Руководство разработчика

11 мая 2004 г.

Оглавление

Введение

Communiware представляет собой развитую систему управлния содержимым Web-сайта. Она предназначена для того чтобы упростить и ускорить разработку сложного сайта, заполнение которого материалами пользователь будет в дальнейшем осуществлять самостоятельно.

Основой Communiware-сайта является онтологическая модель (см. главу 2, которая позволяет описывать объекты, существующие на сайте и отношения между ними в терминах, понятных пользователю. В одних приложениях это могут быть такие термины как Статья, Рубрика, Реплика в дискуссии, в других Фирма, Коммерческое предложение, Заказ.

Как правило, при разработке Communiware-сайта, разработчику не требуется программировать на таких языках программирования как Perl или PHP. Его деятельность ограничивается работой с web-интерфейсами редактирования онтологической модели, и созданием шаблонов, отображающих объекты из базы данных в определенном дизайне.

В данной книге рассматривается создание типичного Communiware-сайта, включающего в себя набор стандартных онтологий и некоторые специфические расширения. Расширению возможностей ядра Communiware, требующему программирования на Perl или SQL посвящена книга Руководство программиста Communiware.

Эта книга является скорее более или менее подробным изложением принципов архитектуры Communiware и работы с ней, сложившимся к версии 0.93, чем справочником, отражающим синтаксические особенности текущей версии.

За подробностями, следует обращаться к Справочному руководству Communiware.

Наиболее свежую информацию, относящуюся к вашей версии следует искать во встроенной документации Communiware на служебном сайте вашего Communiware-сервера (http://*ваш сервер*/default/reference_r).

В первой главе книги мы рассмотрим процесс создания типичного сайта,

не углубляясь особенно в детали. В последующих главах будут подробно рассмотрены отдельные стороны этого процеса.

Глава 1

Быстрый старт.

1.1 Создание сайта Копирование пакетов

1.2 Определение онтологии данного сайта

1.3 Создание шаблонов обрамления

1.3.1 Четыре вида стандартных обрамлений

1.3.2 Библиотека графических элементов и ее использование

1.4 Оформление итемов, представляющих самих себя (материалы)

1.5 Оформление итемов, представляющих списки связанных с ними.

1.6 ErrorDocuments

1.7 Разграничение доступа

1.8 Создание форм ввода

1.8.1 Ограничение вариантов выбора

1.8.2 Почтовые уведомления

Глава 2

Описание онтологической модели

Основным отличием Communiware от других систем управления Web-сайтами, таких как Zope, Midgard или ColdFusion, является наличие унифицированной системы представления знаний о структуре информации.

В то время как другие системы предлагают только набор средств программирования Web-страниц, в Communiware основной упор делается на создание модели той области деятельности заказчика, которую он хочет отразить на Web-сайте, и предоставление заказчику возможности работать с хранящейся на сайте информацией именно в терминах этой модели. А развитые средства Web-программирования предназначены только для того, чтобы обеспечить красивое отображение этой модели в множество Web-страниц.

Эта модель предметной области называется *онтологией* данного сайта. Онтология сайта обычно создается интерактивно с помощью web-форм Ontology development kit, но может быть также выражена в ввиде одного XML-файла использующего Ontology.dtd и редактироваться (или даже создаваться) при помощи обычного XML-редактора.

Коммунивер позволяет использовать одну и ту же онтологию для нескольких сайтов имеющих принципиально разный дизайн. Например, сайты разных софтверных компаний могут использовать одну и ту же онтологию «Сайт софтверной компании», но при этом иметь принципиально разный внешний вид и разное содержание.

Следует различать три разных вещи — *онтологию*, которая представляет собой формализованное описание того, как решать некоторый класс за-

дач средствами Communiware, *Коммунивер-продукт*, представляющий конкретную программную реализацию онтологии и *Коммунивер-сайт*, представляющий собой конкретный экземпляр продукта, заполненный пользовательскими данными. Когда заказчик просит вас разработать ему сайт, он на самом деле просит разработать Коммунивер-продукт, выполняющий определенные функции и снабдить его определенным декором. В сайт он его превратит уже сам, заполнив своей содержательной информацией. Поскольку для заказчика обычно важен только его собственный сайт, ему нет нужды различать понятия сайта и продукта.

2.1 Понятие итема

Любая модель предметной области состоит из определенных объектов и связей между ними. В Communiware в качестве объектов выступают *итемы* — сущности, представляющие собой набор атрибутов, похожие на записи в реляционных БД и объекты в объектно-ориентированном программировании. Мы не используем для них термина «объект», поскольку объекты в ООП обычно обладают своим собственным поведением, в то время как информационные объекты на сайте (статьи, реплики, задачи в системе управления проектом, коммерческие предложения) относительно статичны, и изменяются в основном в результате явных действий пользователя.

Несколько ближе к истине был бы термин «документ», широко применяемый в настольных операционных средах. Тем не менее, мы считаем что и этот термин некорректен. Некоторые итемы (например статьи) безусловно являются документами. Другие могут быть как отдельными частями большого структурированного документа (например, главы в книге, или варианты ответа в опросе), третьи — контейнерами, «папками», объединяющими отдельные документы (рубрики, группы, направления работ).

Таким образом, *итем — это любая информационная сущность, которую в данной онтологической модели нам удобно рассматривать самостоятельно*.

Термин „рассматривать самостоятельно” имеет и буквальный смысл. Любой итем в Communiware может быть указан в URL, и при этом мы получим на экране Web-страницу, посвященную исключительно данному итему[1].

Поскольку объекты в предметной области заказчика бывают разной природы, для их описания на сайте требуется разный набор сведений. Поэтому итемы делятся на *типы*. Каждый тип итема имеет свой собственный набор *атрибутов*, т.е. значений, характеризующих данный информационный

[1]При наличии у нас достаточных прав доступа

объект. Кроме того, для типа итема задается набор *связей*, в которых он может участвовать, и набор *статусов*, т.е. состояний, которые он может принимать в процессе своей жизни на сайте. Иногда также накладываются ограничения на то, из какого состояния в какое он может переходить, и кто имеет право его переводить в это состояние.

Этот набор ограничений называется *workflow*.

Каждый итем в базе данных Communiware имеет уникальное имя или *идентификатор*. Этот идентификатор может представлять собой значимое слово, заданное при создании данного итема. Если при создании итема идентификатор не был задан, то Communiware автоматически генерирует уникальный числовой идентификатор.

Ссылки на итем обычно осуществляются именно по его идентификатору.

2.2 Понятие сайта.

В случае, если бы на одном компьютере всегда функционировал ровно один Web-сайт, жизнь была бы устроена несколько проще. Но в наше время типично наличие на одном компьютере нескольких тематически (и онтологически) совершенно разных сайтов, которые могут иметь даже собственные доменные имена. На типичном сервере разработки Web-студии обычно одновременно существует около десятка одновременно разрабатываемых проектов, а на крупных хостинговых серверах типа `people.mtu.ru`, могут сосуществовать тысячи сайтов.

Поэтому в Communiware предусмотрено понятие *сайта*.

Сайт представляет собой множество взаимосвязанных итемов, объединенных тематической общностью, системой навигации и общей онтологией.

Как правило, все страницы сайта имеют единый стиль оформления. Кроме того, сайт объединяет множество людей, которые имеют права на редактирование содержащейся на нем информации.

Сайт обычно выглядит как некоторое поддерево URL, т.е. все страницы сайта имеют общий префикс, который может быть как уникальным именем хоста, так некоторым именем „директории", разделяя имя хоста с другими сайтами.

Естественно, что сам сайт представлен в Communiware как итем, имеющий такие атрибуты как «имя хоста», «уровень защиты информации» (см. главу **??**), и набор связей с итемами пользователей, имеющих определенные права доступа к информации этого сайта. Если сайт разделяет имя хоста с другими сайтами, то в его итеме хранится также префикс имени директории. Имя хоста вместе с этим именем называется *префиксом сайта*, так

как с него начинается URL всех страниц данного сайта.

Любой итем считается принадлежащим какому-либо сайту. Это отношение принадлежности, так же как и любое другое (см. раздел 2.6.6), обеспечивает наследование прав доступа. Т.е. любой пользователь, имеющий право редактировать итем сайта, имеет право редактировать и все итемы, принадлежащие к данному сайту.

2.3 Представление итема. Шаблоны. Пространство URL.

Сам по себе итем — абстрактная сущность, представляющая собой набор атрибутов, и не имеющая собственного внешнего вида.

Для того чтобы можно было увидеть итем как часть Web-страницы, требуется задать *представление* этого итема.

Способ задания представления в Communiware называется *шаблоном*, и представляет собой, в сущности, HTML-текст, в котором в некоторых местах помещены специальным образом оформленные указания вывести в этом месте атрибут итема, или список связанных итемов. Синтаксис этих указаний рассмотрен в главе 5.

Один и тот же итем можно показывать по разным шаблонам, и таким образом получить, например, дискуссию, порожденную данным итемом, или форму для редактирования итема, или вид для печати.

Кроме шаблонов, которые представляют данный итем как целую Web-страницу, существуют шаблоны, представляющие его как фрагмент страницы, например, представление статьи в списке статей на странице рубрики.

Шаблон хранится в базе данных Communiware как отдельный итем типа «шаблон».

С каждым итемом в Communiware связано *представление по умолчанию*, которое используется тогда, когда представление не указано явным образом. Обычно это представление одинаково для всех итемов данного типа на сайте, но существует возможность определить отдельное представление по умолчанию для всех итемов, создаваемых в некоторой части сайта, или даже для индивидуального итема.

URL любой коммуниверной страницы представляет собой сочетание *префикса сайта*, *идентификатора итема* и *идентификатора шаблона*, например в URL `http://dev.ice.ru/default/guide_r/def_edit` префиксом сайта является `dev.ice.ru/default`, идентификатором итема — `guide_r`, а идентификатором шаблона — `def_edit`.

Если в URL опущен идентификатор шаблона, то используется представление по умолчанию для данного итема. Если опущен и идентификатор итема, то показывается итем сайта.

2.4 Тип содержимого

Итемы могут иметь разный *тип основного содержимого*. Для большинства итемов основным содержимым является некий HTML-текст (про такие итемы говорят, что они имеют тип содержимого Html, но могут быть итемы, весь смысл которых заключается в их связях с другими итемами (например, рубрика), или хранящие всю информацию в атрибутах. Про такие итемы мы говорим, что у них нет основного содержимого, т.е. они имеют тип содержимого None.

В других случаях требуется хранить в итеме не текст, а двоичные данные, например дистрибутив программы или презентацию в формате PowerPoint. Этот тип содержимого называтся Binary, и отличается тем, что данные такого типа Communiware никак не обрабатываются, и отдаются пользователю в точности в том виде, в каком они были загружны на сайт[2].

Отдельным типом содержимого является шаблон Communiware (Template). Это вызвано тем, что шаблон должен не просто показываться, а специальным образом интерпретироваться. Кроме того, при редактировании шаблона проверка корректности его текста проводится по другим правилам, нежели проверка просто Html.

В некоторых случаях требуется хранить текст, который требует специальной интерпретации, но не является шаблоном Communiware. Примером таких итемов может служить *правило устаревания* (см. раздел **??**). Для таких случаев предназначен тип содержимого Code.

И наконец, существуют итемы, содержимое которых имеет табличную природу, например временные ряды, такие как прогнозы погоды или динамика курсов валют. В этих ситуациях хочется уметь фильтровать отсчеты по временным интервалам, но кажется избыточным хранить каждый отсчет как отдельный итем. В таких случаях используется тип содержимого Table. Содержимое итемов с типом содержимого Table хранится как таблица базы данных, одним из компонентов составного первичного ключа которой является идентификатор итема.

[2]Communiware имеет возможность производить полнотекстовый поиск по некоторым типам двоичных файлов

2.5 Атрибуты

Логически итем представляет собой набор именованных значений, называемых атрибутами. Во многих контекстах даже основное содержимое итема можно рассматривать как атрибут с именем `TEXT`. Доступ к свойствам итема в Communiware всегда осуществляется как ссылка на его атрибут.

2.5.1 Типы атрибутов

Атрибуты хранят информацию разной природы, поэтому каждый атрибут характеризуется своим *типом*.

Выделяются следующие типы атрибутов:

STRING текстовая строка. В виде атрибутов этого типа хранятся атрибуты с перечислимым значением, идентификаторы URL и другие значения, предназначенные не только для чтения человеком, но и для интерпретации компьютером. Когда такой атрибут выводится на web-страницу, он показывается там в точности как он хранился в базе. Т.е. если в тексте атрибута содержались специальные для HTML символы, они преобразуются в соответствующие entities.

RICHTEXT Форматированный текст, могущий содержать в себе Html-разметку. Таким способом хранятся заголовки и аннотации итемов. При выводе на экран часть разметки может подавляться. Например, если атрибут выводится как текст гиперссылки, то ссылки, содержащиеся в его тексте, подавлятются.

NUMBER Число. Над числами можно производить арифметические операции и форматировать их при выводе. Communiware поддерживат специальные средства для вывода числа, сопровождаемого счетным словом в корректной форме (см. раздел 6.1.3).

DATE Момент времени, т.е. дата и время. В большинстве случаев дата в Communiware хранится с точностью до секунд. Но в некоторых случаях, если речь идет скажем о дате написания классического произведения, дата известна с существенно меньшей точностью.

В Communiware предусмотрено понятие *точности даты*. Если наряду с атрибутом `X` типа дата существует атрибут `X_ACCURACY` принимающий одно из значений `SECOND`, `MINUTE`, `HOUR`, `MONTH` или `YEAR`, то при выводе даты компоненты, означающие меньшие промежутки времени, подавляются.

Все даты в Communiware имеют стандартное внутреннее представление **ГГГГ.ММ.ДД ЧЧ.МИ.СС**. Это представление позволяет сравнивать даты как строки. Попытка определить атрибут типа DATE, значение которого не соответствует данному шаблону, к успеху, как правило, не приводит. Впрочем, динамические элементы для ввода и вывода дат обладают средствами явного задания формата для ввода.

2.5.2 Атрибуты с перечислимым значением

Многие атрибуты принимают ограниченное множество значений. Таковы, например, названия типов итемов, названия типов связей, статусы. Тем же свойством обладают и многие расширенные атрибуты.

Как правило, такие значения представляют собой идентификатор, т.е. короткое (до 20 символов) слово состоящее из латинских букв, цифр и знака подчерка. По соглашению, такие идентификаторы, так же как и имена атрибутов, пишутся большими буквами.

Эти идентификаторы достаточно легко запоминаемы и удобны при использовании их в шаблонах (при сравнениях, в параметрах фильтров и т.п).

Но поскольку достаточно часто возникает необходимость вывода этих названий на web-страницу пользователю, необходимо иметь возможность расшифровывать эти идентификаторы, выдавая полное название на текущем языке. Поэтому в онтологическую модель помимо идентификатора значения всегда включаются его названия на всех поддерживаемых языках.

Кроме того, для многих из перечислимых значений существует естественным образом заданная (вытекающая из постановки задачи) упорядоченность. Например, для статусов задачи в системе управления проектом естественной является упорядоченность в порядке степени выполненности задачи.

В этих случаях со значением (идентификатором) связывается порядковый номер (ORDNUM), и сравнения на больше/меньше в шаблонах происходят с учетом этого порядкового номера.

2.5.3 Стандартные атрибуты

Существует достаточно широкий набор атрибутов, который имеют все итемы, независимо от их типа. Этот набор включает в себя:

ITEM_ID Уникальный идентификатор данного итем. Все обращения к итему внутри Communiware производятся по этому идентификатору.

TITLE Заголовок итема, т.е. название которое выводится на Web-страницу пользователю. Этот атрибут имеет тип RICHTEXT, т.е. может содержать некоторый набор HTML-разметки.

WRITTEN Дата написания итема, точнее дата создания информационного объекта, представляемого данным итемом. Например, если мы загрузим в Communiware текст «Евгения Онегина», то этому атрибуту мы должны будем присвоить значение 1830 года.

WRITTEN_ACCURACY Точность представления даты написания. Перечислимый атрибут с возможными значениями минута, час, день, месяц, год. Позволяет указать точность с которой известна дата написания.

При форматировании даты с помощью динамического элемента Attr компоненты, меньшие чем указанный период, будут удалены из формата.

PUBLISHED Дата публикации итема на сайте. Обычно известна точно, поэтому поле для задания точности не предусмотрено. Этот атрибут можно редактировать, поскольку дата официальной публикации может отличаться от даты фактического помещения на сайт или в базу данных Communiware. Например, в Московском Либертариуме имеются итемы с датой публикации в 1994 году (год появления первого статического сайта Либертариума) хотя Communiware появилась на этом сайте в 1999 году.

UPDATED Дата последнего изменения итема через интерфейсы редактирования Communiware. Выставляется автоматически. Параллельно с ней обновляется связь LASTEDITED связывающая данный итем с итемом пользователя, вносившего изменения. См также главу 8.8.

LASTCHANGE Дата последнего фактического изменения информации о данном итеме в базе данных В отличие от UPDATED изменяется, например, при восстановлении итема из резервной копии.

STATUS_UPDATE Дата последнего изменения статуса.

ABSTRACT Аннотация. Может содержать HTML разметку. Может использоваться даже для итемов у которых текста как такового нет.

TYPE_ID Тип итема. Теоретически предполагается что этот атрибут остается неизменным в течении всей жизни итема. Но на практике существует возможность его изменения. Например, модератор сайта имеет право превратить реплику или внешнюю ссылку в статью.

Изменение типа итема возможно только если оба типа не имеют расширенных атрибутов или имеют одну и ту же таблицу расширенных атрибутов, и имеют пересекающийся набор статусов, при этом текущий статус допустим для нового типа итема.

STATUS Статус итема. См раздел 2.8.

TEMPLATE_ID Вообще-то это должно быть не атрибутом, а связью. Это ссылка на шаблон по умолчанию для данного итема, т.е. шаблон, который используется для показа данного итема, если в URL явно не указан шаблон.

Присваивание значения этому атрибуту происходит обычно автоматически с использованием механизма стандартных шаблонов (2.3).

SERVER Еще одно исключение из правила «если с итемом связан другой итем, это должно быть связью». Любой итем в базе данных Communiware относится к определенному сайту. Сайт также является итемом. Более того, принадлежность итема к сайту фактически рассматривается как наличие метасвязи BELONGS (см. раздел 2.6.6).

PARAMS Атрибут, который может быть использован разным способом для разных типов. Место для тех ситуаций когда нужно поместить дополнительную информацию, а создание таблицы расширенных атрибутов ради одного атрибута кажется излишним.

OLD_URL Некая информация с семантикой URL. Название OLD_URL сложилось исторически — при импорте Либертариума туда помещалась URL, которую имела эта страница до импорта. Теперь этот атрибут обычно используется для ссылки на документ, доступный на WWW вне текущей базы данных Communiware — в итемах типа внешняя ссылка и в итемах типа Файл (когда файл находится под управлением Communiware, но отдается web-сервером непосредственно с диска)

TEXTSIZE Размер текста итема в символах без учета Html-разметки. Имеет смысл только для итемов с типом содержимого Html.

DATASIZE Размер текста итема в байтах (для всех типов содержимого, кроме таблиц) или число строк таблицы.

LANG Язык итема. Двухбуквенный код языка по ISO 639-1. Язык итема, упомянутого в URL страницы, используется для вывода всех системных сообщений, включая интернационализированные названия перечислимых значений, и форматирования дат.

RESTRICT_TEMPLATES Указывает степень ограничения показа данного итема.

FORMATTING Признак наличия форматирования, не представимого в текстовом формате. В случае если этот атрибут равен единице, текст при редактировании будет отдаваться в формате Html, а не конвертироваться в более удобочитаемое текстовое представление.

2.5.4 Расширенные атрибуты

В достаточно большом числе случаев определенному типу итемов требуется иметь дополнительные атрибуты. Communiware позволяет создать набор расширенных атрибутов (и автоматически заводит для него таблицу в базе).

На расширенные атрибуты имеются следующие ограничения:

1. Имена расширенных атрибутов должны быть уникальны. Т.е. атрибут с данным именем должен храниться ровно в одной таблице.

2. Тип итема должен иметь ровно одну таблицу расширенных атрибутов. Т.е. два типа итема могут иметь либо полностью совпадающий набор расширенных атрибутов, либо не должны иметь ни одного совпадающего расширенного атрибута.

3. Не существует способа удалить расширенный атрибут, кроме как удалить все типы, которые его имеют, и создать их заново.

