JSTL

JSP-страницы, включающие скриптлеты, элементы action (стандартные действия) и пользовательские теги, не могут быть технологичными без использования JSTL (JSP Standard Tag Library). Создание страниц с применением JSTL позволяет упростить разработку и отказаться от вживления Java-кода в JSP. Как было показано ранее, страницы со скриптлетами трудно читаемы, что вызывает проблемы как у программиста, так и у веб-дизайнера, не обладающего глубокими познаниями в Java.

Библиотеку JSTL версии 1.1.2 (**jstl-1.1.2.jar** и **standard-1.1.2.jar**) или более позднюю версию можно загрузить с сайта apache.org. Библиотеки следует разместить в каталоге **/lib** проекта. При указании значения параметра **xmlns** элемента **root** (для JSP-страницы значение параметра **taglib uri=""**) необходимо быть внимательным, так как если адрес будет указан неправильно, JSP-страница не сможет иметь доступ к тегам JSTL. Проверить правильность значение параметра **uri** (оно же справедливо и для параметра **xmlns**) можно в файле подключаемой библиотеки (например **c.tld**). Простейшая JSP с применением тега JSTL, выводящим в браузер приветствие будет выглядеть следующим образом:

```
<!--пример # 3  : правильный jsp-документ: simple.jspx -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
            xmlns:c=http://java.sun.com/jsp/jstl/core
            version="2.0">
<jsp:directive.page contentType=
            "text/html; charset=Utf-8"/>
<html><body>
<c:out value="Welcome to JSTL"/>
</body></html>
</jsp:root>
```

Тег **<c:out/>** отправляет значение параметра **value** в поток **JspWriter**.

JSTL предоставляет следующие возможности:

- поддержку Expression Language, что позволяет разработчику писать простые выражения внутри атрибутов тега и предоставляет “прозрачный” доступ к переменным в различных областях видимости страницы;
- организацию условных переходов и циклов, основанную на тегах, а не на скриптовом языке;
- простое формирование доступа (URL) к различным ресурсам;
- простую интернационализацию JSP;
- взаимодействие с базами данных;
- обработку XML, а также форматирование и разбор строк.

Expression Language

В JSTL вводится понятие Expression Language (EL). EL используется для упрощения доступа к данным, хранящимся в различных областях видимости (**page**, **request**, **application**) и вычисления простых выражений.

EL вызывается при помощи конструкции **“${**имя**}”**.

Начиная с версии спецификации JSP 2.0 / JSTL 1.1, EL является частью JSP и поддерживается без всяких сторонних библиотек. С версии web-app 2.4 атрибут

isELIgnored по умолчанию имеет значение **true**. В более ранних версиях необходимо указывать его в директиве **page** со значение **true**.

EL-идентификатор ссылается на переменную, возвращаемую вызовом **PageContext.findAttribute(**имя**)**. В общем случае переменная может быть сохранена в любой области видимости: **page**(**PageContext**), **request**(**HttpServletRequest**), **session** (**HttpSession**), **application** (**ServletContext**). В случае если переменная не найдена, возвращается **null**. Также возможен доступ к параметрам запроса через предопределённый объект **paramValues** и к заголовкам запроса через **requestHeaders**.

Данные приложения, как правило, состоят из объектов, соответствующих спецификации JavaBeans, или представляют собой коллекции, такие как **List**, **Map**, **Array** и др. EL предоставляет доступ к этим объектам при помощи операторов **".**" и **"[]"**. Применение этих операторов зависит от типа объекта. Например:

```
<c:out value="${student.name}"/>
```

```
<!--пример # 4 : правильный jsp-документ : simple2.jspx -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          xmlns:c="http://java.sun.com/jsp/jstl/core"
          version="2.0">
<jsp:directive.page contentType=
                              "text/html; charset=Utf-8"/>
<html>
<head><title>Простое использование EL</title></head>
<body>
<c:set var="login" value="Бендер" scope="page"/>
<c:out value="${login} in Rio"/>
<br/>
<c:out value="Бендер в байтовом виде: ${login.bytes} "/>
</body></html>
</jsp:root>
```

С помощью оператора **".**" можно вызывать некоторые методы класса, к которому принадлежит объект. Вызов **login.bytes** в переводе на обычную Java означает **login.getBytes()**.

В результате запуска этого документа в браузер будет выведено:

Бендер in Rio
Бендер в байтовом виде: [B@edf730

Операторы в EL поддерживают наиболее часто используемые манипуляции данными.

Типы операторов:

Стандартные операторы отношения:
== (или **eq**), **!=** (или **neq**), **<** (или **lt**), **>** (или **gt**), **<=** (или **le**), **>=** (или **ge**).
Арифметические операторы: **+**, **-**, *****, **/** (или **div**), **%** (или **mod**).
Логические операторы: **&&** (или **and**), **||** (или **or**), **!** (или **not**).
Оператор **empty** – используется для проверки переменной на **null**, или "пустое значение". Термин "пустое значение" зависит от типа проверяемого объекта. Например, нулевая длина для строки или нулевой размер для коллекции.

Например:

```
<c:if test="${not empty user and user.name neq 'guest'}>
  User is Customer.
</c:if>
```

Автоматическое приведение типов

Данные не всегда имеют тот же тип, который ожидается в EL-операторе. EL использует набор правил для автоматического приведения типов. Например, если оператор ожидает параметр типа **Integer**, то значение идентификатора будет приведено к типу **Integer** (если это возможно).

Неявные объекты

JSP-страница всегда имеет доступ ко многим функциональным возможностям сервлета, создаваемым Web-контейнером по умолчанию. Неявный объект:

- **request** – представляет запрос клиента. Обычно объект является экземпляром класса, реализующего интерфейс **javax.servlet.http.HttpServletRequest**. Для протокола, отличного от HTTP, это будет объект реализации интерфейса **javax.servlet.ServletRequest**. Область видимости в пределах страницы.
- **response** – представляет ответ клиенту. Обычно объект является экземпляром класса, реализующего интерфейс **javax.servlet.http.HttpServletResponse**. Для протокола, отличного от HTTP, это будет объект реализации интерфейса **javax.servlet.ServletResponse**. Область видимости в пределах страницы.
- **pageContext** – определяет контекст JSP-страницы и предоставляет доступ к неявным объектам. Объект класса **javax.servlet.jsp.PageContext**. Область видимости в пределах страницы.
- **session** – создается контейнером в соответствии с протоколом HTTP и является экземпляром класса **javax.servlet.http.HttpSession**, предоставляет информацию о сессии клиента, если такая была создана. Область видимости в пределах сессии.
- **application** – контейнер, в котором исполняется JSP-страница, является экземпляром класса **javax.servlet.ServletContext**. Область видимости в пределах приложения.
- **out** – содержит выходной поток сервлета. Информация, посылаемая в этот поток, передается клиенту. Объект является экземпляром класса **javax.servlet.jsp.JspWriter**. Область видимости в пределах страницы.
- **config** – содержит параметры конфигурации сервлета и является экземпляром класса **javax.servlet.ServletConfig**. Область видимости в пределах страницы.