

Развесим ярлыки или — Три слоя, слой Второй

Андрей Терешко

Tag — ярлык по русски. Но link (связь) — часто тоже переводят как "ярлык". Поэтому, чтобы не путаться, буду дальше писать — тег. Так принято.

Раньше, во времена оные, web-мастерам приходилось тестировать свои html-страницы, обычно, в двух браузерах — NC и IE. Схватка между двумя "аллигаторами" заставляла web-мастеров лавировать (заниматься "слаломом") в использовании фирменных тегов того или иного браузера. Не использовать новые теги было бы грешно, а использовать — значит обречать на неработоспособность при просмотре страницы в том или ином браузере. Но проблема разрешилась сама собой.

Теперь нет нужды дожидаться, когда производители сделают новые теги совместимыми, да и самих тегов, их новой функциональности не надо ждать. Теги можно делать самим!

Эта технология называется JBP (Java Browser Page) (Copyright). Требуется скачать и установить JBP-plugin на стороне клиента (браузера). Этот плагин и будет обрабатывать любые новые теги. Код для обработки, в виде java классов, передается с сервера в браузер, при обращении к странице, которая включает новые теги. Возможности — неограниченные.

Вы ничего не слышали о Java Browser Page? Хорошо — Вы считаете, что такой технологии еще нет или она не будет эффективна, так как потребуется время для на загрузки java-классов, да запуск JVM в браузере, а не все браузеры "одинаково полезны" - то есть не все поддерживают java?

Окей — разместим JVM на сервере, там же будем производить и обработку новых тегов, а браузеру будем передавать только чистый "стандартный" html код.

Web-мастер такой "подмены" (перенос интерпритации тегов из браузера (клиента) на сервер) даже и не заметит. Ему будет важен результат, который — одинаков.

Разработка html-страницы, таким образом, разбивается на три части:

Web-дизайнер, пользуясь стандартным html-кодом (а другого он не знает и знать не хочет) создает html-страницу — именно такую, какую и будет, в конечном счете, видеть пользователь в окне своего браузера. Web-дизайнер подбирает цвета, расположение текста, картинок, таблиц, шрифты — то есть занимается художественной компоновкой html-страницы, или, иначе говоря, — дизайном.

Web-программист (или "тег-мастер"(Copyright)) изучая html-код, который ему подсунил Web-дизайнер, заменяет стандартные html-теги, собственными тегами, обработка которых порождает заменяемый стандартный html-код. Зачем это надо? Стандартный html-код - статичен. Если это таблица — то в стандартном html-коде уже задано, сколько в ней столбцов и строк — задан и текст. Тег-мастер использует новые теги, чтобы упростить и увеличить функциональность (и динамичность) создания html-страницы.

Если тег-мастеру не хватает тегов, он их заказывает у java-программиста. Поэтому теги и называются — заказные теги (custom tag). Самому тег-мастеру знать java не требуется. Java-программист пишет "обработчики заказных тегов" — вопрошая их в javaBeans. Именно java-программисту требуются знания и стандартного html-кода (стандартных тегов) и знание заказных тегов и, само собой — знание языка java. Бедный java-программист (не шутка).

Этот — серверный вариант обработки заказных тегов — уже(!) существует и называется JSP with Custom Tag (Java Server Page с заказными тегами). Разработат фирмой Sun. Текущая версия — 1.1.

Если тег-мастер "умудрится" так закодировать html-страницу, что ни одного стандартного html-тега на ней не использует, то эта html-страница становится абсолютно независимой страницей — независимой от стандартного кода html. То есть, все теги на ней будут custom. Эта страница станет "вещью в себе".

Почему "вещь в себе"? Дело в том, что может поменяться (даже полностью) как стандартный язык html-кода, так и обработчики (javaBeans) заказных тегов, да и сам

Развесим ярлыки или — Три слоя, слой Второй

язык, на котором написаны эти обработчики может измениться или стать вообще другим. А страница с заказными тегами останется. Да не просто останется, а продолжит свое функционирование.

Языком для описание заказных тегов стал язык XML! Вот где явно видима поступь XML по миру. XML проникая всюду, порождает вавилонское столпотворение языков описания всего, что можно вообще описать. Дашь XML в массы!

Имея страницу с заказным тег-кодом, можно "на выходе", меняя только обработчики, получать различную интерпретацию заказного тег-кода: в виде стандартного html-файла; в виде pdf-файла; в виде doc-файла; в виде MP3 файла (если заказной тег-код описывает (кодирует) музыку, а обработчики таких заказных тегов могут "генерировать" файл в формате MP3). Возможности неисчерпаемые — как "электрон".

Нельзя сказать, что идея трехслойного мира была создана Sun. Это мировоззренческая идея, которая издавна применяется в различных областях. Например в кино и театре.

Первый слой — это то, что получается "на выходе", например html-страница.

