```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      web-jsptaglibrary_2_1.xsd">
```

Зарегистрировать адрес URI библиотеки пользовательских тегов **mytag-lib.tld** для приложения можно двумя способами:

1. Указать доступ к ней в файле **web.xml**, для чего следует указать после **<welcome-file-list\>**:

```
<jsp-config>
   <taglib>
      <taglib-uri>/WEB-INF/mytaglib.tld</taglib-uri>
      <taglib-location>/WEB-INF/mytaglib.tld
      </taglib-location>
   </taglib>
</jsp-config>
```

2. Прописать URI библиотеки в файле-описании (**.tld**) библиотеки и поместить этот файл в папку **/WEB-INF** проекта. В таком случае в файле **web.xml** ничего прописывать не требуется. Преимуществом данного способа является то, что так можно использовать библиотеку во многих приложениях под одним и тем же адресом URI. Естественно, в этом случае TLD-файл должен размещаться не в локальной папке проекта, а, например, в сети Интернет как независимый файл.

Непосредственное использование в странице JSP созданного и зарегистрированного простейшего тега выглядит следующим образом:

```
<!-- пример # 2 : вызов простого тега : demotag1.jsp -->
<HTML><HEAD>
<%@ taglib uri="/WEB-INF/mytaglib.tld"
             prefix="mytag" %>
      </HEAD>
            <BODY>
                  <mytag:getinfo/>
            </BODY>
</HTML>
```

В результате выполнения тега клиент в браузере получит следующую информацию:

```
Size = (3)
```

Тег с атрибутами

Тег может содержать параметры и передавать их значения для обработки в соответствующий ему класс. Для этого при описании тега в файле ***.tld** используются атрибуты, которые должны объявляться внутри элемента **tag** с помощью элемента **attribute**. Внутри элемента **attribute** между тегами **<attribute>** и **</attribute>** могут находиться следующие элементы:

- **name** – имя атрибута (обязательный элемент);
- **required** – указывает на то, всегда ли должен присутствовать данный атрибут при использовании тега, который принимает значение **true** или **false** (обязательный элемент);

- **rtexprvalue** – показывает, может ли значение атрибута быть JSP-выражением вида **${expr}** или **<%=expr%>** (значение **true**) или оно должно задаваться строкой данных (значение **false**). По умолчанию устанавливается **false**, поэтому этот элемент обычно опускается, если не требуется задавать значения атрибутов во время запроса (необязательный элемент).

Соответственно для каждого из атрибутов тега класс, его реализующий, должен содержать метод **set**ИмяАтрибута**()**.

В следующем примере рассматривается простейший тег с атрибутом **firstname**, который выводит пользователю сообщение:

// пример # 3 : тег с атрибутом : HelloTag.java

```
package test.mytag;
import javax.servlet.jsp.tagext.TagSupport;
import java.io.IOException;

public class HelloTag extends TagSupport {
    private String firstname;

     public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
     public int doStartTag() {
        try {
        pageContext.getOut().write("Hello, " + firstname);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return SKIP_BODY;
    }
}
```

В файл **mytaglib.tld** должна быть помещена следующая информация о теге:

```
<tag>
  <name>hello</name>
  <tag-class>test.mytag.HelloTag</tag-class>
  <body-content>empty</body-content>
  <attribute>
      <name>firstname</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
```

Использовать созданный тег в файле **demotag2.jsp** можно следующим образом:

пример # 4 : вызов тега с передачей ему значения : demotag2.jsp

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core"
                  prefix="c" %>
```

```
<%@ taglib uri="/WEB-INF/mytaglib.tld" prefix="mytag"%>
<%@ page
            language="java"
            contentType="text/html; charset=CP1251"
            pageEncoding="CP1251"
      %>
<HTML><HEAD>
      <TITLE>demotag2.jsp</TITLE>
      </HEAD>
            <BODY>
      <c:set var="login" value="Bender"/>
      <mytag:hello firstname="${login}" />
            </BODY>
</HTML>
```

При обращении по адресу:

http://localhost:8080/FirstProject/demotag2.jsp

в браузер будет выведено:

Hello, Бендер

Тег с телом

Как и в обычных тегах, между открывающим и закрывающим пользовательскими тегами может находиться тело тега, или **body**. Пользовательские теги могут использовать содержимое элемента **body-content**. На данный момент поддерживаются следующие значения для **body-content**:

- **empty** – пустое тело;
- **jsp** – тело состоит из всего того, что может находиться в JSP-файле. Используется для расширения функциональности JSP-страницы;
- **tagdependent** – тело интерпретируется классом, реализующим данный тег. Используется в очень частных случаях.

Когда разрабатывается пользовательский тег с телом, то лучше наследовать класс тега от класса **BodyTagSupport**, реализующего в свою очередь интерфейс **BodyTag**. Кроме методов класса **TagSupport** (суперкласс для **BodyTagSupport**), он имеет методы, среди которых следует выделить:

void doInitBody() – вызывается один раз перед первой обработкой тела, после вызова метода **doStartTag()** и перед вызовом **doAfterBody()**;

int doAfterBody() – вызывается после каждой обработки тела. Если вернуть в нем константу **EVAL_BODY_AGAIN**, то **doAfterBody()** будет вызван еще раз. Если **SKIP_BODY**, то обработка тела будет завершена;

int doEndTag() – вызывается один раз, когда отработаны все остальные методы.

Для того чтобы тело было обработано, метод **doStartTag()** должен вернуть **EVAL_BODY_INCLUDE** или **EVAL_BODY_BUFFERED**; если будет возвращено **SKIP_BODY**, то метод **doInitBody()** не вызывается.

В следующем примере рассматривается класс обработки тега, который получает значения атрибута **num** (в данном случае методом установки значения для