Введение в XML

XML – технология, предназначенная для представления текстовых данных в высокоструктурированном виде, пригодном для программной обработки.

XML-документы представляют собой систему элементов, каждый элемент имеет имя и оформляется парой из открывающего и закрывающего тэгов:

```
<имя_элемента>
  . . .                    - тело элемента
</имя_элемента>
```

Тело элемента может включать текст и вложенные элементы. Элемент может содержать атрибуты, которые оформляются в открывающем тэге в виде пар `имя_атрибута = 'значения_атрибута'` (в одинарных или двойных кавычках). Пример элемента с атрибутами:

```
<имя_элемента  атрибут 1 = "значение 1" атрибут 2 = 'значение 2'>
   тело
</имя_элемента>
```

Имя элемента может быть на любом языке. Считается недопустимым отсутствие закрывающего и открывающего тэгов, кавычек у значений атрибутов, наложение элементов. Пример недопустимого описания:

```
<элемент 1>
   <элемент 2>
</элемент1>
   <элемент 2>
```

Элементы с пустым телом допускается оформлять в виде `<элемент/>` это эквивалентно `<элемент></элемент>`.

У документа должен быть только один корневой элемент, все остальные должны быть расположены так или иначе в рамках его тела. Таким образом, любой XML-документ – дерево элементов.

Кроме элементов спецификация XML определяет еще несколько специфичных понятий, которые могут использоваться при формировании XML-документа:

1. ссылки на сущности (entity references);
2. ссылки на символы (character references);
3. комментарии (comments) – предназначены для внедрения в XML-документ информации, которая не считается частью данных, переданных в составе XML-документа; оформляются как `<!-- текст комментария -- >`;
4. секции CDATA (CDATA sections) – используются для оформления фрагментов текста, которые содержат множество символов, запрещенных спецификацией для употребления в литеральной форме в составе текста; ; оформляются как `<![CDATA[текст]]>`;
5. объявление типа документа (document type declaration);
6. инструкции обработки (processing instruction);
7. XML-объявление (XML declaration) – строка вида `<?xml version="версия" encoding="кодировка" ?>`, с которой может начинаться XML-документ. Атрибут `version` обязательный, указывает версию спецификации XML, которой соответствует документ. Сейчас это версии 1.0 и 1.1. Атрибут `encoding` необязательный, указывает кодировку, в которой записан документ (по умолчанию – UTF-8), имена кодировок указываются в соответствии со стандартом IANA;
8. текстовые объявления (text declaration),

Данные составляющие XML-документа совместно с открывающими тэгами (start tags) и закрывающими тэгами (end tags) формируют разметку (mark up), которая позволяет представить текст в высокоструктурированном виде. В формировании разметки также участвуют тэги пустых элементов (empty-elements tags).

Спецификация XML запрещает использовать символы `&`, `<`, `>`, `"`, `'` в их литеральной форме везде, кроме разделителей разметки (ссылки на сущности и символы, а также тэги) или внутри комментариев, инструкций обработки и секции CDATA. Во всех остальных случаях (например, в теле элемента или в значении атрибута) эти символы должны заменяться на встроенные esc-последовательности: `&` – `&` `<` – `<` `>` – `>` `"` – `&quot;` `'` – `&apos;`.

Документ XML составлен из текстовых данных и разметки. Спецификация указывает ряд требований к разметке, не фиксируя при этом четко ни набор элементов, ни их атрибуты, ни правила вложенности элементов. Это сделано для того, чтобы можно было эти ограничения описывать с помощью специальных средств, в зависимости от того, данные какой предметной области должны будут представляться в виде XML в рамках приложения. В силу этого XML можно считать не столько языком, сколько технологией представления текста в структурированном виде.

Ограничения на структуру XML-документов могут задаваться с помощью языка DTD (Document Type Definition) либо с помощью языка XML-схем (XML Schema). Можно сказать, что языки DTD и XML Schema – это мета-языки, позволяющие определить некоторый конкретный XML-язык. В XML-документ можно внедрить ссылку на схему или DTD, которым этот документ должен соответствовать.