4. Не существует Web-интерфейса, позволяющего добавить атрибут в существующую таблицу расширенных атрибутов. Это можно сделать только из командной строки сервера.

2.5.5 Табличные атрибуты

У итемов с типом содержимого Table кроме расширенных атрибутов есть еще *табличные атрибуты*.

Табличные атрибуты отличаются от всех прочих атрибутов тем, что каждому итему соответствует более одного значения такого атрибута.

Как минимум один из табличных атрибутов должен быть *ключевым* (TABLEKEY).

Комбинация значений всех ключевых атрибутов определяет значения всех остальных,*содержательных* (TABLEVALUE) табличных атрибутов данного итема. В случае использования итемов с табличным содержимым для хранения временных рядов ключевым атрибутом обычно является время, к которому относится то или иное значение.

Каждому типу итема с содержимым Table соответствует одна *таблица содержимого*, ключом которой является комбинация идентификатора итема и всех ключевых табличных атрибутов, имеющихся у данного типа итема.

2.6 Связи

Как правило, информационные объекты интересны не сами по себе, а в своем взаимоотношениии с другими объектами. Например, на большинстве сайтов наличие информации об определенном человеке совершенно не интересно, если не известно, что этот человек является автором таких-то документов, или обладает правами на редактирование таких-то разделов, или является ответственным за выполнение таких-то задач.

Связи между итемами обычно несут гораздо больше информации чем атрибуты итема. *В Communiware атрибут никогда не используется для хранения ссылки на другой итем. Если нужно выразить отношение между объектами, для этого используется связь*[3].

Связи в Communiware, также как и итемы, имеют различные типы. В отличие от итемов и отношений в реляционной модели, связи не имеют атрибутов. Связь однозначно определяется парой итемов, которые она определяет, и своим типом.

2.6.1 Активные и пассивные итемы в связи.

Итемы в связи не равноправны. Один из них считается *активным*, а другой *пассивным*. Эти названия происходят от активного и пассивного залога в грамматике. Например, автор *написал* (активный залог) статью, а статья *написана* автором. Поэтому в связи AUTHOR итем персоны автора будет активным, а итем статьи — пассивным.

Обычно в отношениях типа один ко многим, пассивным считается тот конец связи, которого много, а активным тот, который один. Но иногда удобнее считать активным тот конец, которого много. Подробнее об этом в разделе 2.6.2. В отношениях типа предок-потомок, обычно активным считается предок (который породил), а пассивным — потомок (который поро-

[3]Из этого правила есть два исключения, рассмотренные в разделе 2.5.3

жден). В отношениях часть-целое, соответственно активным является целое (которое включает часть), а пассивной — часть (включенная в целое).

2.6.2 Простые и иерархические связи

Достаточно часто нас интересуют не только непосредственные отношения между итемами, но и отношения опосредованные, проходящие по цепочке связей. Наиболее характерным примером таких отношений является связь часть-целое. Если объект A является частью объекта B (например, глава 12 входит во второй том книги) а объект B — частью объекта C (второй том является частью книги), то объект A является частью объекта C.

Такие отношения называются *транзитивными* или *иерархическими* и представляются в Communiware с помощью иерархических связей.

В противоположность этому у простых связей наличие у A связи с B, и у B связи с C не означает наличия связи между A и C. Например, если у нас есть книга C, автором которой является B, а в тексте итема B находится его биография, автором которой является A, это не означает, что A является автором книги C.

В большинстве объектных систем программирования поиск всех потомков по иерархии требовал бы рекурсивного обхода дерева.

В Communiware рекурсия не применяется. При создании связи информация о всех интересующих нас путях (цепочках транзитивных связей) помещается в таблицу *транзитивного замыкания* (ITEM_REL), и может быть извлечена оттуда одним запросом. Это приводит к существеному ускорению показа иерерахических структур за счет увеличения затрат времени на их модификацию. Выбор данного подхода определяется тем, что в типичных web-приложениях визуализация требуется на три порядка чаще, чем модификация.

Иерархические связи в Communiware могут иметь тип отношения не только 1:N, но и M:N, т.е. граф, образованный этими связями, может иметь точки слияния (подрубрика, которая входит одновременно в две рубрики более высокого уровня). Недопустимы только циклические цепочки связей (приводящие к бесконечной рекурсии).

2.6.3 Двунаправленные связи

Существуют (правда довольно редко встречаются), связи в которых мы не можем выделить различные роли двух участников связи. Примером такой связи в стандартной онтологии является связь SEEALSO (См. также).

Если читателю документа A может быть интересен документ B, то с достаточно высокой вероятностью читателю документа B также понадобится информация из документа A.

Для таких случаев предусмотрено понятие *двунаправленной связи*. Фактически, двунаправленная связь эквивалентна наличию двух связей — от A к B и от B к A, которые создаются и удаляются одновременно.

Двунаправленная связь в Communiware не может быть иерархической и не может участвовать в метасвязи, поскольку это привело бы к возникновению циклических путей, а Communiware на данный момент работает только с ациклическими графами.

2.6.4 Множественность отношения

Важным свойством связи является множественность отношения. Например, часть может быть частью только одного целого, а целое — состоять из нескольких частей. Поэтому связь целое-часть имеет множественность 1:N. Связь автор-произведение имеет множественность M:N, так как у произведения может быть несколько соавторов, и у каждого автора может быть несколько работ.

Таким образом, выделяются четыре вида множественности отношения: 1:1, 1:N, N:1 и M:N.

2.6.5 Метасвязи

Иногда, отношения аналогичные транзитивным связям, возникают автоматически в силу наличия каких-то других отношений. Например, отношение «является модератором» между персоной и рубрикой и отношение «является рубрикой для» между этой рубрикой и некоторой статьей, приводит к возникновению у данной персоны права редактировать данную статью. Подобные автоматически возникающие отношения в Communiware называются *метасвязями*. Метасвязь определяется как такое множество типов связи, что существование пути между двумя итемами, проходящего только по связям из данного множества, эквивалентно существованию некоего интересующего нас отношения между этими итемами.

В некоторых случаях связь может входить в метасвязь *неявно*. Это означает, что записи в таблице транзитивного замыкания, соответствующих этому пути не создается, но функции Communiware, работающие с путями работают так, как будто этот путь существует.

Различение явного и неявного вхождения связи в метасвязь позволяет

существенно уменьшить объем таблицы транзитивного замыкания, и ускорить работу с ней.

В настоящий момент не существует способа включить связь в метасвязь неявно с помощью средств редактирования онтологии, поэтому неявное включение поддерживается только для авторизующих метасвязей, непосредственно поддерживаемых ядром.

2.6.6 Метасвязь BELONGS и отношение принадлежности

Особую роль в Communiware играет метасвязь BELONGS (принадлежит к).

Эта метасвязь является, так сказать, скелетом сайта, и ее наличие определяет принадлежность итема к той или иной части сайта.

Эта метасвязь *неявно* входит во все авторизующие метасвязи, т.е. по ней наследуются права доступа (см. раздел 2.7).

Кроме того, в метасвязь BELONGS неявно входит отношение принадлежности итема к сайту (как указано в разделе 2.5.3, принадлежность итема к сайту хранится вообще не как связь, а как атрибут итема). Т.е. с точки зрения системы прав доступа, все итемы данного сайта имеют метасвязь BELONGS с итемом этого сайта.

2.6.7 Упорядоченность соседей

Соседи некоторого итема по некоторой связи часто должны быть определенным образом упорядочены. Обычно порядок существенен для множества связей, в которых данный итем является активным. Например, порядок глав в книге однозначно определен волей автора, а порядок статей в рубрике — волей редактора.

Иногда порядок, в котором итемы показываются в каком-либо списке, однозначно следует из каких-либо свойств самих итемов. Например, статьи могут быть показаны в хронологическом порядке их публикации. Но часто порядок расположения объектов задается пользователем вручную, исходя из каких-то неформализуемых критериев.

Порядковый номер в списке однородных объектов не является имманентным свойством итема. Как мы уже видели, статья может входить в две и более рубрики одновременно (или в рубрику и сборник статей, опубликованный как книга), и в каждом случае иметь свое место, назначенное редактором рубрики (составителем сборника).

Поэтому порядковый номер (ORDNUM) в Communiware считается свойством связи, а не свойством итема. Порядковый номер должен быть уни-

кальным в пределах множества связей данного типа с данным активным итемом.

2.7 Права доступа

Web-сайт является инструментом для коллективной работы с информацией. У него есть множество пользователей, которые эту информацию читают, редакторы (как правило, более одного) которые эту информацию на сайт вносят и модифицируют, разработчики, которые имеют право менять способы представления (шаблоны) и так далее. Набор ролей, которые может играть пользователь на сайте, крайне разнообразен.

В такой ситуации неизбежно возникает проблема разграничения прав доступа между пользователями сайта.

Пользователи сайта делятся на *анонимных* и *зарегистрированных*.

Естественно, что иметь право делать что-либо, что не доступно другим пользователям, может только зарегистрированный пользователь.

Поскольку зарегистрированному пользователю соответствует итем типа «персона», права доступа естественным образом выражаются в виде связей между этим итемом и теми итемами, на которые пользователю даются права.

Поэтому связь в Communiware может иметь признак *авторизующая*. Авторизующие связи обычно не являются иерархическими, и никогда не являются двунаправленными.

Поскольку давать права на каждый итем в индивидуальном порядке было бы слишком трудоемко, обычно для контроля прав доступа используются *авторизующие метасвязи*.

Авторизующей считается любая метасвязь, в которую включена хотя бы одна авторизующая простая связь.

Все авторизующие метасвязи неявно включают метасвязь BELONGS, т.е. все права доступа на некоторый итем распространяются и на его потомков по BELONGS, в том числе все права на итем сайта распространяются на все итемы данного сайта.

С очевидностью, это не касается простых авторизующих связей. Т.е, если некоторое право дается простой авторизующей связью, а не порождаемой ей метасвязью, то это право распространяется только на тот итем, на который оно явно дано.

В стандартной онтологии примером такой связи является связь `DEVELOPER`. Проверку наличия у пользователя прав разработчика (напри-

мер, прав на редактирование онтологии) следует всегда производить в контексте итема сайта.

2.7.1 Права на чтение

Поскольку Communiware исходно предназначена для публичных web-сайтов, по умолчанию считается что право на чтение некоторого итема имеют все пользователи, включая и анонимных.

Для того чтобы ввести ограничения на чтение итема, требуется явным образом дать некоторым пользователям право его читать, создав связь ALLOWEDUSERS между ними и этим итемом.

Связь ALLOWEDUSERS входит в метасвязь READS, которая распространяет действие ограничения на всех потомков по BELONGS.

Если есть персоны, имеющие явно установленное право на чтение, то подразумевается, что у всех остальных этого права нет.

2.7.2 Право на запись

Другим фундаментальным правом является право на запись, т.е. на редактирование всех атрибутов и связей данного (существующего) итема.

Это право дает метасвязь WRITES, в которую входят простые авторизующие связи MODERATES и DEVELOPER. Последняя может быть привязана только к корню сайта, и дает также права на изменение онтологии и шаблонов.

Поскольку было бы странным давать пользователю право редактировать то, что он не может прочитать, право на запись автоматически предполагает право на чтение.

2.7.3 Прочие права

В реальных задачах существует множество ролей пользователей, у которых должно быть больше прав, чем просто читать, но меньше чем редактировать все. Например, можно представить себе онтологию, в которой у некоторого круга пользователей есть право помещать статьи в определенную рубрику, и редактировать свои статьи, но нет права изменять атрибуты самой рубрики и редактировать чужие статьи[4].

Для создания таких промежуточных прав можно создать свою авторизующую связь (и, возможно, соотвествующую ей авторизующую метасвязь).

[4]примерно эту семантику имеет связь TRUSTEDUSERS в стандартной онтологии

Описание того, какие конкретно возможности предоставляются пользователям, обладающим данным правом, производится с помощью шаблонов редактирования (см главу 8) и системы workflow (см. раздел 2.8).

2.7.4 Группы пользователей

В некоторых случаях имеется довольно много пользователей, которые должны иметь достаточно большие права на сайте. Например, на интранет-сайте организации права на редактирование нескольких не связанных между собой рубрик должны иметь все сотрудники определенного отдела. В этом случае индивидуальное управление правами доступа каждого из сотрудников весьма трудоемко.

Поэтому в Communiware предусмотрено понятие *группы пользователей*. Пользователи включаются в группы с помощью связи `INCLUDED_USER_GROUP`, которая входит во все авторизующие метасвязи. Поэтому права, данные всей группе в целом распространяются на всех членов группы. Заметим, что хотя наличие у группы хотя бы одной связи `MODERATES` приводит к тому что все члены группы связаны с ней метасвязью `WRITES`, так как любой фрагмент пути по некоторой метасвязи сам является путем по этой же метасвязи, это не дает всем членам группы права редактировать саму группу, так как права дает только такой путь по авторизующей метасвязи, который содержит авторизующую простую связь.

Права, которые назначаются через простые авторизующие связи, не имеющие соответствующих метасвязей, не могут быть назначены группам пользователей, так как, поскольку метасвязи, соединяющей пользователя и итем на которой даны права не существует, не существует и права. Такие права как `DEVELOPER` должны даваться индивидуально.

При создании прав следует также учитывать этот момент и, при наличии необходимости назначать права группам, создавать соответствующую авторизующую метасвязь.

Кроме организации пространства субъектов доступа, группы пользователей могут быть использованы в качестве адресатов писем, посылаемых динамическим элементом `MailTo` (см. раздел 9.4) и подписчиков системы дайджестов (см. главу 10).

2.8 Статусы и workflow

Статус итема — это некоторое состояние, которое итем может принимать в процессе существования на сайте. С точки зрения языка шаблонов

Рис. 2.1: Граф workflow

статус — это атрибут с перечислимым значением (см. раздел 2.5.2).

Список возможных значений статуса свой для каждого типа итемов, хотя един для всех сайтов, использующих данный тип итема[5]. Значения статуса упорядочены, причем упорядоченность задается явным образом разработчиком онтологии. Поэтому, если типы A и B имеют статусы X и Y, то вполне возможна ситуация, что для типа A $X < Y$, а для типа B, $X > Y$.

Как и другие атрибуты с интернационализированным значением, статусы имеют интернационализированные названия. Доступ к этим названиям возможен как стандартным способом (TXT(STATUS), раздел 3.4), так и с использованием специального вычислимого атрибута ITEM_STATUS.

В некоторых случаях имеет смысл закладывать в онтологию ограничения на переходы между статусами, т.е. указать, что итем со статусом A может быть переведен в B и C (но не D и E) пользователем с правами X, а пользователем с правами Y — только в B.

Такой набор переходов между статусами называется *workflow*. Он может быть легко представлен графически с помощью ориентированного графа, в котором вершинам соответствуют статусы, а дугам — допустимые переходы между ними. Причем дуги помечены правами пользователя, дающими ему право осуществить этот переход (Рис. 2.1).

Под правами пользователя здесь понимается наличие определенной (обычно авторизующей, см раздел 2.7) связи между пользователем и из-

[5]обычно это означает, что все эти сайты используют одну и ту же онтологию, хотя некоторые из них могут использовать в то же время и какие-то другие онтологии

меняемым итемом. В стандартный набор инструментария разработчика онтологий входит скрипт `workflow`, позволяющий редактировать этот набор ограничений.

Если для типа итема не задано ни одного ограничения, считается что любой пользователь, имеющий права на изменение этого итема, может его перевести из любого статуса в любой.

Динамический элемент EditField автоматически учитывает workflow, если оно существует, и показывает только те статусы, переход в которые возможен в данной ситуации.

2.9 Типы итемов и их свойства

В предыдущих разделах мы рассмотрели разнообразные свойства итемов, неоднократно упоминали о том, что итемы бывают различных типов, но нигде не остановились на том, что же такое тип итемов, и какими свойствами он обладает.

Тип итема хранится в его атрибуте `TYPE_ID` определяет следующие свойства итема:

- Набор расширенных атрибутов (вернее имя таблицы расширенных атрибутов (см. раздел 2.5.4).
- Тип содержимого (см. раздел 2.4).
- Представление по умолчанию (см раздел 2.3).
- Для итемов с типом содержимого Table — набор табличных атрибутов, вернее имя таблицы содержимого (см раздел 2.5.5).
- Множество связей, в которых данный итем может участвовать как пассивный, и множество связей, где он может участвовать как активный.
- Множество представлений, которые применимы для данного итема. Обычно понятие применимости касается тех шаблонов, которые могут быть использованы в качестве представления по умолчанию в тех случаях, когда нас не устраивает стандартное для сайта, и шаблонов дайджестов, которые могут быть использованы для рассылки обновлений в части базы данных, связанной с данным итемом (см главу 10)

Тип итема хранится в базе данных Communiware как итем типа `ITEM_TYPE` и вся вышеперечисленная информация хранится как его атрибуты и связи. Исключение составляет набор допустимых типов связи, которые нельзя выразить как связь типа итема с типом связи, так как тип связи итемом не является[6].

[6] по чисто техническим причинам

Глава 3

Система контекстов

Шаблоны или представления в Communiware играют двоякую роль. С одной стороны, они определяют внешний вид страницы, т.е. содержат HTML-разметку, задающую дизайн сайта, а с другой стороны они определяют набор информации, который будет на странице показан.

Этот набор включает в себя атрибуты некоторых итемов. Тот набор информации, который доступен для визуализации в некоторой точке страницы, называется *контекстом*. Обработка коммуниверного шаблона Коммунивер-сервером представляет собой фактически последовательность операций по созданию и модификации контекстов, и только последней фазой этой обработки является подстановка значений из этих контекстов в HTML-текст шаблонов.

3.1 Контексты итемов

Наиболее распространенным случаем контекста является контекст некоторого итема. Именно такой контекст создается перед началом обработки шаблона после анализа URL (так как в URL присутствует идентификатор некоторого итема). Как правило, именно контексты итемов создаются фильтрами (вернее, контекст, отличный от контекста итема можно создать только SQL- или Perl-фильтром).

Контекстом итема является любой контекст, содержащий атрибут `ITEM_ID` значением которого является идентификатор существующего итема.

В контексте итема всегда доступны все атрибуты данного итема. Даже

если при создании контекста они не были туда помещены (например, фильтром), при обращении к атрибуту Communiware автоматически извлечет его из базы данных, так как известен идентификатор этого итема.

Атрибуты (в том числе и групповые) кэшируются в контексте, поэтому повторное обращение к тому же атрибуту в том же контексте не потребует операций с базой данных.

Особым случаем контекста итема является контекст формы редактирования итема. Эти контексты подробно рассмотрены в разделе 8.1.

3.2 Контексты, не содержащие итемов.

В некоторых специальных случаях приходится изображать в шаблоне списки сущностей, которые не являются итемами. Например, список атрибутов итема или список картинок, включенных в Html-текст итема. В этих случаях создаются контексты, не содержащие в себе `ITEM_ID`.

Такие контексты состоят только из тех атрибутов, которые в них были помещены при их создании. Автоматическое извлечение дополнительных атрибутов в таких контекстах невозможно.

Особым случаем здесь является контекст формы создания итема. Так как итем еще не создан, то у него нет ни идентификатора, ни значений атрибутов, которые можно извлечь из базы. Это сближает контекст формы создания с контекстами не-итемов. С другой стороны, этот контекст обладат рядом общих свойств с контекстом итема, так как ему предстоит после заполнения пользователем формы и обработки ее системой постинга, превратиться в итем.

3.3 Глобальный контекст

Кроме атрибутов итема существует ряд атрибутов, значение которых не меняется в ходе всего формирования страницы. Такие атрибуты называются атрибутами *глобального контекста*. В основном, это атрибуты, значения которых определяются параметрами HTTP-запроса.

Например, если запрос сформирован в результате заполнения пользователем формы поиска (см. раздел 9.1) то все заполненные поля этой формы превращаются в атрибуты глобального контекста, и их значения доступны в любой точке шаблона.

Кроме того существует ряд *„магических"* атрибутов, которые создаются ядром Communiware до начала обработки шаблона. Перечислим эти атрибуты:

AUTHOR_ID идентификатор текущего зарегистрированного пользователя. Если страница генерируется для анонимного пользователя, этот атрибут не определен.

USER совпадает с `AUTHOR_ID` для зарегистрированного пользователя, и содержит строку АНОНИМ для анонимного пользователя. Не следует использовать этот атрибут ни для каких целей, кроме вывода на экран. Для проверок на права доступа и тому подобных вещей следует использовать `AUTHOR_ID`.

