# 第4章　JSP基础

# 主要内容

1. JSP简介
2. JSP基本语法
3. JSP内置对象

# 4.1 JSP简介

## ❖JSP简介

JSP是Java Server Page的缩写，它是对Servlet的扩展，其目的是简化建立和管理动态网站的工作。

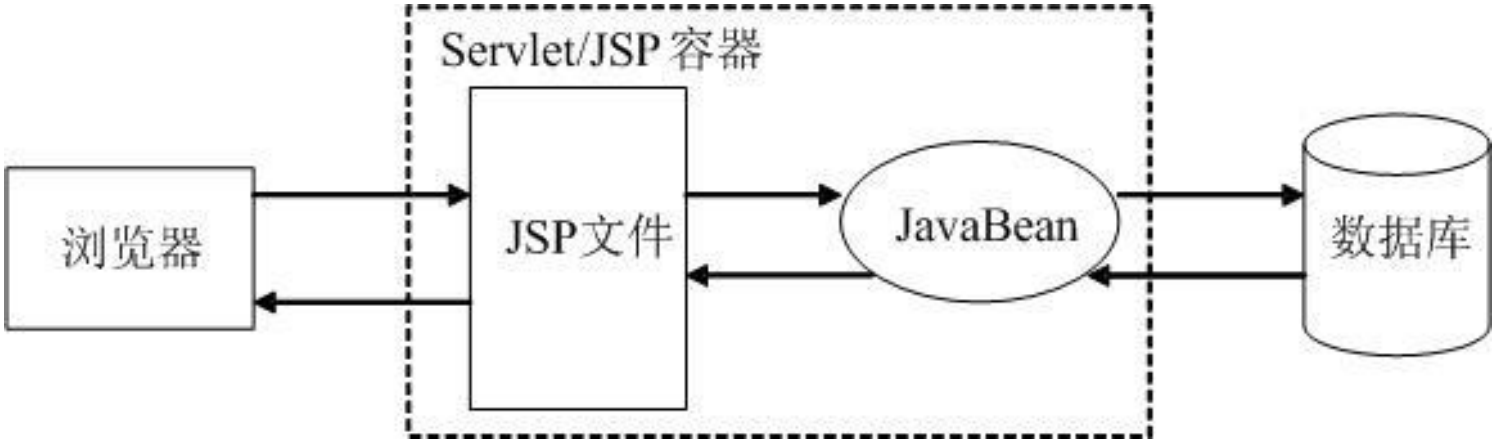
和ASP类似，它们都是在普通的网页文件中嵌入脚本代码来产生动态内容，在JSP文件中嵌入的是Java代码和JSP标记。

# 4.1 JSP简介

## ❖JSP优点

JSP的优点是它能把HTML编码和业务逻辑有效的分离。通常，JSP负责生成动态的HTML页面，业务逻辑由其它可重用的组件（如Servlet或JavaBean）和Java程序来实现，JSP通过Java程序片段访问这些业务组件。

# 4.1 JSP简介




JSP访问服务器端可重用组件

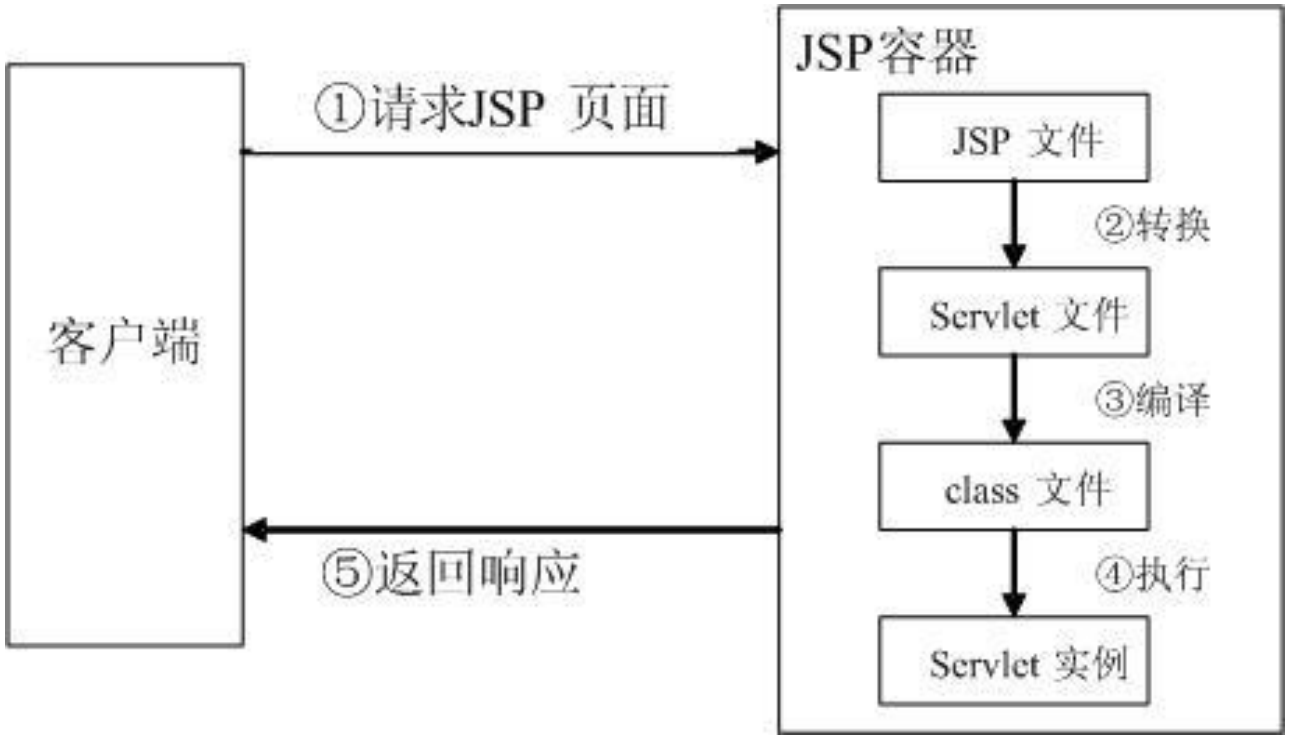
# 4.1 JSP简介

## ❖JSP文件执行过程

当服务器收到Web客户端对一个JSP文件的请求时，

1. JSP容器首先检查JSP页面的语法是否正确。
2. 将它转换成Servlet源文件并对其进行编译。
3. Servlet容器加载转换后的Servlet类，然后实例化一个该类的对象来处理客户端的请求。
4. 请求处理后，响应对象被JSP容器接收，将HTML格式的响应信息发送到客户端。

# 4.1 JSP简介




JSP文件的执行过程

# 主要内容

1 JSP简介

2 JSP基本语法

3 JSP内置对象

# 4.2 JSP基本语法

## ❖JSP构成

一个JSP页面由**元素**和**模板数据**组成。元素是必须由JSP容器处理的部分，而模板数据会被JSP容器忽略（如JSP页面中的HTML代码），这些内容将会被直接发送到客户端。

在JSP 规范中，元素有三种类型：**指令元素**、**脚本元素**和**动作元素**。

# 4.2 JSP基本语法

## ❖指令元素

JSP指令（Directive）用来设置和整个JSP页面相关的属性，如网页的编码方式和脚本语言等，它不会产生任何输出到当前的输出流中。

常用的三种指令为page、include和taglib，其语法形式：

<%@ 指令名 属性=“值” %>

## page指令

page指令作用于整个JSP页面，定义了许多与页面相关的属性。该指令的语法形式为：

<%@ page 属性1=“值1”属性2=“值2” .....%>

# 4.2 JSP基本语法

| 属性 | 描述 |
| --- | --- |
| language=”scriptLanguage” | 指定脚本元素中使用的脚本语言。 |
| import=”importList” | 指定导入的Java包名或类名列表，该列表用逗号分隔。在JSP文件中，可以多次使用该指令来导入不同的包。 |
| session=”true\|false” | 指定在JSP页面中是否可以使用session对象，默认值是true |
| errorPage=”error_url” | 指定当JSP页面发生异常时，将转向哪一个错误处理页面 |
| isErrorPage=”true\|false” | 指定当前的JSP页面是否是处理异常的网页，默认值是false |
| contentType=”ctinfo” | 指定用于响应的JSP页面的MIME类型和字符编码 |
| isELIgnored=”true\|false” | 用于定义在JSP页面中是否执行或忽略EL表达式 |
| pageEncoding=”peinfo” | 指定JSP页面使用的字符编码。如果设置了这个属性，则JSP页面的字符编码使用该属性指定的字符集。如果没有设置这个属性，则使用contentType属性指定的字符集。如果这两个属性都没有设定，则使用字符集“ISO-8859-1”。 |

## include指令

include指令用于在JSP页面中包含其它文件，该文件可以是JSP页面、HTML文件或文本文件。

使用了include指令的JSP页面在转换时，JSP容器会在其中插入所包含文件的文本或代码，因此这些代码也会被执行。

include指令的语法形式为：

<%@ include file=”relativeURL” %>

## taglib指令

taglib指令允许页面使用用户定制的标签，其语法形式为：

<%@ taglib (uri=”tagLibURI” | tagdir=”tagDir”)  prefix=”tagPrefix” %>

- uri属性用来指定标签库的标识符，可以是绝对的或者相对的URI。
- tagdir属性则使用安装在/WEB-INF/tags目录或其子目录下的标签文件。
- prefix表示在JSP网页中引用这个标签库中的标签时的前缀。

# 4.2 JSP基本语法

## ❖脚本元素

脚本元素包括三个部分：

**声明**

**脚本段**

**表达式**

## 声明

脚本元素中的“声明”用于声明在JSP页面的脚本语言中使用的变量和方法，它必须是完整的声明语句，遵照Java语言的语法。

声明不会在当前的输出流中产生任何输出，它以<%!开始，以%>结束，其语法形式如下：

<%! declarations %>

# 4.2 JSP基本语法

可以在一个声明中声明多个变量和方法，也可以使用多个声明来完成。下面是两个使用声明的例子：

1) <%! int i; %>

2) <%! public String f(int i) {

if(i<3) return “i<3”; else return “i>=3”; }

%>

## 脚本段

脚本段是在请求处理期间要执行的Java代码段。脚本段可以产生输出，并将输出发送到客户端，也可以是一些流程控制语句。脚本段以<%开始，以%>结束，它的语法形式如下：

<% scriptlet %>

## 表达式

脚本元素中的“表达式”是Java语言中完整的表达式。在请求处理时，这些表达式会被计算，计算的结果将被转换为字符串插入到当前的输出流中。表达式以<%=开始，以%>结束，它的语法形式如下：

```
<%= expression %>
```

# 4.2 JSP基本语法

## ❖JSP注释

在JSP页面中，可以使用两种类型的注释：

一种是HTML注释，这种注释可以在客户端看到；

另一种是为JSP页面本身所做的注释，通常是给程序员看的，称为JSP注释。

## HTML注释

HTML注释的语法形式如下：

<!-- comments -->

在HTML注释中，可以包含动态的内容，这些动态的内容将被JSP容器处理，然后将处理的结果作为注释的一部分。

```
<!-- 这是注释部分 -->
<!--1+1 = <%= 1+1%> -->
```

## JSP注释

JSP注释的语法形式如下：

<%-- comments --%>

这种注释对开发人员是非常有用的，可以在JSP页面中对代码的功能做注释。另外，在脚本段中也可以使用Java语言本身的注释机制，例如：

<% //comments %>

<% /*comments*/ %>

# 主要内容

1 JSP简介

2 JSP基本语法

3 JSP内置对象

# 4.3 JSP内置对象

## ❖内置对象简介

为了简化JSP页面的开发，在JSP规范中定义了一些内置对象，这些对象主要用于向客户端产生输出、进行会话跟踪和获得请求参数等。

内置对象不需要由JSP页面的编写者实例化，它们由JSP容器实现和管理。

# 4.3 JSP内置对象

## JSP中的内置对象

| 内置对象 | 类型 |
| --- | --- |
| request | javax.servlet.HttpServletRequest |
| response | javax.servlet.HttpServletResponse |
| pageContext | javax.servlet.jsp.PageContext |
| application | javax.servlet.ServletContext |
| out | javax.servlet.jsp.JspWriter |
| config | javax.servlet.ServletConfig |
| page | java.lang.Object |
| session | javax.servlet.http.HttpSession |
| exception | java.lang.Throwable |

# 4.3 JSP内置对象

## ❖request对象

request对象代表请求对象，它对应于HttpServletRequest的子类。

当客户端请求一个JSP页面时，JSP容器会将客户端的请求信息包装在这个request对象中，包括编码方式、请求方式、请求参数的名称和参数值等信息。

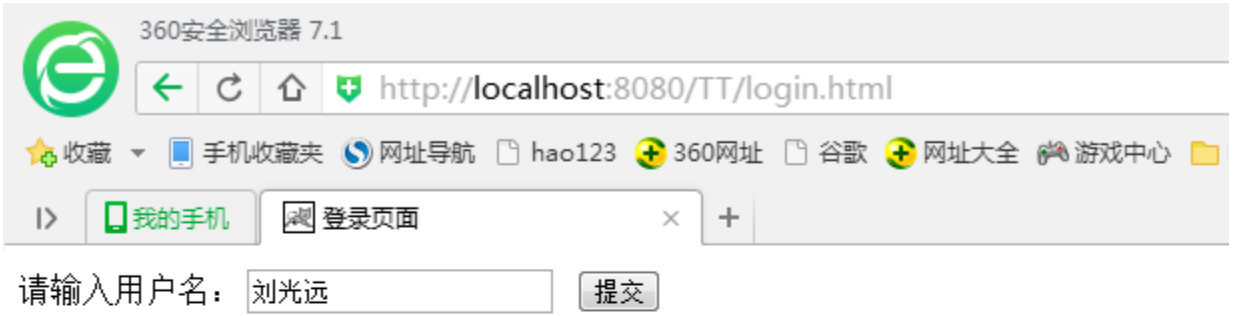
# 4.3 JSP内置对象

【例4-1】request对象的使用。

登录页面login.html

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>登录页面</title>
   </head>
   <body>
      <form action="request.jsp" method="post">
            请输入用户名：<input type="text" name="name"/>
            <input type="submit" value="提交"/>
      </form>
   </body>
</html>
```

# 运行效果

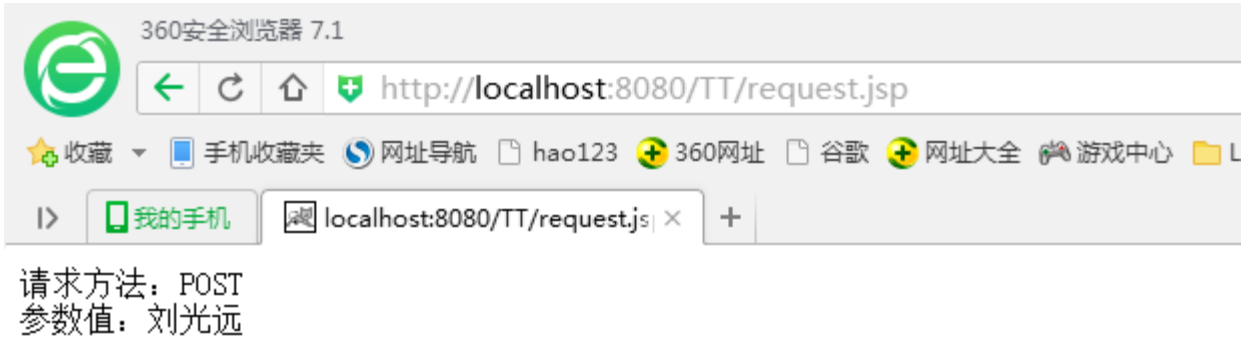

360安全浏览器 7.1
http://localhost:8080/TT/login.html
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
网址大全
游戏中心
我的手机
登录页面
请输入用户名：
刘光远
提交

# 4.3 JSP内置对象

请求参数信息显示页面request.jsp

```
<%@ page language="java" pageEncoding="UTF-8"%>
<html>
<body>
    <%request.setCharacterEncoding("UTF-8"); %>
    请求方法：<%= request.getMethod() %><br />
    参数值：<%= request.getParameter("name") %><br />
</body>
</html>
```

# ❖运行效果


360安全浏览器 7.1
http://localhost:8080/TT/request.jsp
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
网址大全
游戏中心
我的手机
localhost:8080/TT/request.js
请求方法：POST
参数值：刘光远

# 4.3 JSP内置对象

## ❖response对象

response对象代表响应对象，对应于HttpServletResponse的子类。

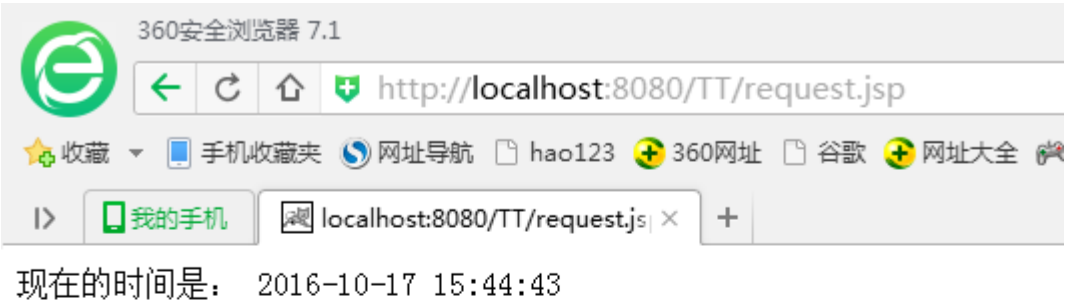
和request对象一样，response对象由JSP容器生成，可以通过它来设置HTTP状态代码、响应头，也可以把页面请求重定向到另一个页面。

# 4.3 JSP内置对象

【例4-2】response对象的使用。

```
<%@ page language="java" pageEncoding="UTF-8"%>
<html>
<body>
    <% response.setHeader("Refresh","5"); %>
    现在的时间是：
    <%= java.text.DateFormat.getDateTimeInstance().format(
        new java.util.Date())%>
</body>
</html>
```

## ❖运行效果


360安全浏览器 7.1
http://localhost:8080/TT/request.jsp
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
网址大全
我的手机
localhost:8080/TT/request.js
现在的时间是： 2016-10-17 15:44:43

# 4.3 JSP内置对象

## ❖session对象

session对象表示当前用户的会话信息，对应于HttpSession的子类。

在session对象中保存的信息，可以在同一会话中的所有页面中访问到。

# 4.3 JSP内置对象

【例4-3】session对象的使用。

first.jsp用于保存信息到session对象中

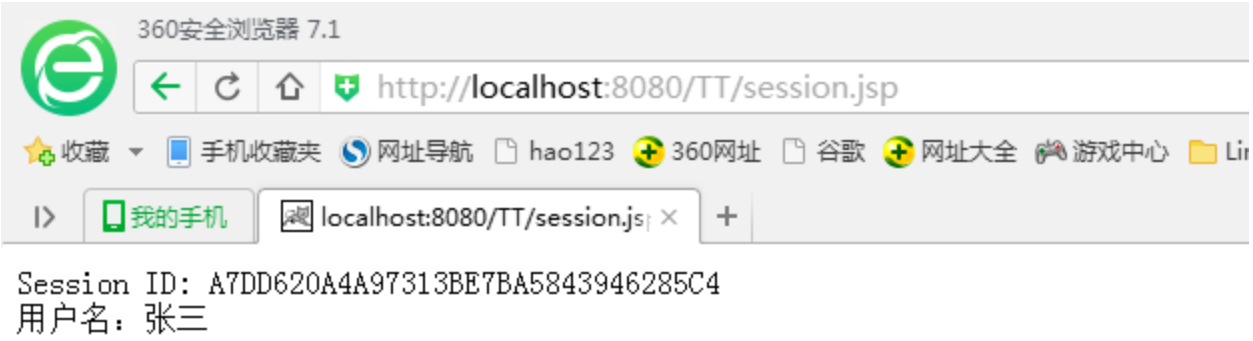
```
<%@ page language="java" pageEncoding="UTF-8"%>
<html>
<body>
<%
        session.setAttribute("username","张三");
        response.sendRedirect("session.jsp");
%>
</body>
</html>
```

# 4.3 JSP内置对象

session.jsp用于显示session对象中保存的信息

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
<body>
        Session ID: <%= session.getId()%><br/>
        用户名：<%= session.getAttribute("username")%>
</body>
</html>
```

## ❖运行效果


360安全浏览器 7.1
http://localhost:8080/TT/session.jsp
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
网址大全
游戏中心
我的手机
localhost:8080/TT/session.js
Session ID: A7DD620A4A97313BE7BA5843946285C4
用户名：张三

# 4.3 JSP内置对象

## ❖application对象

application对象实现了ServletContext接口，它代表Web应用本身，整个Web应用共享同一个application对象。可以使用application对象的setAttribute()方法将一个值放入某个属性，该属性的值对整个Web应用有效，因此它的每个JSP页面（或Servlet）都可以通过getAttribute()方法访问这个属性。

# 4.3 JSP内置对象

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
   <body>
  <%
            Integer count;
            synchronized (application) {
              count = (Integer) application.getAttribute("count");
              if (count == null)
                        count = new Integer(0);
              count = new Integer(count.intValue() + 1);
              application.setAttribute("count", count);
            }
  %>
        你是本网站的第<%= count.intValue()%>个访问者！
 </body>
</html>
```

## ❖运行效果


360安全浏览器 7.1
http://localhost:8080/TT/first.jsp
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
网址大全
我的手机
localhost:8080/TT/first.jsp
你是本网站的第1个访问者!

# 4.3 JSP内置对象

## ❖out对象

out对象的类型是javax.servlet.jsp.JspWriter，该类是从java.io.Writer类派生的，以字符流的形式向客户端输出数据。

out对象可以看成是带有缓冲区的PrintWriter对象，缓冲区的大小默认是8kb，通过page指令的buffer属性能够对这个值进行调整。

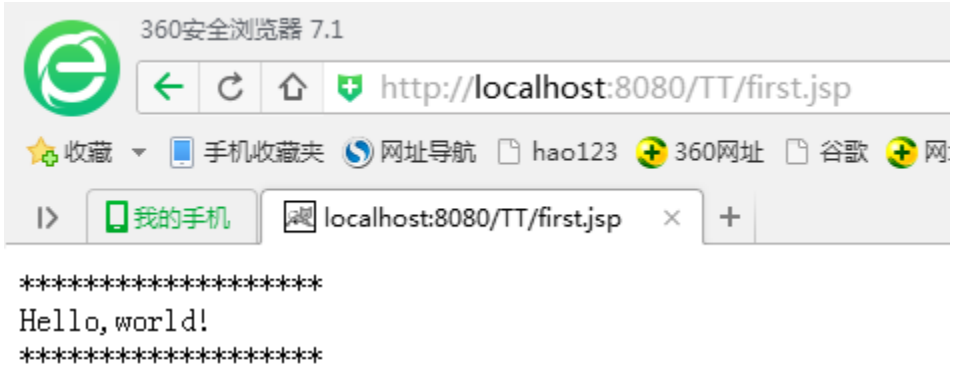
## out对象的常用方法

| 方法名 | 描述 |
| --- | --- |
| println()/print() | 向客户端输出数据。 |
| clear() | 清除缓冲区的内容。如果缓冲区已经被刷新，该方法将抛出IOException异常。 |
| clearBuffer() | 清除缓冲区的内容。与clear()方法不同的是，即使缓冲区已经被刷新，该方法也不会抛出IOException异常。 |
| flush() | 输出（即刷新）缓冲区。 |
| close() | 刷新缓冲区，关闭输出流。 |
| getBufferSize() | 获得使用的缓冲区的大小。 |
| isAutoFlush() | 判断out对象是否进行自动刷新。 |
| getRemaining() | 获得缓冲区中未使用的字节数。 |

# 4.3 JSP内置对象

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ page import="java.io.IOException"%>
<html>
   <body>
       <%! public void printStar(JspWriter out) {
             try {
                out.println("**********<br>");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }%>
        <%
            printStar(out);
            out.println("Hello,world!<br>");
            printStar(out);
         %>
   </body>
</html>
```

## ❖运行效果


360安全浏览器 7.1
http://localhost:8080/TT/first.jsp
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
我的手机
localhost:8080/TT/first.jsp
**********
Hello,world!
**********

# 4.3 JSP内置对象

## ❖pageContext对象

pageContext对象代表页面上下文，该对象主要用于访问页面共享数据，也可以使用它来访问request、session和application范围的属性。

# 4.3 JSP内置对象

| 方法名 | 描述 |
| --- | --- |
| **获取其它内置对象** | |
| getRequest() | 获得当前的请求对象。 |
| getResponse() | 获得当前的响应对象。 |
| getSession() | 获得当前的会话对象。 |
| getServletContext() | 获得当前的Servlet上下文对象。 |
| getOut() | 获得当前的输出流对象。 |
| getServletConfig() | 获得当前的ServletConfig实例。 |
| getPage() | 获得当前的page对象(在Servlet环境中，它是一个Servlet实例)。 |
| getException() | 获得当前的异常对象。 |

# 4.3 JSP内置对象

| 获取和保存属性 | |
|---|---|
| setAttribute(String name,Object value) | 将属性name保存在当前的页面范围内。 |
| getAttribute(String name) | 获取保存在当前页面范围内的属性name。 |
| setAttribute(String name,Object value, int scope) | 将属性name保存在scope指定的范围内，scope的取值可以是PAGE_SCOPE、REQUEST_SCOPE、SESSION_SCOPE或APPLICATION_SCOPE。 |
| getAttribute(String name,int scope) | 获取scope指定范围内的属性name。 |
| removeAttribute(String name,int scope) | 删除scope指定范围内的属性name。 |
| removeAttribute(String name) | 删除所有范围内的名为name的属性。 |
| findAttribute(String name) | 按照page、request、session和application范围的顺序搜索名为name的属性。 |

# 4.3 JSP内置对象

| 请求转发 | |
|---|---|
| forward(String relativeUrlPath) | 将请求转发给其它Web资源 |
| include(String relativeUrlPath) | 将其它资源的响应信息包含在当前响应中 |

# 4.3 JSP内置对象

## ❖exception对象

exception对象用来处理错误异常，是java.lang.Throwable类的一个实例。

exception对象配合page指令一起使用，该指令中errorPage和isErrorPage属性可以为页面提供错误处理机制。

exception对象的主要方法有：

- getMessage()：返回错误信息。
- printStackTrace()：以标准错误的形式输出一个错误和错误的堆栈。
- toString()：以字符串的形式返回一个对异常的描述。

# 4.3 JSP内置对象

【例4-4】exception对象的使用。

本实例是通过使用 page 的errorPage属性和isErrorPage属性处理 JSP 页面的异常。功能如下：

1）从compute.html网页文件接受用户输入的两个整数，单击【计算】按钮，会将数据提交给check.jsp进行检查和计算。

2）check.jsp文件首先对用户提交的数据进行检查，如果用户输入的数据不能被解析为整数（如包含非数值型字符）或者除数为0时，就会抛出异常，这时将会跳转到错误处理页面error.jsp。如果用户输入的数据正确，则计算并输出结果。

```
<html>
  <head>
    <title>整数除法运算</title>
  </head>
  <body>
    <center>
        ------整数除法运算--------
        <form method="post" action="check.jsp">
          <table border="0">
            <tr>
              <td>被除数：</td>
              <td><input type="text" name="divided" /></td>
              <td>除数：</td>
              <td><input type="text" name="divide" /></td>
            </tr>
          </table>
          <p>
            <input type="submit" name="compute" id="compute" value="计算" />
          </p>
        </form>
    </center>
  </body>
</html>
```

## ❖运行效果


整数除法运算 - Windows Internet Explorer
http://localhost:
百度一下，你就知道
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
收藏夹
整数除法运算
------整数除法运算--------
被除数：
32
除数：
4
计算
Internet
100%

# 4.3 JSP内置对象

## check.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"
        errorPage="error.jsp" %>
<html>
   <head>
            <title>计算结果</title>
   </head>
   <body>
            <center>正确的运算结果是：<hr>
            <%
               String divided = request.getParameter("divided"); //接受被除数参数
               String divide = request.getParameter("divide");//接受除数参数
               int i = Integer.parseInt(divided);
               int j = Integer.parseInt(divide);
               out.println(i + "/" + j + "=" + i / j);
             %>
            </center>
   </body>
</html>
```

## ❖运行效果（正确）


计算结果 - Windows Internet Explorer
http://localhost:
百度一下，你就知道
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
收藏夹
计算结果
正确的运算结果是：
32/4=8
完成
Internet
100%

## error.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"
isErrorPage="true" %>
<html>
        <head>
                <title>处理错误信息页面</title>
        </head>
        <body>
                <center>错误信息</center>
                <hr>
                <%
                    out.print(exception.toString());
                %>
        </body>
</html>
```

## ❖运行效果（发生异常）


处理错误信息页面 - Windows Internet Explorer
http://localhost:
百度一下，你就知道
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
收藏夹
处理错误信息页面
错误信息
java.lang.NumberFormatException: For input string: "abcd"
完成
Internet
100%

# 4.3 JSP内置对象

## ❖其它对象

config对象代表当前JSP页面的配置信息，它实现了ServletConfig接口，因此可以通过config对象获得在web.xml文件中指定的配置参数。一般情况下，JSP页面不需要进行配置，所以这个对象很少使用。

page对象是当前页面转换后的Servlet类的实例，一般很少使用这个对象。

# 本讲小结

JSP是对Servlet的扩展，其目的是简化建立和管理动态网站的工作，它通过在网页文件中嵌入脚本代码来产生动态内容。

本讲学习了如何使用JSP进行Java Web应用程序的开发。知识点包括JSP页面的构成、指令元素、脚本元素和动作元素以及内置对象等。