JSP入门

本章内容：共3小节，11个知识点

- 第1节：概述
- 第2节：页面元素及内置对象概念
- 第3节：Servlet与JSP作用总结

本章目标

- 理解JSP的基本概念；
- 理解JSP的运行过程和本质；
- 掌握JSP页面元素的用法；

第1节【概述】

- 知识点1：动态网页的执行原理
- 知识点2：JSP的功能与特性
- 知识点3：JSP的执行流程解析

知识点1【动态网页的执行原理】

- 上一章使用Servlet生成动态网页，我们发现：任何内容，不管是静态的还是动态的，都使用out.println一行一行输出;
- 执行步骤如下：
  - 容器初始化Servlet实例，根据请求方法，调用相应的doXXX方法，并初始化请求和响应对象，作为doXXX方法的参数使用;
  - 执行doXXX方法后，将响应对象中的数据流写到客户端;
  - 客户端浏览器将收到的数据进行解析，显示给用户;
- 问题来了：Servlet中生成动态页面太麻烦了，其实大部分都是静态内容，也要一行一行输出。
- 解决办法：JavaEE提供了新的动态页面组件JSP，可以方便的生成动态页面。

这样一来，是不是Servlet就没啥用了？No，No，No，Servlet还很有用，可以用来编写控制逻辑，后续会学习哦~~~~

知识点2【JSP的功能与特性】-1

- JSP（Java Server Pages）是JavaEE规范中的Web组件，用来编写动态页面；
- JSP运行在服务器端，本质是Servlet；【后续会解释】
- JSP文件以.jsp为后缀，在Eclipse的工程目录中存在WebContent目录下；
- JSP文件可以直接在浏览器中访问；
- JSP文件中的内容就是 HTML+Java代码，静态部分使用HTML和文本即可，动态部分使用Java代码；

知识点2【JSP的功能与特性】-2

- JSP文件中的内容就是 HTML+Java代码：

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
</head>
<body>
 我是一个JSP页面<br>
<%System.out.println("在JSP中可以编写Java代码");%>
你的IP地址是：<%=request.getRemoteAddr() %>
</body>
</html>
```

Java代码写到JSP中需要特定的格式，下节介绍。

课堂案例：index.jsp

JSP中的指令元素，后续学习。

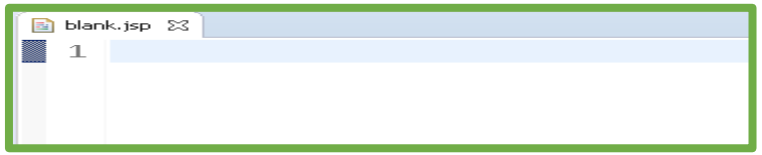
JSP中的静态部分都使用HTML及文本即可，比Servlet简单很多。

JSP中的动态部分使用Java代码，<%%>为脚本片段，<%=%>为输出表达式，后续介绍。

知识点3【JSP的执行流程解析】-1

课堂案例：
blank_jsp.java

- 为了理解JSP的执行流程，我们编写一个空白的jsp，blank.jsp，该文件中内容为空。



- 在浏览器中通过http://127.0.0.1:8080/chapter03/blank.jsp访问，流程如下：

翻译 → 编译 → 运行

- 1、翻译：Web服务器找到blank.jsp，对其进行翻译，生成blank_jsp.java文件；查看路径： 工作空间D:\eclipse-workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\chapter03\org\apache\jsp
- 2、编译：服务器将blank_jsp.java编译成类文件，翻译和编译的过程遵守Servlet规范，因此说JSP的本质也是Servlet；
- 3、实例化并提供服务：服务器实例化类，调用类中的_jspService方法提供服务

知识点3【JSP的执行流程解析】-2

课堂案例：
blank_jsp.java

- 打开blank_jsp.java，发现即使一个空白的JSP文件，翻译成Java文件后，也有很多代码，关键代码是_jspService方法：

