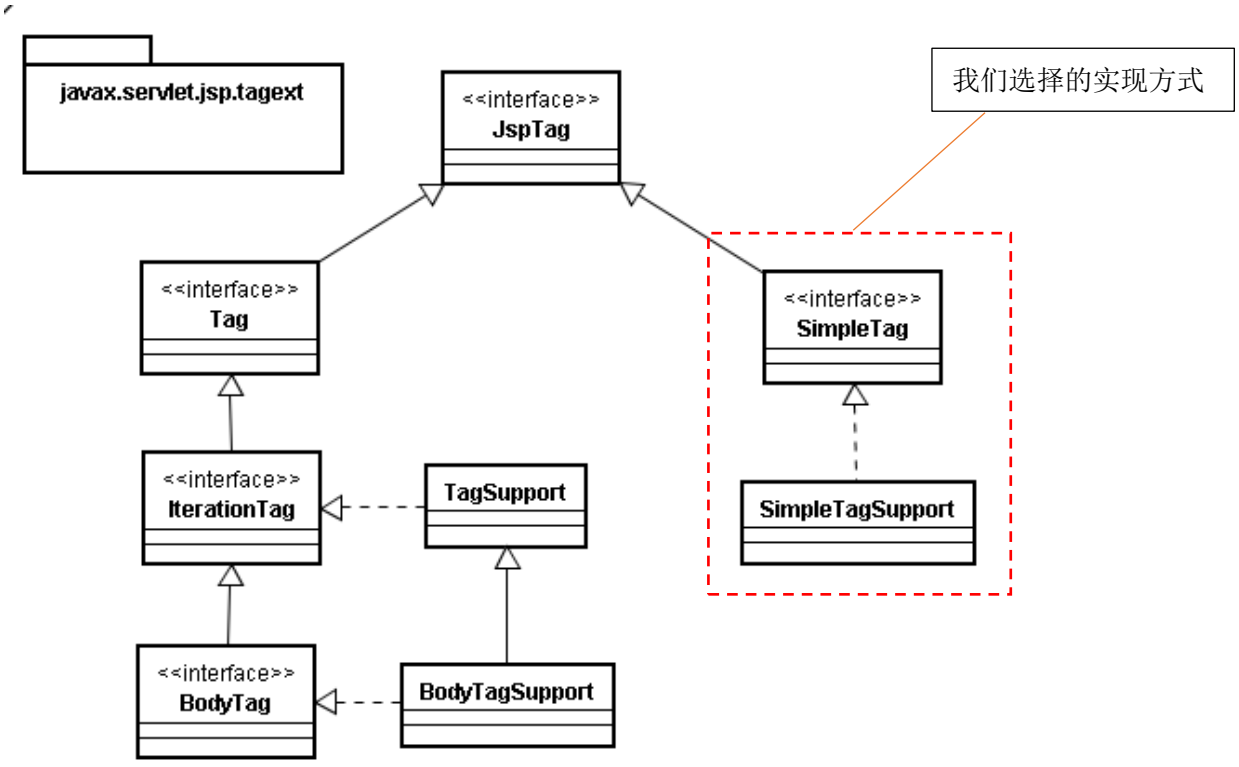
自定义标签库（Tag library）

简介

- JSP 标签库技术可以让我们定制自己的标签。
- 我们前边讲解了 JSP 动作标签，动作标签本质上就是一段 Java 代码，在 JSP 页面被转换为 Servlet 期间，JSP 引擎解析到 JSP 文件就会将动作标签转换为我们预先定义好的 Java 代码。
- 同样，自定义标签实际上一个实现了特定接口的 Java 类，封装了一些常用功能。在运行时，标签将被响应的 Java 代码所代替。多个标签就构成了标签库。
- 简单来说，标签库就是让我们以标签的形式在 JSP 中调用 Java 程序。
- 完成一个自定义标签，需要两个步骤：
  - 1.编写标签处理器类（Tag Handle Class）
  - 2.编写标签库描述文件（Tag Library Descriptor）

