин представляло собой дерево (была связной).

Поддерживаемые операции с моделью

API модели построен на основе образца проектирования “Команда” [1]. Модель поддерживает следующие операции над данными:

- **Создание новой вершины. Для создания необходимо предоставить название вершины, задать родительскую вершину и все данные на одном из допустимых языков. Система может модифицировать предложенное название для сохранения его уникальности среди дочерних вершин. Созданная вершина содержит один вид, в который заносятся переданные данные.**
- **Редактирование данных. Для операции задается язык и значения полей. В результате для указанной вершины создается новый вид или обновляется значение существующего.**

- **Удаление вершины. Требуется только идентификатор вершины. В результате удаляется вершина, а так же дочерние вершины и все связанные с ней виды и ресурсы.**
- **В случае файловой системы это приводит к рекурсивному удалению каталога, поэтому время выполнения операции пропорционально числу дочерних вершин.**
- **Добавление ресурса. Для нового ресурса задается родительская вершина и все параметры, которые его описывают: язык, название, ссылка (для соответствующего типа ресурсов), файл.**
- **Удаление ресурса. Для удаления ресурс задается родительской вершиной, языком и индексом. С точки зрения вида, ресурсы представляют упорядоченный список, поэтому это правомерно.**
- **Структурные операции (копирование и перенос). По причинам, которые описаны ниже, объединены в одну команду. Операция задается исходной вершиной, вершиной, которая станет родительской для копии исходной, и флагом, предписывающим удалять или нет исходную вершину после копирования. Перенос вершины на концептуальном уровне можно представить как копирование с последующим удалением исходного поддерева, хотя эти операции имеет смысл реализовывать по разному.**

На исходные данные налагается несколько важных условий. Прежде всего, родительская вершина не должна совпадать с исходной или быть ее дочерней вершиной (на любом уровне иерархии), иначе операция переноса теряет смысл. Следующее условие возникает в связи с ограничениями на множество языков: родительская вершина должна позволять все те языки интерфейса, которые позволяет исходная. Если хотя бы одно из этих условий не выполнено, операция не осуществляется.

Существенная разница между переносом и копированием заключается в процедуре замены ссылок. Так, при переносе новые страницы отождествляются со старыми, поэтому все ссылки, построенные с использованием цепочки имен, автоматически заменяются. При этом требуется проанализировать все существующие документы. Уникальные идентификаторы страниц сохраняются. При копировании замен ссылок не производится, и генерируются новые идентификаторы.

Структурные операции требуют достаточно большого количества ресурсов. Они могут приводить к копированию директорий в файловой системе, имеющих

большую вложенность. В то же время операция переноса выполняется значительно быстрее, поскольку не требует замены ссылок и перенос директорий в большинстве операционных систем выполняется эффективно, а в случае хранения объектов в базе перенос приводит к изменению одной строчки (информации о наследовании).

Скорость выполнения операций по копированию и переносу одинакова при различных способах хранения и является обратно пропорциональной числу всех дочерних вершин. В то же время эти операции в случае работы с базой данной выполняются в несколько раз быстрее.

Отображение данных

Отображение данных является одним из наиболее важных сервисов верхнего уровня, основная задача которого – предоставить произвольному пользователю доступ к материалам портала по протоколу HTTP.

Выбор технологии

Рассматривалось несколько вариантов реализации отображения материалов в рамках технологий, используемых в портале. При этом решение должно было обладать максимальной гибкостью, для чего требовалась поддержка XSLT как способа разделения дизайна и данных, и страницы должны были генерироваться на основе запросов клиента (естественно, с возможностью кэширования). В основном, выбор производился между следующими возможностями:

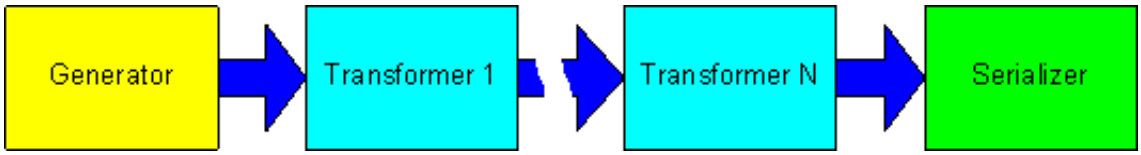
- **Использование технологии портлетов, предоставляемой Jakarta Jetspeed – одного из программных продуктов, используемых в портале.**
- **Использование Apache Cocoon.**

Оба варианта удовлетворяли предъявленным требованиям, но в результате в качестве основы был выбран Cocoon благодаря возможности более простой и изящной реализации. Это можно объяснить тем, что архитектурно Apache Cocoon приспособлен как раз для таких задач (публикация XML данных), в то время как JetSpeed является более низкоуровневым проектом и предназначен, в основном, для разработки сайтов, использующих технологию портлетов. При этом на сами портлеты ограничений практически не накладывается, соответственно нет и высокоуровневых средств, организующих работу с XML данными.