```
public void _jspService(final javax.servlet.http.HttpServletRequest request, final javax.servlet.http.HttpServletResponse response)
throws java.io.IOException, javax.servlet.ServletException {
    final javax.servlet.jsp.PageContext pageContext;
    javax.servlet.http.HttpSession session = null;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    javax.servlet.jsp.JspWriter _jspx_out = null;
    javax.servlet.jsp.PageContext _jspx_page_context = null;
    try {
      response.setContentType("text/html");
      pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
       _jspx_out = out;

}
```

方法有请求和响应两个参数，和Servlet的doXXX方法完全一样

方法体中声明并初始化了一系列的对象，包括out,session等，这就是以后学习的内置对象了。

JSP文件中所有HTML、文本以及Java代码等内容，都将被翻译成Java代码插入到这个位置。

知识点3【JSP的执行流程解析】-3

课堂案例：
blankNew.jsp

- 定义blankNew.jsp，加入简单内容；

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
</head>
<body>
我是blankNew.jsp文件<br>
<%
for(int i=0;i<10;i++){
System.out.println("在控制台打印");
}%>
您的IP地址：<%=request.getRemoteAddr() %>>
</body>
</html>
```

知识点3【JSP的执行流程解析】-4

课堂案例：
blankNew_jsp.java

- 查看blankNew_jsp.java文件的变化

```
public void _jspService(final javax.servlet.http.HttpServletRequest request, final javax.servlet.http.HttpServletResponse response)
      throws java.io.IOException, javax.servlet.ServletException {
  final javax.servlet.jsp.PageContext pageContext;
  javax.servlet.http.HttpSession session = null;
  final javax.servlet.ServletContext application;
  final javax.servlet.ServletConfig config;
  javax.servlet.jsp.JspWriter out = null;
  final java.lang.Object page = this;
  javax.servlet.jsp.JspWriter _jspx_out = null;
  javax.servlet.jsp.PageContext _jspx_page_context = null;
  try {
    response.setContentType("text/html; charset=utf-8");
    pageContext = _jspxFactory.getPageContext(this, request, response,null, true, 8192, true);
    _jspx_page_context = pageContext;
    application = pageContext.getServletContext();
    config = pageContext.getServletConfig();
    session = pageContext.getSession();
    out = pageContext.getOut();
     jspx_out = out;
    out.write("\r\n");
    out.write("<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\" \"http://www.w3.org/TR/html4/loose.dtd\">\r\n");
    out.write("<html>\r\n");
    out.write("<head>\r\n");
    out.write("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">\r\n");
    out.write("<title>Insert title here</title>\r\n");
    out.write("</head>\r\n");
    out.write("<body>\r\n");
    out.write("我是blankNew.jsp文件<br>\r\n");
for(int i=0;i<10;i++){
System.out.println("在控制台打印");
}
    out.write("\r\n");
    out.write("您的IP地址： ");
    out.print(request.getRemoteAddr() );
    out.write("\r\n");
    out.write("</body>\r\n");
    out.write("</html>");
```

可见，JSP中所有 内容都将被翻译到_jspService的方法中，插入到那段固定的代码后。静态内容都使用out输出，和Servlet中的输出是一样一样的！所有的Java代码都直接翻译到对应位置。由此可见，JSP的本质就是一个Servlet，不过是服务器翻译生成了Java类，不用我们编写。

本节总结提问【概述】

- 为什么要使用JSP组件?
- JSP执行的流程是什么?

本节总结【概述】

- Servlet生成动态页面比较繁琐，使用JSP生成动态页面比较便捷，因为其中的静态内容可以使用HTML生成；
- JSP的执行过程是：翻译-编译-实例化-提供服务；
- JSP的本质就是Servlet，不过是服务器将JSP进行了翻译和编译；可以说，JSP也是一个Java类；

第2节【页面元素及内置对象概念】

- 知识点1：脚本元素
- 知识点2：表达式元素
- 知识点3：模版元素
- 知识点4：声明元素
- 知识点5：内置对象概念

知识点1【脚本元素】

课堂案例：
testElements.java

- 脚本元素可以用来包含任意Java代码；
- 格式为：<%Java代码%>，例如：

```
<%System.out.println("这是脚本元素"); %>
```