URL_PREFIX То что должно идти в URL страниц данного Communiware-сайта до идентификатора итема.

SCRIPT_PREFIX То что должно идти в URL интерфейсных скриптов Communiware до имени скрипта

PIC_PREFIX То, что должно идти в URL картинок и бинарных файлов до идентификатора итем.

USER_STATUS Определен, если текущий пользователь — зарегистрированный. Содержит его атрибут STATUS.

USER_AGENT Строка идентификации пользовательского браузера.

BROWSER_NAME Результат анализа ядром названия пользовательского браузера. Обычно "Mozilla"или "MSIE", реже "Opera"или "Lynx".

BROWSER_VERSION Номер версии браузера как вещественное число, если его оказалось возможным определить в результате анализа строки USER_AGENT.

OS_VERSION Для MSIE — версия Windows под которой он работает (в этом случае знание разновидности ОС важно для корректной настройки WYSIWYG плагина). В случае других пользовательских агентов и, тем более, других ОС может быть не определен.

ACCEPT_TYPE Значение HTTP-заголовка Accept-Type присланного браузером, в виде списка значений MIME-типов, который может быть использован как фильтр, или в операции сравнения множеств.

CURRENT_SERVER Идентификатор Communiware-сайта, для которого генерируется страница. Может не совпадать с атрибутом SERVER текущего итем.

CURRENT_TEMPLATE Идентификатор шаблона страницы, обработка которого производится. Заметим, что атрибут итема TEMPLATE_ID содержит идентификатор шаблона по умолчанию для данного итем.

REMOTE_ADDR IP адрес клиента. В отличие от параметра CGI-environment REMODE_ADDR, учитывает возможность наличия фронтэнда, и указывает первый IP-адрес за фронтэндом (а в CGI-переменной в конфигурации с фронтэндом всегда будет 127.0.0.1).

FORWARDED_FOR Список адресов прокси, следущих после REMOTE_ADDR, если они таковые сообщили.

RANDOM Случайное вещественное число от 0 до 1. См также функцию контекста RANDOM(n). Реально этот атрибут следует рассматривать как функцию контекста без аргументов, а не как атрибут, поскольку все остальные атрибуты глобального контекста не меняют своего значения в процессе обработки страницы, а этот — возвращает новое значение при каждом обращении.

Кроме того, в глобальный контекст помещаются как атрибуты все Cookie, присланные пользовательским браузером.

При этом поле формы всегда имеет приоритет над cookie (т.е. если есть поле формы, то cookie с таким же именем игнорируется), магический атрибут — над полем формы, а атрибут итема — над атрибутом глобального контекста.

3.4 Функции контекста и вычислимые атрибуты

Далеко не всегда значение атрибута контекста удобно использовать в его исходном виде. Например, при выводе на экран типа итема или значения статуса удобнее получить название на текущем языке, а не идентификатор.

В ряде других случаев требуются результаты некоторой обработки информации, или информация, зависящая не только от текущего контекста, но и от каких-то других параметров, например признак наличия определенного права у текущего пользователя на текущий итем (параметром в данном случае является имя права).

Для получения подобной информации существуют функции контекста. На функции контекста очень похожи *групповые атрибуты*, характеризующие свойства не текущего итема, а группы итемов, связанных с текущим,

которые, в силу их близкого родства с фильтрами мы рассмотрим в разделе **??**.

Определены следующие функции контекста:

TXT(АТРИБУТ) где атрибут перечислимого типа (т.е. имеет таблицу значений). Выдает описание текущего значения данного атрибута на текущем языке. (см. раздел 2.5.2).

TXT(значение:АТРИБУТ) выдает описание указанного значения на текущем языке, рассматривая его как значение указанного атрибута. Например, `TXT(TOPIC:TYPE_ID)` всегда выдаст название типа TOPIC, независимо от значения TYPE_ID в текущем контексте.

NAME(АТРИБУТ ИЛИ ТИП СВЯЗИ) выдает интернационализированное название данного атрибута или типа связи.

ENV(переменная) Выдает значение указанной переменной среды. В принципе, при разработке Коммунивер-сайтов редко приходится использовать столь низкоуровневые средства программирования, как обращения к переменным среды. Большинство переменных среды, которые могут заинтересовать разработчика, уже доступны как атрибуты глобального контекста.

TYPE(АТРИБУТ) Выдает тип атрибута (NUMBER, STRING, RICHTEXT, DATE)

RIGHTS(keyword) проверяет права доступа текущего пользователя (раздел 2.7). В качестве ключевого слова может фигурировать как имя связи (не обязательно авторизующей), так и ключевое слово `ACCESS`, которое означает что следует проверить наличие любой авторизующей связи.

RIGHTS(keyword,user) проверяет права указанного пользователя.

RIGHTS(keyword,user,item) проверяет права указанного пользоватля на указанный итем.

ACTIVE(linkname,item_id) true, если данный итем имеет связь указанного типа с указанным, и является в данной связи активным.

PASSIVE(linkname,item_id) true, если данный итем имеет связь указанного типа с указанным, и является в данной связи пассивным.

LEN(АТРИБУТ) Длина значения атрибута в символах. Для атрибутов типа RICHTEXT — без учета разметки.

RANDOM(число) Выдает целое случайное число от 0 до указанного числа. Заметим, что существует еще магический атрибут RANDOM без параметра, выдающий вещественное случайное число от 0 до 1.

VALUE(имя) Возвращает значение атрибута, имя которого передано в качестве параметра. Благодаря дополнительному раунду подстановки в аргументах функций контекста (см. раздел 5.2 это позволяет получить значение атрибута, имя которого хранится в другом атрибуте.

3.5 Изменение контекста в ходе обработки шаблона

3.5.1 Стэк контекстов

На протяжении формирования Web-страницы контекст не остается неизменным. Например, на странице рубрики может быть показан список статей в ней. Формирование каждого элемента списка производится в контексте итема соответствующей статьи, а заголовок страницы и аннотация рубрики выводятся в контексте самой рубрики.

Наиболее распространенным способом изменения контекста является создание контекста другого итема. Единственным способом создания контекста является создание списка итемов с помощью фильтра. Для этого применяются динамические элементы `List` и `Loop`, подробно описанны в разделе 4.1.

Контексту итема на странице как правило соответствует прямоугольный блок, внутри которого могут быть размещены такие же блоки контекстов — элементов списка. Их можно рассматривать как „наложенные” на этот блок. Внутри элементов списка могут содержаться другие списки, порождающие блоки-контексты следующего уровня. Таким образом, в некоторой точке страницы может существовать целая „стопка” контекстов — текущий, объемлющий и так далее.

В данной точке шаблона можно обратиться к атрибутам любого из существующих в ней контекстов. При указании просто имени атрибута мы обращаемся к атрибуту текущего контекста. Если к имени атрибута (или спецификации группового атрибута или функции контекста — после закрывающей круглой скобки) добавить *модификатор контекста* #1, мы полу-

чим атрибут объемлющего контекста, если #2 — объемлющего контекста второго уровня и так далее.

3.5.2 Параметризованные шаблоны

В разных частях страницы могут различаться не только итемы, информация о которых выводится, но и шаблоны, по которым формируется вывод. В случае включения шаблона фрагмента с помощью динамического элемента `Include` (см. раздел 5.7.2), контекст итема не меняется, но фрагмент страницы показывается по другому шаблону. Динамическому элементу Include можно передать параметры, которые будут помещены в текущий контекст в виде атрибутов с именами `ARG1`, `ARG2` и так далее.

Эти атрибуты помещаются в текущий контекст, так что если нам потребуется создать внутри фрагмента список, и тем сменить текущий контекст, то обращаться к параметрам Include потребуется как к атрибутам объемлющего контекста.

3.5.3 Изменение глобального контекста

В некоторых случаях требуется на основании значений текущего контекста принять решение, результатами которого нужно будет воспользоваться в совсем другой части шаблона. В этом случае логичным способом передачи информации являтся помещение ее в глобальный контекст в виде атрибута.

Кроме того, нам может понадобится дать значение по умолчанию атрибуту глобального контекста, который может существовать, а может и не существовать. Например, в формах поиска атрибуты, содержащие результаты пользовательского ввода могут быть не определены, если пользователь еще не нажал кнопку submit в форме, и могут быть пустыми, если пользователь не заполнил соответствующе поле, в то время как для правильного поведения шаблона требуется, чтобы в этих случаях атрибут имел некоторое непустое значение.

Для этих целей предназначен динамический элемент `Define`.

Его синтаксис следующий:

`<:Define` *`АТРИБУТ выражение`* [*`mun`*] `[ifabsent|ifempty]:>`

где *АТРИБУТ* — имя создаваемого атрибута глобального контекста, выражение — выражение, вычисляющее его значение (см. раздел 5.3), необязательный параметр *mun* задает тип нового атрибута (см. раздел 2.5.1), и последний необязательный параметр определяет условия, при которых этот атрибут требуется определить. Условие `ifabsent` выполняется только если в контексте (как текущем, так и глобальном) вообще отсутствует атрибут

с данным именем, а `ifempty` — в тех случаях, когда атрибут либо отсутствует, либо значение его равно пустой строке.

В случае формы поиска `ifabsent` соответствует ситуации, когда форма еще не была заполнена и отправлена на сервер, а `ifempty` — ситуации, когда соответствующее поле было оставлено пустым.

Поведение динамического элемента `Define` в случае, когда атрибут с таким именем уже существует в глобальном контексте, а модификаторы `ifempty` и `ifabsent` не указаны, не определено[1]. Точно так же не определено состояние контекста, если в шаблоне были использованы два динамических элемента `Define` с одинаковым именем.

В силу особенностей реализации парсера шаблонов, атрибут, определенный с помощью динамического элемента `Define` не доступен выше по тексту шаблона от места его определения. Это считается известной ошибкой, и, вероятно будет исправлено в одной из следующих версий Communiware.

3.6 Процедурные вставки

Далеко не все операции с контекстом, которые может потребоваться выполнить на сложном web-сайте выразимы с помощью простейших средств условной обработки, описанных в разделе 5.6 и динамического элемента Define.

Особенно это касается обработки результатов заполнения форм пользователем. Хотя для работы с контекстом форм редактирования итемов существуют дополнительные средства, описанные в главе 8.

Поскольку язык шаблонов Communiware предназначен для описания представлений, а не для реализации алгоритмов, сложные процедуры обработки следует писать на других языках, и вызывать в процессе обработки шаблона как *процедурные вставки*.

В настоящий момент поддерживаются процедурные вставки только на языке Perl.

По соображениям безопасности процедурные вставки не могут быть отредактированы через web-интерфейсы. Они должны быть оформлены как модули Perl и установлены на сервере системным администратором.

---

[1]Это означает, что Communiware вправе делать с этим динамическим элементом все что угодно — изменить значение атрибута, выдать синтаксическую ошибку, превратить атрибут в списочное значение, дописав новое значение в конец списка. Вплоть до версии 0.932 работает первый вариант поведения

Более подробная информация о написании процедурных вставок содержится в «Руководстве программиста Communiware».

Ряд полезных процедурных вставок входит в дистрибутив Communiware. Информацию об их использовании можно найти в «Справочном руководстве Communiware» и во встроенной документации на служебном сайте Communiware.

Вызов процедурной вставки из шаблона осуществляется с помощью динамического элемента `Do`

Его синтаксис следующий:

```
<:Do стадия функция параметры...:>
```

Где *стадия* — этап обработки HTTP-запроса, на котором необходимо вызвать эту процедурную вставку, *функция* — имя процедурной вставки и *параметры* то, что нужно передать ей.

Как правило, в параметрах передается информация об особенностях поведения данной процедурной вставки в данном шаблоне. Атрибуты контекста нет необходимости передавать таким способом, так как процедурной вставке доступен как для чтения, так и для изменения текущий контекст.

Определены следующие *стадии*:

immediate в момент генерации web-страницы. Эта стадия определена для любого шаблона.

postrequest После окончания генерации страницы, когда она уже отправлена пользователю[2].

prepost Перед началом обработки результатов заполнения формы. Определена только для динамических элементов `Do` расположенных внутри форм редактированя итемов (см. главу 8) и форм почтовой рассылки и подписки на дайджесты (см. главы 9 и 10).

postpost После окончания обработки формы, когда транзакция с базой данных уже завершена. Может быть использована для вызова внешних программ, которым должны быть доступны результаты изменений. Определена в тех же случаях, что и `prepost`.

preedit Перед началом обработки того из контекстов постинга, внутри которого употреблен динамический элемент `Do`. В случае если в результате обработки одного из предыдущих контекстов произошла ошибка, и обработка формы отменена, данная вставка может быть не выполнена. Определена в тех же случаях, что и `prepost`. Следует заметить

[2] в текущей версии Communiware не реализована

что в формах подписки и рассылки всегда есть ровно один контекст, в отличие от форм редактирования.

postedit После обработки конкретного контекста, внутри транзакции. Может быть использована только в формах редактирования итемов. То что процедурная вставка выполняется внутри транзакции означает, что

1. Самой процедурной вставке видны изменения, произведенные в результате заполнения формы.
2. Вызванным из нее внешним программам изменения не видны.
3. Ошибка в процессе выполнения процедурной вставки приводит к откату транзакции и аннулированию всех изменений, внесенных данной формой.

Глава 4

Списки и фильтры

4.1 Списки в шаблонах

На web-страницах достаточно часто встречаются списки однородных сущностей — статей в рубрике, результатов поиска и так далее. Поэтому в Communiware формирование списков — один из базовых механизмов.

Список в Communiware — единственный способ показать множество итемов, отличных от итема, указанного в URL. Даже если это множество состоит ровно из одного элемента. Пока мы не рассматриваем задачу создания формы редактирования итема, которой посвязена глава 8, список — единственный способ сформировать контекст итема средствам языка шаблонов.

Основным параметром, который требуется задать для формирования списка является *фильтр*. Фильтр это некоторое условие, позволяющее отобрать из базы Communiware множество итемов. Явное перечисление идентификаторов итема тоже рассматривается как фильтр.

4.2 Особенности контекста списка

В контексте элемента списка кроме атрибутов текущего итем и общих для всей страницы атрибутов глобального контекста присутствует ряд дополнительных атрибутов.

Во-первых, фильтр может вернуть дополнительные поля, которые не являются атрибутами итем. Они будут доступны в контексте, как атрибуты с теми именами, которые им были даны автором фильтра. Обычно такие

атрибуты возвращаются SQL- и Perl-фильтрами (см. раздел 4.10).

Во-вторых, в контексте элемента списка создаются несколько атрибутов, отражающих положение текущего итема в списке.

Кроме этого, в контексте элемента списка всегда присутствует атрибут SEQ_NO — порядковый номер элемента в списке, и ODD, принимающий значение 1 если текущий элемент — нечетный и 0 в противном случае. Этот атрибут позволяет легко реализовать „полосатые" таблицы.

В случае, если список был разрезан на несколько страниц, атрибут SEQ_NO имеет значение порядкового номера на текущей странице, а атрибут TOTAL_SEQ_NO — общего порядкового номера во всем списке.

В-третьих, при наличии в условиях фильтра операции поиска по связи вниз (см. раздел 4.7.4) в контексте элемента списка могут появиться атрибуты `ORDNUM` — порядковый номер (поставленный редактором) данного итема среди потомков того же предка, `DISTANCE` — минимальное расстояние по цепочки связи от итема, потомков которого мы ищем до найденного (только для поиска по иерархической связи на всю глубину) и `PARENT_ID` — идентификатор ближайшего предка *В-четвертых*, дополнительные атрибуты могут быть добавлены в контекст операциями группировки и сортировки списка (см. разделы 4.3 и 4.4).

4.3 Сортировка списков

Фильтр, задающий список итемов задает только критерий отбора. Он как правило, ничего не говорит о том, в каком порядке должны быть показаны отобранные итемы.

Порядок показа является свойством списка, а не фильтра.

Динамические элементы, показывающие список, позволяют задать способ сортировки итемов в качестве отдельного параметра.

В простейшем случае, этот параметр представляет собой просто список имен атрибутов (через запятую), по значениям которых следует отсортировать итемы в списке. Если требуется отсортировать список по *убыванию*, то перед именем соответствующего атрибута надо поставить знак минус.

Можно (но не обязательно) использовать знак плюс для сортировки по возрастанию.

В значительной части случаев в качестве критерия сортировки используется атрибут `ORDNUM`, который является не свойством итема, а свойством связи между двумя итемами. Этот атрибут помещается в контекст элемента списка всеми фильтрами, выполняющими поиск соседей по определенной связи.

4.3.1 Древовидная сортировка

Достаточно часто список в Communiware представляет собой множество итемов, имеющее внутреннюю иерархическую структуру, например иерархия рубрик web-сайта. Для показа иерархических структур в Communiware не используется рекурсивный обход дерева (тем более, что это часто и не дерево, а граф более общего вида)

В этих случаях формулируется фильтр, отбирающий потомков (предков) по иерархии на всю глубину, а потом множество отобранных итемов упорядочивается специальным образом.

Этот способ сортировки называется *древовидной сортировкой*. Задается она как

`t:`*`идентификатор1`*`,`*`идентификатор2`*`,`*`дополнительные критерии`*.

Здесь *идентификатор1* — имя атрибута контекста, содержащего идентификатор текущего итема (обычно `ITEM_ID`), *идентификатор2* — имя атрибута контекста, содержащего идентификатор итема-предка по интересующей нас связи (в обычных фильтрах использующих операцию `<--` он называется `PARENT_ID`), а *дополнительные критерии* — критерии для сортировки потомков одного и того же итема.

В результате древовидной сортировки формируется следующий порядок итемов: сначала идет корень иерархии (если он есть), затем его первый по заданному порядку потомок, потом его потомки, затем следующий потомок корня и так далее.

При этом в контекст каждого элемента помещаются следующие атрибуты:

LEVEL числовой атрибут, указывающий растояние от текущего итема до корня иерархии.

PREV_SIBLING идентификатор предыдущего итема на том же уровне иерархии (атрибут не определен, если данный итем первый из потомков своего предка).

NEXT_SIBLING идентификатор следующего потомка того же предка. Не определен, если данный итем последний из потомков своего предка на том же уровне иерархии.

UNCLES список нулей и единиц длиной LEVEL, в котором единица означает, что у предка на данном уровне есть сосед снизу (следующий потомок того же предка) и 0, если данный предок — последний потомок своего предка.

4.4 Группировка списков

В некоторых случаях просто сортировки списка недостаточно. Необходимо каким-либо образом выделить группы однородных элементов. Особенно часто эта задача возникает в списках с *древовидной сортировкой*, поскольку выделение групп элементов с одинаковым значенем `LEVEL` позволяет визуально изобразить древовидность.

Процесс визуального выделения групп элементов в Communiware называется *группировкой списков*.

4.4.1 Виды группировки

Существуют три способа группировки списка

группировка по тэгу В начале каждой группы вставляется определенный HTML-тэг, а в конце — соответствующий закрывающий.

Именно этот вариант группировки чаще всего используется для отображения древовидной сортировки. Иерархическое представление данных можно оформить, например, посредством тэга `<blockquote>` или, если каждый элемент оформлен как `<DT>`, как вложенные `<DL>`.

Для облегчения подобных операций, при тэговой группировке по числовому атрибуту, закрывающий тэг выводится столько раз, на сколько уменьшилось значение атрибута группировки.

группировка по шаблону Между группами помещается указанный шаблон. Он обрабатывается в контексте, в котором гарантированно присутствуют только значение атрибута группировки в следующей группе, и его значение в предыдущей, сохраненное как `PREV_`*`имя атрибута`*.

„невидимая" группировка Никаких специальных видимых элементов между группами не создается, но в контекст элементов списка помещаются те же самые атрибуты, что и при других видах группировки, что позволяет разработчику шаблона их самостоятельно проанализировать, и вывести необходимые визуальные элементы самому.

4.4.2 Особенности контекста группированных списков

В результате группировки в контексте элемента списка появляется атрибут с именем `PREV_`*`имя атрибута группировки`* и значением, равным значению этого атрибута в предыдущем элементе списка (кроме первого

элемента. Переход из группы в группу можно легко отследить по несовпадению значения этого атрибута со значением атрибута группировки.

4.4.3 Синтаксис группировки

Общий синтаксис группировки имеет вид:

АТРИБУТ:*mun*(*параметр*)

В случае группировки по тегу тип имеет значение `tag` а параметр — имя и атрибуты тэга. Например,

`LEVEL:tag(DL)`

в древовидно-отсортированном списке приведет к формированию вложенной пары тегов `<DL>...</DL>` для каждого следующего значения `LEVEL`. Если при этом элементы списка будут оформлены как `<DT>`, получится визуальное представление иерархии.