XML-документы, удовлетворяющие общим требованиям спецификации XML, называются well-formed. Если документ удовлетворяет также ограничениям, определяемым предметной областью, то он называется valid.

Пространства имен XML

XML-документы могут конструироваться из элементов и атрибутов, определенных в разных схемах или DTD. Так как разработка схем может вестись независимо, при совместном их использовании возникает опасность конфликта имен элементов и атрибутов. Для преодоления этой трудности был придуман механизм пространств имен XML (XML namespaces) – с каждой схемой сопоставляется некоторый глобально-уникальный идентификатор, задаваемый в формате uri. Идея заключается в том, что идентификатор пространства имен совместно с именем элемента или атрибута из этого пространства имен образует глобально-уникальное имя, так как имена элементов и атрибутов в пределах одного пространства имен уникальны.

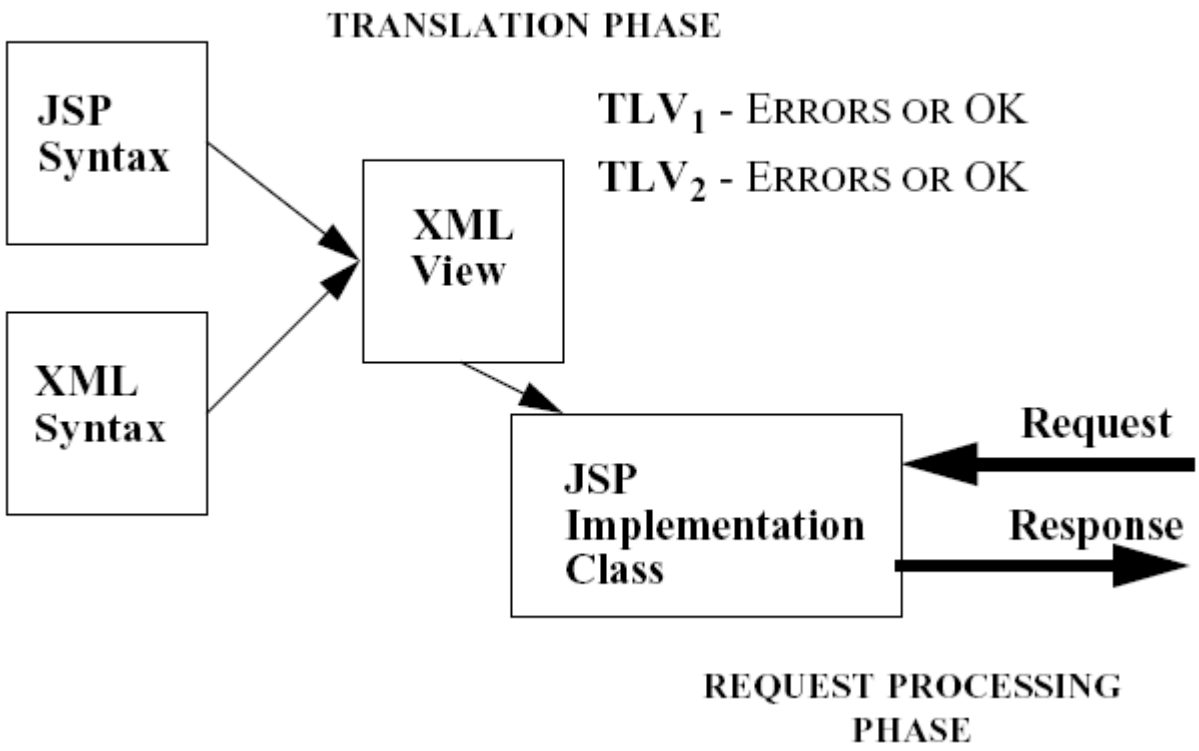
Чтобы избавить от необходимости квалификации каждого элемента и атрибута сравнительно длинным uri, вместо них используют короткий префикс, который вводится в XML-документе с помощью атрибута вида `xmlns:prefix`. Для объявления пространства имен по умолчанию используется атрибут с именем `xmlns`.