  - 服务器翻译脚本元素时，将把其中Java代码直接翻译到_jspService方法中，如果语法错误，将在浏览器中提示错误；

知识点2【表达式元素】

课堂案例：
testElements.java

- 表达式元素用来向页面输出动态内容；
- 格式为：<%=Java代码%>，例如：

```
您的IP地址：<%=request.getRemoteAddr() %>
```

  - 服务器翻译表达式元素时，将把其中Java代码部分的返回值使用out.write语句输出，例如：

```
out.write("您的IP地址：");
out.print(request.getRemoteAddr() );
```

知识点3【模版元素及注释元素】

课堂案例：
testElements.java

- 模板元素指JSP中静态HTML或者XML内容；
- 在JSP中可以使用注释元素，有三种情况：
  - 格式为<%--JSP注释--%>；JSP的注释只有在源代码中可见，翻译时已经忽略；
  - 在JSP中，除了使用JSP注释外，还可以使用HTML注释，<!--HTML注释-->，HTML注释会被返回到客户端，但是不显示到页面中；
  - JSP中的Java代码部分，可以使用Java注释；Java注释会翻译到.java文件中，但是编译时忽略；

知识点4【声明元素】

课堂案例：
testElements.java

- 如果需要在JSP文件中定义类的成员变量或方法，可以使用声明元素，格式为<%! 声明语句%>
- 例如：

```
<%!
  private String path="WEB-INF";
  public void readPropertiesFile(){
  }
%>
```

- 声明元素被翻译到Java类中，而不是_jspService方法中；
- 实际中使用不多；

知识点5【内置对象的概念】

课堂案例：
testElements.java

- 内置对象指的是在JSP中可以直接使用的对象，不需要声明，直接使用固定的名字使用即可；例如<%=request.getRemoteAddr()%>中的request就是内置对象；

问：不用声明，对象从哪里来的?

答：服务器在翻译编译JSP时默认声明创建的。都在_jspService方法的参数和方法体内。

问：那为啥就可以直接用呢？

答：我们自己写的JSP中的所有内容，都被服务器翻译在内置对象声明创建后，当然就可以直接用啦。

问：有哪些内置对象呢？

问：一共有9个，比如request,response,out,session等，查看一个JSP翻译成的Java文件就很清楚了。

JSP九大内置对象

request	请求对象	类型 javax.servlet.ServletRequest	作用域 Request
response	响应对象	类型 javax.servlet.SrvletResponse	作用域 Page
pageContext	页面上下文对象	类型 javax.servlet.jsp.PageContext	作用域 Page
session	会话对象	类型 javax.servlet.http.HttpSession	作用域 Session
application	应用程序对象	类型 javax.servlet.ServletContext	作用域 Application
out	输出对象	类型 javax.servlet.jsp.JspWriter	作用域 Page
config	配置对象	类型 javax.servlet.ServletConfig	作用域 Page
page	页面对象	类型 javax.lang.Object	作用域 Page
exception	例外对象	类型 javax.lang.Throwable	作用域 page

JSP中的九大隐含对象可分为4类：

- 1. 与输入/输出有关的对象： request、response、out
- 2. 与属性作用域有关的对象：session、application、pageContext
- 3. 与Servlet 相关对象：page、config
- 4. 与错误处理有关的：exception

作用域范围从小到大顺序：

- page---->request---->session---->application

- page 当前页面有效(页面跳转后无效)
- request 同一次请求有效（请求转发后有效，重定向后无效）
- session 同一次对话有效（同一个浏览器在退出关闭之前都有效）
- application 全局有效（整个项目）