В случае группировки по шаблону тип группировки `include`, а параметр — имя шаблона.

В у „невидимой" группировки парамтеров нет. Например,

`TYPE_ID:attr()`

Т.е. указание пустых скобок обязательно.

4.5 Динамические элементы для создания списков

4.5.1 List

Динамический элемент `List` формирует в текущей точке страницы список итемов, возвращенных заданным фильтром, каждый элемент которого оформлен по указанному шаблону фрагмента.

Синтаксис

`<:List` *`filter template sort group cut`*`:>`

Где *фильтр* — спецификация фильтра (см раздел4.7. Спецификация фильтра должна быть передана в динамический элемент как один параметр. Т. е., если она содержит пробелы, ее надо взять в двойные кавычки.

template — имя шаблона, по которому будут оформлены элементы списка.

sort — критерий сортировки списка (см. раздел 4.3).

group — критерий группировки (см. раздел 4.4).

cut — количество элементов списка, которые необходимо показать на данной странице. См. раздел4.6

Все параметры ДЕ List являются позиционными. Т.е. если необходимо пропустить какой-либо из промежуточных параметров, но указать последующий, нужно обозначить пустой параметр с помощью двух двойных кавычек.

Например,

```
<:List {TOPIC<-} def_material_e +ORDNUM  20:>
```

В случае если все параметры от последнего значимого, до последнего возможного не задаются, можно закрыть динамический элемент сразу после последнего значимого параметра:

```
<:List {TOPIC<-} def_material_e:>
```

Параметры *filter* и *template* являются обязательными.

4.5.2 Loop

Динамический элемент `Loop` отличается от `List` только тем, что шаблон оформления элемента списка не хранится в отдельном итеме, а указывается как тело этого блокового динамического элемента.

Все параметры, за исключением параметра *template* полностью совпадают с параметрами ДЕ `List`.

Например:

```
<:Loop {TOPIC<--}:>
<:Attr TITLE:>
<:EndLoop:>
```

4.6 Разрезание длинных списков на страницы

В Web-дизайне достаточно часто возникает ситуация, когда список получается слишком длинный и его имеет смысл показывать постранично.

Для этой цели в динамических элементах `List` и `Loop` предусмотрен параметр „число элементов”.

В этих случаях бывает необходимо предусмотреть возможность формирования ссылки на следующую страницу списка.

Формированием таких ссылкок занимается динамический элемент `Continuation`.

Этот динамический элемент обязательно должен быть употреблен в том же локальном контексте, что и список, на другую страницу которого он отправляет, и, в текущей версии, ниже по тексту шаблона, чем сам список.

Динамический элемент `Continuation` имеет две формы:

<:Continuation next:>, который формирует ссылку на следующую страницу и <:Continuation start:>, формирующий ссылку на первую страницу.

На первой странице списка <:Continuation start:> не показывается, а на последней не показывается <:Continuation next:>. Возможность формирования ссылки на предыдущую страницу или на произвольную страницу по номеру в настоящий момент динамическим элементом Continuation не поддерживаются.

Дополнительные параметры динамического элемента Continuation позволяют задать текст ссылки, шаблон, по которому будет формироваться страница продолжнения (очень часто требуется на страницах продолжения не показывать тех элементов оформления, которые есть на первой) и список атрибутов контекста, которые требуется пробросить с текущей страницы. Это бывает важно для тех списоков, которые формируются по результатом сабмита формы пользователем, например результатов полнотекстового поиска.

В этом случае указание пробрасываемых атрибутов непосредственно в динамическом элементе Continuation предпочтительнее использования динамического элемента PassParams (см. раздел **??**), так как эти атрибуты будут добавлены только к ссылке на страницу продолжения, а не ко всем URL на странице.

Полный синтаксис динамического элемента Continuation следующий:

<:Continuation *тип текст ссылки шаблон атрибуты-тэга атрибуты контекста*:>

Параметр *тип* принимает значения **start** или **next**. Параметры *атрибуты-тэга* и *атрибуты контекста* представляют собой списки пар имя=значение через запятую. Впрочем, если в списке атрибутов контекста присутствует элемент, содержащий только имя, без знака равенства и значения, будет использовано текущее значение в контексте.

Существующая реализация обмена информацией между Continuation и динамическими элементами, генерирующими списки, позволяет использовать на странице только один список, для которого есть ссылки на страницы продолжения.

Но иногда возникает необходимость использовать обрезание списка без формирования страниц продолжения. Например, при показе на всех страницах сайта (в том числе и тех, где есть списки с продолжением) списка пяти последних новостей.

В таких случаях в параметре *обрезание* динамических элементов List и Loop можно использовать конструкцию *число*!. С такими списками нельзя связать динамический элемент Continuation, но зато они ведут себя кор-

ректно на страницах, где есть другие списки с `Continuation`.

4.7 Синтаксис фильтров

Перейдем, наконец, к описанию того, как именно задаются фильтры. Текст фильтра представляет собой несколько *условий* разделенных запятыми и заключенных в фигурные скобки - { и }. Допустим фильтр не содержащий ни одного условия. Вот два примера синтаксически правильных фильтров:

```
{}
{TYPE_ID = 'TOPIC', (PART, TOPIC) <--}
```

Первый из них не содержит никаких условий вообще (что допустимо), а второй - два условия: `TYPE_ID = 'TOPIC'` и `(PART, TOPIC) <--`. Фильтр отбирает итемы, удовлетворяющие *всем* приведенным условиям, т.е. условия объединяются по логическому И.

4.7.1 Атрибуты контекста и атрибуты итемов

Напомним некоторые понятия, употребимые в Communiware. Фильтр выполняется в определенном контексте, т.е. множестве поименованных и имеющих значения *атрибутов контекста*. Напомним, что контекст определен во время выполнения фильтра, и может изменяться со временем. В отличие от атрибутов контекста *атрибуты итема* - это поименованные значения связанные с конкретным итемом. В текущем контексте могут отсутствовать какие-то из используемых фильтром атрибутов. В этом случае они считаются неопределенными. Понятия атрибутов контекста и атрибутов итемов активно используются при описании различных типов условий, возможных в фильтрах.

4.7.2 Условия отношения

К этой группе условий относятся условия, накладываемые на значеня атрибутов отбираемых итемов: „дата публикации не ранее чем ...", „адрес электронной почты равен ..." и т.п. Условия отношения имеют один из двух вариантов синтаксиса:

```
<атрибут итема> <операция> <значение>
<атрибут итема> <операция> (<значение>, <значение>,...)
```

Операция - это

`=` – равенство,

`!=` или `<>` – неравенство,

`>` – больше,

`<` – меньше,

`>=` – больше или равно,

`<=` – меньше или равно,

`like` – операция текстовой „похожести”, такая же, как в языке SQL.

Как видно, в правой части условия может находиться либо единичное значение, либо список значений, разделенных запятыми. В этом случае условие трактуется, как логическое `ИЛИ` для всех значений из списка. Условие `ITEM_ID = ('aaa', 'bbb', 'ccc')` означает, что атрибут ITEM_ID в искомом множестве итемов должен быть равен или 'aaa', или 'bbb' или 'ccc'.

Значение в правой части условий отношения может иметь следующие варианты.

Название атрибута контекста – действительно, название атрибута контекста, например `ITEM_ID, PUBLISHED, SYSDATE`.

Числовой литерал – последовательность цифр 0–9: `1, 456, 789001`.

Строковый литерал – последовательность произвольных символов, заключенная в апострофы: `'пример строкового литерала'`.

Литерал типа дата – предназначен для указания в тексте фильтров константных дат, имеет вид 'YYYY.MM.DD HH.mm.SS'. Обязательно указание всех частей даты, до секунд включительно. Пример литерала типа дата: `'2001.04.01 12.24.37'`.

Выражение над датой – в качестве значений в условиях язык описания фильтров допускает простые выражения над датами имеющие вид `<название атрибута контекста> [+-] <число>`. Таким образом можно выразить „дату на 10 дней большую, чем. . . ”. Пример условия с выражением над датой: `PUBLISHED > SYSDATE - 10`. Число не обязательно должно быть целым. Числа с дробной частью можно применять для указания нецелого количества суток.

Несколько примеров фильтров, использующих условия отношения.

{} „Минимальный" фильтр. Так как никаких ограничивающих условий не указано, то он отбирает *все* итемы, имеющиеся в момент его выполнения[1].

{TYPE_ID = 'TOPIC'} Отбираются все итемы типа TOPIC.

{TYPE_ID = ('TOPIC', 'PART')} Отбираются все итемы типа TOPIC или типа PART.

{ITEM_ID like '%test%'} Отбираются итемы, идентификатор которых содержит в себе строку `test`, например `testname, svrtest5`...

{SERVER = 'default', PUBLISHED > SYSDATE - 10} Отбираются итемы принадлежащие сайту `default` и опубликованные в последние десять дней.

{PUBLISHED > PUBLISHED} Хотя и не очень осмысленный, но вполне работоспособный фильтр. Отбираются итемы, у которых значение атрибута PUBLISHED (левая часть условия) больше, чем значение атрибута PUBLISHED текущего контекста (правая часть), т.е. все, опубликованные позднее текущего.

Два важных замечания относительно условий отношения с атрибутами контекста в правой части.

1. Если при выполнении фильтра атрибут контекста, входящий в правую часть условия не определен, то все это условие „редуцируется" и не оказывает никакого влияния на результат работы фильтра. Например, если при работе последнего из приведенных выше примеров не определен атрибут контекста PUBLISHED, то фильтр возвратит множество всех итемов, как если бы его текст был `{}`.

2. Если в правую часть условия входит один из атрибутов контекста ITEM_ID, CURRENT_ITEM_ID, CURRENT_SERVER, CURRENT_TEMPLATE, TYPE_ID или SERVER и он не определен, то такая ситуация воспринимается как ошибка — если уж ITEM_ID (и другие атрибуты из этого списка) используется, то он *обязан* быть определен.

[1]Строго говоря не вообще все, а все итемы, имеющиеся на данном Communiware-сервере. См. в разделе 4.7.5 описание условия sameserver.

4.7.3 Условия определенности атрибутов

Иногда нас не интересует конкретное значение атрибутов итемов, отбираемых фильтром. Нас могут интересовать итемы, у которых некоторый атрибут *определен*, т.е. имеет какое-то значение, или наоборот *не определен*, т.е. про это значение ничего не известно.

Пока к условиям определенности атрибутов относится единственное условие `has`, проверяющее определенность атрибута итема.

has

Синтаксис этого условия:

```
has ITEMATTR_NAME
```

где `ITEMATTR_NAME` — имя некоторого атрибута итема. Условие отбирает только те итемы, у которых указанный атрибут существует и определен.

4.7.4 Условия связи

Описанные выше условия отношения задают требования к значениям атрибутов итема. *Условия связи* позволяют выразить требования к достижимости одних итемов из других по определенным типам связей. Синтаксис условий связи следующий:

```
<типы связей> <операция> <идентификаторы итемов>
```

<*Типы связей*> - это или название одиночного типа связи, или перечисление через запятую нескольких типов связей, заключенных в скобки. Например: `TOPIC, (TOPIC, PART)`.

<*Идентификаторы итемов*> - или „пусто”, или одиночное значение, или список значений, разделенных запятыми и заключенный в скобки. Значением здесь может быть либо литеральная строка, либо название атрибута контекста. Если правая часть условия не указана (пусто), то ее значением принимается ITEM_ID, т.е. идентификатор текущего итема.

Перед списком значений может быть указан так называемый *квантор*. Квантор — это одна из строк `any` или `all`. Назначение кванторов описано ниже.

<*Операция*> указывает на „критерий достижимости” искомых итемов из итемов заданных в правой части условия.

`<-` отбирает ближайших *потомков* заданных итемов по указанным типам связей.

`->` операция обратная `<-`. Отбирает ближайших *предков* заданных итемов по указанным типам связей.

`<--` отбирает итемы-потомки, достижимые из указанных в правой части по заданным типам связей.

`-->` отбирает итемы-предки, из которых достижим хоть один из заданных итемов по указанным типам связей.

Для условий связи с атрибутами контекста в правой части действует правило, похожее на используемое в условиях отношения: если в правой части условия связи указаны только атрибуты контекста, и все они не определены, то все условие „редуцируется", и не оказывает влияния на работу фильтра. И, точно так же, ошибочным является использование в правой части условия связи одного из атрибутов контекста ITEM_ID, CURRENT_ITEM_ID, CURRENT_SERVER, CURRENT_TEMPLATE, TYPE_ID или SERVER — если этот атрибут не определен.

Квантор `any` является умолчательным, т.е. его можно не указывать. Он означает, что отбираются итемы, связанные указанным образом *с любым* из итемов, перечисленных в списке. Квантор `all` означает, что отбираются итемы, каждый из которых связан *со всеми* итемами из списка.

Примеры фильтров с условиями связи.

`{TOPIC <- 'default'}`
Отбирает итемы, ближайшие соседи итема 'default' по связи TOPIC.

`{TOPIC <- ITEM_ID}`
То же, но для „текущего" итема, идентификатор которого находится в атрибуте контекста ITEM_ID.

`{TOPIC <-}`
В точности то же, что и предыдущий пример. Пустая правая часть неявно заменяется на ITEM_ID.

`{TOPIC <--}`
Все, а не только ближайшие потомки текущего итема по связи TOPIC.

`{(TOPIC, PART) <-- ('default', ITEM_ID)}`
Все потомки текущего итема и итема 'default' по связям типов TOPIC и PART.

`{(TOPIC, PART) <-- all('default', ITEM_ID)}`
: Итемы, *каждый* из которых имеет предком по связям типов TOPIC и PART и текущий итем и итем 'default'.

`{AUTHOR -> ('default', ITEM_ID)}`
: Все авторы текущего итема и итема 'default'.

`{AUTHOR -> all('default', ITEM_ID)}`
: Авторы, которые *одновременно* являются авторами текущего итема и итема 'default'.

4.7.5 Предопределенные условия

Кроме описанных выше условий отношения и связи в тексте фильтра возможно также употребление так называемых *предопределенных условий*.

sameserver

Условие `sameserver` отбирает итемы принадлежащие тому же серверу, что и текущий. То же условие может быть (более длинно) записано, как

```
{SERVER = CURRENT_SERVER}
```

Для условия `sameserver` используется следующее умолчание: оно неявно включается в фильтр, если этот фильтр, после редукции всех условий с неопределенными атрибутами, не содержит ни одной операции связи. Таким образом, при использовании только условий отношения все отобранные итемы будут принадлежать к текущему сайту.

Если у вас в фильтре присутствует поиск по связи, но вы уверены что все выбранные итемы будут принадлежать текущему сайту, крайне рекомендуется указывать это условие. На серверах с большим количеством сайтов, например на массовых хостингах это условие может ускорить работу фильтра на несколько порядков.

allservers

Условие `allservers` играет роль обратную, по отношению к `sameserver`. А именно, в результат работы фильтра попадают итемы *со всех серверов*. Другими словами, это условие отменяет неявное включение условия `sameserver` в фильтр.

Одновременное указание в фильтре условий `sameserver` и `allservers` является ошибкой.

Права доступа — allowed-only и include-forbidden

Эти два условия служат для ограничения (или наоборот — расширения) набора итемов, отбираемых фильтром, в зависимости от прав доступа текущего пользователя.

Условие `include-forbidden` отменяет проверку прав вообще. Т.е. при его указании фильтр будет отбирать даже итемы, которые обычно недоступны по чтению текущему пользователю. Условие `allowed-only` напротив, „включает" проверку прав доступа. Нельзя одновременно указывать оба этих условия.

Если фильтр не содержит ни одного из этих условий, то выполнение или нет проверки прав производится в соответствии со следующими правилами.

1. Права *не* проверяются, если текущий пользователь — не аноним, и имеет статус SUPERUSER.
2. Права *не* проверяются, если среди условий фильтра есть условие

   `BELONGS <linkop> ITEM_ID`

   где `<linkop>` — любое условие связи.
3. Права *не* проверяются, если среди условий фильтра есть

   `<LINKTYPE> <linkop> AUTHOR_ID`

   где `LINKTYPE` — название типа авторизующей связи а `linkop` — любое условие связи.
4. Права *не* проверяются, если сервер, к которому принадлежит текущий итем не требует авторизации.
5. Иначе — права проверяются, даже при отсутствии условия `allowed-only`.

4.7.6 Условия поиска

Строго говоря множественное число (условия) в заголовке употреблены напрасно. Имеется только один тип условия поиска — `contains`. Это условие позволяет отбирать итемы, текст которых удовлетворяет некоторому *образцу поиска*. Иначе говоря, `contains` позволяет производить полнотекстовый поиск среди итемов.

Замечание относительно полнотекстового поиска: в качестве поисковой машины в Communiware используется программный продукт `MnogoSearch`. Из этого следует, во-первых, что для возможности полнотекстового поиска должна периодически запускаться утилита `cmwsidx`, входящая в состав Communiware, во-вторых, что изменения в тексте какого-либо итема отразятся в данных поисковой системы только после очередной индексации посредством `cmwsidx`, и в-третьих — некоторые ограничения на возможности самого поиска. В частности не поддерживается поиск по словам в пределах одной фразы и по „близости" слов.

Условие поиска имеет один из следующих двух форматов:

```
contains CTX_ATTR_NAME
cpntains 'search expression'
```

Т.е. *образец поиска*, в соответствии с которым отбираются итемы, либо является значением некоторого атрибута контекста, либо задан явно, как строка в одиночных апострофах. Пустой или неопределенный образец поиска приводит к ошибке.

В простейшем случае *образец поиска* состоит просто из искомых слов, разделенных пробелами, например `communiware server`. Такое условие приводит к поиску итемов, текст которых содержит *все* указанные слова. При желании (например для увеличения читабельности образца поиска) можно между словами указать операцию `и` (или, что то же самое — `and` или `&`). Обращаем внимание на то, что слова в образце поиска указываются „как есть", без заключения в апострофы или кавычки. Таким образом следующие фильтры

```
{contains 'Gates Bill'}
{contains 'Gates and Bill'}
{contains 'Gates и Bill'}
{contains 'Gates & Bill'}
```

полностью равноценны, о отбирают итемы, текст которых содержит одновременно слова Bill и Gates.

Для указания слов, которые *не* должны содержаться в тексте итемов надо перед такими словами указать операцию отрицания — одно из `не`, `not` или `^`. Например фильтр

```
{contains 'Gates не Bill'}
```

будет искать итемы, текст которых содержит слово Gates, но не содержит слово Bill.

Если между словами в образце поиска указать операцию **или** (а также `or` или `|`), то, как и следует ожидать, фильтр будет отбирать итемы, содержащие хотя бы одно из этих слов. Пример:

```
{contains 'Gates или Bill'}
```

Такой фильтр отберет итемы, текст которых содержит хотя бы одно из слов Bill и Gates.

Операция *не* имеет высший приоритет, затем, по старшинству, идет *и*, и, наконец *или*. Таким образом фильтр

```
{contains 'Gates не Bill или Linus Torwalds'}
```

отберет итемы, текст которых либо одновременно содержит слова Linus и Torwalds, либо содержит слово Gates, но не содержит слово Bill.

Для изменения интерпретации образца поиска можно группы слов и операций заключать в круглые скобки, и применять к этим выражениям в скобках те же операции *не*, *и* и *или*. Таким образом можно строить достаточно сложные образцы поиска:

```
{contains '(Gates не Bill) и
  (Linus или Torwalds) и
    не (Microsift или Linux)'}
```

Вполне допустимо в одном образце поиска использовать любое — русское, английское или символьное — представление операций. Таким образом фильтр из предыдущего примера может быть переписан как

```
{contains '(Gates не Bill) and
  (Linus | Torwalds) и
    ^ (Microsift or Linux)'}
```

без изменения его смысла.

Нормализация и стоп-слова

При обработке поискового запроса Communiware проводит так называемую *нормализацию* слов, входящих в запрос. Нормализация здесь — это приведение слова к „канонической", „основной" грамматической форме и к нижнему регистру символов. Такая же нормализация производится и при индексировании текстов итемов. Таким образом фильтр

```
{contains 'СтолА'}
```

отберет итемы, содержащие слова *стол*, *столы*, *столами* и т.п.