Второй слой, зажатый между Первым и Третьим — это стабильный слой, слой с "заказными тегами", вещь в себе, например jsp-страница.

Третий слой — это "обработчики" — те, кто обрабатывает "заказные теги" (слой Второй), порождая "на выходе" слой Первый.

Что есть фильм? Изображение на целулоиде киноплёнки или в магнитных доменах видеокассеты? — это всего лишь выходной формат — слой Первый, видимый. Он создан ("сгенерирован") на основе сценария (или книги) — "вещи в себе", которая может быть написано на языке и в такой форме, которая и слыхом не слыхивала о кино. (Сценарий, обычно, "выдает", что писался для кино, но вот текст книги, старинной книги, например "Война и мир", уж точно о кино ничего не знает и знать не хочет). Вот этот - независимый ни от чего текст — и есть слой Второй.

Обработчики этого Второго слоя могут быть разными. В результате их работы — на выходе - будет либо фильм, либо школьное сочинение, либо опера или балет. "Обработчики" (их код) и образуют Третий слой.

Однако, после выхода фильма, уже сам фильм, его форма, "застывает" — уходя во Второй слой. И при просмотре фильма, его влияние на людей — слой Первый, зависит уже от той жизни, которой живет народ в настоящее время. "Обработчиками" — слой

Третий — становятся реалии этой жизни народа.

Что есть история? — Неизменная(!) часть произошедших событий — слой Второй. Наши "знания о истории"(слой Первый) формируются обработчиками-историками (слой Третий). При замене обработчиков-историков(Третьего слоя) — наши представления об истории, (что же на самом деле произошло?), т.е. слой Первый — также меняются.

Известный в широких кругах, академик Фоменко, попытался, с использованием некоторых математических методов, исследовать разнообразные исторические тексты (слой Первый), представленные в виде летописей и других различных древних текстов. Он попытался реконструировать (воссоздать) историю — слой Второй. При этом, он выдвинул гипотезу, что этот Второй слой довольно статичен и описывает не такой уж большой временной промежуток "настоящей истории". А "растяжка" и "смещение по времени" этой "настоящей истории" — получены в результате замены "обработчиков" — создателей летописей и старинных текстов. Его работы в области истории подобны тому, как по фильму, балету, опере и школьному сочинению (все Первые слои) воссоздать полный текст "Евгения Онегина" (слой Второй).

Что есть живое существо? Набор довольно постоянных генов — Геном — слой Второй. И в зависимости от внешних условий (слой Третий — обработчики) происходит проявление свойств организма — слой Первый. Есть живые существа, которые, в зависимости от природных условий и численности вида, меняют свой пол! Есть такие животные, которые в сухую пору, делящуюся до полугода, превращаются в камень, а в сезон дождей — вновь оживают. Есть животные, могущие задерживать рождения потомства на месяцы и годы, в зависимости от природных условий.

Что есть наш мир? — Совокупность неизменных(!) законов (слой Второй)? С помощью каких "заказных тегов" он описан? Кем? — уже известно. Наше представление о мире (слой Первый) базируется на интерпретации (слой Третий) различных физических явлений, опытов. Глядя на Мир (слой Второй, из "заказных тегов"), изучая его через различные браузеры (теоретические или с помощью молотка-синхрофазотрона), подставляя различные "обработчики" — парадигмы, мы пытаемся познать эти "заказные теги мира".

Хотя слой Второй, из заказных тегов, и неизменен, статичен, — работа обработчиков

Развесим ярлыки или — Три слоя, слой Второй

(Третий слой) этих заказных тегов дает на выходе (Первый слой) разнообразное динамическое содержание.

Да и сами обработчики могут "на выходе" породить код, также состоящий из "заказных тегов"(возможно и другого языка, другого типа), который опять будет передан на обработку обработчикам (возможно совсем другого типа). И так далее, далее, далее ... — главное, чтобы пользователь у браузера не уснул, дожидаясь конца обработки.

В этой статье я привожу примеры Второго слоя — код jsp-страницы. Для того, чтобы увидеть Первый слой, Вам достаточно посмотреть код html в браузере (View Page Source). Третий слой ("обработчики") обычно скрыт в java-jar файлах — библиотеках, содержащих javaBeans и находящихся на сервере.

Такое деление на слои, конечно, относительно. Относительно JSP технологии. Сам же браузер использует другую "тройку" слоев: Первый слой — то, что видит пользователь на мониторе. Второй слой — html код страницы, доставленный с сервера. Третий слой — "обработчики" — находится в "родном" коде браузера. И при одной и той же html-странице, но используя разные "родные браузер-обработчики" (NN или IE), можно получить, и обычно получаем, разные результаты на "выходе" — вплоть до "пустого" экрана при просмотре какой-нибудь html-страницы в каком-нибудь одном из браузеров.