本节总结提问【页面元素及内置对象概念】

- 脚本元素、表达式元素的作用是什么?
- 有几种注释?
- 声明元素有什么不同?
- 内置对象是什么意思？如何使用?
- 9大内置对象，4个作用域范围

本节总结【页面元素及内置对象概念】

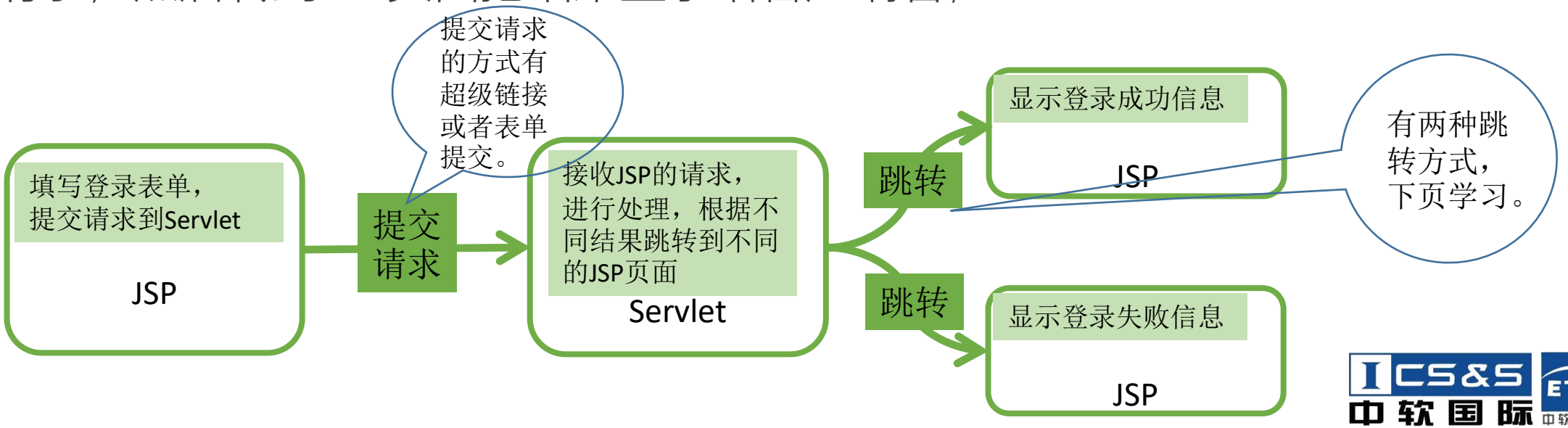
- JSP中的Java代码都可以直接写到脚本元素中，即<%%>中；
- 如果要向浏览器输出动态内容，可以使用表达式元素<%=%>；
- JSP中有三种注释，分别是JSP注释、HTML注释、Java注释；
- <%!%>声明元素只能用来编写声明代码，翻译到Java类中，而不是_jspService方法中；
- 内置对象指的是服务器已经声明并赋值的对象，可以直接使用；

第3节【Servlet与JSP的作用】

- 知识点1：Servlet与JSP的作用总结
- 知识点2：Servlet与JSP之间跳转的方式
- 知识点3：请求属性的使用

知识点1【Servlet与JSP的作用总结】

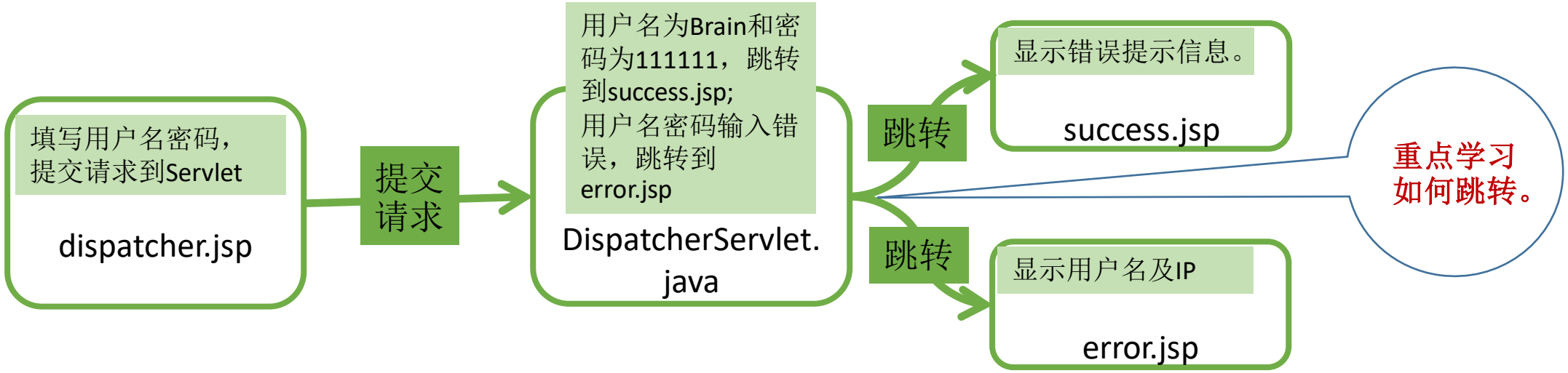
- Servlet和JSP都可以生成动态页面；然而，显然JSP更适合生成动态页面，因为其中的静态部分可以直接使用HTML即可；
- 那么问题来了：Servlet有什么作用呢?
- 实际应用中，Servlet是不会用来生成动态页面的，而是会用来接收来自JSP的请求，处理请求，然后转到JSP页面把结果显示给客户端看；