С точки зрения Apache Cocoon, система публикации состоит из набора компонент, каждый из которых предназначен для выполнения определенных действий. Существует несколько типов компонент, из которых для нас наиболее важны генераторы (Generators) и преобразователи (Transformers). Генераторы отвечают за выдачу данных, трансформаторы – за их модификацию.

Еще одна особенность Cocoon – наличие карты сайта. Карта сайта (sitemap)

отвечает за отображение интернет адресов (url) в цепочку компонент (pipeline), в обязанности которой входит выдача документа пользователю. Цепочку схематично изображена на рисунке.



Любая цепочка начинается с компонента, предоставляющего данные (Generator). Сами данные являются XML документом, поэтому есть возможность (и смысл) использовать событийную модель (SAX) для связи между компонентами. Далее события проходят через цепочку преобразователей, и окончательно попадают в компонент, преобразующий их в формат, понимаемый клиентом (обычно это HTML).

Несмотря на кажущееся усложнение процесса (генерации XML с последующим преобразованием в HTML) данная модель очень удобна для внесения изменений. В частности, XSLT преобразование в ней представимо как одна из модификаций SAX событий (один из компонентов с ролью Transformer), благодаря чему есть возможность обработки данных до и после трансформации (возможно, не одной). Это используется, например, при замене ссылок.

Кроме того, потоковая (событийная) модель обладает тем преимуществом, что клиент может получить часть результата, в то время как исходные данные еще не переданы полностью. Это особенно заметно на больших страницах, которые, в противном случае, должны были бы пройти все стадии преобразований перед выдачей клиенту, что привело бы к большой временной задержке. Кроме того, такой подход требовал бы больше ресурсов машины, так как была бы необходимость хранения всех промежуточных результатов в оперативной памяти.

Соответствие URL и материалов

Внимание привязке материалов к адресам уделялось по следующим причинам:

- **URL должны отражать иерархию материалов, поскольку этот подход привычен для пользователей**
- **Полезно, если URL имеют читабельный вид, поскольку в этом случае их легче запомнить**
- **URL должен включать в себя информацию о языке.**

Альтернативой последнему пункту могло быть сохранение информации о языке в пользовательских сессиях. В этом случае один адрес соответствовал бы нескольким видам одной вершины, что является неудобным, прежде всего для обмена ссылками.

Стоит отметить, что возможность настройки адресов появилась как следствие выбора Apache Cocoon в качестве основы сервиса, поскольку JetSpeed имеет фиксированный формат адресов.

В результате был выбран следующий формат:

/<код языка>/<имя первой вершины>/.../<имя последней вершины>.html

Здесь код языка – строчка из двух букв (ru, en, ..., согласно ISO 639-1), затем цепочка имен вершин, соответствующая пути в дереве от вершины непосредственно под корнем до назначения, и завершается это стандартным для WEB расширением “html”.

Дополнительно реализована возможность просмотра исходного XML документа, на основе которого генерируется данная страница. Документ доступен по тому же адресу, что и сама страница, за исключением расширения – вместо .html используется .xml. К сожалению, это еще не позволяет осуществлять XSLT преобразование на стороне клиента, поскольку после преобразования требуется дополнительная обработка полученного документа, которая может осуществляться только на сервере.

Содержимое исходного документа

Как уже упоминалось, исходный XML документ должен содержать все необходимые для страницы данные, при этом XSLT используется лишь для их оформления. В нашем случае он состоит из трех частей: карты сайта, выбранного материала и дополнительной информации, такой как ссылки на переключение языков.

Карта сайта представляет собой фрагмент дерева материалов, в котором содержатся все вершины текущего пути (от корня до выбранного материала) и их непосредственные потомки. Вершины, входящие в путь, при этом помечены. Такой подход позволяет не включать в XML лишнюю информацию, и в то же время всегда можно отобразить текущий путь, разделы первого уровня и подразделы текущего материала.