Наличие в образце поиска таких распространенных слов, как *так*, *что*, *потому*, или, для английского языка, *the*, *by*, *where*, как правило бесполезно. Эти слова содержатся в большинстве текстов и никак не помогают в поиске. Поэтому в Communiware хранится список таких слов (они называются „стоп-слова"), и при обнаружении их в образце поиска они „убираются", и не участвуют в поиске. Иначе говоря фильтр

```
{contains 'дом который построил джек'}
```

равносилен

```
{contains 'дом построил джек'}
```

Возможна ситуация, когда все слова в образце поиска являются стоп-словами. Такой случай рассматривается как ошибочный — так же, как и полное отсутствие образца поиска.

4.8 Хранимые фильтры и их уточнение

Как говорилось выше фильтры могут записываться непосредственно в тексте шаблонов — в динамических элементах List и Loop. Такие фильтры будем называть литеральными. Вот пример работоспособного шаблона, использующего литеральные фильтры.

```
<:Surround *blanc_decor:>
<:Loop "{TOPIC <-}" "+ORDNUM,TITLE,ABSTRACT":>
  <:Attr ORDNUM:>
  <:ItemLink ITEM_ID @TITLE:>
  <:Attr ABSTRACT:><br>
  <br>
<:EndLoop:>
<:EndSurround:>
```

При просмотре итема по этому шаблону мы увидим список его непосредственных потомков итема по связи TOPIC, упорядоченных по ORDNUM. Однако тексты фильтров могут храниться и в базе данных. Такие фильтры будем называть *хранимыми*. Далее описаны особенности использования хранимых фильтров.

4.8.1 Создание и редактирование хранимых фильтров

Создание и редактирование хранимых фильтров выполняется с помощью входящего в Communiware скрипта `filter`.

Запуск скрипта filter

Для запуска скрипта `filter` можно просто указать в поле для набора ulr вашего бровсера `http://serverurl/scripts/filter`. `serverurl` — это url используемого Communiware-сервера, `scripts` — url, по которому располагаются все Communiware-скриты. Это значение задается при конфигурации Communiware, но как правило равно умолчательному значению `scripts`.

Второй путь для запуска скрипта `filter` — воспользоваться динамическим элементом `Script` для формирования ссылки на этот скрипт из какого-либо шаблона. Например

```
<:Script filter "Редактировать фильтр":>
```

Скрипту `filter` могут быть переданы параметры в query-string, т.е. части url после первого знака вопроса. В частности скрипт реагирует на задание следующих параметров в query-string:

SERVER — Communiware сервер на котором будет тестироваться / создаваться фильтр.

FILTER_ID — идентификатор того фильтра, который мы хотим редактировать.

ITEM_ID — идентификатор итема, в контексте которого будет производиться тестирование фильтра.

Таким образом реальный url вызова скрипта мог бы выглядеть как

```
http://www.mycmwserver.ru/scripts/filter?
   FILTER_ID=GetHelp&ITEM_ID=default&SERVER=kms
```

Разбивка на две строки приведена, конечно, только для удобства печати.

Поля формы скрипта filter

После запуска скрипта `filter` на экране бровсера появляется форма с полями, позволяющими выполнять создание, удаление, тестирование и редактирование фильтра. Рассмотрим эти поля.

Идентификатор — в этом поле находится имя текущего редактируемого фильтра. Если изменить имя в этом поле, а затем нажать кнопку „сохранить", то фильтр будет сохранен под новым именем.

Очистить форму для создания нового фильтра — по нажатию этой кнопки происходит очистка всех полей формы (кроме поля „Контекст для тестирования"). Если нужно создать новый фильтр „с нуля", а не изменением существующего, то работу целесообразно начать с нажатия этой кнопки.

Перейти к фильтру — эта кнопка и следующее за ней drop-down меню служат для перехода к редактированию какого-либо другого фильтра. Сначала нужно выбрать имя нужного фильтра в меню, а затем нажать на кнопку. После этого скрипт `filter` прочитает указанный фильтр так же, как если бы было указано `FILTER_ID=...` в query-string.

Запрос — в этом поле вводится полный текст фильтра, включая объемлющие фигурные скобки.

Разрешить переписывание — значение этого чекбокса существенно только при работе с SQL-фильтрами (см. п. 4.10).

Аргументы — в этом поле задается информация об аргументах фильтра. Если фильтр имеет параметры, то значение этого поля указывается как имена параметров, разделенных запятыми. Однако имеет значение только количество аргументов, но не их названия. (Для SQL-фильтров это не так - см. п. 4.10.2.) Таким образом указание в поле *Аргументы* `VAL1,VAL2` или `PUBLISED,TITLE` полностью равноценны. Последним значением в списке аргументов может быть указана звездочка - *. Смысл ее — „все остальные параметры". Подробнее параметризация фильтров описана ниже.

Значения для тестирования — определяет значения параметров фильтра при тестировании. Поле „Аргументы" определяет *количество* аргументов фильтра, а поле „Значения для тестирования" — те значения, которые должны быть сопоставлены каждому из аргументов при тестировании[2].

Атрибуты для извлечения — как говорилось выше фильтры могут не только отбирать нужное множество итемов, но и сразу же извлекать

[2]Для программистов: „Аргументы" это аналог формальных параметров подпрограммы, а „Значения для тестирования" — фактических.

из базы данных какие-либо атрибуты этих итемов (см. п. 4.2). В данном поле задается список атрибутов итемов которые желательно при тестировании извлечь с помощью фильтра, а также порядок сортировки. Имена извлекаемых атрибутов разделяются запятыми. Сортировка по возрастанию атрибута обозначается знаком „+" перед именем извлекаемого атрибута, а сортировка по убыванию — знаком „-".

Таким образом значение поля „Атрибуты для извлечения" равное

```
TITLE,+EMAIL,-UIN
```

показывает, что при тестировании фильтра мы хотим извлечь атрибуты итема `TITLE`, `EMAIL` и `UIN`, причем набор возвращенных итемов должен быть отсортирован по возрастанию значения атрибута `EMAIL` а в пределах каждой группы `EMAIL` — по убыванию атрибута `UIN`.

Контекст для тестирования — как и следует из названия задает тот контекст, в котором будет производиться тестирование фильтра. Контекст, как и обычно, это множество поименованных значений. Важнейшие для тестирования это `ITEM_ID`, `AUTHOR_ID`, `CURRENT_SERVER`. Умолчания для значений этих атрибутов скрипт пытается определить из query-string. Для определения значения атрибута `CURRENT_SERVER` используется поле `SERVER` из query-string. Кроме того возможно задавать значения произвольных атрибутов контекста, необходимых для тестирования фильтра.

Значения атрибутов тестового контекста задаются в форме

```
ATTRNAME=ATTRVALUE
ATTRNAME=VALUE1,VALUE2 LIST/
ATTRNAME=VALUE1,VALUE2 LIST/TYPE
```

Все, что расположено в строке после первого знака равенства является значением атрибута `ATTRNAME`. Возможно также задавать *списочные значения* атрибутов добавляя в конце строки конструкцию `LIST/` или `LIST/TYPE`, где `TYPE` — один из типов данных Communiware: `NUMBER, DATE, STRING, RICHTEXT`. В первом случае скрипт будет пытаться определить тип атрибута контекста по его имени, а во втором тип атрибута указывается явно.

Символы запятых, входящие в *значения* атрибутов надо предварять обратной косой чертой, т.е. записывать как `\,`.

Тест — нажатие на эту кнопку приводит к выполнению редактируемого фильтра с параметрами и в контексте заданными соответствующими полями формы. При выполнении тестового запуска фильтра выводится тот SQL-запрос, который был порожден трансляцией исходного текста фильтра в заданном контексте.

Если фильтр возвращает непустое множество итемов, то выводится также таблица с результатами, возвращенными фильтром.

Ограничить размер вывода — при больших количествах итемов, возвращенных фильтром неудобно просматривать всю таблицу результатов „одним куском". Это поле устанавливает максимальное число записей из результатов теста, показываемых одновременно.

Если фильтр возвращает больше записей, чем установлено данным полем, то в начале и в конце таблицы результатов выводятся кнопки для „листания" таблицы вперед и назад.

План выполнения — кроме собственно выполнения SQL-запроса, порожденного фильтром скрипт `filter` позволяет узнать *как* будет выполняться этот запрос. Для получения этой информации применяются специфичные для применяемых SQL-серверов средства. Для PostgreSQL это команда explain, для Oracle — explain plan. Результат выполнения этих команд также серверо-специфичен, так что для полного его понимания надо изучить соответствующие разделы руководства применяемого SQL-сервера.

Пример плана выполнения для PostgreSQL:

```
Hash Join  (cost=4.83..7.45 rows=1 width=48)
  ->  Seq Scan on item  (cost=0.00..2.41 rows=41 width=2
  ->  Hash  (cost=4.83..4.83 rows=1 width=24)
        ->  Index Scan using passive_rel on item_rel
              item_rel_1 (cost=0.00..4.83 rows=1 width=2
```

Комментарий — как и следует ожидать, в это поле вводится в произвольной форме комментарий к скрипту, описывающий его назначение и особенности.

Сохранить — при нажатии этой кнопки все изменения, сделанные при редактировании фильтра (текст фильтра, параметры, комментарий) будут занесены в базу данных Communiware. При выполнении этой команды значение поля „Идентификатор" не должно быть пустым.

Преобразовать в SQL — по нажатию этой кнопки производится преобразование фильтра из „обыкновенного" в SQL-фильтр (см. п. 4.10). При этом в поле „запрос" записывается преобразованный в SQL запрос фильтра. Вполне возможно, что полученный SQL-текст и параметры фильтра могут потребовать „ручной" доводки.

Удалить фильтр — эта кнопка выводится только если редактируется существующий фильтр (не только что созданный), и этот фильтр не используется ни в одном шаблоне. Как и можно предположить, после нажатия на эту кнопку фильтр будет безвозвратно удален, будьте осторожны!

Редактирование и тестирование фильтра

При создании нового или редактировании существующего фильтра выполняется следующая последовательность действий.

- Для создания нового фильтра нажатием на кнопку „Очистить форму" очищаются все поля формы скрипта `filter`.
- Для редактирования существующего — его имя выбирается из меню, после чего нажимается кнопка „Перейти к фильтру".
- В полях „Запрос", „Аргументы" и „Комментарий" вводятся или редактируются соответственно текст фильтра, описание его параметров и комментарий.
- Для тестирования работы фильтра заполняются поля „Значения для тестирования", „Атрибуты для извлечения" и „Контекст для тестирования". Устанавливается желаемое значение максимума выводимых записей, после чего нажимается кнопка „Тест".
- Для анализа выполнения сгенерированного SQL-запроса нажимается кнопка „План выполнения".
- Для сохранения нового состояния фильтра после редактирования заполняется (при создании нового фильтра!) поле „Идентификатор" и нажимается кнопка „Сохранить".

  Заметим, что при изменении значения поля „Идентификатор" при редактировании существующего фильтра он будет сохранен под новым именем, а фильтр со „старым" именем останется неизменным.
- Для удаления фильтра нажимается кнопка „Удалить фильтр".

4.8.2 Параметризация хранимых фильтров

Как уже говорилось при описании скрипта `filter`, хранимые фильтры могут иметь набор позиционных параметров, значения которых указываются в шаблоне, в месте вызова фильтра. Опишем подробнее работу с параметрами фильтров.

Набор параметров указывается в поле *Аргументы* при работе со скриптом `filter`. Имена параметров разделяются запятыми. Однако имеет значение только количество аргументов, но не их названия. (Для SQL-фильтров это не так - см. п. 4.10.2.) Таким образом указание в поле *Аргументы* `VAL1,VAL2` или `PUBLISED,TITLE` полностью равноценны. Последним значением в списке аргументов может быть указана звездочка - *. Смысл ее — „все остальные параметры".

При использовании хранимого фильтра в шаблоне значения параметров указываются после имени фильтра в скобках. При выполнении хранимого фильтра значения параметров подставляются в атрибуты контекста с именами `ARG1`, `ARG2`. Значения параметров соответствующих звездочке записываются в атрибут контекста `ARGREST` как строка из значений, разделенных запятыми.

Предположим, что мы хотим создать хранимый фильтр, отбирающий итемы заданного типа, имеющие дату публикации больше, чем указанная. Тогда при создании и тестировании фильтра мы можем задать следующие значения:

```
Идентификатор:
  uf_example1
Запрос:
  {TYPE_ID = ARG1, PUBLISHED > ARG2}
Аргументы:
  TID,PUB
Значения для тестирования:
  TOPIC, 2001.04.01 00.00.00
```

Описанный метод параметризации хранимых фильтров подходит для случая, когда параметризуются значения в правой части условий отношения, т.е. атрибуты контекста. Однако часто нужно параметризовать и другие части условий. Например типы связей в условиях связи или даже *названия* атрибутов итема в условиях отношения. В этих случаях применяется другой способ параметризации — текстовая подстановка. Дело в том, что в тексте хранимого фильтра перед выполнением производится такая же подстановка значений атрибутов контекста на место конструкции `@ATTRNAME`,

что и в динамическом элементе `Subst`. Поэтому для параметризации „неатрибутов" в тексте хранимого фильтра нужно использовать `@ARG1`, `@ARG2`, `@ARGREST` и т.п. Обращаем внимание на то, что при этом выполняется *текстовая* подстановка, т.е. апострофы, там, где они нужны в тексте фильтра, надо указывать самим.

Предположим, что мы хотим создать фильтр, отбирающий потомков текущего итема по (нескольким!) указанным типам связей. Тогда поля скрипта `filter` можно заполнить следующим образом:

```
Идентификатор:
  uf_example2
Запрос:
  {(@ARGREST) <-}
Аргументы:
  *
Значения для тестирования:
TOPIC,PART
Контекст для тестирования:
ITEM_ID=default
```

Теперь этот фильтр может быть использован где-нибудь в шаблоне примерно так: `<:Loop "uf_example2(TOPIC,PART)TITLE":>...`

4.8.3 Уточнение условий в хранимых фильтрах

Использование хранимых фильтров в шаблонах позволяет выразить следующее: „хочу получить из итемов, отбираемых таким-то хранимым фильтром удовлетворяющие следующему условию ...". Т.е. возможно добавить к условиям фильтра еще какие-то условия, задав их непосредственно в тексте шаблона.

Для этого в тексте шаблона (в динамических элементах `Loop` и `List`) после имени хранимого фильтра нужно указать вертикальную черту (|), а затем, в фигурных скобках, дополнительные условия на языке описания фильтров. Например конструкция

```
<:Loop "{uf_example2(TOPIC,PART)
                  | PUBLISHED > SYSDATE - 10}":>
```

возвратит список итемов — непосредственных потомков текущего по связям TOPIC и PART (результат работы `uf_example2`), но только те из них, которые опубликованы за последние десять дней.

4.9 Агрегатные функции над фильтром

Довольно часто нам требуется получить некоторую интегральную характеристику набора итемов, например их количество или последнюю дату публикации.

В Communiware, как и в реляционных базах данных, такие характеристики называются *агрегатными функциями*. Агрегатные функции это разновидность функций контекста (см. раздел 3.4), которые возвращают характеристинку некоего связанного множества итемов, а не самого текущего контекста.

Communiware поддерживает следующие функции контекста `COUNT`, `MIN`, `MAX`, `AVG` и `SUM`

Синтаксис функций контекста следующий:

имя`(`*атрибут*`,`*условие*`...)`

Где *атрибут* это атрибут над которым вычисляется функция, т.е, например, тот, максимальное значение которого мы хотим получить. В случае функции `count` обычно в качестве атрибута используется `ITEM_ID`, а *условие* — любое условие фильтрации, описанное в разделе 4.7. Условий может быть перечислено через запятую сколькуо угодно.

При подстановке через `@` агрегатные функции, так же как групповые атрибуты или функции контекста с аргументами, следует заключать в фигурные скобки вместе с параметрами.

Групповые атрибуты, описанные в следующем разделе, фактически являются частным случаем агрегатных функций и внутренне реализованы именно через них.

Примеры использования агрегатных функций

```
<:Counter COUNT(ITEM_ID,TOPIC<--,TYPE_ID='ARTICLE') стат(ья!

эквивалент CHILDREN(TOPIC,ARTICLE)

<:Subst "Средний размер статьи на этом сайте " .
  "@{AVG(TEXTSIZE,TYPE_ID='ARTICLE')} символов":>
```

4.9.1 Групповые атрибуты

Напомним, что среди возможных модификаций имен атрибутов контекста, используемых, например, в динамическом элементе `<:Attr:>` есть так

называемые *групповые* или *qualified* атрибуты, значения которых содержат некоторую информацию сразу о некотором множестве итемов связанных с данным. Например, атрибут `LASTPUBLISHED(TOPIC)` представляет собой максимальное значение атрибута `PUBLISHED` среди итемов, связанных с текущим связью типа `TOPIC`.

Внутренняя реализация получения групповых атрибутов состоит в автоматическом формировании текста некоторого фильтра и выполнении этого фильтра для получения нужного значения, так что можно сказать, что групповые атрибуты — это ближайшие родственники фильтров.

Синтаксис групповых атрибутов

```
NAME(LINK,[TYPE],[STATUS],[FROM_DATE],[TO_DATE])
```

где

NAME — одно из допустимых имен группового атрибута, описанных далее.

LINK — название типа связи. Если за этим названием следует знак – (минус), то в рассмотрение принимаются только ближайшие соседи тиема по указанной связи. Если название типа связи предварено знаком * (звездочка), то в рассмотрение принимаются только итемы только того Communiware сайта, к которому принадлежит текущий итем.

TYPE — или название типа итемов, или список таких типов, разделенных запятыми и заключенных в квадратные скобки.

STATUS — или значение статуса итемов, или список значений статусов, заключенный в квадратные скобки.

FROM_DATE и TO_DATE — даты в стандартном для Communiware формате, т.е. YYYY.MO.DD HH.MI.SS, за которыми может следовать восклицательный знак. Даты *без* восклицательного знака обозначают значения атрибута `PUBLISHED`, а *с* восклицательным знаком — значения атрибута `UPDATED`.

Если какой-либо из аргументов группового атрибута „в середине” не указывается, то на его месте можно указать одиночную звездочку.

Некоторые из возможных имен групповых атрибутов[3]:

[3]Все имена существующих групповых атрибутов можно просмотреть с помощью скрипта `showattrs`.

CHILDREN — общее количество итемов, удовлетворяющих условию группового атрибута.

TREESIZE — суммарный объем текстов итемов.

LASTORDNUM — максимальный порядковый номер связи.

Примеры использования групповых атрибутов:

```
<:Attr "CHILDREN(TOPIC)":> - общее количество потомков
                             текущего итема по связи
                             типа TOPIC
<:Attr "CHILDREN(*TOPIC-)":> - то же, но только ближайшие
                             потомки и только на текущем
                             сервере
<:Attr "TREESIZE(TOPIC,[TOPIC,ARTICLE],*,2001.04.01
                        00.00.00,2002.04.01 23.59.59)":>
                             - Суммарный объем текста итемов
                             с типами TOPIC и ARTICLE, потом
                             данного по связи TOPIC и
                             опубликованных в указанном
                             диапазоне дат
```

4.9.2 Агрегатные функции и фильтры

Как говорилось выше для получения значений групповых атрибутов и агрегатных функций автоматически формируется и выполняется специальный фильтр, извлекающий необходимые значения. В этой секции описаны свойства фильтров, позволяющие при выполнении фильтра получать не атрибуты некоторого множества итемов, а агрегатные значения, такие, как сумма, общее количество или среднее значение какого-либо атрибута.

Доступ к этим функциям можно получить либо при программировании процедурных вставок и специальных фильтров на языке Perl, либо воспользовавшись интерфейсом разработчика для тестирования фильтров.

Для извлечения агрегатных значений нужно указать специальные, описанные ниже, конструкции либо в списке полей *Атрибуты для извлечения* скрипта `filter` либо в поле `QUERY` таблицы `ATTRIBUTE` при создании нового атрибута. Непосредственно в шаблонах, например в динамических элементах `Loop` или `List` агрегатные атрибуты указывать *нельзя*.

Итак, синтаксис агрегатного атрибута:

```
name(ATTR_NAME)
```

name — одна из агрегатных функций.

ATTR_NAME — имя атрибута итема.

Определены следующие агрегатные функции:

count — общее количество итемов, возвращаемых фильтром, имеющих непустое значение указанного атрибута.

max — максимальное указанного значение атрибута, среди итемов, возвращенных фильтром.

min — максимальное указанного значение атрибута, среди итемов, возвращенных фильтром.

sum — сумма значений указанного атрибута для итемов, возвращенных фильтром.

avg — среднее значение указанного атрибута для итемов, возвращенных фильтром.