知识点2【 Servlet与JSP之间跳转的方式】-1

课堂案例：
DispatcherServlet.java

- 让我们用一个简单例子来理解跳转方式。【用于登录功能】



- 跳转方式一：响应重定向，响应接口中提供了该方法。

方法声明	方法描述
void sendRedirect(java.lang.String location)	应重定向到location，相当于客户端重新请求location所在的资源；

知识点2【 Servlet与JSP之间跳转的方式】-2

课堂案例：
DispatcherServlet.java
error.jsp
success.jsp

- 跳转方式一：响应重定向，响应接口中提供了该方法。

- 第二种跳转方式：请求转发，RequestDispatcher接口定义了请求转发的方法；

方法声明	方法描述
forward(ServletRequest request, ServletResponse response)	将请求转发到服务器上的其他资源，包括其他的Servlet，JSP等；

- 要使用forward方法，需要先获得RequestDispatcher对象；请求接口中提供了获得该对象的方法：

方法声明	方法描述
RequestDispatcher getRequestDispatcher(java.lang.String path)	使用path返回一个RequestDispatcher 对象

```
else{
//                      response.sendRedirect("success.jsp");
                        request.getRequestDispatcher("success.jsp").forward(request, response);
                }
```

注意，路径是相对于当前请求路径的相对路径。

课堂案例：
dispatcher.jsp

```
<form name = "myform" action = "DispatcherServlet" method = "post">
                        <tr><td>用户名</td>
                        <td><input type = "text" name = "name"></td>
                       </tr>
                        <tr>
                                <td>密码</td>
                                <td><input type = "password" name = "password"></td>
                        </tr>
                        <tr>
                                <td><input type = "submit" value = "提交"/></td>
                                <td><input type = "reset" value = "重置"/></td>
                        </tr>
</form>
```

com.chinasofti.chapter03.section03包下

课堂案例：
DispatcherServlet.java

```
public void doPost(HttpServletRequest request,HttpServletResponse response)  throws ServletException,IOException{
                request.setCharacterEncoding("UTF-8");// 设置浏览器向服务器发送的字符编码
        response.setContentType("text/html;charset= utf-8");// 响应 设置服务器返回给浏览器时的字符编码
                PrintWriter out = response.getWriter();
                String userName = request.getParameter("name");
                String pass = request.getParameter("password");
                if(userName.equals("Brain")&&pass.equals("111111")){
        request.getRequestDispatcher("success.jsp").forward(request, response);
                }else{
                        //RequestDispatcher rd = request.getRequestDispatcher("error.jsp");
                        //rd.forward(request, response);
                         response.sendRedirect("error.jsp");
                }
}
```

跳转页面

课堂案例：
error.jsp
success.jsp

success.jsp

success!!!

error.jsp

error!!!