相关接口

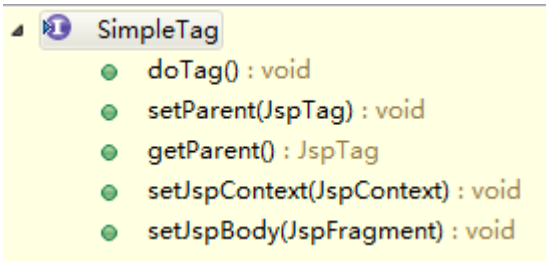


- 在自定义标签的 API 中，最大的接口为 JspTag 接口，该接口是所有标签中最大的一个接口，所有的标签类都是该接口的后代。
- 由上图可以看出，自定义标签的 API 一共有两个分支：
  - 第一个分支是 Tag，该接口比较古老，一般我们不会通过该分支实现自定义标签。
  - 第二个分支是 SimpleTag，该接口实现自定义标签比较简单，一般我们都是通过实现该接口来完成自定义标签的处理器类。

通过 SimpleTag 接口实现

1.编写标签处理器类（Tag Handle Class）

- 上文我们说到，标签库就是让我们以标签的形式在 JSP 页面中调用 Java 程序，既然是 Java 程序那我们就先来编写一个标签处理器类。
- 首先我们需要编写一个类来实现 SimpleTag 接口，实现之前先来看一下这个接口。



  - SimpleTag 接口中共有 5 个抽象方法
    1. doTag() 标签执行时被调用的方法，我们主要编写的方法。
    2. setParent(JspTag) 设置父标签的方法
    3. getParent() 获取父标签的方法
    4. setJspContext(JspContext) 设置 pageContext 的方法
    5. setJspBody(JspFragment) 设置标签体的方法
- 实现 SimpleTag

```
public class MyTag implements SimpleTag {
    @Override
    public void doTag() throws JspException, IOException {
        System.out.println("Hello World!!!");
    }

    @Override
    public void setParent(JspTag parent) {}
    @Override
    public JspTag getParent() {return null;}
    @Override
    public void setJspContext(JspContext pc) {}
    @Override
```

```
    public void setJspBody(JspFragment jspBody) {}
}
```

2.编写标签库描述文件（Tag Library Descriptor）

- 编写完标签处理器类，还需要在项目中对该类进行注册，才可以在 JSP 中使用我们刚刚编写的标签。
- 每一个标签库描述文件对应一个标签库。
- 标签处理器类实际上就是一个 XML 文件，这个 XML 文件有一点特殊，它的扩展名是以 tld 结尾的，一般我们会放在 WEB-INF 文件夹下。
- 步骤：

1. 在 WEB-INF 下创建一个 xml 文件命名为 taglib.tld

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xsd"
        version="2.1">

        <description>Atguigu 1.0 library</description>
        <display-name>Atguigu</display-name>
        <tlib-version>1.0</tlib-version>
        <short-name>a</short-name>
        <uri>http://www.atguigu.com/tag/atguigu</uri>

        <tag>
            <name>hello</name>
            <tag-class>com.atguigu.web.tag.MyTag</tag-class>
            <body-content>empty</body-content>
        </tag>
</taglib>
```

2. taglib.tld 结构

3. 这里的 tld 文件只是给出了一个基本的结构，有一部分内容由于尚未使用，所以在 tld 文件中没有体现出来。

3.在 JSP 中使用标签

- 编写完处理器类和 tld 文件后，标签就可以在我们的项目中使用了。
- 使用步骤：
  1. 在 JSP 页面中引入标签库
     - 格式：<%@ taglib uri=" " prefix="" %>
       - 使用 taglib 指令引入一个标签库<%@ taglib %>
       - uri 属性需要标签库的唯一约束，也就是我们标签库的 uri 属性
       - prefix 属性配置的是标签库在页面中的前缀，一般和 short-name 一致
     - 例如:我们要在页面中引入我们刚刚创建好的标签

```
<%@taglib uri="http://www.atguigu.com/tag/atguigu" prefix="a"%>
```

  2. 在 JSP 页面中使用标签库
     - 标签库引入后就可以像是使用 JSP 动作标签一样使用我们的自定义标签。
     - 标签格式：
       - 有标签体：<库名:标签名></库名:标签名>
       - 自结束标签：<库名:标签名 />
     - 我们刚刚定义的标签没有标签体，可以直接使用，如下：