Функции `count`, `max` и `min` могут быть указаны для любого атрибута итема, а функции `sum` и `avg` — только для числовых атрибутов.

Таким образом, если при редактировании фильтра мы зададим:

```
Запрос:
{TOPIC <--}

Атрибуты для извлечения:
count(ITEM_ID),max(PUBLISHED),sum(TEXTSIZE)
```

то мы можем получить при тестовом запуске фильтра результат вроде такого:

Row N	SUM_TEXTSIZE	COUNT_ITEM_ID	MAX_PUBLISHED
0	544950	206	2002.07.10 10.57.28

Обратите внимание, как формируются имена возвращенных фильтром атрибутов — к имени агрегатной функции в верхнем регистре добавляется символ подчеркивания и имя атрибута этой агрегатной функции.

4.10 Специальные типы фильтров

В этой главе мы рассматривали до сих пор только фильтры, написанные на специальном языке, описанном в разделе 4.7. Однако в Communiware возможно использование и других типов фильтров. К ним относятся литеральные фильтры, в которых отбираемые итемы явно перечислены, фильтры, в которых процедура отбора записана на алгоритмическом языке Perl и, наконец, SQL-фильтры, в которых способ отбора итемов записан как оператор SELECT языка SQL.

Рассмотрим подробнее все эти разновидности фильтров.

4.10.1 Литеральные фильтры

В программировании шаблонов для Communiware может встретиться ситуация, когда набор интересующих нас итемов известен *на этапе разработки*, т.е. может быть записан явно в самом шаблоне. Для предоставления такой возможности в шаблонах можно использовать *литеральные фильтры*.

Литеральный фильтр представляет собой список имен итемов разделенных запятыми и заключенный в круглые скобки. Такое выражение может встречаться везде в шаблонах, где ожидается фильтр, например в динамических элементах `Loop` и `List`.

Пример использования литерального фильтра:

```
<:Loop "(default,someitem,otheritem)":>
  Item Id: <:Attr ITEM_ID:><BR>
<:EndLoop:>
```

Этот фрагмент шаблона приведет к выводу текста

```
  Item Id: default<BR>
  Item Id: someitem<BR>
  Item Id: otheritem<BR>
```

Обратите внимание на то, что значения элементов списка последовательно присваиваются атрибуту `ITEM_ID`. Это поведение литерального фильтра можно изменить добавив после закрывающей скобки какое-либо другое имя атрибута, отделенное от скобки двоеточием, например:

```
<:Loop "(red,yellow,blue):COLOR":>
  Color: <:Attr COLOR:><BR>
<:EndLoop:>
```

4.10.2 SQL-фильтры

Как говорилось выше „обычные"[4] фильтры перед выполнением транслируются в SQL-запрос к серверу. Результат выполнения этого запроса и есть результат фильтра.

Однако возможны случаи, когда выразительных средств языка описания фильтров недостаточно для определения искомого множества итемов. Например: „найти множество итемов, имеющих не меньше двух потомков по связи типа `TOPIC`". Или: „найти предков по связи типа `AUTHOR` всех итемов являющихся потомками данного по связи типа `TOPIC`". Подобные условия относительно легко могут быть выражены на языке SQL. SQL-фильтры как раз и дают возможность написания фильтров запрос которых написан на SQL, а не на языке описания фильтров.

SQL-фильтры создаются, тестируются и модифицируются с помощью того же скрипта `filter`, что и унифильтры. Однако работа с SQL-фильтрами имеет ряд особенностей.

Обращение к таблицам базы данных Communiware

Для работы с унифильтрами достаточно знания языка описания фильтров и понятия об атрибутах итемов и связях между итемами. Этого недостаточно для написания работоспособного SQL-фильтра, т.к. от разработчика требуется написать корректный SQL-запрос, извлекающий данные из таблиц базы данных „нижележащего" SQL-сервера.

Рассмотрение набора и структуры таблиц, входящих в базу данных Communiware достаточно объемно и выходит за рамки данного документа. Отсылаем интересующихся к „Full Reference Manual"[5]

Приведем, однако, *неполный и очень краткий* перечень таблиц БД Communiware.

ITEM — содержит атрибуты, общие для всех итемов, такие, как TYPE_ID, TITLE, ABSTRACT, SERVER, PUBLISHED, STATUS. Запись в этой таблице существует *для каждого имеющегося итема*.

TEXT (при использовании PostgreSQL эта таблица называется TEXT_ITEM) — содержит тексты итемов.

[4]Как синоним выражению „обычные фильтры" будем далее употреблять термин „унифильтры".

[5]Даже название этого документа пока, к сожалению, условное.

ITEM_LINK — содержит *непосредственные* связи между итемами, т.е. информацию о том, что „этот итем связан связью такого типа с вот этим итемом".

ITEM_REL — содержит информацию о *транзитивном замыкании* связей, т.е. о том, что „из этого итема можно за столько-то шагов добраться до вот этого по связям такого типа".

AUTHOR — данные о зарегестрированных пользователях.

SERVER — данные о имеющихся на данном хосте Communiware-серверах.

А всего в базе данных Communiware несколько десятков таблиц...

Переписывание SQL-фильтра

Так же, как исходный текст унифильтра преобразуется в SQL-текст, так и исходный текст SQL-фильтра перед выполнением может (хотя и не обязан) быть переписан для того, чтобы обеспечить извлечение нужных атрибутов итемов а также необходимый порядок сортировки. Это выполняется специальной компонентой Communiware — „машиной переписывания".

Благодаря наличию этой машины, одни и те же фильтры можно использовать в очень большом числе шаблонов.

К сожалению, машина переписывания имеет свои ограничения. Написание полноценного парсера SQL — сложная задача, решать которую в полном объеме не было необходимости.

В большинстве случаев машина переписывания в состоянии обнаружить фильтр, который не может быть корректно переписан. Но иногда ее логика может дать сбой и в результате переписывания получится некорректный SQL-запрос, или, что еще хуже, запрос, дающий неверный результат. Возможно также, что в результате переписывания существенно ухудшится время выполнения запроса.

Поэтому все вновь разрабатываемые SQL-фильтры следует тестировать на корректное переписывание а также обращать внимание на скорость выполнения запроса и план его выполнения — эти данные доступны при работе со скриптом `filter` так же, как и для унифильтров.

В случае, если переписывание фильтра дает неадекватные результаты, следует убрать отметку с чекбокса „Разрешить переписывание".

В этом случае фильтр будет всегда выполняться в том виде, в каком он был вами введен, а недостающие атрибуты, используемые в шаблонах, извлекаться дополнительными SQL-запросами.

Сортировка при этом будет производиться Communiware, а не сервером баз данных.

Если вы объявили фильтр неперезаписываемым, следует позаботиться о том, чтобы он извлекал наиболее часто используемые атрибуты даже в неперезаписанном виде. В частности, атрибут LASTCHANGE следует извлекать всегда, так как он используется интерпретатором шаблонов при любой обработке контекста итема.

Существенно важным моментом является то, что сортировка неперезаписываемого фильтра возможна только по тем атрибутам, которые он извлекает. Поэтому все атрибуты, которые могут быть использованы в качестве критериев сортировки также следует извлекать.

Поскольку неперезаписываемые фильтры обычно имеют весьма ограниченную область применения — в одном-двух шаблонах в рамках конкретного сайта, разработчик такого фильтра обычно представляет себе, какие атрибуты могут понадобиться.

Запрос SQL-фильтра

Запрос SQL-фильтра при его создании или редактировании вводится в то же поле *Запрос* формы скритпа `filter`, что и для унифильтров. Однако сам запрос при этом представляет собой оператор SELECT языка SQL. Более того: именно по тексту, записанному в поле *Запрос* скрипт `filter` определяет — представлен унифильтр или SQL-фильтр. Из этого, в частности, следует, что изменив текст в поле *Запрос* можно из унифильтра сделать SQL-фильтр и наоборот.

Так же, как и унифильтры SQL-фильтры могут быть параметризованы. На месте всех параметров (значеня которых будут подставляться при выполнении фильтра) в тексте запроса должны быть указаны так называемые „заменители”[6], знаки вопроса. Пример тривиального параметризуемого SQL-запроса:

```
select item_id from item where type_id = ?
```

Как известно язык SQL не чуствителен к регистру символов имен и ключевых слов, так что предыдущий пример мог быть набран как `select ITEM_ID from ITEM where TYPE_ID = ?`, это вопрос предпочтений и читаемости текста.

Существенным моментом при написании запросов SQL-фильтров является то, что эти запросы могу быть написаны на диалекте SQL *специфичном*

[6]Соответствующий английский термин — placeholder.

для используемого SQL-сервера, Oracle или PostgreSQL. С одной стороны это дает возможность использовать больше нестандартных возможностей данного SQL-сервера, но с другой стороны такие серверо-зависимые фильтры не могут быть перенесены на Communiware систему, использующую другой SQL-сервер.

Параметризация SQL-фильтра

Так же, как и унифильтры, SQL-фильтры могут иметь параметры, значения которых будут задаваться при выполнении фильтра, и описание этих параметров вводится в поле *Аргументы* скрипта `filter`.

Однако способ задания параметров для SQL-фильтров другой, чем для унифильтров. Как говорилось выше (стр. 65) для унифильтров существенно только количество приведенных в поле *Аргументы* значений, но не их конкретные имена. Для SQL-фильтров это не так. SQL-фильтр может иметь *позиционные параметры*, значения которых будут переданы фильтру при его использовании в шаблонах и использовать как параметры *атрибуты контекста*. В поле *Аргументы* должно быть приведено описание для каждого placeholder-а (вопросительног знака) встречающегося в тексте запроса.

- Для позиционных параметров таким описанием служит порядковый номер этого параметра начиная с 1.
- Для параметров – атрибутов контекста – описание параметра это просто имя соответствующего атрибута.

Таким образом, если заданы

```
Запрос:
select passive as item_id from item_link
     where active = ? and linktype_id = ?

Аргументы:
  ITEM_ID,1
```

то на место первого placeholder-а при выполнении фильтра будет подставлено текущее значение атрибута контекста `ITEM_ID`, а на место второго — значение первого позиционного параметра.

Если один и тот же параметр должен быть подставлен на место нескольких placeholder-ов, то в поле *Аргументы* следует повторить соответствующий описатель в нужных позициях. Пример:

```
Запрос:
select passive as item_id from item_link
where active = ? and linktype_id = ?
union
select active as item_id from item_link
where passive = ? and linktype_id = ?

Аргументы:
ITEM_ID,1,ITEM_ID,1
```

Этот фильтр отбирает ближайжих соседей текущего итема по связи указанного типа „в обе стороны".

4.10.3 Perl-фильтры

При проектировании Communiware-приложений встречаются ситуации, когда метод отбора нужных итемов проще всего выразить на используемом в ядре Communiware языке программирования Perl. Предполагается, что читатель данной секции знаком с языком и системой программирования Perl.

Фильтры, написанные на Perl представляют собой часть программного кода Communiware. Создание их через Web-интерфейс невозможно, так как они предоставляют слишком большие возможности доступа к операционной системе сервера. Для установки нового Perl-фильтра требуется иметь доступ к командной строке сервера с правами привелигированного пользователя.

Perl-фильтр представляет собой модуль языка Perl, определяющий объект, обычно наследник от `Communiware::Filter`. Имя модуля должно быть в пространстве имен `Communiware::Filter`, например `Communiware::Filter::Pic`.

При использовании фильтра в шаблоне общаяя часть имени не указывается. Например, на вышеупомянутый фильтр ссылкются как `::Pic`.

Имя фильтра, начинающееся с двух двоеточий служит указанием для интерпретатора шаблонов, что следует искать не унифильтр или же SQL-фильтр, а модуль Perl.

Объект-фильтр должен иметь конструктор `new`, создающий объект, которому в качестве параметров передаются все параметры фильтра, указанные в шаблоне, и поддерживать методы `fetchrow_hashref`, `fetchall_arrayref` и `finish`, аналогичные соответствующим методам хэндлов SQL-операторов DBI (см. также документацию на Perl-модуль DBI).

Кроме того, в модуле должна быть определена функция `check`, **не являющаяся методом объекта**, которая получает те же параметры, что и конструктор `new`, и возвращает любое истиное значение, если параметры корректны, и **`undef`**, если параметры указаны с ошибками.

От методов фильтра не требуется полной функциональности соответствующих методов DBI. Достаточно, если `fetchrow_hashref` будет при каждом вызове возвращать ссылку на хэш с атрибутами очередного элемента списка, причем имена атрибутов должны быть в верхнем регистре, а `fetchall_arrayref` — ссылку на массив ссылок на хэши со всеми еще не извлеченными элементами.

Внутри реализации фильтра можно пользоваться функциями модуля `Communiware::Context` для доступа к атрибутам контекста и вывода информации на генерируемую страницу, а также методами объекта `$Communiware::dbh` для прямого доступа к базе данных.

4.11 Использование фильтров в условиях и выражениях

До сих пор мы упоминали только об одной возможности использования фильтров — для организации циклического выполнения некоторого фрагмента шаблона, т.е. об использовании фильтров в динамических элементах `Loop` и `List`.

Однако фильтры могут применяться не только для того, чтобы получить множество итемов и выполнить нечто с каждым из них, но и для ответа на вопрос: а входит ли данный итем в множество итемов, возвращенных фильтром.

Для этого служит условие типа `IN`, которое может встречаться в *выражениях* Communiware — арифметических и логических. Хотя выражения могут встречаться во многих динамических элементах, наиболее естественно использование условия `IN` в динамических элементах, содержащих условия, таких, как `If` или `Cond`.

Синтаксис условия `IN` следующий:

```
value IN filter_spec
```

Value здесь — некоторое значение, строка или число, filter_spec — спецификация фильтра одном из описанных выше форматов, т.е. это может быть имя хранимого фильтра с параметрами, или литеральный унифильтр, или имя Perl-фильтра и т.д.

Условие является истинным, если среди итемов, возвращенных фильтром есть такой, у которого `ITEM_ID` совпадает с value.

При использовании этого условия в качестве первого условия в динамическом элементе If, левый операнд трактуется как имя атрибута, значение которого надо сравнивать, а не как константа. Во всех остальных случаях оно трактуется как константа.

Примеры использования условия `IN`, взятые из дистрибутивного набора шаблонов Communiware.

1. Проверяется совпадение значения атрибута CHILDRENLINK хотя бы с одним из явно заданных значений. Используется литеральный фильтр.

```
<:If ! CHILDRENLINK IN "(BELONGS,PARENT,THREAD)":>
```

2. Проверяется, может ли итем типа, название которого содержится в атрибуте TYPE_ID быть активным в связи типа `TOPIC`. Используется хранимый SQL-фильтр TypeCanBe.

```
<:If TYPE_ID IN TypeCanBe(ACTIVE,TOPIC):>
```

Следует учесть, что выражение, заданное первым аргументом оператора IN сравнивается со значением атрибута ITEM_ID контекстов, возвращаемых фильтром.

Глава 5

Синтаксис шаблонов

5.1 Понятие динамического элемента

Шаблон Communiware представляет собой HTML-текст, в котором местами размещаются инструкции для подстановки в текст страницы значений из контекста. В отличие от статического HTML-текста, который копируется в выходную страницу без изменений, эти инструкции называются *динамическими элементами*.

Весь текст шаблона за пределами динамических элементов является *статическим*, т.е для подстановки в шаблон любой информации из базы данных или из параметров HTTP-запроса требуется использовать тот или иной динамический элемент.

Динамический элемент выделяется в HTML-тексте шаблона угловыми скобками с двоеточиями, например:

```
<:Attr TITLE:>
```

После открывающей угловой скобки с двоеточием всегда следует имя динамического элемента. В именах динамических элементов (как и в большинстве других синтаксических конструкций Communiware) большие и маленькие буквы считаются различными. То есть, в отличие от HTML-тэгов, имена динамических элементов следует писать именно в том регистре, в котором они написаны в документации.

После имени динамического элемента следуют параметры, разделенные пробелами. Если параметр должен содержать пробелы внутри, например является фрагментом HTML-текста, его необходимо заключить в двойные кавычки.

Например,

```
<:ItemLink ITEM_ID "это ссылка":>
```

представляет собой пример правильного употребления динамического элемента `ItemLink`, а

```
<:ItemLink ITEM_ID это ссылка:>
```

неправильно. В данной конструкции слово `,,ссылка''` будет проинтерпретировано как третий параметр динамического элемента, а не как часть второго.

Никакие другие разделители, такие как скобки, не являются ограничителями параметров. Поэтому конструкция

```
<:Loop {TOPIC <--, TYPE_ID = 'ARTICLE'}:>
```

является неправильной. Спецификация фильтра в динамическом элементе `Loop` должна быть передана как первый параметр. Поэтому, если она содержит пробелы, она должна быть вся заключена в кавычки.

```
<:Loop "{TOPIC <--, TYPE_ID = 'ARTICLE'}":>
```

Одинарные кавычки не являются ограничителем параметра, поэтому их можно спокойно использовать в синтаксисе фильтра.

Если необходимо употребить кавычку как часть текста параметра, перед ней нужно поставить обратную косую черту.

```
<:ItemLink ITEM_ID "<img src=\"@ICON\">":>
```

Иногда у динамических элементов используются так называемые *ключевые параметры* , т.е. параметры, интерпретация которых зависит не от их положения среди других параметров, а от ключевого слова, отделенного от собственно значения знаком равенства.

Например,

```
<:Image src=xxx.gif item=yyy:>
```

Эти параметры подчиняются общим правилам разбора, и в кавычки, в случае наличия пробела должен заключаться весь текст параметра, а не только значение.

Правильной будет запись:

```
<:Image src=icon.gif "alt=Всплывающая подсказка":>
```

а не

```
<:Image src="icon.gif" alt="Всплывающая подсказка":>
```

как в HTML.

5.2 Подстановка значений в параметрах динамического элемента

Во всех параметрах динамических элементов можно использовать подстановку атрибутов контекста. Для этого применяется синтаксис `@ИМЯ`.

При обработке параметров динамического элемента такая конструкция заменяется на значение атрибута `ИМЯ`.

Эта замена производится *до того*, как динамический элемент „увидит" свои параметры, но после того, как синтаксическая конструкция будет разделена на отдельные параметры. Поэтому наличие пробелов в значении подставляемых атрибутов не влияет на колчество параметров динамического элемента.

Для того чтобы употребить знак `@` в параметрах динамического элемента как строковую константу, следует защитить его обратной косой чертой.

Например, конструкция

```
<:Header From info@communiware.ru:>
```

приведет к тому, что в значение заголовка From будет подставлено значение атрибута контекста `communiware` (скорее всего не существующего), а для помещения туда E-Mail адреса `info@communiware.ru` следует воспользоваться конструкцией

```
<:Header From info\@communiware.ru:>
```

Именем атрибута после знака `@` считается последовательность из букв, цифр и подчерков, возможно заканчивающаяся модификатором контекста (см. раздел 3.5.1).

В случае, если возникает неоднозначность, например если непосредственно после подставляемого значения следует буква, можно взять имя атрибута в фигурные скобки, чтобы точно отметить его границы — `@{ATTR}`.

Этот синтаксис следует использовать всегда, когда используется подстановка функций контекста, или групповых атрибутов, так как иначе параметры в круглых скобках не будут восприняты как часть спецификации подставляемого атрибута или функции. Например, правильно использовать следующие конструкции:

```
@{CHILDREN(TOPIC)}
@{TXT(TYPE_ID)}
```

и неправильно

```
@CHILDREN(TOPIC)
@TXT(TYPE_ID)
```

Обычно подстановка атрибутов производится ровно один раз, т.е. наличие в подставляемом тексте знака @ никогда не влияет на результаты подстановки. Особым случаем здесь являются аргументы групповых атрибутов и функций контекста. Если в этих аргументах присутствует символ подстановки, то эта подстановка выполняется и вычисление самой функции производится с учетом подставленного значения.

По особому производится подстановка в случае, если параметр динамического элемента интерпретируется как список пар *имя=значение* через запятую (Такие параметры есть у динамических элементов `ItemLink`, `Script` и `Continuation`).

В этих случаях подстановка производится после того, как значение параметра разбивается на компоненты по запятым (для того чтобы наличие запятых в подставляемом тексте не влияло на разбиение), но если один из компонентов при этом представляет собой *только* спецификацию подстановки, он обрабатывается так же как и параметр в целом.