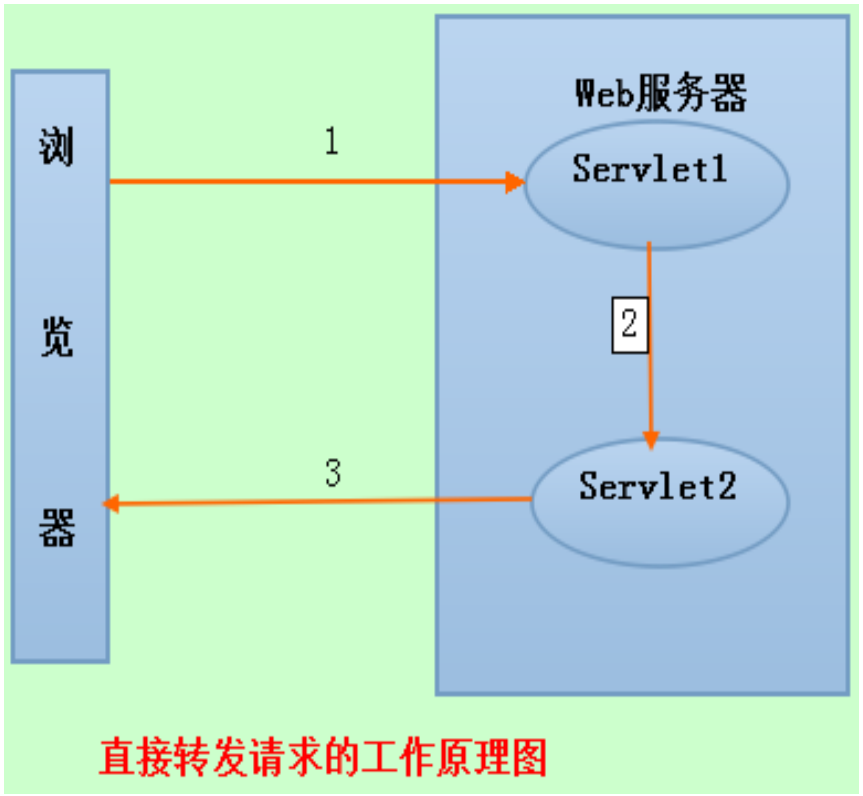
请求转发和重定向

- Forward和Redirect代表了两种请求转发方式：直接转发和间接转发。

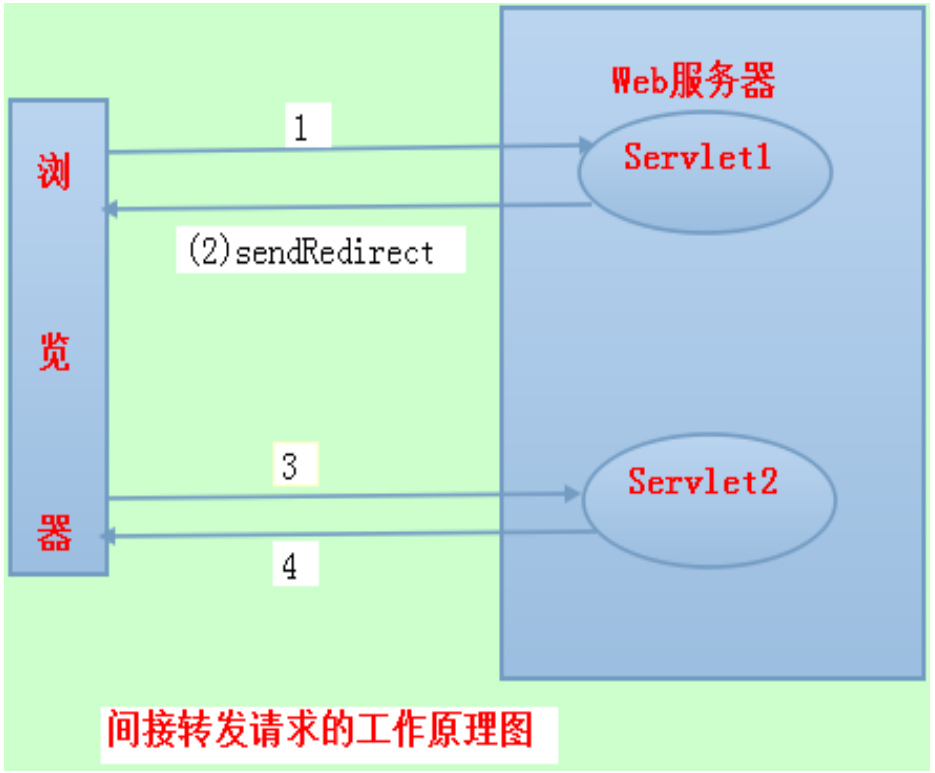
- 直接转发方式（Forward），客户端和浏览器只发出一次请求，Servlet、HTML、JSP或其它信息资源，由第二个信息资源响应该请求，在请求对象request中，保存的对象对于每个信息资源是共享的。