```
<a:hello />
```

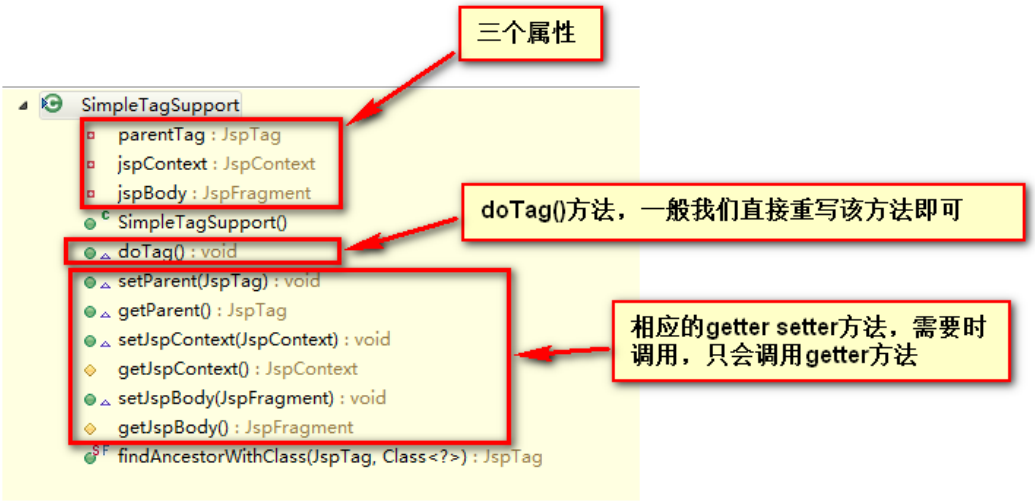
- 这样在我们每次访问这个页面时，控制台中就会输出一个 HelloWorld。

4.运行流程

1. 容器创建处理器类实例后，调用 setJspContext()方法，设置 JspContext。
2. 如果当前标签有父标签则调用 setParent()方法来设置父标签。
3. 如果标签有属性，调用 setXxx()方法设置属性。
4. 如果存在标签体，调用 setJspBody()，设置标签体
5. 调用 doTag()方法，完成标签的主要逻辑

通过 SimpleTagSupport 类实现

- 通过实践发现，直接通过 SimpleTag 接口来编写标签处理器类是有一些麻烦的，因为 SimpleTag 接口中有很多方法需要我们来实现，但是现实使用中，我们并不是所有方法都使用。
- 基于这个原因，我们还可以通过另一种方式来实现标签处理器类，就是 SimpleTagSupport，这个类实现了 SimpleTag 接口，并且接口中的方法都已经被该类实现了，所以我们直接通过继承该类就可以更加方便的实现一个标签处理器类。
- SimpleTagSupport



- 通过继承 SimpleTagSupport 实现标签处理器类，只需要重写 doTag()方法即可。

```
public class MyTag2 extends SimpleTagSupport {
    @Override
    public void doTag() throws JspException, IOException {
```

```
        JspWriter out = getJspContext().getOut();
        out.print("<h1>你好啊，我是自定义标签</h1>");
    }
}
```

- tld 文件

```
<tag>
    <name>tag2</name>
    <tag-class>com.atguigu.web.tag.MyTag2</tag-class>
    <body-content>empty</body-content>
</tag>
```


通过标签实现 if 功能

- 功能要求：
  - 标签形式

    <a:if test=””>
        标签体
    </a:if>
    - 标签名：if
    - 参数：test，接收一个 boolean 值
    - 标签体：网页代码或 EL 表达式
  - 逻辑
    - 如果 test 的值为 true，则显示标签体的内容
    - 如果 test 的值为 false，则不显示标签体的内容
- 实现：
  - 处理器类

```
public class IfTag extends SimpleTagSupport {
    private boolean test;
    public void setTest(boolean test) {
        this.test = test;
    }
    @Override
    public void doTag() throws JspException, IOException {
        if(test){
            getJspBody().invoke(null);
        }
    }
}
```

  - tld 文件

```
<tag>
    <name>if</name>
```

```
    <tag-class>com.atguigu.web.tag.IfTag</tag-class>
    <body-content>tagdependent</body-content>
    <attribute>
        <name>test</name>
        <required>true</required>
        <rtexprvalue>true</rtexprvalue>
    </attribute>
</tag>
```