Это позволяет запоминать целые блоки пар имя=значение в атрибутах контекста, что полезно, если для нескольких динамических элементов следует использовать одинаковые наборы опций.

В некоторых случаях требуется выполнить преобразование подставляемого текста. Например, в некоторых дизайнах заголовки рубрик в главном меню должны быть выведены большими буквами, в то время как храниться в базе и выводиться на других страницах они должны в смешаном регистре.

Для этого существуют *модификаторы подстановки*. Модификатор подстановки это символ, который расположен между символом подстановки `@` и именем атрибута, и инструктирует Communiware выполнить определенные преобразования с подставляемым значением.

Определены следующие модификаторы подстановки:

_ Переводит все символы подставляемого значения в нижний регистр. (мнемоника _ это что-то внизу).

^ Переводит все символы подставляемого значения в верхний регистр. (мнемоника — стрелочка вверх).

& Выполняет HTML-эскейпинг подставляемого значения, заменяя все специальные символы на сответствующие entities. (мнемоника — entity начинаются с символа &). В большинстве случаев Communiware самостоятельно разбирается в каких случаях требуется эскейпинг, а в каких нет, но иногда, например, при конструировании тэгов вручную с помощью динамического элемента `Subst` этот вид модификации может пригодиться.

% Выполняет URL-эскейпинг подставляемого значения. Заменяет все специальные символы на %xx, где xx — шестнадцатиричный код символа. Мнемоника очевидна. Используется в основном при конструировании JavaScript-кода, поскольку динамические элементы `Script` и `ItemLink` выполняют эскейпинг самостоятельно.

***** Преобразует значение типа RICHTEXT в представление „текст с выделениями” (см. раздел **??**).

5.3 Выражения

Выражения это способ сконструировать нужное значение посредством арифметически или строковых операций.

Выражения в Communiware допустимы только в тех динамических элементах, синтаксис которых это явно позволяет. Это `Define`, `Subst`, `Header` и `Use`.

Выражения состоят из операндов (констант), знаков операций и круглых скобок, изменяющих приоритет операций. Каждый из этих элементов выражения должен быть оформлен как отдельный параметр динамического элемента. То есть знаки операций и скобки должны быть отделены от операндов пробелами.

В выражениях Communiware не предусмотрено использование переменных (атрибутов контекста). Поэтому значения атрибутов контекста должны подставляться в операнды выражений с использованием механизма текстовой подстановки, описанного в разделе 5.2.

По этой же причине в выражениях не используется информация о типах атрибутов, использованных в качестве операндов. Тип операнда определяется по его внешнему виду.

В выражениях определены следующие операции:

+ Сложение. Складывать можно два числа или дату с числом. При прибавлении числа к дате число интерпретируется как количество дней. При необходимости прибавить к дате несколько часов или минут выразите этот интервал как долю от суток.

- Вычитание. Определено над числами, датой и числом и двумя датами. Результат вычитания двух дат — число (вещественное) дней.

***** Умножение. Определено только над числами

/ Деление. Определено только над числами. Результат — вещественное число.

div Целочисленное деление. Определено только над числами.

mod Остаток от деления.

. Строковая конкатенация. Определена над любыми аргументами, но они будут проинтерпретированы как строки.

x Строковое повторение. Левый операнд — строка (т.е. любой), правый — целое число. Результат — повторенная соответствующее число раз строка.

5.4 Условия

В отличие от выражений, результатом которых является строка или число или дата, результатом вычисления условия является логическое значение «да» или «нет».

Условия в Communiware во многом похожи на выражения. Точно так же, как и в выражениях, в условиях операции и операнды должны быть отдельными параметрами динамического элемента.

Условие состоит из ряда *элементарных условий*, объединеных логическими операциями *и*(`&&`) и *или*(`||`) . Условие может также сопровождаться логической операцией *отрицания*(`!`). Знак отрицания записывается перед условием, к которому он применяется, как отдельный знак операции (отдельным параметром динамического элемента).

Наиболее высокий приоритет из всех логических операций имеет операция отрицания. Затем идет операция `&&` и последней — `||`.

Для изменения порядка вычисления условий можно использовать круглые скобки, записываемые как отдельные параметры динамического элемента.

В Communiware определены следующие элементарные условия:

конастанта Имеет значение истина, если указанная константа не равна 0 или пустой строке. Поскольку при подстановке несуществующего атрибута результатом является пустая строка, этот тип условия можно использовать для проверки существования атрибута. Кроме того, при подстановке группового атрибута `CHILDREN`, эта операция будет давать значение ложь, если у итема нет потомков с требуемыми свойствами (атрибут `CHILDREN` имеет значение 0). Аналогично ведет себя данное условие при подстановке функции `RIGHTS`.

=,>,<,>=,<=,!= Обычные операции сравнения. Определены над всеми типами операндов. В случае чисел и дат используется естественный для этих типов порядок, в случае строк — лексикографический порядок.

IN Левым аргументом этой операции является константа, правой — спецификация фильтра. Результатом будет *истина*, если один из идентификаторов итема, возвращенных данным фильтром равен указанной константе.

:=: Сравнение множеств. Оба операнда этой операции представляют собой список значений через запятую. Результатом будет *истина*, если эти списки содержат в точности один и тот же набор значений (без учета повторений и пустых элементов.

5.5 Комментарии и пробелы в шаблонах

Шаблон Communiware может содержать HTML-комментарии. Так как комментарии предназначены для человека, редактирующего текст, а не для пользователя, просматривающего текст через браузер, то при обработке шаблона они удаляются, и содержащиеся в них динамические элементы не обрабатываются. Это позволяет временно исключить из обработки большие фрагменты шаблона.

В некоторых случаях (например, при встраивани в шаблон кода баннерной системы) требуется, чтобы в сформированном HTML присутствовали комментарии специального вида. Этого можно добиться, формируя последовательность символов, начинающую HTML-комментарий, с помощью строковых выражений в динамическом элементе `Subst`.

Например:

```
<:Subst "<" . "!-- это комментарий --" . ">":>
```

Во время отладки иногда удобно видеть комментарии в шаблонах. Отключить удаление комментариев можно с помощью *отладочной куки* COMMENTS.

С точки зрения спецификации SGML одним из приложений которого является HTML, несколько подряд идущих пробелов эквивалентно одному, а символ перевода строки эквивалентен пробелу.

Поэтому при обработке шаблонов Communiware выполняет сжатие пробелов.

Исключением являются пробелы, находящиеся внутри HTML-тэгов `script` и `pre`.

5.6 Условное выполнение фрагментов шаблона

Далеко не всегда для создания требуемого внешнего вида страницы достаточно выполнить подстановку тех или иных атрибутов контекста. Достаточно часто приходится выводить или не выводить не или иные фрагменты шаблона в зависимости от выполнения некоторых условий.

Для этой цели в Communiware предусмотрены два динамических элемента `If` и `Cond`. Оба они представляют собой примеры *блоковых* динамических элементов. Блоковый динамический элемент, подобно блоковому SGML-тэгу, имеет открывающую и закрывающую часть, между которыми располагается некоторый фрагмент шаблона. В отличие от тэгов, у которых закрывающая часть имеет строго единообразный синтаксис *</имя тэга>*, в качестве закрывающей части блокового динамического элемента используется отдельное ключевое слово. Это ключевое слово не является „полноценным" динамическим элементом, т.е. рассматривается как синтаксическая ошибка, если встретилось без соответствующего открывающиего динамического элемента.

Параметры закрывающего динамического элемента всегда игнорируются. Их можно использовать для записи комментариев, указывающих к какому из соответствующих открывающих динамических элементов относится соответствующая закрывающая конструкция.

5.6.1 Динамический элемент If

Динамический элемент `If` имеет синтаксис:

```
 <:If условие:>
фрагмент 1
<:Else:>
фрагмент 2
<:EndIf:>
```

В случае если выполняется *условие*, обрабатывается *фрагмент 1*, иначе *фрагмент 2*.

Допустим сокрашенный вариант, в котором отсутствует ключевое слово `<:Else:>` и второй фрагмент шаблона. В этом случае если условие не выполняется, соответствующий фрагмент будет просто проигнорирован.

Условие в динамическом элементе `If` трактуется несколько отличным образом от всех остальных динамических элементов.

Первый аргумент первого условия интерпретируется не как константа, а как имя атрибута контекста. При этом, если этот атрибут имеет перечислимый тип, для которого определен ORDNUM, например `STATUS`, то сравнение на больше-меньше производится в соответствии с этим ORDNUM.

Если условие состоит из имени связи или ключевого слова ACCESS, то это условие интерпретируется как функция контекста `RIGHTS`.

Такая трактовка сохраняется даже если первое условие предваряется знаком отрицания.

Если необходимо использовать в качестве единственого простого условия условие, первый операнд которого — константа, то следует взять это условие в скобки.

Например

```
<:If TYPE_ID IN AllowedTypes:>
```

проверяет наличие значения атрибута `TYPE_ID` текущего контекста в результате фильтра `AllowedTypes`, а

```
<:If ( TYPE_ID IN AllowedAttrs ):>
```

проверяет наличие литеральной строки `TYPE_ID` в результате фильтра `AllowedAttrs`.

5.6.2 Динамический элемент Cond

Динамический элемент `Cond` позволяет организовать более сложное ветвление, чем простое условие да/нет, как в динамическом элементе `If`.

Этот элемент представляет собой последовательность блоков `Case`, каждый из которых содержит условие и фрагмент шаблона. Например:

```
<:Cond:>
<:Case @A = @B:>
Show <:Attr SOMETHING:>
<:EndCase:>
<:Case @A > @B:>
Sbow <:Attr SOMETHING_ELSE:>
<:EndCase:>
<:Case otherwise:>
<!--Show nothing-->
<:EndCase:>
<:EndCond:>
```

Эти блоки проверяются последовательно, и обрабатывается первый блок, условие в котором истино. Если такого блока не нашлось, весь `Cond` заменяется на пустую строку.

Поэтому распространенной практикой является указывать в последнем блоке заведомо истиное условие, например непустую строковую константу `otherwise`.

Внутри динамического элемента `Cond` могут присутствовать только блоки `Case`, пробелы (переводы строки) и SGML-комментарии.

Наличие там любого другого содержимого (за пределами блоков `Case`) считается синтаксической ошибкой.

5.7 Повторное использование шаблонов

Оформление типичного сайта обычно включает в себя многочисленные фрагменты страниц, которые должны выглядеть одинаково если не на всех, то хотя бы на многих страницах данного сайта.

На всех сайтах, использующих одинаковую онтологию, встречаются фрагменты шаблонов, которые отвечают за функциональность и выглядят абсолютно одинаково даже на сайтах с принципиально различным дизайном.

Communiware предоставляет средства для повторного использования фрагментов кода на языке шаблонов, формирующих идентичные или аналогичные фрагменты страниц.

Такие фрагменты должны быть оформлены в виде отдельных шаблонов, которые затем включаются в другой шаблон с помощью динамических элементов `Include` и `Surround`.

5.7.1 Типы шаблонов

Переиспользуемые шаблоны, генерирующие фрагменты страниц, отличаются от шаблонов-представлений итемов по крайней мере тем, что результатом их применения является не целостный HTML-документ, а некоторый произвольный его фрагмент.

Поэтому в Communiware предусмотрено понятие *типа шаблона*. Тип шаблона хранится в расширеном атрибуте`TEMPLATE_TYPE` итема шаблона.

Он может быть одним из:

PAGE (шаблон страницы) Результатом обработки такого шаблона обязательно должен быть целостный HTML-документ (или документ в какой-либо другой DTD).

Такой шаблон можно назначать в качестве представления по умолчанию или ссылаться на него в URL, например с помощью динамического элемента `ItemLink`.

MAIL (шаблон дайджеста) Также раскрывается в целостный HTML-документ. Используется системой дайджестов (см. главу 10). Отличия этих шаблонов от шаблонов страниц подродно описаны в разделе **??**.

ELEMENT (шаблон фрагмента) Раскрывается более-менее произвольную в последовательность HTML-тэгов. Единственное ограничение — все тэги, которые в этом шаблоне открыты, в нем же должны быть и закрыты. Используется в динамических элементах `Include` и `List`.

CONTAINER (шаблон обрамления) Используется в тех случаях когда нужно поместить некое стандартное обрамление вокруг части текущего шаблона.

Таким обрамлением часто бывает, например, стандартная шапка и подвал сайта.

Так как для таких шаблонов также требуется соблюдение условия *„что здесь открыто, то здесь и закрыто*", наличие двух типов шаблона для вставки фрагмента и для обрамления позволяет гарантировать отсутствие незакрытых тэгов в отдаваемой пользователю странице.

5.7.2 Включение фрагментов

Динамический элемент `Include` позволяет включить в текущее место генерируемой страницы шаблон фрагмента.

Первым параметром этого динамического элемента является идентификатор включаемого шаблона. Все остальные параметры добавляются в текущий контекст как аргументы данного шаблона (см. раздел 3.5.2).

Кроме того, фрагменты шаблона можно включать с помощью динамического элемента `List` (см раздел 4.5.1), который выводит каждый элемент списка по указанному именованному шаблону.

5.7.3 Обрамляющие шаблоны

Несколько более сложный случай представляют собой шаблоны-контейнеры.

В этих шаблонах должно быть предусмотрено место для включения обрамляемого содержимого. Это место отмечается с помощью динамического элемента `Content`.

Включаются такие шаблоны с помощью блокового динамического элемента `Surround`. Содержимое блока между `Surround` и `EndSurround` включается на то место, где во включаемом шаблоне обрамления присутствовал динамический элемент `Content`.

5.7.4 Экспорт и импорт шаблонов между сайтами

Как уже упоминалось ранее, шаблоны фрагментов могут быть общие для нескольких сайтов. Поэтому Communiware предусматривает возможность использования на одном сайте шаблонов, принадлежащих другому сайту. Но, поскольку язык шаблонов Communiware предоставляет разработчику развитые средства управления доступом к информации, возможность показать любой итем по любому шаблону представляет собой риск несанкционированного доступа к информации. Особенно это актуально для хостинговых серверов, где человек, имеющий права разработчика на одном из сайтов может не иметь прав доступа к информации на соседнем.

Поэтому предусмотрены средства ограничения возможностей использования шаблонов с чужих сайтов.

Во-первых, шаблон, который требуется использовать на других сайтах, должен быть объявлен *экспортируемым*. (вообще говоря, это распространяется не только на шаблоны. Экспортируемым может быть любой итем). Для этого необходимо установить в 0 значение атрибута

RESTRICT_TEMPLATES.

Во-вторых, на сайте должно быть явно разрешено использование шаблонов с соседнего сайта. Для этого необходимо установить связь USE_TEMPLATES между итемами сайтов. Установить такую связь может только пользователь, обладающий правами на редактирование итема того сайта, на котором будет разрешено использовать чужие шаблоны.

Глава 6

Визуализация данных

6.1 Визуализация атрибутов контекста

Основное назначение системы шаблонов Communiware — вывод данных из базы на web-страницу. Поэтому в языке шаблонов предусмотрен достаточно богатый набор динамических элементов для вывода атрибутов текущего контекста.

6.1.1 Динамический элемент Attr

Любой атрибут контекста может быть выведен на страницу при помощи динамического элемента `<:Attr:>`.

Этот динамический элемент выводит значение атрибута в его "естественном" виде — строки показываются пользователю так, как они хранятся в базе, форматированный текст показывается с той разметкой, которая в него была внесена и так далее.

Динамический элемент `<:Attr:>` позволяет задать способ форматирования для своего содержимого. Для большинтства типов атрибутов используются стандартные способы форматирования, определенные в ANSI стандарте на язык C — для строк и чисел поддерживаются спецификаторы формата функции `printf`, а для дат — функции `strftime`.

Поскольку в стандарте не определен такой тип данных, как форматированный текст, для него нами был разработан собственный набор спецификаторов формата.

Он включает в себя следующие спецификаторы:

p

Удалить все тэги, вызывающие разрывы строк (параграфы, таблицы и так далее). Используется в тех случаях когда нужно встроить текст в строку.

l

Удалить гиперссылки. Обычно используется когда текст выводится как составная часть текста другой гиперссылки, так как HTML не допускает вложеных ссылок.

i

Удалить все картинки. Это делает размер текста более предсказуемым. Этот спецификатор формата заменяет встроенные картинки их альтернативными текстами.

a

Удалить всю разметку. Этот спецификатор формата удаляет картинки совсем. Если вы хотите удалить всю разметку, но сохранить альтернативные тексты на их месте, используйте `ai`.

t

Аналогично `a`, но еще и заменяет entity, представляющие специальные символы, такие как < на сами эти символы. Используется в очень специальных случаях, например при формировании с помощью шаблона не-HTML вывода.

В отличие от спецификаторов формата для других типов, спецификаторы формата для типа `RICHTEXT` не нуждаются в префиксе &. Их можно комбинировать между собой, например с спецификация `li` удалит из текста картинки и гиперссылки.

Кроме того, для данного типа допустима *пустая* спецификация формата, означающая, что текст следует выводить без изменений.

Общий синтаксис динамического элемента Attr таков:

`<:Attr` *имяформат*`:>`

Первым параметром этого динамического элемента является *имя* атрибута, а не его значение, так как для форматирования атрибута необходима информация о его типе.

Если формат не указан, то для атрибутов типа строка используется `%s`, для чисел — `%g`, для дат — `%e %B %Y %H:%M` и для форматированных текстов — `p`.

Для того чтобы вывести форматированный текст со всей разметкой с помощью этого динамического элемента, необходимо указать пустую спецификацию формата явно. Это единственное место в Communiware, где параметр динамического элемента со значением пустая строка не эквивалентен отсутствию параметра.

6.1.2 Динамический элемент Subst

Кроме динамического элемента `<:Attr:>` для вывода значений атрибутов на web-страницу служит динамический элемент `<:Subst:>`.

В отличие от `<:Attr:>`, который формирует некоторое "удобочитаемое" представление значения с учетом типа атрибута, `<:Subst:>` просто подставляет значение в результирующую страницу.

Параметры динамического элемента `<:Subst:>` интерпретируются как выражение (см. раздел 5.3). Это позволяет при необходимости явным образом отформатировать подставляемые значения.

Динамический элемент `<:Subst:>` является практически единственным способом подставить значение внутрь HTML-тэга, так как динамические элементы не могут располагаться внутри тэга, а `<:Subst:>` позволяет сформировать тэг целиком.

6.1.3 Вывод чисел

6.1.4 Вывод основного содержимого

Историческое назначение динамического элемента `<:Text:>` — вывод основного содержимого итема с типом содержимого `Html`. Но при развитии он приобрел много особенностей форматирования, полезных для любого атрибута типа "форматированный текст", поэтому ныне он поддерживает вывод и других атрибутов, и даже позволяет выбрать один из списка.

Простейший способ вывести основное содержимое итема — использовать `<:Text:>` без параметров:

```
<:Text:>
```

При этом основное содержимое итема будет выведено с сохранением всего форматирования, которое в нем есть.

Если Вам требуется вывести, например, часть текста новости в списке новостей, заменив иллюстрации на их альтернативные тексты и создать гиперссылку на полный текст новости (где в шаблоне, скорее всего, содержится вызов `<:Text:>` без параметров), Вы можете указать, сколько строк (очень приблизительно) следует оставить от текста. Ссылка на полный вариант будет проставлена автоматически:

```
<:Text default_lines=4 format=i :>
```

Это приведет к результату примерно следующего вида:

```
Первые четыре строки
содержимого итема
с иллюстрацией
[иллюстрирующая фотография]...>>
```

При этом двойная галочка будет ссылкой на страницу этой новости, отформатированной ее шаблоном по умолчанию. Поскольку разные браузеры форматируют текст по-разному, понятие строки очень условно. Строкой считается 60 "приблизительно символов"(символ по той же причине понятие тоже очень условное).

Умолчания у `<:Text:>` вполне осмысленные, и обычно не приходится указывать много параметров. Но настроек у него много, и в качестве развесистого примера можно привести следующий:

```
<:Text default_lines=4 max_lines=6 format=il
cut_mark=include(link_more) do_link=ifcut glue_link=no
link_text=Подробности... link_template=with_relevant
link_style=class=more attr=ABSTRACT,TEXT :>
```

Этот фрагмент будет выводить аннотацию, если она есть, а если нету — основное содержимое. В тексте все иллюстрации будут заменены на их альтернативные тексты и все ссылки на их тексты. Если результат длиннее шести строк, то будут выведены первые четыре, иначе текст будет выведен целиком. Если текст будет обрезан, то к нему будет приписан результат работы шаблона link_more, а после завершения форматирования обрезанного куска — ссылка с текстом “Подробности” и классом more, указывающая на представление текущего итема по шаблону with_relevant. Если текст не будет обрезан, то ни результата шаблона link_more, ни ссылки не будет.