Введенные префиксы используются для квалификации элементов и атрибутов в виде `prefix:name`. Элементы без префиксов считаются относящимися к пространству имен по умолчанию. Префиксы пространств имен и пространства имен по умолчанию можно вводить в любом элементе. При этом действие вводимых префиксов и умолчаний распространяется на все вложенные в данный элемент элементы и атрибуты, если только в них не было сделано переопределение.

XML-синтаксис описания JSP-страниц.

Стандартный синтаксис JSP ориентирован на ручную разработку JSP-страниц и появился, когда ни XML-технологий, ни инструментария для работы с JSP не было.

XML-синтаксис ориентирован на программную генерацию исходного кода JSP-страниц, например, по визуальным моделям. Сервер приложений использует XML-представление для всех JSP-страниц на этапе трансляции. При этом JSP-страница, записанная в стандартном синтаксисе, преобразуется в эквивалентное XML-представление (XML View).



В качестве корневого элемента используется элемент `<jsp:root>` следующего формата:

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="версия">
      тело
</jsp:root>
```

Значение атрибута `version` может быть 1.2 или 2.0. Он указывает версию технологии JSP, которую должен поддерживать сервер приложений. Все содержимое JSP-страницы (за исключением необязательного пробельного материала, комментариев и объявления XML) должно располагаться в теле элемента `<jsp:root>`. Таким образом, общий шаблон JSP-страницы в XML-синтаксисе такой:

```
<?xml version="1.0" encoding="кодировка" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="версия"
xmlns:…="…" … >
</jsp:root>
```

Директивы оформляется XML-элементами вида `<jsp:directive.`*имя_директивы*`>`. Атрибуты те же, что и в стандартном синтаксисе. У всех XML-элементов, обозначающих директивы, пустое тело. Примеры:

```
<jsp:directive.include file="menu.html" />
<jsp:directive.page import="java.util.*" />
```

Директива `taglib` в XML-синтаксисе заменена атрибутом вида `xmlns:`*`префикс`*`="`*`uri`*`"` либо `xmlns:`*`префикс`*`="urn:jsptagdir:`*`каталог`*`"`. Первый используется, если в `taglib` используется атрибут `uri`, второй – если используется атрибут `tagdir`. Если в директиве `taglib` используется атрибут `uri`, и он был задан относительным URL, то используется атрибут `xmlns:`*`префикс`*`="urn:jsptld:`*`uri`*`"`. Эти атрибуты используются для корневого элемента `<jsp:root>`.

Пример импорта библиотек действий в стандартном синтаксисе:

```
<%@taglib prefix="jhc" uri="http://jh.com/commons" %>
<%@taglib prefix="mydb" uri="/my/db" %>
<%@taglib prefix="myc" tagdir="/WEB-INF/tags/myc" %>
```

В XML-синтаксисе это будет выглядеть так:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:jhc="http://jh.com/commons" xmlns:mydb="urn:jsptld:/my/db"
xmlns:myc="urn:jsptagdir:/WEB-INF/tags/myc" version="2.0">
      …
</jsp:root>
```

Скриптовые элементы оформляются XML-элементами следующего вида:

```
<jsp:declaration>
      <!-- Текст объявления -->
</jsp:declaration>
<jsp:scriptlet>
      <!-- Текст скриптлета -->
</jsp:scriptlet>
<jsp:expression>
      <!-- Текст выражения -->
</jsp:expression>
```

Когда обычное выражение используется для формирования значения атрибута действия на этапе обработки запроса, то в стандартном синтаксисе это может быть записано так:

```
<префикс:имя_действия имя_атрибута="<%= выражение %>" …>
</префикс:имя_действия>
```

В этом случае в XML-синтаксисе элемент `<jsp:expression>` использовать нельзя, поскольку спецификация XML запрещает указывать элемент в качестве значения атрибута. В XML-синтаксисе этот фрагмент будет записан так:

```
<префикс:имя_действия имя_атрибута="%=выражение%" …>
      …
</префикс:имя_действия>
```