JSP технология, в настоящее время, насколько я в курсе, не может получать на выходе документ(файл) в формате, например, PDF. Она под это "не заточена". Но если очень хочется, имея статичное описание документа в формате jsp с использованием заказных тегов, получать "на выходе" либо html файл, либо PDF файл, то можно самому попытаться это сделать. В Интернете достаточно java парсеров (систакических анализаторов), которые можно, довольно легко, приспособить к "разбору" Вашего jsp файла с заказными тегами. Обработчиками должны быть Ваши javaBeans, использующие библиотеку создания PDF файлов. И, имея парсер (parser), jsp-файл и обработчи(handler), можно, одним взмахом руки, точнее, одной строчкой, получить файл на выходе в требуемом формате:

```
Pdf myPdfDocument = parser.parse("http://myserver/mydocument.jsp", handler);
```

Приведу несколько примеров Второго слоя — то есть, файлов jsp:

Пример от Sun, использования заказного тега (файл, обычно, с расширением jsp):

```
<html>
<%@ taglib uri="/tlds/simpleDB.tld" prefix="q" %>
<q:queryBlock connData="conData1">
  <q:queryStatement>
    SELECT account,balance...
  </q:queryStatement>
  The top 10 accounts and balances are:
  <table>
    <tr><th>ACCOUNT</th><th>BALANCE</th></tr>
    <q:queryCreateRows from="1" to="10">
      <td><q:queryDisplay field="ACCOUNT"/></td>
      <td><q:queryDisplay field="BALANCE"/></td>
    </q:queryCreateRows>
  </table>
</q:queryBlock>
</html>
```

Мои пояснения:

Строка `<%@ taglib uri="/tlds/simpleDB.tld" prefix="q" %>` указывает путь к библиотеки описания заказных тегов (`simpleDB.tld`) и префикс (удобное краткое наименование) для нее.

Заказной тег `<q:queryBlock connData="conData1">` передает своему обработчику, то есть обработчику тега `queryBlock`, параметр со значением (`connData="conData1"`) . `"conData1"` — это наименование базы данных.

Заказной тег `<q:queryStatement>` имеет "тело" тега, которое содержит одну строку — `"SELECT account,balance..."`. Параметров этот тег (`queryStatement`) своему обработчику не передает.

Строка `"The top 10 accounts and balances are:"` будет, обычно, передаваться в браузер без изменения.

Тег `<table>` — не является "заказным тегом" — это стандартный html-тег для описания таблицы. Для него обработчик на сервере не вызывается!

Заказной тег `<q:queryCreateRows from="1" to="10">` передает своему обработчику два параметра (`from` и `to`) и содержит в своем теле два других заказных тега - `queryDisplay`, которые передают своему обработчику параметр `field`.

Развесим ярлыки или — Три слоя, слой Второй

Весь код, заключенный между `<q:queryBlock ...` и `</q:queryBlock>` — это тело заказного тега `queryBlock`.

При обработке этого заказного тега(`queryBlock`) на сервере, будут обработаны, также на сервере, последовательно, все заказные теги, входящие в тело заказного тега `queryBlock`. Результат будет передан в браузер пользователя в виде файла, содержащего только стандартные (а не заказные) `html`-теги. А уже сам браузер обработает эти стандартные теги (в данном случае — `<html>`, `<table>`, `<tr>`, `<th>` и `<td>`).

Обработчик заказного тега может прочитать свое тело и изменить его или пропустить дальнейшую обработку. Например, в данном случае, при обработке заказного тега `queryBlock`, при отсутствии найденных записей, обработчик может пропустить дальнейшую обработку тела заказного тега, чтобы не показывать пустую таблицу, или вывести сообщение о том, что записи не найдены и уже затем пропустить дальнейшую обработку тела заказного тега. Как именно поступит обработчик заказного тега зависит только от кода конкретного `javaBean`, который и производит обработку конкретного заказного тега.

В вышеприведенном примере от Sun, используется смешение заказных тегов и стандартных `html`-тегов. Обычно так и происходит при кодировании `jsp`-страницы. Однако, если есть охота использовать страницы с заказными тегами не только для генерации `html`-страниц, но, например и для генерации отчетов в виде `pdf` файлов и т.п., то "чистота тегов", то есть использование только заказных тегов, желательна. Хотя, конечно, обработчик заказного тега, перед выводом, может читать тело своего заказного тега и "вычищать" его от не нужных в данный момент тегов, но код обработчика тогда будет, возможно, излишне усложнен.