Параметры `<:Text:>`:

default_lines=*N*

Если текст обрезан, выводятся первые `default_lines` его строк; если этот параметр не определен или равен 0, обрезания не происходит и ссылка на полный текст не выводится.

max_lines=*K*

Если текст состоит менее чем из `max_lines` строк, он будет обрезан; по умолчанию (или при равенстве 0) `max_lines` равно `default_lines`. Указание `max_lines` без указания `default_lines` — синтаксическая ошибка.

format=*формат*

Спецификация форматирования текста. Такая же, как для `<:Attr:>`

cut_mark=*текст_или_шаблон*

Если текст был обрезан, к нему будет добавлено то, что указано в этом параметре (по умолчанию “...”). Если значение имеет вид `include(`*`спецификация_шаблона`*`)`, будет вставлен результат обработки этого шаблона, иначе текст будет вставлен как указан. При сохранении форматирования значение `cut_mark` добавляется

сразу к обрезанному тексту, до закрытия открытых к моменту обрезания тегов.

do_link=(always|ifcut|never)

При указании `do_link=always` (это значение по умолчанию) ссылка "полностью" показывается всегда, независимо от того, был ли обрезан текст. При указании `do_link=ifcut` ссылка будет показана только если текст был обрезан. При указании `do_link=never` ссылка не будет выведена никогда.

glue_link=(yes|no)

При указании `glue_link=yes` (это значение по умолчанию) ссылка будет добавлена к обрезанному тексту сразу после `cut_mark`, при указании `glue_link=no` — после закрытия тегов.

6.2 Формирование гиперссылок

6.2.1 ItemLink

6.2.2 LinkBox

6.2.3 Script

6.3 Формирование inline-изображений

6.3.1 Image

6.3.2 IncludeVirtual

6.4 JavaScript и стили в шаблонах

Lib

6.5 Работа с HTTP-протоколом

Header

Глава 7

Система виртуальных контекстов}

7.1 Стандартные виртуальные контексты, что они должны уметь.

7.2 Система декорирования

7.2.1 Набор стандартных шаблонов оформления

7.2.2 Стилевое оформление

7.2.3 Стандартные JavaScript-функции

Глава 8

Редактирование итемов.

8.1 Контексты постинга. Соотношение между контекстом и формой

Особенности контекста во время визуализации формы постинга. `<:Post:>`,`<:Item:>`

8.2 Фазы обработки постинга. Процедурные вставки на разных фазах

8.3 Ввод и редактирование атрибутов и связей.

EditField

8.3.1 Редактирование форматированного текста

8.3.2 Редактрование дат

8.4 Вычисления времени обработки

`<:Use:>`

8.5 Проверка условий, которым удовлетворяют введенные данные.

`<:Check:>`

8.6 Действия (сохранение, удаление etc).

`<:Input:>` Возможность разных действий с разными контекстами внутри одной формы

8.7 Редирект по завершении обработки.

8.8 История изменений

Глава 9

Формы, не изменяющие итемов.

9.1 Поиск и фильтрация

`<:Input:>`

9.2 настройки

`<:Cookie:>`

9.3 Login

9.4 MailTo

Глава 10

Система дайджестов и правил устаревания

10.1 Схема работы системы дайджестов

Дайджесты в Communiware — это периодические рассылки обновлений материалов по электронной почте. Пользователь может подписаться на тот или иной дайджест сам, а при наличии достаточных полномочий — подписать других пользователей. На момент написания этого документа (Tue Feb 26 23:42:08 MSK 2002) достаточными являются полномочия на редактирование (`WRITES`) итема, на который производится подписка.

Дайджест, получаемый пользователем, определяется следующими параметрами:

предмет подписки

итем, на который подписывается пользователь. Это *не* тот итем, обновления которого отслеживаются, это тот итем, *в контексте которого* производится отбор материалов для рассылки. Например, при подписке на комментарии к статье этим итемом будет эта статья, хотя обновления самой статьи в рассылку не попадут (вернее, не окажутся поводом для рассылки).

вариант подписки

принцип отбора материалов для рассылки. Это итем типа `DIGEST_VARIANT`. Расширенных атрибутов не имеет. Его текст — спецификация фильтра, который будет выполнен в контексте предмета подписки и который должен отобрать материалы, составляю-

щие дайджест. Контекст, в котором выполняется этот фильтр, содержит, помимо ID итема подписки, еще два атрибута типа дата: PERIOD_START и PERIOD_END. Для корректной работы фильтр должен учитывать эти атрибуты и отбирать только те итемы, которые изменились (в смысле, заложенном в фильтр его автором) в период с PERIOD_START до PERIOD_END. Вариант подписки должен быть пассивным в связи REPRESENTABLE с типом предмета подписки. Кроме того, стандартные шаблоны управления подпиской выбирают только те варианты подписки, которые принадлежат текущему сайту. Это позволяет создать для сайта набор подписок, характерных именно для него. В то же время при явном задании варианта подписки можно воспользоваться вариантом с сайта, с которого можно использовать шаблоны. Это позволяет воспользоваться непосредственно вариантом, предоставляемым тем или иным пакетом, а не его копией, и тем самым гарантировать соответствие версии этого варианта версии установленного пакета.

адресат

это зарегистрированный пользователь сервера Communiware, который подписан на некоторый итем по некоторому варианту и шаблону с некоторой периодичностью и форматом. Для того, чтобы получать дайджесты, подписанный пользователь, разумеется, должен сообщить серверу свой email (атрибут EMAIL итема типа AUTHOR). Можно подписывать также группы пользователей. В момент рассылки демон рассылки раскроет группу вместе со всеми подгруппами в список пользователей, email'ы которых известны серверу.

Помимо определяющих параметров, у дайджеста есть еще параметры настройки:

шаблон дайджеста

шаблон, представляющий отобранные для дайджеста материалы. Это итем типа DIGEST_TEMPLATE, в целом совместимого с типом TEMPLATE. Особенности шаблонов дайджестов описаны ниже в разделе 10.3. Шаблон дайджеста должен быть активным в связи DIGEST_PRESENTATION с вариантом подписки, к которому он применяется. В отличие от варианта подписки, шаблоны дайджеста выбираются стандартными шаблонами управления подпиской не только с текущего сайта — использование стандартного представления ничему не противоречит. Если на некотором сайте требуется не предоставлять возможности подписываться по некоторому стандартному (то есть скопированному из установленного пакета) варианту подписки через некоторый стандартный шаблон, достаточно разо-

рвать эту связь, если она скопировалась из пакета при создании сайта вместе с вариантом подписки.

периодичность рассылки

одно из предопределенных значений, определяющих периодичность, с которой будет рассылаться дайджест. Стандартные предопределенные значения включают `HOURLY` (ежечасно), `DAILY` (ежедневно), `WEEKLY` (еженедельно) и `MONTHLY` (ежемесячно). Демон рассылки обрабатывает подписки с интервалом, соответствующим периоду, выставляя атрибут контекста `PERIOD_START` в значение `PERIOD_END` предыдущего запуска, а `PERIOD_END` — в момент времени, на несколько минут (сколько — определяется параметром `GRACE` конфигурации сервера) более ранний, чем момент запуска, чтобы предоставить авторам или редакторам материалов возможность исправить ошибки, замеченные сразу после создания или изменения материала. Поддержка новых значений требует вмешательства администратора сервера, чтобы внести новые значения в базу данных и сконфигурировать соответствующие им запуски демона рассылки.

формат

одно из предопределенных значений, определяющих формат письма, которое получит адресат. Стандартные предопределенные значения — `TEXT/PLAIN` и `TEXT/HTML`. В текущей практике шаблон дайджеста возвращает на выходе тело в формате `TEXT/HTML`, которое преобразуется при необходимости в `TEXT/PLAIN` с помощью программы `w3m`. Таким образом, оба формата генерируются с помощью одного шаблона дайджеста. Поддержка новых значений требует вмешательства разработчиков ядра Commnuniware.

Работа системы дайджестов состоит из двух независимых этапов — управления подписками и рассылки.

На этапе управления подписками, подробно описанном ниже (см. 10.2) пользователь или разработчик сайта настраивает, какие дайджесты, когда и в каком виде этот пользователь (или другие пользователи) будет получать. Доступные пользователю возможности настройки определяются разработчиком сайта. Пользователь может быть подписан на один и тот же итем по одному и тому же варианту только по одной комбинации шаблон-периодичность-формат. Однако это проверяется только на уровне хранения подписок, так что можно быть подписанным на тот же итем по тому же варианту иным способом как члену группы. При разработке был использован принцип менеджер имеет право определять, что и когда рассылать подчиненным, но пользователь лучше знает, в каком формате он умеет читать

письма, поэтому в случае, когда пользователь подписан на один и тот же итем по тем же варианту и шаблону и с той же периодичностью как пользователь и как член группы, приоритет имеет формат из записи, соответствующей пользователю. Также и при смене подписки другому пользователю менеджер может установить любые параметры, но если пользователь уже подписан на этот итем по этому варианту, формат будет взят из имеющейся записи.

На этапе рассылки демон рассылки, запускаемый с той или иной периодичностью, отбирает все подписки, настроенные на эту периодичность, и по каждой из них рассылает дайджесты за соответствующий период. При этом учитывается только наличие подписки на момент запуска демона, то есть если пользователь был подписан с месячной периодичностью за семнадцать секунд до запуска демона, он получит новости за месяц. Соответственно, если он отказался от ежемесячной рассылки за семнадцать секунд до запуска демона, он не получит новости за весь месяц. Демон рассылки учитывает права доступа каждого получателя на чтение той информации, которая попадает в отобранное фильтром варианта подписки множество и предоставляет шаблону дайджеста (см. раздел 10.3) только тот набор, который разрешено читать будущим получателям результата обработки этого шаблона. У разработчика сайта нет способа влиять на работу демона рассылки. Подчеркнем, что в отличие от действия динамического элемента `<:MailTo:>` () отбор информации для рассылки производится с правами *получателя*, а не того, кто проводит настройку.

10.2 Управление подписками

Управление подписками пользователей производится либо разработчиком сайта, либо пользователем согласно настройкам, которые предоставил ему разработчик, но всегда по некоторому активному действию пользователя (сабмиту формы (раздел **??**) или постингу (раздел **??**)) и *с его полномочиями*. Разработчик использует для управления подписками динамический элемент `<:Subscribe:>`.

10.2.1 Динамический элемент `<:Subscribe:>`

Динамический элемент `<:Subscribe:>` используется для непосредственного управления подпиской пользователя или пользователей или для предоставления пользователю возможности управлять ее параметрами. `<:Subscribe:>` допустим только внутри формы, обычной (`<:Form:>`) или

формы постинга (`<:Post:>`).

Синтаксис:

`<:Subscribe` *именованные параметры*`:>`

Допустимые параметры:

item=*ID предмета подписки*

Обязателен. Допустимы подстановки как времени визуализации, так и времени постинга, что позволяет подписывать пользователей на создаваемый итем. Преобразуется в скрытое поле формы, не визуализируется никак.

variant=*ID варианта подписки*

Обязателен. Допустима подстановка только времени визуализации. Преобразуется в скрытое поле формы, не визуализируется никак.

addressee=*спецификация фильтра*

Указывает набор адресатов. Представляет из себя спецификацию фильтра. Допустимы подстановки как времени визуализации, так и времени постинга, но следует помнить, что фильтр выполняется *во время и в контексте постинга*. Отсутствие этого параметра эквивалентно указанию `(@AUTHOR_ID)`.

template=*ID шаблона дайджеста*

Допустима подстановка как времени визуализации, так и времени постинга. Отсутствие этого параметра приводит к созданию меню выбора шаблона дайджеста из набора шаблонов, активных в связи `DIGEST_PRESENTATION` с указанным вариантом подписки. При этом значениями полей являются ID этих шаблонов, а визуализируются их названия (`TITLE`). Если при создании меню оказалось, что указанный вариант подписки представим ровно одним шаблоном, меню не генерируется и заголовок этого шаблона не показывается, однако разделитель меню (параметр `delimiter`) используется так, как если бы меню генерировалось, дабы не нарушать верстку страницы.

period=*значение периодичности*

Допустимый набор значений — предопределенные значения периодичности плюс значение `NEVER`, означающее не посылать и приводящее к прекращению рассылки указанным адресатам дайджеста, определяемого указанными предметом и вариантом подписки. Отсутствие этого параметра приводит к созданию меню выбора периодичности из всех возможных значений периодичности плюс `NEVER`.

format=*формат писем*

Допустимый набор значений — предопределенные значения формата получаемых адресатами писем. Отсутствие этого параметра приво-

дит к созданию меню выбора из этого набора значений.

delimiter=*разделитель меню*

Строка, вставляемая как разделитель между автоматически генерируемыми меню из вышеописанного набора.

Во всех случаях автоматического создания меню если на момент визуализации известно, что адресат один, умолчательные значения заполняются из существующей подписки этого адресата на указанный итем по указанному варианту, если такая существует.

10.2.2 Принудительная подписка

Если в `<:Subscribe:>` указаны все параметры подписки, то в форме не будет задаваться никаких вопросов, а после ее обработки подписка будет настроена так, как указано в этих параметрах. Это позволяет разработчику сайта формировать *принудительные подписки* пользователей на изменения в определенных итемах. Подписать пользователя на некоторую рассылку имеет право сам этот пользователь и пользователь, имеющий право редактирования предмета подписки. Таким образом, разработчик может предоставить менеджеру некоторого проекта возможность принудительно подписывать участников этого проекта на те или иные изменения в проекте, включив в шаблон управления проектом динамический элемент `<:Subscribe:>` с указанием в параметре `addressee` фильтра, определяющего участников проекта. Следует помнить, что этот фильтр выполняется в момент *настройки подписки*, а не в момент рассылки. Поэтому для того, чтобы менеджеру не приходилось переназначать подписки при каждом изменении состава участников проекта, следует завести группу участников проекта и указывать в параметре `addressee` фильтр, определяющий ее. При таком подходе подписана в момент настройки подписки будет группа, а ее состав будет раскрыт демоном рассылки в момент рассылки. В предположении, что группа участников проекта — группа (тип итема `GROUP`, активная в (гипотетической) связи `PARTICIPANT` с проектом, соответствующий фрагмент шаблона, пригодного в том числе и для вновь создаваемых проектов, будет выглядеть примерно так:

```
<:Subscribe item=%ITEM_ID
"addressee={TYPE_ID = 'GROUP', PARTICIPANT ->}":>
```

10.2.3 Стандартный шаблон подписки

Стандартный шаблон подписки формирует меню выбора подписок следующим образом:

Формируется список вариантов подписки, применимых к типу текущего итема (т.е. пассивных в связи REPRESENTABLE с этим типом), принадлежащих текущему сайту.

В каждом элементе этого списка вызывается <:Subscribe:> с параметрами item со значением ID текущего итема, addressee со значением ID пользователя и variant со значением ID варианта подписки, соответствующего текущему элементу списка.

Таким образом, для предоставления на сайте стандартных подписок следует скопировать (вместе со связями) все варианты стандартных подписок с сайта default, а для предоставления нестандартных следует либо отредактировать связи скопированных вариантов подписки (например, привязав их к другим шаблонам), либо создать на этом сайте другие варианты подписки и привязать их к нужным типам и шаблонам.

10.3 Особенности шаблонов дайджестов

Шаблон дайджеста похож на шаблон страницы, но обладает рядом особенностей.

Во-первых, страница, которую генерирует этот шаблон, будет просматриваться пользователем при чтении почты, возможно off-line, поэтому рекомендуется избегать графических элементов оформления, или даже подавлять иллюстрации в текстах, если в шаблон дайджеста включается полный текст итем.

Во-вторых, страница, сгенерированная шаблоном дайджеста, может быть сконвертирована в текст еще на сервере, причем решение об этом принимает пользователь, и у разработчика нет никаких средств запретить это. Поэтому важные гиперссылки должны быть представлены таким образом, чтобы URL была видна в тексте. Т.е. осмыслено применять конструкции вида:

```
<H1><:Attr TITLE:><H1>
<:ItemLink ITEM_ID "<TT>@URL_PREFIX/@ITEM_ID</TT>":>
```

вместо применяемой обычно в шаблонах страниц

```
<H1><:ItemLink ITEM_ID @TITLE:></H1>
```

Во-третьих, это итем отдельного типа DIGEST_TEMPLATE, совместимого по структуре с типом TEMPLATE, но имеющего право участвовать в других связях. Иной у него и набор возможных типов шаблона (TEMPLATE_TYPE), о которых ниже.

В-четвертых, контекст, в котором генерируются дайджесты, содержит ряд дополнительных атрибутов: PERIOD_START и PERIOD_END, которые

задают временной интервал, обновления за который должны посылаться, и DIGEST_TITLE, содержащий заголовок варианта подписки. Кроме того, этот контекст не содержит ряда атрибутов, существующих в контексте HTTP-запроса — в основном тех, которые специфичны именно для протокола HTTP, таких как настроечные куки и информация о браузере, но может не содержаться и информации о получателе (см. подраздел 10.3.1).

В-пятых, в тексте шаблона для формирования основного списка следует использовать фильтр ::Mail, который получает от демона рассылки результаты выполнения фильтра, указанного в тексте варианта подписки, без повторного выполнения этого фильтра. Кроме того, результаты выдаваемые фильтром ::Mail отфильтрованы на предмет итемов, недоступных для чтения.

10.3.1 Шаблоны дайджестов типа MAIL

Это обычные почтовые рассылки Communiware. Они рассчитаны преимущественно на общедоступные сайты. В них генерируется минимальное число писем — для всех подписавшихся на один и тот же итем с одинаковыми параметрами (исключая формат и, естественно, адресата) фильтр, отбирающий итемы для рассылки, выполняется один раз (он, разумеется, должен отключать проверку прав и не должен зависеть от адресата, поскольку в контексте его выполнения всякая информация о пользователе отсутствует). Затем адресаты группируются по признаку одинаковых прав на чтение *результатов работы этого фильтра*, и для каждой такой группы выполняется шаблон, получающий в фильтре ::Mail результаты работы фильтра варианта подписки, отфильтрованные по праву чтения для этой группы. Тем самым для общедоступного сайта, где нет ограничений на чтение материалов, шаблон выполняется единожды и получает в фильтре ::Mail все итемы, отобранные вариантом подписки, а для сайта, на котором есть необщедоступные материалы, это уже не так. Поскольку адресатов у сгенерированного письма много и поскольку права на прочие материалы у членов такой группы могут различаться, шаблон выполняется в анонимном контексте, то есть в нем нет атрибута AUTHOR_ID, и следовательно нет ни информации об адресате, ни возможности извлечь информацию, недоступную анониму (исключая, разумеется, атрибуты предмета подписки и итемов, предоставленных фильтром ::Mail).

Шаблоны дайджестов типа MAIL позволяют минимизировать нагрузку на сервер, поэтому если есть возможность, пользоваться следует именно ими, при необходимости формируя фильтр варианта подписки так, чтобы он извлекал всю необходимую информацию (например, для рассылки дай-

джестов статей — название рубрики, в которую входит данная статья, если эта рубрика не общедоступна).

10.3.2 Шаблоны дайджестов типа `MAILPERS`

Как следует из названия этого типа шаблонов дайджестов, основное их назначение — формирование дайджестов, зависящих от информации об адресате. Они характерны тем, что в контексте выполнения такого шаблона присутствует атрибут `AUTHOR_ID`, позволяющий извлечь информацию об адресате, что шаблон выполняется с правами этого пользователя, и что фильтр варианта подписки также выполняется в присутствии атрибута `AUTHOR_ID` и с правами этого пользователя, что позволяет ему не отключать проверку прав и зависеть от атрибута `AUTHOR_ID`, например, мои задачи. Разумеется, следует разнести ответственность за персонализированное оформление и персонализированный отбор данных, и в будущих версиях Communniware, видимо, так и будет сделано.

Однако за большее удобство приходится платить. При использовании `MAILPERS`-шаблонов для каждого адресата отдельно выполняются фильтр варианта подписки и шаблон, что создает дополнительную нагрузку на сервер. Поэтому пользоваться персонализированными шаблонами без крайней необходимости не следует.

10.4 Правила устаревания

Предметный указатель