Шаблон оформляется в теле стандартного действия `<jsp:text>`. Спецификация XML запрещает в явном виде использовать символы `<` и `&` как в теле элементов, так и в составе значений атрибутов. Если шаблон содержит множество таких символов (например, при генерации HTML-ответов), то проще использовать секции CDATA в теле действия `<jsp:text>`, например:

```
<jsp:text>
      <![CDATA[
            <html><head></head><body></body></html>
      ]]>
</jsp:text>
```

Действие `<jsp:text>` используется для оформления неструктурированных с точки зрения спецификации XML шаблонов (например, простого текста или HTML). При генерации XML-содержимого можно обойтись и без действия `<jsp:text>`, поскольку любые XML-элементы, которые не являются стандартными для JSP, будут восприняты как шаблон и выведены в ответ как текст.

EL-элементы в XML-синтаксисе изменений не претерпевают. Они используются точно в такой же форме как для формирования значений атрибутов действий, так и для динамической генерации ответа.

Элемент `<jsp:root>` был обязательным корневым элементом в JSP 1.2. В JSP 2.0 в качестве корневого элемента JSP-страницы, записанной в XML-синтаксисе (JSP-документа) может быть любой XML-элемент, лишь бы JSP-документ соответствовал спецификации XML. Например, в качестве корневого можно использовать стандартные JSP-элементы (например, `<jsp:text>`) или любой XML-элемент, который является частью шаблона. При этом связывание префикса с идентификатором пространства имен (например, при подключении библиотеки действий или при использовании нескольких пространств имен в ходе генерации ответов в формате XML) может выполняться не только элементом `<jsp:root>`, но и любым другим XML-элементом, необязательно корневым, лишь бы это удовлетворяло спецификации XML Namespaces.

Динамические части ответа могут генерироваться самыми различными способами, как и в стандартном синтаксисе. Для этого можно использовать скриптовые элементы, EL-

элементы, а также стандартное действие <jsp:element> в сочетании с <jsp:attribute> и <jsp:body>, а также действия, определенные программистом.

```
<jsp:element name="${content.headerName}"
xmlns:jsp="http://java.sun.com/JSP/Page">
      <jsp:attribute name="lang">
            ${content.lang}
      </jsp:attribute>
      <jsp:body>
            ${content.body}
      </jsp:body>
</jsp:element>

<table indent="${table.indent}">
      <row>${table.value}</row>
</table>
```

При генерации ответов в формате XML бывает необходимо иметь возможность выдать как часть ответа XML-объявление и объявление типа документа (DOCTYPE declaration). Это удобно сделать с помощью специального стандартного действия, которое может употребляться только в JSP-документах. Оно называется <jsp:output> и его семантика эквивалентна семантике инструкции output языка XSLT. Атрибуты этого действия:

- omit-xml-declaration – необязательный атрибут, указывающий, следует ли выводить XML-объявление в составе ответа (false) или нет (true);
- doctype-root-element – указывает имя корневого элемента, которое будет использовано при генерации объявления типа документа. Атрибут необязателен, но указать его можно только в сочетании с атрибутом doctype-system;
- doctype-public – указывает публичный идентификатор, который будет использован при генерации объявления типа документа. Публичный идентификатор – это некий широко известный идентификатор, связанный с DTD-описанием некоторого XML-языка, который в данном случае будет использован для генерации ответа. Атрибут необязателен, указывать его можно только в сочетании с атрибутом doctype-system;
- doctype-system – указывает системный идентификатор, который будет использован при генерации объявления типа документа. Системный идентификатор представляет собой URL, по которому может быть получено DTD-описание некоторого XML-языка, который в данном случае будет использован для генерации ответа.

Объявление типа документа генерируется только тогда, когда указан атрибут doctype-system (данный атрибут необязателен). Если этот атрибут указан, то также должен быть задан атрибут doctype-root-element, содержащий имя корневого элемента соответствующего XML-языка.