Второй пример, от фирмы Allaire Corporation. Рассылка почты. Адреса берутся из базы:

```
<%@ taglib uri="jruntags" prefix="jrun" %>
<jrun:sql datasrc="source" id="x">
    SELECT Name, Email FROM Customers
</jrun:sql>
<jrun:param id="x" type="allaire.taglib.QueryTable"/>
<jrun:foreach group="<%= x %>">
    <jrun:sendmail host="mail.yourserver.com"
```

```
sender="yourname@yourcompany.com"
recipient="<%= x.get("Email") %>"
subject="The JRun Tag Library!"/>
Dear <%= x.get("Name") %>, The JRun Tag Library easily enables
database-driven e-mailing!
</jrun:sendmail>
</jrun:foreach>
```

Мои пояснения:

Используется библиотека заказных тегов "jruntags" с префиксом "jrun".

Заказной тег `<jrun:sql ...>` передает своему обработчику параметр со значением (`datasrc="source"`). "source" — это наименование базы данных. `id` - это идентификатор запроса, он получает наименование "x". Тег содержит тело, состоящее из одной строки: "SELECT Name, Email FROM Customers" — выборки всех записей (с полями Name и Email) из таблицы Customers. Естественно эту строку тег-мастер может изменять по своему усмотрению, заменяя условия запроса.

Заказной тег `<jrun:param id="x" type="allaire.taglib.QueryTable"/>` обрабатывается своим обработчиком, преобразуя параметр `id` в тип `allaire.taglib.QueryTable`. Обработчик этого тега фактически совершает вот что:

```
allaire.taglib.QueryTable x =( allaire.taglib.QueryTable) PageContext.getAttribute("x");
```

В этой строке, в `getAttribute("x")` упоминается атрибут "x", ранее полученный при обработке заказного тега `<jrun:sql datasrc="source" id="x">`.

Весь код, заключенный между `<jrun:foreach ...` и `</jrun:foreach` — это тело заказного тега `foreach`.

Заказной тег `<jrun:foreach group="<%= x %>">` передает своему обработчику параметр со значением: `group="<%= x %>"`. Обработчик этого заказного тега ("foreach" — для области, для всего радиуса действия) производит циклическую обработку для всех записей из таблицы, значение которой содержится в переменной "x", типа `allaire.taglib.QueryTable`.

Заказной тег `<jrun:sendmail .../>` передает своему обработчику значение параметров `host`, `sender`, `recipient` и `subject`. Часть из этих параметров задается тег-мастером при написании страницы с заказными тегами, а параметр "recipient" вычисляется при выполнении этой страницы: `recipient="<%= x.get("Email") %>"`.

Развесим ярлыки или — Три слоя, слой Второй

Тело заказного тега `<jrun:sendmail .../>` содержит одну строку:

```
Dear <%= x.get("Name") %>, The JRun Tag Library easily enables database-driven e-mailing!
```

Именно эта строка и будет, очевидно, использована обработчиком тега при создании содержимого письма. Кто именно "Дорогой" ("Dear") — определится во время выполнения. То есть, содержимое письма будет персональным для каждого получателя.

Еще некоторые заказные теги от Allaire Corporation, входящие в JRun и представленные фирмой на обсуждение для включения в стандарт будущих версий JSP:

`jrun:sendmsg` — посылка асинхронных сообщений используя JMS.

`jrun:transaction` — выполнение распределенных транзакций с использованием EJB и JTA.

`jrun:servlet` — выполнение сервлета с параметрами.

`jrun:form` — работа с html формами.

`jrun:if` — выполнение блока с условием.

Третий пример использования заказных тегов (почти "Hello World"):

```
<%@ taglib uri="/sky/primaMaster.tld" prefix="studio" %>
<studio:newWord1 id="universe" attempt="1">
  < studio:sky number ="1">
    Голубое
  </ studio:sky >
  < studio:star number ="mach">
    Блестят и сияют
  </ studio:star>
  ...
  < studio:planet name ="the Earth">
    < studio:ocean number = "4"/>
    < studio:continent number ="6"/>
    ...
  </ studio:planet>
  < studio:people number ="2">
    Плодитесь и размножайтесь.
  </ studio:people>
```

```
</studio:newWordl>
```

"Обработчик" этого заказного тега (newWordl) уже отработал, а результат Вы, уважаемый читатель, можете увидеть сами, оглянувшись вокруг.

1. Ресурсы

- <http://java.sun.com/products/jsp/> - JSP от Sun.
- <http://www.allaire.com/> — Jsp с заказными тегами. JRun 3.0 Developer Edition is freely available for download.
- <http://edocs.beasys.com/wlac/portals/docs/tagscontents.html> — заказные теги.
- <http://jakarta.apache.org/taglibs/index.html> — заказные теги.
- <http://jsptags.com/> — заказные теги.
- http://sourceforge.net/project/?group_id=1282 — open source library of custom tags.
- <http://www.caucho.com/> — JSP Resin (free download).
- http://java.sun.com/aboutJava/communityprocess/jsr/jsr_052_jsptaglib.html - Standard Tag Library (В стадии разработки).