Сами данные представлены в виде, аналогичном используемому при их хранении в файловой системе. Фактически документ строится на основе этого формата путем удаления ненужных частей, то есть значений полей на языках, не соответствующих отображаемому языку документа.

Данный формат подходит так же для отображения заглавной страницы. При этом считается, что заглавная страница соответствует корневому документу дерева, который не имеет содержания. Таким образом, на заглавной странице доступен только список корневых разделов.

Исходный документ в рамках Cocoon создается специально написанным генератором, в обязанности которого, в первую очередь, входит разбор пути и соединения нужных фрагментов (карты, данных материла), которые он получает

от модели данных. Помимо этого, осуществляются некоторые преобразования, связанные с анализом языка интерфейса и данных. Поскольку текущий язык интерфейса фиксирован, производится оценка на допустимость каждой вершины в данном языке, и при необходимости, если язык данных не совпадает с языком интерфейса, это упоминается в карте.

```
<site:site xmlns:site="http://isir.ccas.ru/site">
 <site:langs>
  <site:lang site:name="ru" site:link="/jetspeed/cocoon/site/ru/members.html" />
  <site:lang site:name="en" site:link="/jetspeed/cocoon/site/en/members.html" />
 </site:langs>
 <site:map>
  <site:item site:title="Начало" site:active="true" site:current="false">
   <site:item site:title="Об Академии" site:active="false" site:current="false" />
   <site:item site:title="Президиум РАН" site:active="false" site:current="false" />
  </site:item>
 </site:map>
 <site:content>
 <page order="1">
  <title lang="ru">Академия наук и Российское государство в новых социально-
экономических условиях</title>
  <menu-title lang="ru">Об Академии</menu-title>
  <abstract lang="ru"/>
  <content lang="ru"><p>Понятие "власть" неотделимо от понятия "государство".
                Теория государства считает, что власть является сущностью
                государства</p></content>
  <page-title lang="ru">Российская Академия Наук</page-title>
 </page>
 </site:content>
</site:site>
```

Ссылки на переключение языков

Карта сайта

Поля текущей страницы

Данный формат является недостаточным для отображения карты сайта, поэтому существует дополнительный генератор, привязанный к определенному адресу, рисующий карту целиком.

Шаблоны и возможные представления

В самом простом случае можно выделить три различных представления:

- **отображение материала**
- **отображение карты**
- **отображение заглавной страницы**

Каждому из них соответствует свой шаблон. При этом первому и последнему представлению соответствует стандартный генератор, карта рисуется с помощью специального генератора.

Такая конфигурация поддерживается изначально, то есть существуют три

шаблона с фиксированными именами, каждый из которых отвечает за соответствующее представление.

Дополнительно можно потребовать возможность переопределения шаблонов для определенных вершин. Такие переопределения действуют рекурсивно, то есть к самой вершине и всем ее потомкам будет применяться указанный шаблон.

Кроме того, достаточно просто реализуется возможность отображения материалов в измененном виде по запросу пользователя, например, так называемой “версии для печати”. Для этого достаточно определить критерий, по которому можно определить, что пользователь запросил измененный вид (чаще всего это будет параметр в URL), и затем по этому критерию в карте сайта или применять специальный шаблон, или передавать параметр в стандартный шаблон, который должен его учитывать.

Рассмотрим особенности самих шаблонов. Прежде всего, следует отметить, что именно шаблонами определяется, каким образом отображаются поля материала. Фактически, шаблоны задают семантику полей.

Следующей особенностью является локализация. Принято, что независимо от языка интерфейса применяется один и тот же шаблон. При этом язык задается параметром, который в этот шаблон передается.

Соответственно, этот параметр может учитываться в шаблоне для локализации интерфейса. Локализацию текстовых строчек производить нежелательно, поскольку это правильнее делать с помощью соответствующего компонента. Однако язык может учитываться при выводе в документ ссылок на графические изображения, поскольку они могут содержать надписи.

Компонент, выполняющий локализацию, является стандартным для Cocoon. Он принадлежит к типу преобразователей (Transformer), и в нашем случае выполняется после XSLT преобразования. Его работа основана на следующем: он ищет в исходном документе специальный тэг, значение которого заменяется на основе словаря для данного языка. Например, его можно настроить так, что фрагмент

<i18n:text>print version</i18n:text>

после преобразования будет заменен на то, что хранится в словаре для фразы “print version”. Именно такой фрагмент содержится в XSL файле.