При использовании действия <jsp:output> требуется обеспечить, чтобы генерируемое XML-объявление шло перед любым другим содержимым, а генерируемое объявление типа документа шло после XML-объявления, но до корневого элемента. Атрибут omit-xml-declaration по умолчанию имеет значение true, если корневым элементом JSP-документа является <jsp:root>, и false в противном случае. (Такое поведение логично, поскольку в JSP 2.0 элемент <jsp:root> используется в JSP-документах, генерирующих не XML-ответы, а текст или HTML, где XML-объявление не нужно. Если же JSP-документ генерирует XML-ответ, то в качестве корневого используется, как правило, не <jsp:root>, а тот XML-элемент, который будет корневым в генерируемом XML-ответе, и в этом случае XML-объявление желательно.) Следует отметить, что указанное для JSP-документа XML-объявление не будет выведено в генерируемый ответ.

Пример.

Это JSP-страница.

```
<html xmlns:jsp="http://java.sun.com/JSP/Page">
<jsp:output doctype-root-element="html" doctype-public="-//W3C//DTD XHTML
Basic 1.0//EN" doctype-system="http://www.w3.org/TR/xhtml-basic/xhtml-
basic10.dtd" />
      <body>
            <h1>Example XHTML document</h1>
      </body>
</html>
```

Генерируется следующий ответ.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"  SYSTEM
"http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html>
      <body>
            <h1>Example XHTML document</h1>
      </body>
</html>
```

JSP-документы, в отличие от обычных JSP-страниц, имеют расширение не `jsp`, а `jspx`. По умолчанию сервер приложений считает все файлы с расширением `jspx` JSP-документами, а файлы с расширением `jsp` либо JSP-документами, либо обычными JSP-страницами, в зависимости от того, используется ли элемент `<jsp:root>` как корневой или нет (это сделано для совместимости с JSP 1.2).

И стандартные, и определяемые программистом действия оформляются одинаковым образом как в стандартном, так и в XML-синтаксисе. При этом в XML-синтаксисе следует обратить внимание на то, что выражения вида `%=выражение%` могут использоваться только для формирования значений атрибутов действий, а не значений атрибутов произвольных XML-элементов, то есть, например, `<a href="%=url%">Click me</a>` – неверно и должно быть выражено так:

```
<jsp:text><![CDATA[<a href =”]]</jsp:text>
<jsp:expression>url</jsp:expression>
<jsp:text><!CDATA[">Click me</a>]]></jsp:text>
Если версия JSP 1.2 или ниже, иначе
<a href="${url}">Click me</a>
```

Если JSP-страница представлена в XML-синтаксисе, она может содержать ссылку на формальное описание ее структуры, выраженное в виде DTD, либо в виде XML-схемы. Спецификация JSP 2.0 требует выполнять проверку соответствия JSP-документа DTD еще до трансляции. Проверка соответствия схеме необязательна.

Действия jsp:element, jsp:attribute и jsp:body.

Эти действия используются для генерации HTML- или XML-элементов (которые являются частью генерируемого ответа) в условиях, когда имена этих элементов, имена и значения их атрибутов и содержимое тела заранее неизвестны и определяются динамически на этапе обработки запроса выражениями или EL-элементами. Обобщенный пример использования этих действий выглядит так:

```
<jsp:element name="…">
      <jsp:attribute name="…">
      </jsp:attribute>
      <jsp:attribute name="…">
      </jsp:attribute>
      …
      <jsp:body>
      </jsp:body>
</jsp:element>
```

Вложенные действия <jsp:attribute> и <jsp:body> необязательны. Их наличие определяется тем, есть ли у генерируемого элемента атрибуты и тело. Вложенных действий <jsp:attribute> может быть несколько, <jsp:body> может быть не более одного.

Обязательные атрибуты name действий <jsp:element> и <jsp:attribute> позволяют указать имена генерируемого элемента и его атрибута, их значение может формироваться скриптовыми выражениями или EL-элементами.

Значение атрибута генерируемого элемента указывается в теле действия <jsp:attribute> (при этом можно использовать выражения, EL-элементы, вложенные действия и т.д.). При этом необязательный атрибут trim указывает, сохранять (false) или нет (true, по умолчанию) пробельный материал (пробелы, табуляции, символы перевода строки и возврата каретки) в начале и конце значения атрибута. Удаление пробелов выполняется на этапе трансляции.

Действия <jsp:attribute> и <jsp:body> можно использовать не только в теле действия <jsp:element>, но и в теле действий, определенных программистом, для динамического формирования значений их атрибутов и тела. То же самое относится и к стандартным действиям. При этом существуют определенные естественные ограничения:

1) если действие, значение атрибута которого формируется действием <jsp:attribute>, не позволяет указать для него выражения или EL-элементы, а тело действия <jsp:attribute> его содержит, то происходит ошибка трансляции;
2) если в теле действия содержится хотя бы одно действие <jsp:attribute>, а действия <jsp:body> нет, то у соответствующего действия не будет тела;
3) действие <jsp:body> нельзя использовать в теле действия, не имеющего тела по определению, а также в теле следующих стандартных действий: <jsp:body>, <jsp:attribute>, <jsp:scriptlet>, <jsp:expression>, <jsp:declaration>;
4) действие <jsp:attribute> нельзя использовать для формирования значений атрибутов действий <jsp:element> и <jsp:attribute>.

XML-представление JSP-страницы (XML View).

Для любого исходного JSP-файла, будь то JSP-страница или JSP-документ, на этапе трансляции формируется XML-представление, на основе которого и выполняется генерация PIC. Преобразование JSP-страниц в XML-представление ведется так:

1) обработать все директивы <%@include%>, включив соответствующее содержимое в состав исходной JSP-страницы;
2) включить элемент <jsp:root> как корневой элемент с соответствующим атрибутом xmlns:jsp и преобразовать все директивы <%@taglib%> в соответствующие атрибуты xmlns: элемента <jsp:root>;
3) преобразовать все объявления, скриптлеты и выражения в соответствующие им XML-элементы;
4) преобразовать все выражения, используемые для формирования значений атрибутов действий в соответствии с синтаксисом XML;
5) создать элементы <jsp:text> для всех фрагментов шаблона;
6) добавить к каждому XML-элементу в XML-представлении атрибут jsp:id, который облегчает поиск возникающих при трансляции ошибок.

Преобразование JSP-документов в JSP-представление ведется так:

1) обработать все директивы <%@include%>, включив соответствующее содержимое в состав исходного JSP-документа;
2) если элемент <jsp:root> отсутствует, то включить его как корневой элемент с атрибутом xmlns:jsp и преобразовать все директивы <%@taglib%> в его атрибуты xmlns:;

3) установить значение атрибута `pageEncoding` директивы `page` в значение UTF-8. В случае отсутствия директивы `page` с атрибутом `pageEncoding` она добавляется;
4) установить значение атрибута `contentType` директивы `page` в то значение, которое будет использовано сервером приложений при генерации ответа (см. вопросы локализации). В случае отсутствия директивы `page` с атрибутом `contentType` она добавляется;
5) добавить к каждому элементу атрибут `jsp:id`.

Пример JSP-документа

Страница для вывода списка клиентов в виде XML-файла:

```
<?xml version="1.0" encoding="UTF-8"?>
<clientella                                  xmlns="http://pulsar.com/clientella"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:c="http://java.sun.com/jsp/jstl/core" version="2.0">
    <!-- to change the content type or response encoding change the following
line -->
    <jsp:directive.page                    contentType="text/html;charset=UTF-8"
import="jdbcsample.dao.ClientDAO"/>
    <c:set var="clients" value="%= new ClientDAO().findAllClients() %"/>
    <c:forEach var="client" items="${clients}" varStatus="loop">
        <client id="${client.id}"
                name="${client.lastName} ${client.firstName}">
            <place>${client.place}</place>
            <email>${client.email}</email>
        </client>
    </c:forEach>
</clientella>
```

Результат:

```
<?xml version="1.0" encoding="UTF-8" ?>
<clientella xmlns="http://pulsar.com/clientella" version="2.0">
    <client name="Petrov Ivan" id="1">
        <place>Vladimir</place>
        <email>petrov@mail.ru</email>
    </client>
    <client name="Ivanov Sergei" id="2">
        <place>Moscow</place>
        <email>ivanov@mail.ru</email>
    </client>
</clientella>
```