- 间接转发方式（Redirect）实际是两次HTTP请求，服务器端在响应第一次请求的时候，让浏览器再向另外一个URL发出请求，从而达到转发的目的。

知识点2【 Servlet与JSP之间跳转的方式】-3



直接转发请求的工作原理图



间接转发请求的工作原理图

思考

- 请求参数是自动被封装到了请求对象中，因此只要做了请求转发，就能够在下一个资源的请求中获取；如果是在Servlet中新定义的，或者通过调用其他资源获得的数据，如何传递到其他的Servlet或JSP呢？

知识点3【请求属性的使用】-1

- 如果需要在Servlet，JSP之间跳转时，同时把一些自定义的、或者通过数据库查询的、或者调用其他资源获得的数据传递到下一个资源时，就可以把这些数据设置为请求的属性即可。
- 请求接口中定义了一系列与属性有关的方法。

方法声明	方法描述
void setAttribute(java.lang.String name, java.lang.Object o)	将任意类型对象设置为请求的属性，指定一个名字；
java.lang.Object getAttribute(java.lang.String name)	通过属性的名字，获取属性的值；
void removeAttribute(java.lang.String name)	通过属性的名字，删除属性；

知识点3【请求属性的使用】-2

课堂案例：
DispatcherServlet.java
success.jsp

- 修改DispatcherServlet.java，将一个字符串数组设置为属性进行传递【无实际意义，只为理解该知识点】

```
//    将一个字符串型数组作为请求属性传递
                String[] addrs={"BeiJing","ShangHai","GuangZhou"};
                request.setAttribute("city", addrs);
```

- 修改success.jsp，可以获取请求属性进行显示

```
<%
 String[] city=(String[])request.getAttribute("city");
 for(String c:city){
%>
<%=c %>
<%} %>
```

在实际应用中，请求属性使用特别广泛。当需要在组件之间传递一些数据，只在请求范围内使用时，就可以使用请求属性。

本节总结提问【Servlet与JSP的作用总结】

- Servlet和JSP分别有什么作用?
- Servlet和JSP之间跳转有几种方式，有何区别?
- 什么是请求属性？和请求参数有啥区别?

本节总结【 Servlet与JSP的作用总结】

- JSP往往用来生成动态页面，而Servlet虽然可以生成动态页面却过于麻烦，往往用来接收JSP的请求，处理请求，然后跳转到不同JSP页面进行结果显示；
- Servlet和JSP之间的跳转有两种方式，分别是响应重定向和请求转发；
- 响应重定向相当于客户端重新发出请求，之前的请求不再保存；请求转发是把当前请求转发到下一个资源；比较常用的是请求转发；
- 请求参数是用户提交请求时，自动封装到请求对象中的一些输入信息，都是String类型；
- 请求属性可以是任意类型的对象，可以用setAttribute方法将对象作为属性存储到请求对象中；

本章总结

- 本章主要对JSP进行快速入门；
- JSP的本质也是一个Java类，不过是服务器进行翻译编译；遵守Servlet规范，所以也可以说JSP的本质就是Servlet；
- JSP的执行流程经过 翻译-编译-实例化-提供服务 几个步骤；
- JSP文件包括HTML及Java代码，根据不同的需要，可以使用页面元素、表达式元素、注释元素、声明元素；
- 服务器总是把JSP文件按照一定的规范进行翻译，除了声明元素外，所有JSP中的内容都翻译到_jspService方法体中，该方法总是定义一系列的对象，称为内置对象，可以在编写JSP时直接使用，比如request/response/out等。
- JSP主要用来生成动态页面，Servlet用来接收请求并处理请求，根据结果跳转到不同的JSP显示结果；
- 有两种跳转方法：响应重定向、请求转发；
- 请求属性可以用来在组件之间共享对象；

本章作业

- 作业1:
- 题目：实现登录页面，输入用户名密码都为admin（用户名和密码不区分大小写），跳转到成功页面，否则调转到失败页面;

  难度：中
- 作业2:
- 题目：分别应用重定向和请求转发两种实现页面跳转。

  难度：中