Еще один преобразователь предназначен для замены ссылок на вершины (как упоминалось, они хранятся в специальном формате) в реальные адреса. Он выполняется после XSLT преобразования, но перед локализацией. Кроме того, используются различные алгоритмы для поиска ссылок в XHTM и HTML полях. В первом случае достаточно разбирать значения атрибутов, поскольку ссылка может храниться только в атрибуте HREF тэга A. В случае HTML поля используется поиск и замена на основе регулярных выражений, что более ресурсоемко.

Дополнительные замечания.

Данный сервис может использоваться не только по прямому назначению. Иногда полезно, например, на его основе собрать сайт как набор HTML файлов или скриптов для дальнейшего использования.

Кэширование данных осуществляется сравнительно просто встроенными в Apache Cocoon средствами.

Атрибутно-полнотекстовый поиск

Поиск является очень удобным для посетителей средством доступа к нужной им информации. Несмотря на эффективность современных поисковых машин, у них есть несколько существенных ограничений. Во-первых, они индексируют только открытую для внешнего мира часть сайта. Индекс сайта, как правило, запаздывает на несколько дней. Предоставляется только полнотекстовый поиск. В связи с этим, если не рассматривается статический открытый сайт с большим временем обновления, система поиска приобретает актуальность.

Для более узкой задачи – непосредственно полнотекстового индексирования – существуют стандартные методы (например, инвертированный индекс) и их реализации. С их помощью можно поддерживать актуальность индекса своего сайта. Однако, они не подходят для индексирования сложноструктурированных данных (например, БД); в таких случаях приходится разрабатывать индивидуальные решения.

С точки зрения внутренней поисковой машины страница имеет большое количество атрибутов (например, автор, время создания и последнего изменения, связь материала с рубриками). Эти атрибуты позволяют не только более качественно осуществлять поиск (например, можно найти все публикации определенного автора), но и представление результатов, поскольку система может определять близкие по содержанию материалы, например, на основе каталогов или рубрикаторов

Основные понятия

Ресурс - данные, для которых строится индекс. Примеры ресурсов: тестовый файл, HTML документ.

Атрибут - контекст вхождения слова в текст ресурса. Каждое слово в тексте может иметь единственный связанный с ним атрибут, уточняющий его значение. Атрибуты используются для уточнения запросов и ранжирования. Пример атрибута – “заголовок документа”.

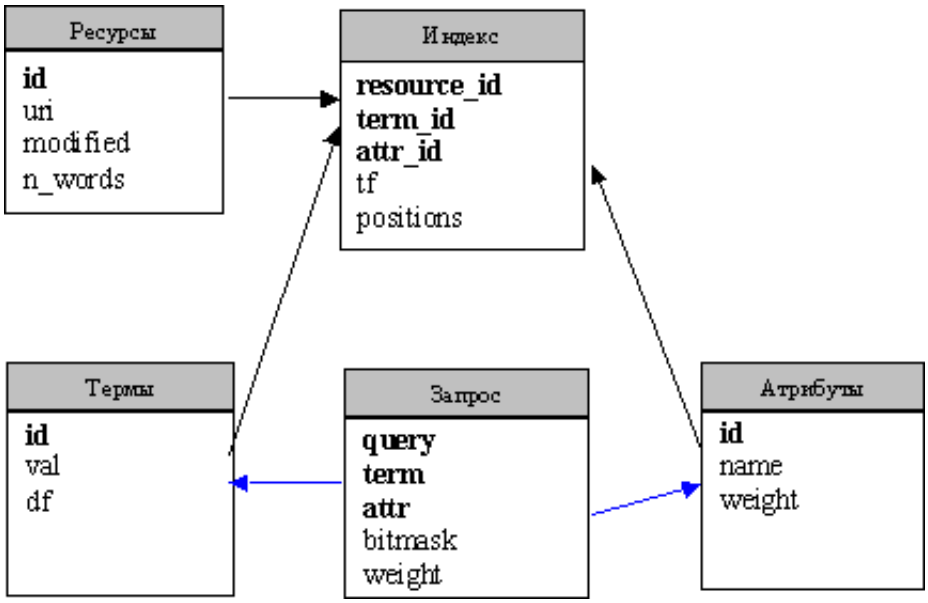
Индекс - информация о ресурсе, предназначенная для поиска.

Терм - последовательность символов алфавита, окруженная разделителями,

встреченная в тексте ресурса.

Схема базы данных

На схеме приведены таблицы базы данных, предназначенные для хранения индекса и сопутствующей информации.



Назначение полей схемы приведено в следующей таблице: