Tarena Technology
达内科技

Tarena Teaching System

Servlet和JSP（上）

学员用书

Java企业应用及互联网
高级工程师培训课程

达内集团教学研发部 编著

目　录

Unit01 1
1. Servlet 基础 3
1.1. Web 应用的演变 3
1.1.1. 单机程序 3
1.1.2. 网络程序-主机终端 3
1.1.3. 网络程序-两层 CS 架构 3
1.1.4. 两层 CS 架构优缺点 4
1.1.5. 网络程序-三层 CS 架构 4
1.1.6. 三层 CS 架构的工作流程 4
1.1.7. 三层 CS 架构的优缺点 5
1.1.8. 网络程序-BS 架构 5
1.1.9. BS 架构的优势 5
1.2. 什么是 Servlet 6
1.2.1. 什么是 Servlet？ 6
1.2.2. 扩展 Web 服务器功能 6
1.2.3. 组件规范 6
1.2.4. Servlet 组件运行原理 7
1.3. 如何开发一个 Servlet 7
1.3.1. 安装 Tomcat 7
1.3.2. 开发 Servlet 的步骤 7
1.3.3. 开发 Servlet 的步骤（续） 8
1.4. 常见错误及解决方式 8
1.4.1. 404 错误 8
1.4.2. 404 错误的解决方法 8
1.4.3. 容器如何找到 service 方法 9
1.4.4. 405 错误 9
1.4.5. 405 错误的解决方法 9
1.4.6. 500 错误 10
1.4.7. 500 错误的解决方法 10
2. HTTP 协议 10
2.1. HTTP 协议 10
2.1.1. 什么是 HTTP 协议 10
2.1.2. Part1：如何通信 11
2.1.3. Part2：数据格式 11
2.1.4. 请求数据包组成 11

达内IT培训集团

2.1.5. 响应数据包组成 12
2.2. Servlet 如何处理 HTTP 协议 13
2.2.1. 如何控制通信数据 13
2.2.2. HttpServletRequest 对象 13
2.2.3. HttpServletResponse 对象 13
经典案例 14
1. 安装和配置 Tomcat 14
2. 手动开发第一个 Web 项目 23
3. 使用工具开发 Web 项目 30
4. 输出请求数据包中的内容 36
5. 向客户端输出中文 41
课后作业 44
Unit02 46
1. Servlet 工作原理 48
1.1. Servlet 如何获取请求参数 48
1.1.1. 获取请求参数值的方法 48
1.1.2. getParameter 方法 48
1.1.3. getParameterValues 方法 48
1.2. 请求方式 49
1.2.1. 为什么区分请求方式 49
1.2.2. 请求方式的种类 49
1.2.3. GET 请求方式 49
1.2.4. GET 请求方式的特点 50
1.2.5. POST 请求方式 50
1.2.6. POST 请求方式的特点 50
1.3. 如何处理中文参数 51
1.3.1. 为什么表单提交中文会出现乱码 51
1.3.2. 解决 POST 方式时的乱码问题 51
1.3.3. 解决 GET 方式时的乱码问题 51
1.4. Servlet 如何输出中文 52
1.4.1. 为什么返回的页面会有乱码 52
1.4.2. 如何解决输出内容的乱码 52
1.5. Servlet 如何访问数据库 52
1.5.1. 使用 JDBC 技术访问数据库 52
1.6. Servlet 如何运行 53
1.6.1. Servlet 运行的详细步骤 53
经典案例 54
1. Servlet 获取请求参数值 54

2. 请求方式 GET 和 POST 的区别 56
3. 处理 POST 请求中的中文参数值 60
4. 处理 GET 请求中的中文参数值 63
5. 员工管理——添加员工信息 66
6. 员工管理——使用 JDBC 添加员工信息 70
7. 员工管理——使用 JDBC 查询所有员工信息 76
课后作业 80
Unit03 82
1. 容器对路径的处理 84
1.1. 重定向 84
1.1.1. 什么是重定向 84
1.1.2. 重定向原理 84
1.1.3. 重定向原理（续） 84
1.1.4. 如何重定向 85
1.1.5. 重定向特点 85
1.2. Servlet 容器如何处理请求资源路径 85
1.2.1. 什么是请求资源路径 85
1.2.2. Web 服务器对请求地址的处理过程 86
1.2.3. 匹配 Servlet 规则-精确匹配 86
1.2.4. 匹配 Servlet 规则-通配符匹配 86
1.2.5. 匹配 Servlet 规则-后缀匹配 87
1.2.6. 无匹配的 Servlet 的处理 87
1.3. 一个 Servlet 实现多请求 87
1.3.1. 为什么要将多 Servlet 合并 87
1.3.2. 使用后缀匹配模式修改 web.xml 88
1.3.3. 分析请求资源后分发 88
2. Servlet 特性 88
2.1. Servlet 的生命周期 88
2.1.1. 什么是 Servlet 的生命周期 88
2.1.2. 生命周期的四个阶段 89
2.1.3. Servlet 生命周期原理图 90
2.1.4. Servlet 核心接口和类 90
2.1.5. Servlet 接口 91
2.1.6. Servlet 核心类 91
2.2. ServletContext 91
2.2.1. 什么是 Servlet 上下文 91
2.2.2. 如何获得 Servlet 上下文 92
2.2.3. Servlet 上下文的作用及特点 92

达内IT培训集团

2.3. Servlet 线程安全问题 92
2.3.1. 为什么会有线程安全问题 92
2.3.2. 如何保证 Servlet 的线程安全 93
经典案例 94
1. 员工管理——使用 JDBC 删除员工信息 94
2. 员工管理——使用 JDBC 修改员工信息 98
3. 容器对 URI 的处理 105
4. 员工管理——单一控制器实现员工信息管理 109
5. Servlet 的生命周期 117
6. ServletContext 对象（Servlet 上下文对象）..... 121
7. Servlet 线程安全 125
课后作业 130
Unit04 131
1. JSP 基本语法 133
1.1. JSP 的由来 133
1.1.1. 为什么有 JSP 规范 133
1.1.2. 什么是 JSP 133
1.2. JSP 编写规范 133
1.2.1. 如何编写 JSP 133
1.2.2. JSP 页面中的 HTML 代码 134
1.2.3. JSP 页面中的注释 134
1.2.4. JSP 页面中的 Java 代码 134
1.2.5. JSP 表达式 135
1.2.6. JSP 小脚本 135
1.2.7. JSP 声明 136
1.2.8. JSP 页面中的指令 137
1.2.9. Page 指令 137
1.2.10. Include 指令 137
1.2.11. JSP 页面中的隐含对象 138
2. JSP 运行原理 139
2.1. JSP 运行原理 139
2.1.1. JSP 是如何运行的 139
2.1.2. JSP 如何转换为 Java 139
2.1.3. 如何将静态页面转化为动态页面 139
经典案例 140
1. JSP 基本元素练习 140
2. JSP 指令练习 141
3. 员工管理——使用 JSP 实现员工信息列表 1 145

4. 员工管理——使用 JSP 实现员工信息列表 2 148
课后作业 152
Unit05 154
1. 转发 156
1.1. 转发 156
1.1.1. 什么是转发 156
1.1.2. 如何实现转发 156
1.1.3. 绑定数据到 request 对象 156
1.1.4. 获得转发器 157
1.1.5. 转发 157
1.1.6. 转发的原理 157
1.1.7. 转发的特点 158
1.1.8. 转发和重定向的区别 158
2. JSP 开发常见问题 159
2.1. 异常处理 159
2.1.1. 编程式的异常处理 159
2.1.2. 容器中的声明式处理 159
2.2. 路径问题 159
2.2.1. 什么是路径 159
2.2.2. 什么是相对路径 160
2.2.3. 相对路径（续） 160
2.2.4. 什么是绝对路径 160
2.2.5. 绝对路径（续） 161
2.2.6. 路径的处理技巧 161
经典案例 162
1. 员工管理——使用转发、JSP 实现完整的增删改查 162
2. 员工管理——添加异常处理 175
3. 路径的练习 178
4. 员工管理——注册 181
课后作业 194

达内IT培训集团

Servlet和JSP(上)

Unit01

知识体系.....Page 3

Servlet 基础	Web 应用的演变	单机程序
		网络程序-主机终端
		网络程序-两层 CS 架构
		两层 CS 架构优缺点
		网络程序-三层 CS 架构
		三层 CS 架构的工作流程
		三层 CS 架构的优缺点
		网络程序-BS 架构
		BS 架构的优势
	什么是 Servlet	什么是 Servlet？
		扩展 Web 服务器功能
		组件规范
		Servlet 组件运行原理
	如何开发一个 Servlet	安装 Tomcat
		开发 Servlet 的步骤
		开发 Servlet 的步骤（续）
	常见错误及解决方式	404 错误
		404 错误的解决方法
		容器如何找到 service 方法
		405 错误
		405 错误的解决方法
		500 错误
		500 错误的解决方法
HTTP 协议	HTTP 协议	什么是 HTTP 协议
		Part1：如何通信
		Part2：数据格式
		请求数据包组成
		响应数据包组成
	Servlet 如何处理 HTTP 协议	如何控制通信数据

达内IT培训集团

		HttpServletRequest 对象
		HttpServletResponse 对象

经典案例.....Page 14

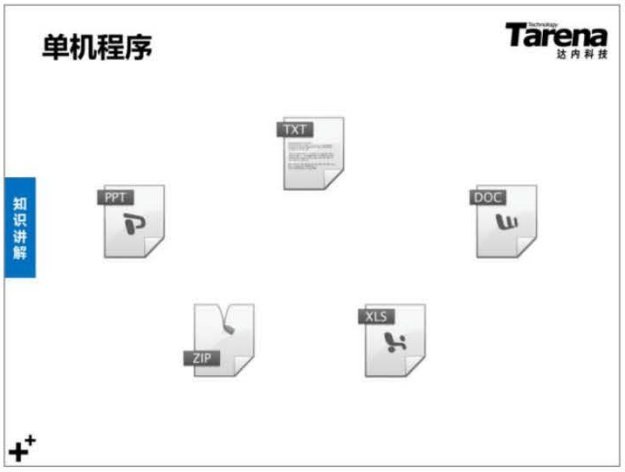
安装和配置 Tomcat	安装 Tomcat
手动开发第一个 Web 项目	开发 Servlet 的步骤
使用工具开发 Web 项目	开发 Servlet 的步骤(续)
输出请求数据包中的内容	HttpServletRequest 对象
向客户端输出中文	HttpServletResponse 对象

课后作业.....Page 44

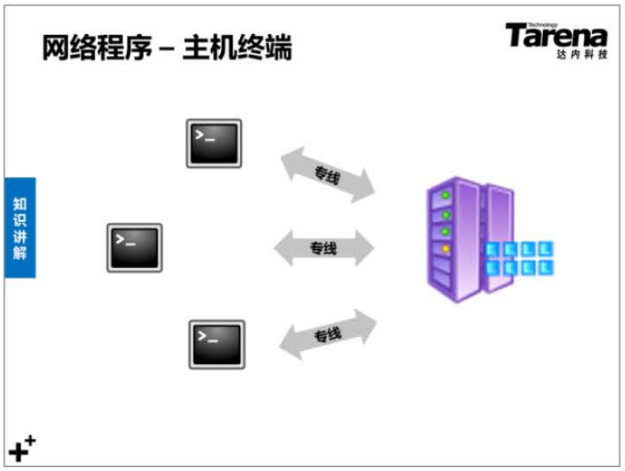
1. Servlet 基础

1.1. Web 应用的演变

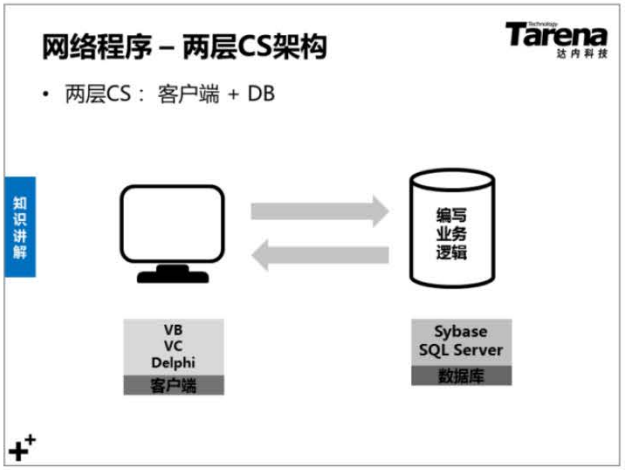
1.1.1. 【Web 应用的演变】单机程序



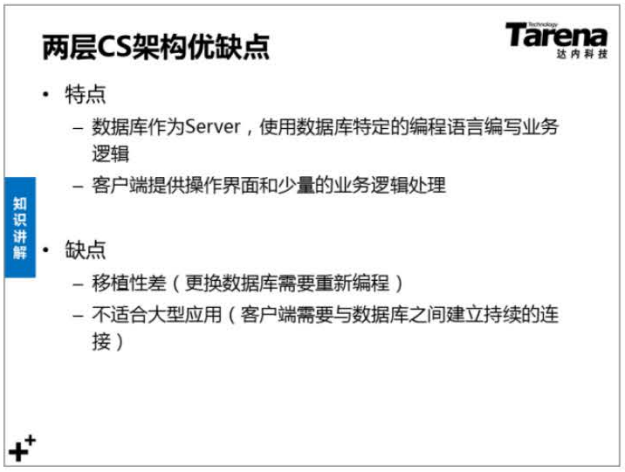
1.1.2. 【Web 应用的演变】网络程序-主机终端



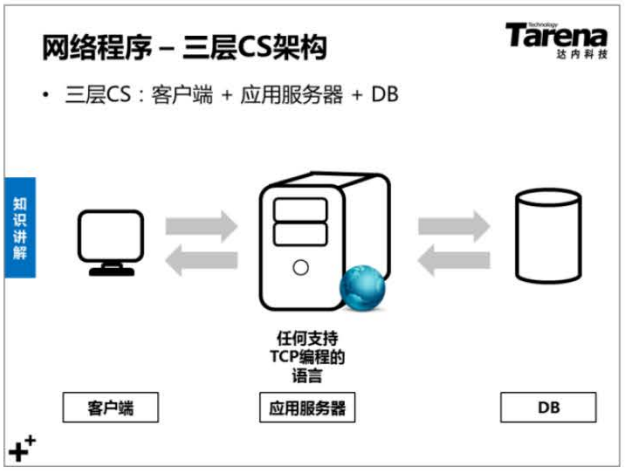
1.1.3. 【Web 应用的演变】网络程序-两层 CS 架构



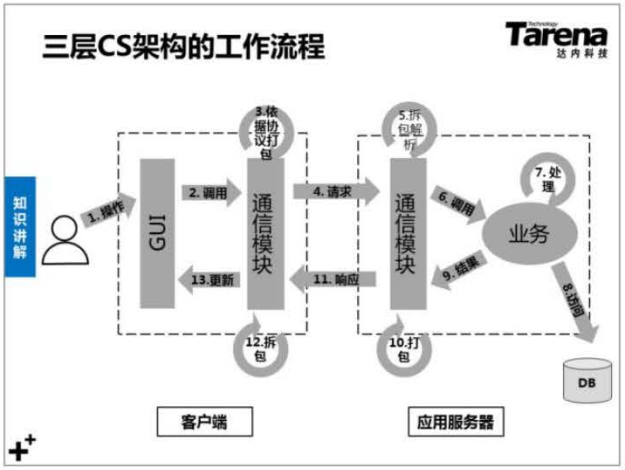
1.1.4. 【Web 应用的演变】两层 CS 架构优缺点



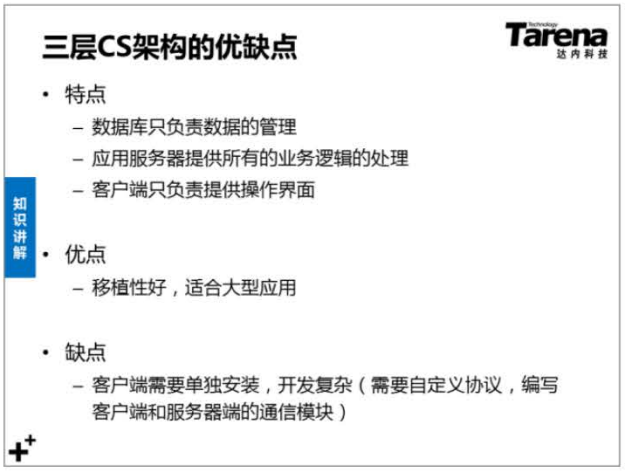
1.1.5. 【Web 应用的演变】网络程序-三层 CS 架构



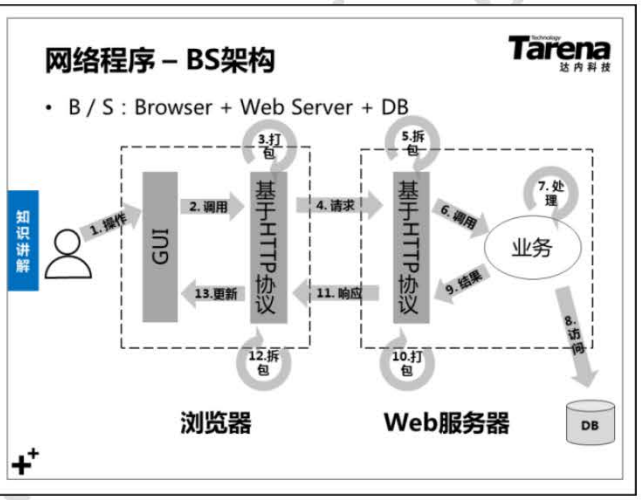
1.1.6. 【Web 应用的演变】三层 CS 架构的工作流程



1.1.7. 【Web 应用的演变】三层 CS 架构的优缺点



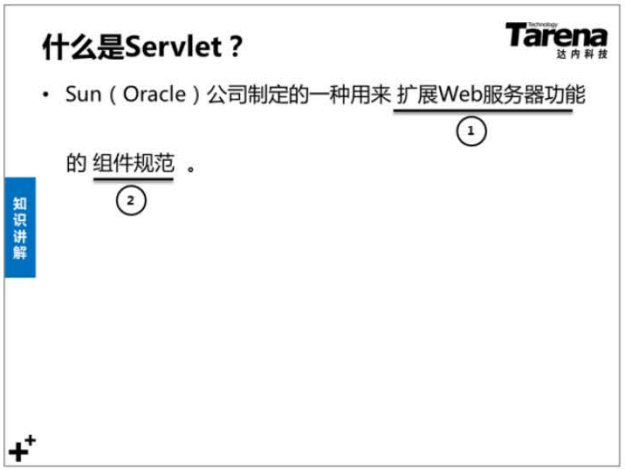
1.1.8. 【Web 应用的演变】网络程序-BS 架构



1.1.9. 【Web 应用的演变】BS 架构的优势

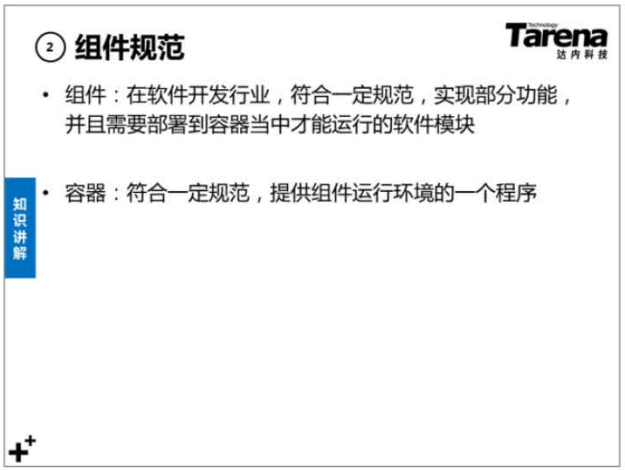
1.2. 什么是 Servlet

1.2.1. 【什么是 Servlet】什么是 Servlet?

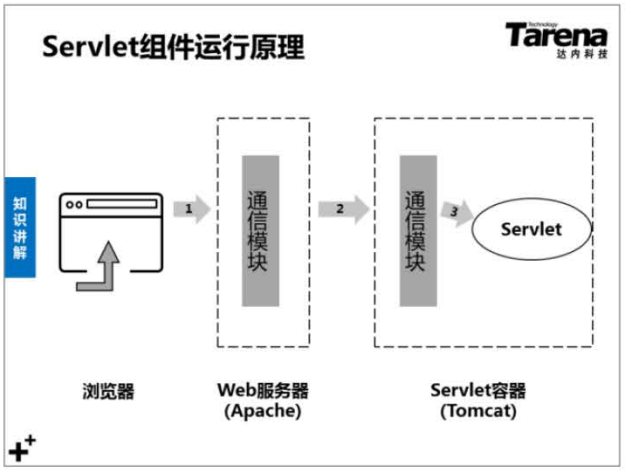


1.2.2. 【什么是 Servlet】扩展 Web 服务器功能

1.2.3. 【什么是 Servlet】组件规范



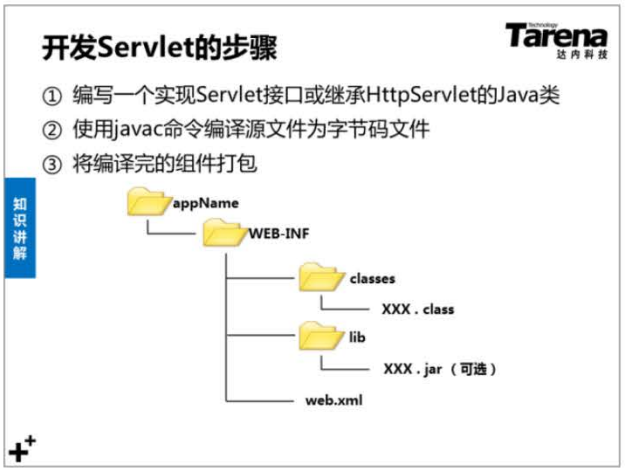
1.2.4. 【什么是 Servlet】Servlet 组件运行原理



1.3. 如何开发一个 Servlet

1.3.1. 【如何开发一个 Servlet】安装 Tomcat

1.3.2. 【如何开发一个 Servlet】开发 Servlet 的步骤



1.3.3. 【如何开发一个 Servlet】开发 Servlet 的步骤（续）

开发Servlet的步骤(续)

解说识知

④ 部署：将appName整个文件夹拷贝到Tomcat的webapps文件夹下

⑤ 启动Tomcat并访问Servlet：在地址栏中输入符合一定规范的地址

http:// ip:port / appName / servlet的URL

（URL在web.xml描述文件中可以找到）

1.4. 常见错误及解决方式

1.4.1. 【常见错误及解决方式】404 错误

404错误

解说识知

- 404数字是什么？
  - 是服务器执行完客户端的请求以后，返回给客户端的一个执行结果的状态编码
- 产生的原因：Web服务器（容器）根据请求地址找不到对应资源。

  如：

  ① 地址错误（拼写不正确，字母大小写错误）

  ② web.xml文件中的两个<servlet-name>不一致

  ③ 工程没有部署

  ④ Web应用程序部署结构没有遵守Servlet规范

1.4.2. 【常见错误及解决方式】404 错误的解决方法

404错误的解决方法

解说识知

- 按照 http : // ip:port / appName / url-pattern规则检查请求地址，区分大小写
- 检查web.xml文件中的<servlet>和<servlet-mapping>节点中的两个<servlet-name>是否一致
- 只有部署以后的工程才能访问
- 检查工程结构是否符合规范

1.4.3. 【常见错误及解决方式】容器如何找到 service 方法

1.4.4. 【常见错误及解决方式】405 错误

1.4.5. 【常见错误及解决方式】405 错误的解决方法

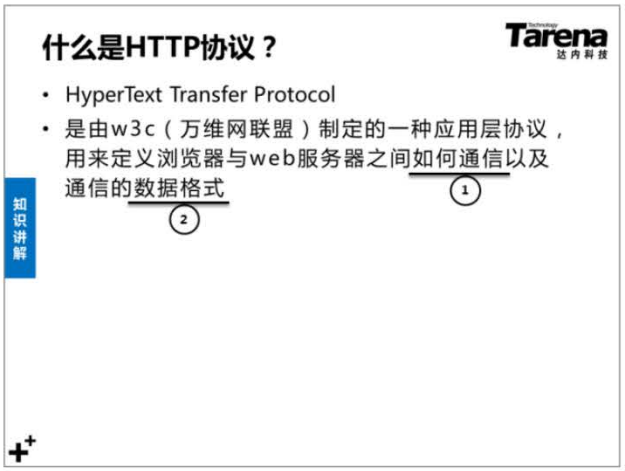
1.4.6. 【常见错误及解决方式】500 错误

1.4.7. 【常见错误及解决方式】500 错误的解决方法

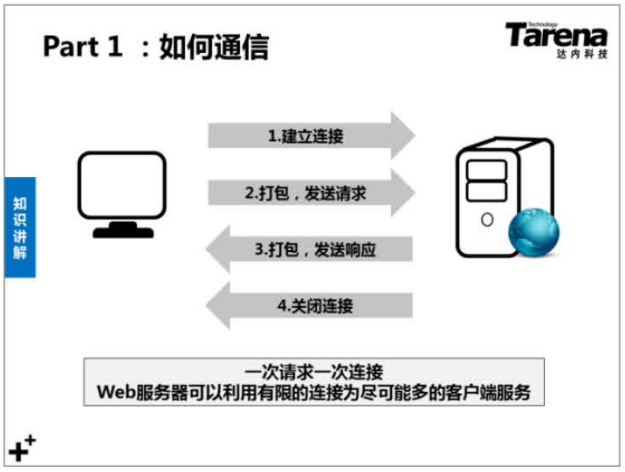
2. HTTP 协议

2.1. HTTP 协议

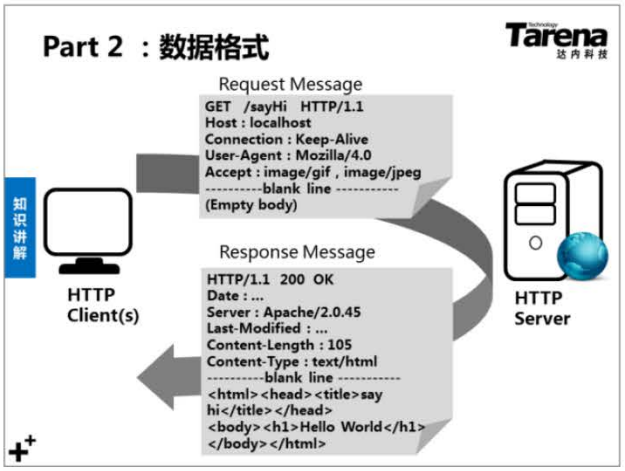
2.1.1. 【HTTP 协议】什么是 HTTP 协议



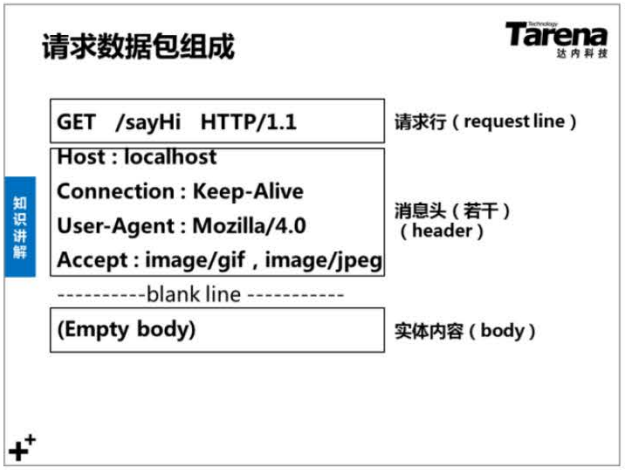
2.1.2. 【HTTP 协议】Part1：如何通信



2.1.3. 【HTTP 协议】Part2：数据格式



2.1.4. 【HTTP 协议】请求数据包组成



请求数据包组成（续1）

解详识知

- 请求行：请求方式+请求资源路径+协议版本
- 消息头（若干）：消息头是一些键值对，一般由w3c定义。通信的双方通过消息头来传递一些特定的含义，比如，浏览器可以发送User-Agent消息头告诉Web服务器，浏览器的类型和版本。大部分为自动生成，某些时候自己需要添加消息头。
- 实体内容：只有当请求方式为post时，实体内容才会有数据。（即请求参数）

2.1.5. 【HTTP 协议】响应数据包组成

响应数据包组成

解详识知

```
HTTP/1.1  200  OK
Date : ...
Server : Apache/2.0.45
Last-Modified : ...
Content-Length : 105
Content-Type : text/html
----------blank line -----------
<html><head><title>say hi</title>
</head><body><h1>Hello World
</h1></body></html>
```

状态行（status line）

消息头（若干）（header）

实体内容

响应数据包组成（续1）

解详识知

- 状态行：协议类型+版本+状态码+状态描述
- 消息头（若干）：Web服务器返回一些消息头给浏览器。例如返回Content-Type消息头，告诉浏览器服务器返回的数据类型和字符集。
- 实体内容：程序处理的结果

2.2. Servlet 如何处理 HTTP 协议

2.2.1. 【Servlet 如何处理 HTTP 协议】如何控制通信数据

如何控制通信数据

- 当Web容器收到一个HTTP请求时，通信数据由Web容器负责封装和提供，这些信息被解释为两个对象
- 与请求数据对应的是HttpServletRequest类型的对象
- 与响应数据对应的是HttpServletResponse类型的对象

解详识知

2.2.2. 【Servlet 如何处理 HTTP 协议】HttpServletRequest 对象

HttpServletRequest对象

- HttpServletRequest对象代表客户端的请求，当客户端通过HTTP协议访问服务器时，请求中的所有消息都封装在这个对象中，通过这个对象的相关方法可以获取请求数据。
- 作用
  - 读取和写入HTTP请求数据（请求行、消息头等）
  - 取得和设置Cookies
  - 取得路径信息
  - 标识HTTP会话
  - 实现请求转发

解详识知

2.2.3. 【Servlet 如何处理 HTTP 协议】HttpServletResponse 对象

HttpServletResponse对象

- HttpServletResponse对象代表提供给客户端的响应，封装了HTTP的响应数据。通过这个对象可设置状态行、消息头、实体内容。
- 作用：
  - 设置对客户端的输出内容
  - 设置响应的状态码
  - 设置浏览器的解码方式
  - 设置Cookies
  - 实现重定向

解详识知

经典案例

1. 安装和配置 Tomcat

- **问题**

分别在 Linux 和 windows 系统下实现 Tomcat 的安装和配置。

- **步骤**

在 Linux 系统下安装、配置 Tomcat。

步骤 0：安装 JDK 及配置 JAVA_HOME、PATH 等环境变量

到 Oracle 官方网站下载 JDK 的 Linux 专用安装程序后，执行安装，假定安装路径为“usr/local/jdk1.7”。安装结束后配置环境变量的过程如图-1 所示，在终端中输入“gedit /home/soft01/.bashrc”，其代表的含义是使用 gedit 编辑 soft01 这个用户下的环境变量。

图 - 1

在文件中输入图-2 所示的第 9,10,11 行代码：

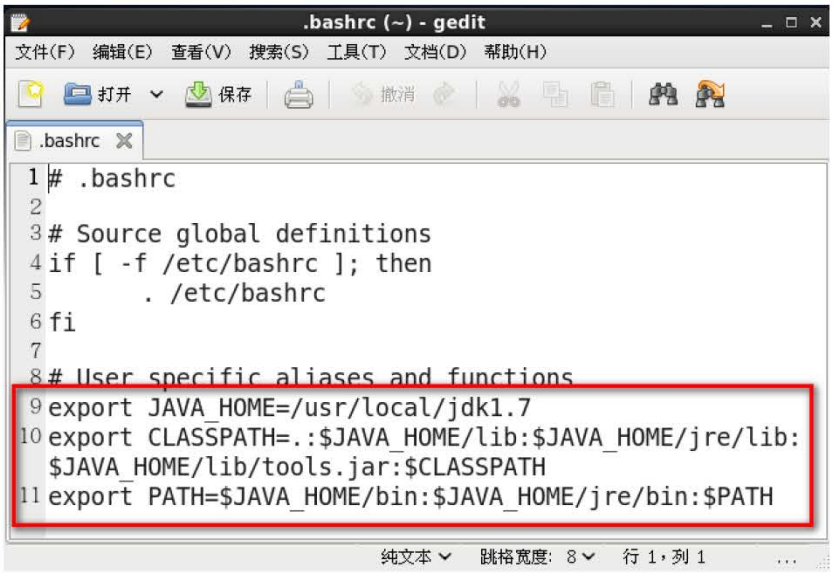


图 - 2

其含义为：

1）新建“JAVA_HOME”环境变量，记录的是 jdk1.7 安装的根路径；

2）修改“CLASSPATH”环境变量，末尾添加原有 CLASSPATH 的值，以防覆盖；

3）修改“PATH”环境变量，记录“jdk1.7/bin“这个路径，末尾添加原有 PATH 的值，以防覆盖。

具体完整代码为：

```
export JAVA_HOME=/usr/local/jdk1.7
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/jre/lib:$JAVA_HOME/lib/
tools.jar:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH
```

在终端中输入“java -version ”，查看是否配置成功，结果如图-3 所示：

图 - 3

步骤一：下载及解压 Tomcat

在 tomcat.apache.org 网站下载 Tomcat7.0 程序的安装包。Linux 系统选择 tar.gz 格式。如图-4 所示：

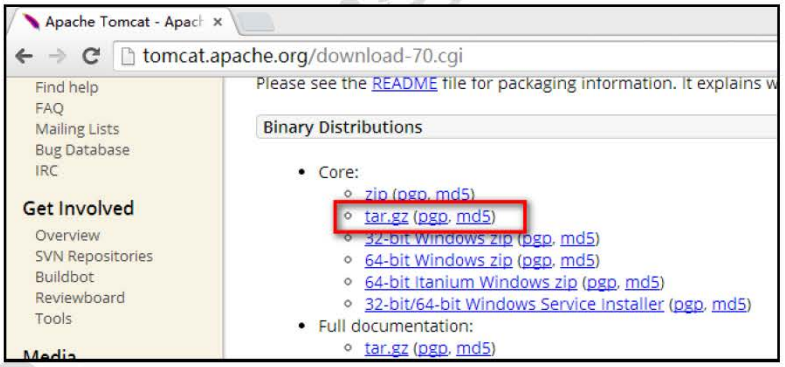


图 - 4

下载后的文件如图-5 所示：

图 – 5

将此文档解压到 “/home/soft01/” 或其子目录下，假定本次解压位置为 “/home/soft01/java”，如图-6 所示：

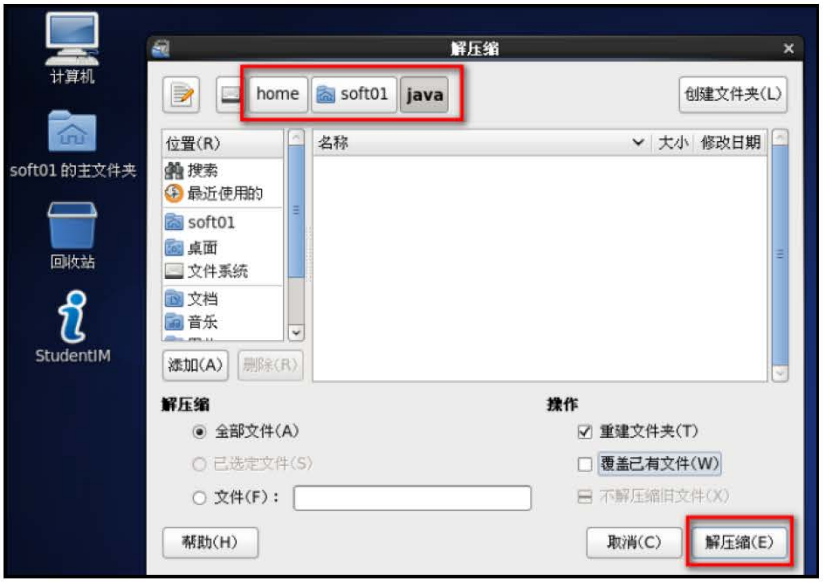


图 - 6

解压后的结果如图-7 所示，保证 tomcat-apache-7.0.6 文件夹下面包含 Tomcat 文件。

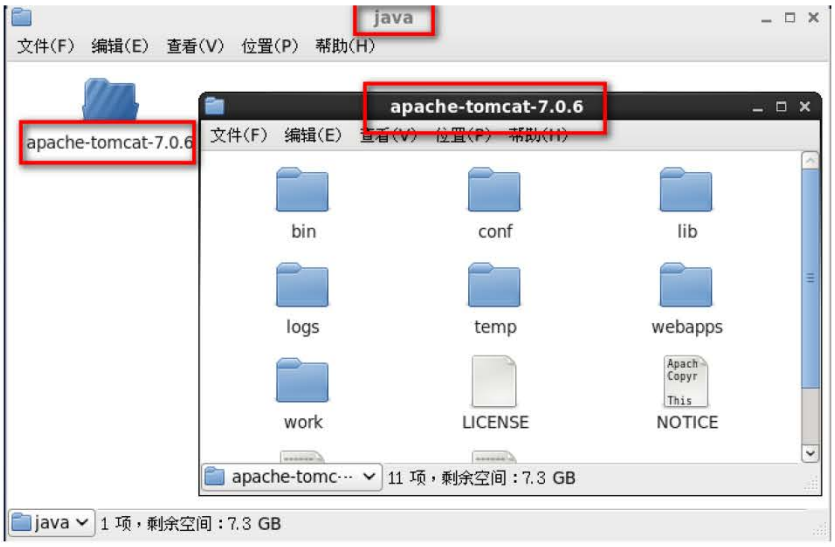


图 - 7

步骤二：启动 Tomcat

在终端中输入如下命令：“cd /home/soft01/java/apache-tomcat-7.0.6/bin” 进入到 tomcat 的安装目录的 bin 下面。如图-8 所示，可以通过 “pwd” 命令查看当前位置是否正确。

图 - 8

继续在终端中输入“sh startup.sh”命令，如图-9 所示：

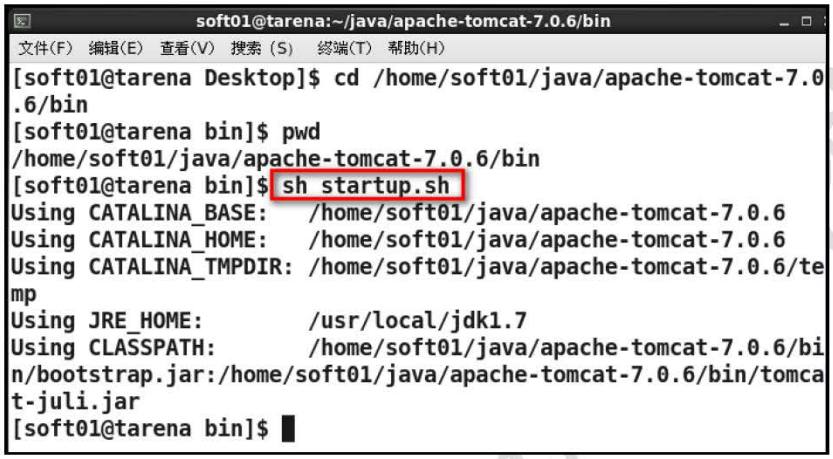


图 - 9

为检验 Tomcat 的确已经启动成功，打开浏览器输入 http://localhost:8080 查看结果。如图-10 所示为启动成功。

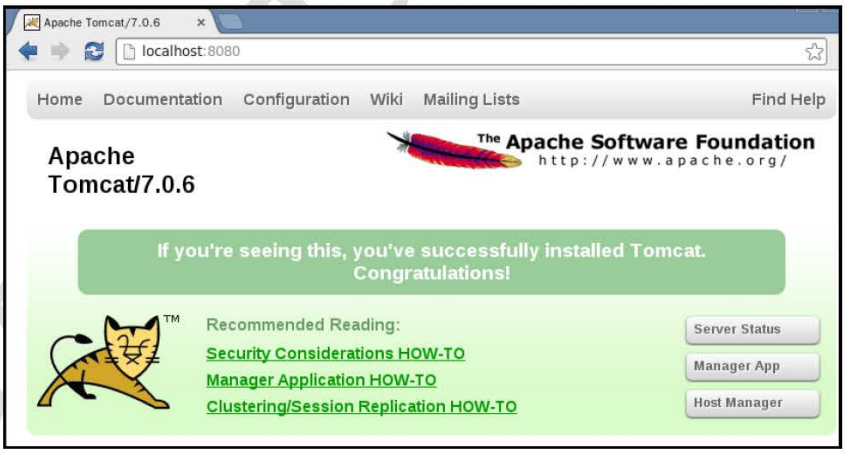


图 - 10

步骤三：关闭 Tomcat

在终端中输入“sh shutdown.sh”命令，停止 Tomcat 服务。如图-11 所示：

```
soft01@tarena:~/java/apache-tomcat-7.0.6/bin
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
mp
Using JRE_HOME:        /usr/local/jdk1.7
Using CLASSPATH:       /home/soft01/java/apache-tomcat-7.0.6/bi
n/bootstrap.jar:/home/soft01/java/apache-tomcat-7.0.6/bin/tomca
t-juli.jar
[soft01@tarena bin]$ sh shutdown.sh
Using CATALINA_BASE:   /home/soft01/java/apache-tomcat-7.0.6
Using CATALINA_HOME:   /home/soft01/java/apache-tomcat-7.0.6
Using CATALINA_TMPDIR: /home/soft01/java/apache-tomcat-7.0.6/te
mp
Using JRE_HOME:        /usr/local/jdk1.7
Using CLASSPATH:       /home/soft01/java/apache-tomcat-7.0.6/bi
n/bootstrap.jar:/home/soft01/java/apache-tomcat-7.0.6/bin/tomca
t-juli.jar
[soft01@tarena bin]$
```

图 - 11

再次在浏览器中输入“http://localhost:8080”查看页面效果，则显示“该页无法显示”的提示。

在 Windows 系统下安装、配置 Tomcat。

步骤 0：安装 JDK 及配置 JAVA_HOME、PATH 等环境变量

到 Oracle 官方网站下载 JDK 的 Windows 专用安装程序后，执行安装，假定安装路径为“d:\java\jdk1.7”，则配置环境变量过程如下：

“我的电脑”右键“属性” → 高级系统设置 → 环境变量。如图-12，图-13，图-14所示：

图 - 12

图 - 13

图 – 14

对“系统变量”做以下三个操作：

新建系统变量，变量名为“JAVA_HOME”，变量值为“D：\java\jdk1.7”。即 JDK 的安装路径的根路径。如图-15 所示：

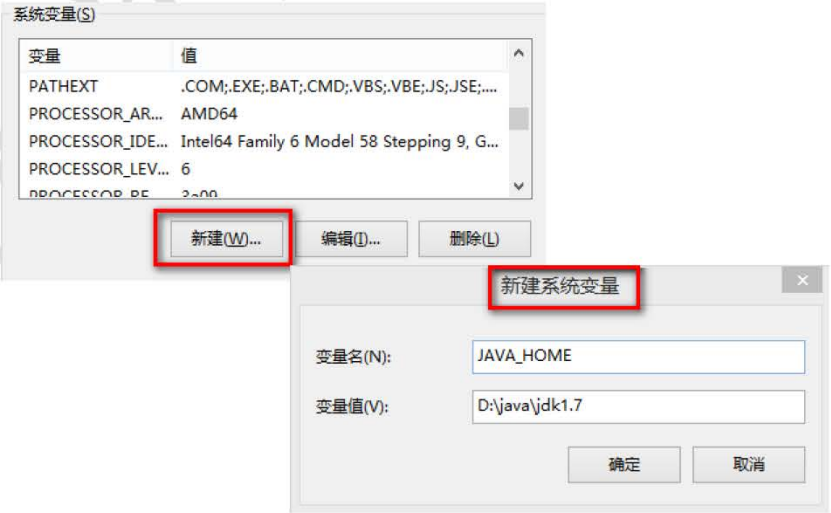


图 – 15

新建系统变量，变量名为“CLASSPATH”，变量值为“.;%JAVA_HOME%\lib;%JAVA_HOME\lib\tools.jar”。第一个为“.”点，代表当前路径，即在当前路径下寻找类，后面为安

装的 jdk 中的 jar 文件的路径，中间用 “；“分号隔开（Linux 系统用”：“冒号分隔）如图-16 所示：

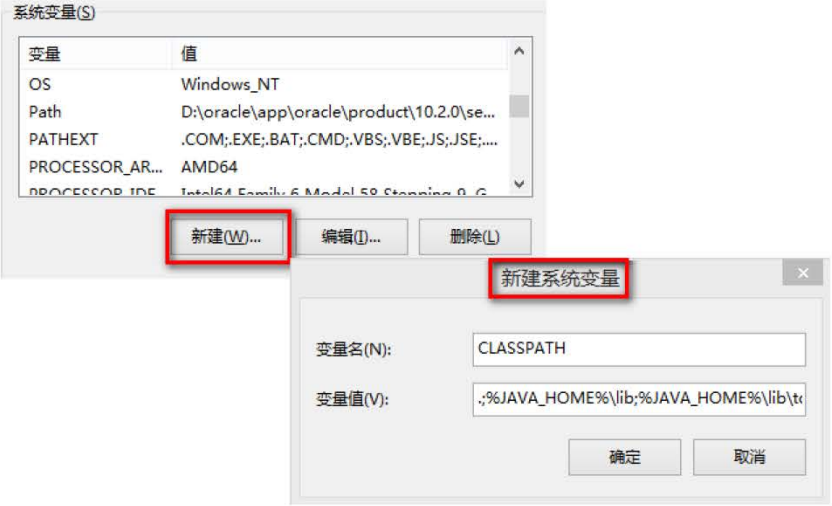


图 - 16

修改原有系统变量 “Path ”，将光标移动到变量值的最前面（注意，一定不要删除原有的变量值）,添加” %JAVA_HOME%\bin;”。一定要以分号结束，作为与原有值的分隔标识。安装的 Tomcat 的 bin 目录下有很多可执行程序和命令，记录这个地址后，不管在任何目录下输入 java 和 javac 命令，系统都能找到对应的命令并执行。

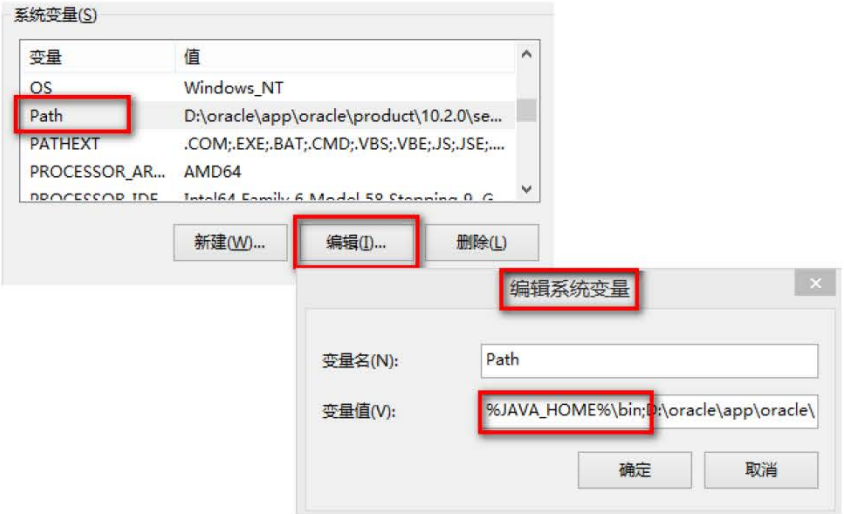


图 - 17

通过以上三个环境变量的设置以后，为了验证 JDK 是否配置成功，则通过在命令行中输入命令来检测。在“运行”窗口中输入“cmd”进入到命令行界面，输入“java -version”查看输出结果，如果提示“java version 1.7.X_XX”则代表配置成功。如图-18 所示：

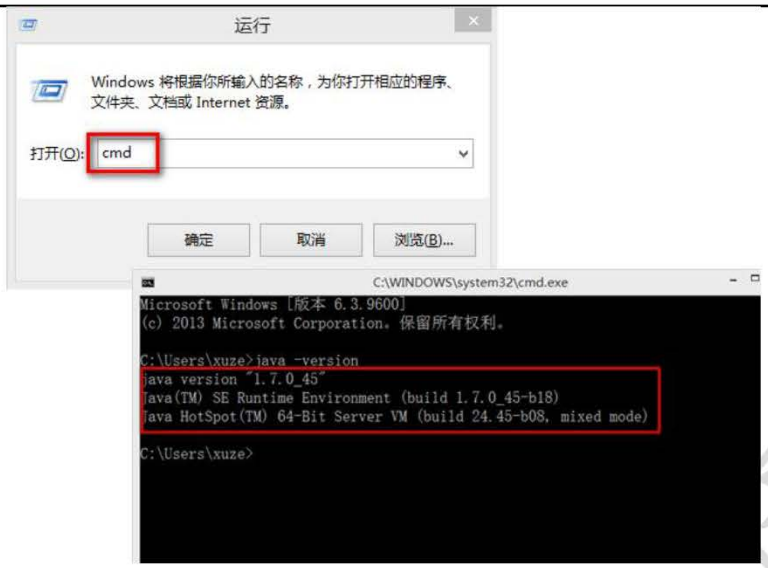


图 - 18

步骤一：下载及解压 Tomcat

在 apache 网站的指定下载页面，选择适合 Windows 操作系统的 Tomcat 版本后下载。如图-19 所示：

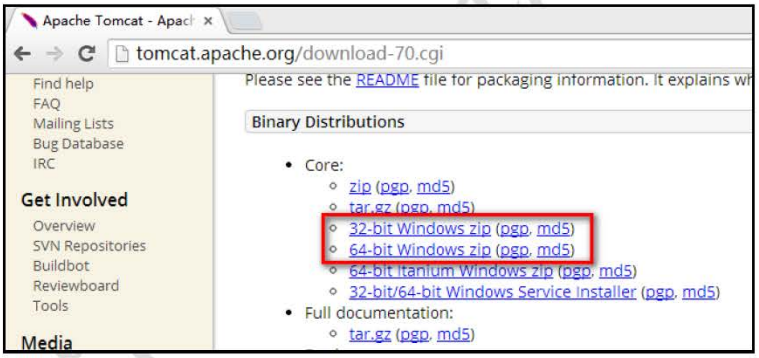


图 - 19

将下载的文件解压缩到指定路径。假定本次解压路径为“d:\java\tomcat7”，结果如图-20 所示。注意“tomcat7”下面不再包含子文件夹，直接包含 tomcat 的各文件。

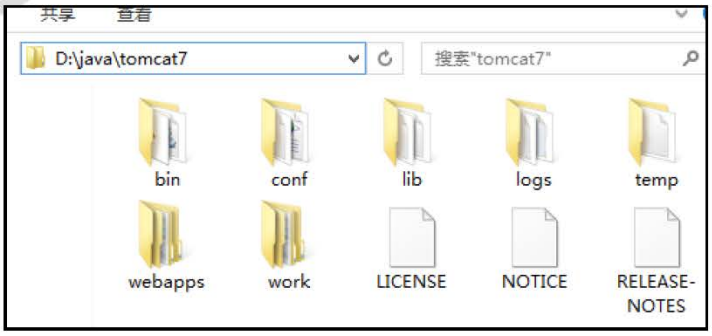


图 -20

步骤二：启动 Tomcat

使用命令行工具，输入“d:”切换到 D 盘，再输入“cd d:\java\tomcat7\bin“进入到 tomcat 文件夹中的 bin 目录下，输入“startup”命令即可启动 Tomcat。如图-21,22 所示：

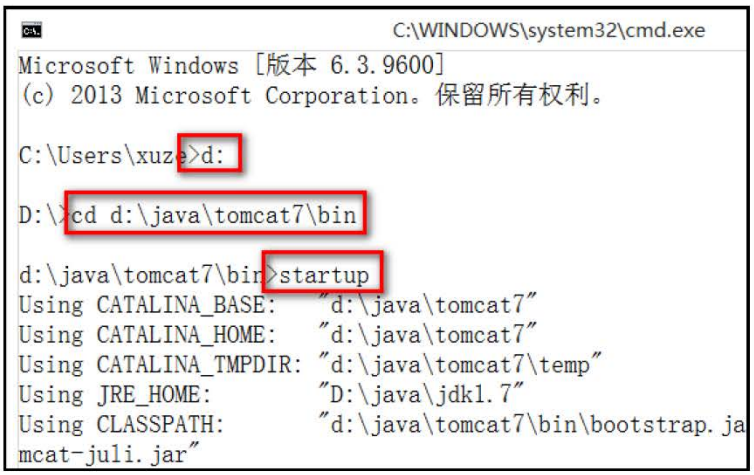


图 – 21

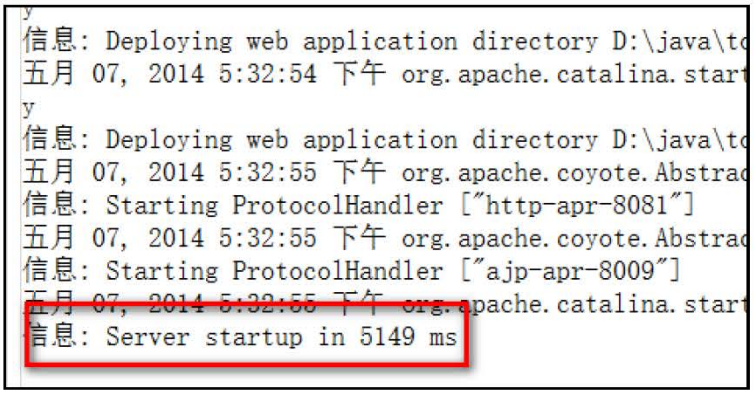


图 - 22

打开浏览器输入”http://localhost:8080“验证启动成功。如图-23 所示：

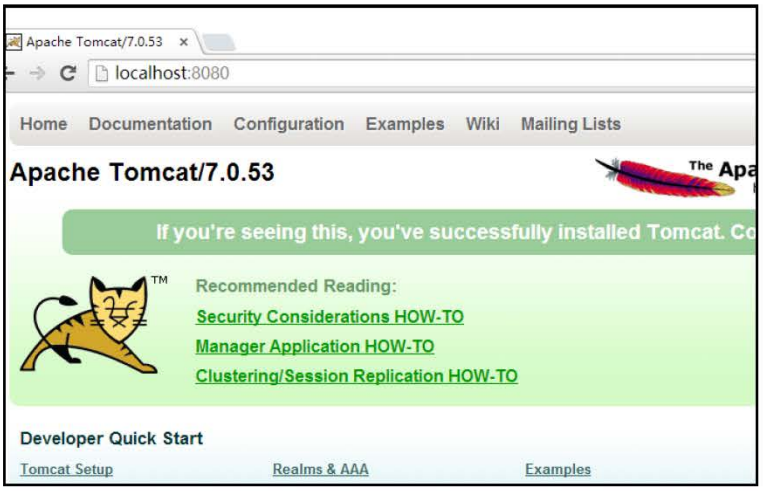


图 - 23

步骤三：关闭 Tomcat

在命令行中输入“shutdown”命令可以关闭 Tomcat。如图-24 所示：

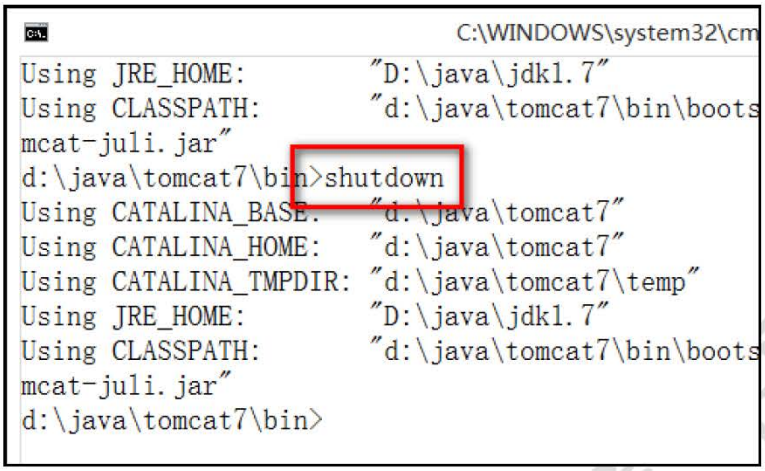


图 - 24

2. 手动开发第一个 Web 项目

- **问题**

使用文本编辑工具开发第一个 Servlet 程序。

- **步骤**

开发 Servlet 程序的步骤如下

1. 编写 java 源文件，文件中的类必须实现 Servlet 接口或继承 HttpServlet 类
2. 编译 java 源文件，生成 class 文件
3. 打包，将编译后的文件按照规定目录结构进行整理
4. 部署，将目录结构复制到 Web 容器的指定位置
5. 启动 Web 容器，输入符合规范的地址访问 Servlet

不管是在 Linux 还是 Windows 系统下，按照以上五步即可完成 Web 程序的开发，区别只在于系统不同则实现对应步骤的命令不同。

Linux 下开发第一个 Web 程序。

步骤一：编写源文件

新建一个 HelloServlet.java 的文件，假定保存路径为 “/home/soft01/java”。如图-25 所示：

达内IT培训集团

图 – 25

文件的代码如图-26 所示：

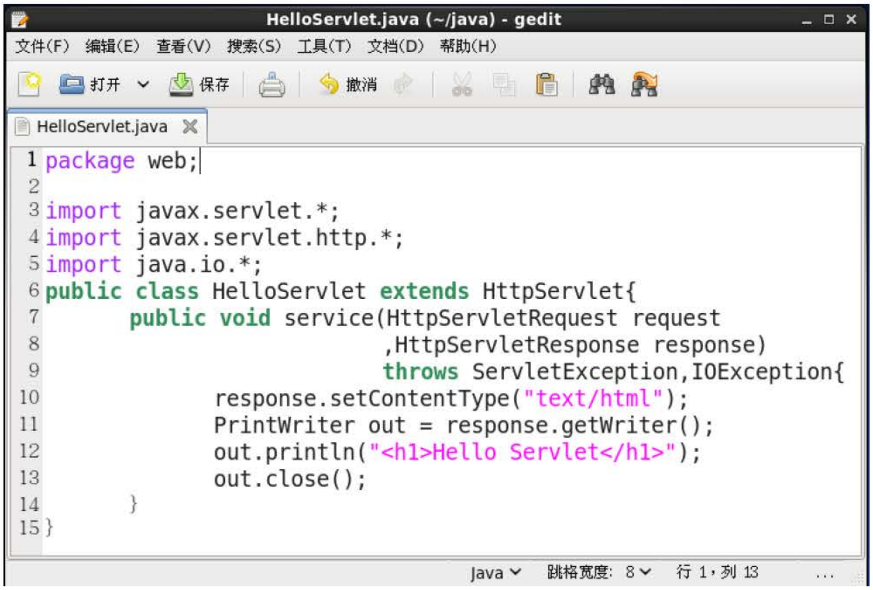


图 - 26

步骤二：编译 HelloServlet.java 文件

进入终端，输入“cd /home/soft01/java”命令，切换当前目录到 HelloServlet.java 文件所在的位置；输入“javac –d . HelloServlet.java”命令进行编译，-d 参数代表编译结果的保存位置，“.”点符号代表当前目录，中间都有空格隔开，即编译 HelloServlet.java 文件，并将结果保存在当前目录；由于 java 源文件中使用了 javax.servlet.*以及 javax.servlet.http.*这两个包中的类，而这些类存在于我们解压的 Tomcat 目录中 lib 下的 servlet-api.jar 文件中，因此为了能够正确编译，需要在 javac 命令中添加“-cp”参数，说明一下这个 servlet-api.jar 文件所在的位置，所以 javac 命令调整为如下内容：

“javac –cp /home/soft01/java/apache-tomcat-7.0.6/lib/serlvet-api.jar –d . HelloServlet.java”

查看编译结果，会发现在当前目录中多了一个名字叫 web 的文件夹（源代码中有 package web 这句代码）及文件夹下的 HelloServlet.class 文件。

编译过程及结果如图-27，图-28 所示：

```
soft01@tarena:~/java
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[soft01@tarena Desktop]$ cd /home/soft01/java
[soft01@tarena java]$ pwd
/home/soft01/java
[soft01@tarena java]$ javac -cp /home/soft01/java/apache-
tomcat-7.0.6/lib/servlet-api.jar -d . HelloServlet.java
[soft01@tarena java]$
```

图 – 27

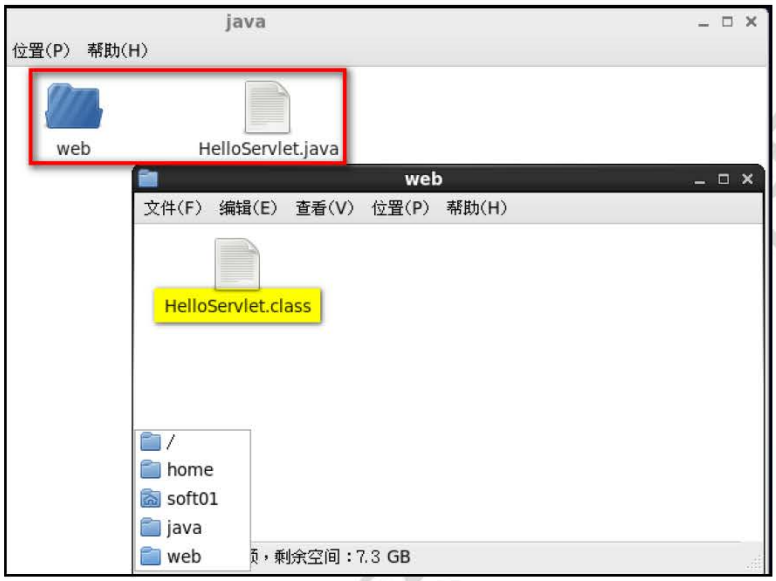


图 – 28

步骤三：打包

打包即将工程各组件、文件以规定的目录结构进行整理。首先是在“/home/soft01/java”目录下创建整个应用的一个文件夹，暂定名称叫 firstweb(可变)，firstweb 内包含一个叫做“WEB-INF”的文件夹，且名字不可变，“WEB-INF”下面包含一个叫做“classes”的文件夹，将刚刚编译好包含 HelloServlet.class 的 web 文件夹整体拷贝到“classes”文件夹下；同时，在“WEB-INF“文件夹下与“classes”同级别，添加一个名字叫做“web.xml”(名称不可变)的文件。具备了“WEB-INF”、“classes”、“web.xml”之后，打包步骤基本完成。生成的目录结构如图-29 所示：

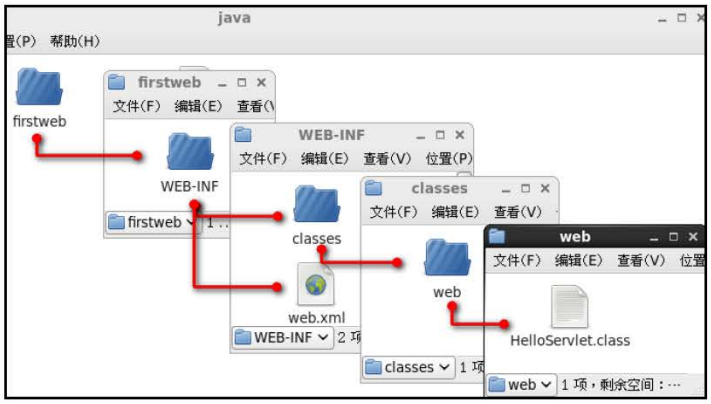


图 – 29

编写web.xml文件时，内容可以参考tomcat安装目录下webapps/examples/WEB-INF/web.xml 文件。拷贝文件第一行，<web-app>、<servlet>、<servlet-mapping>节点即可，修改文件内容，与图-30保持一致。其中，两个servlet-name的名字要一致，servlet-class 填写的是编译后的 HelloServlet 这个类的完整名称，带上包名。url-pattern内的值一定更要以斜杠“/”开头。

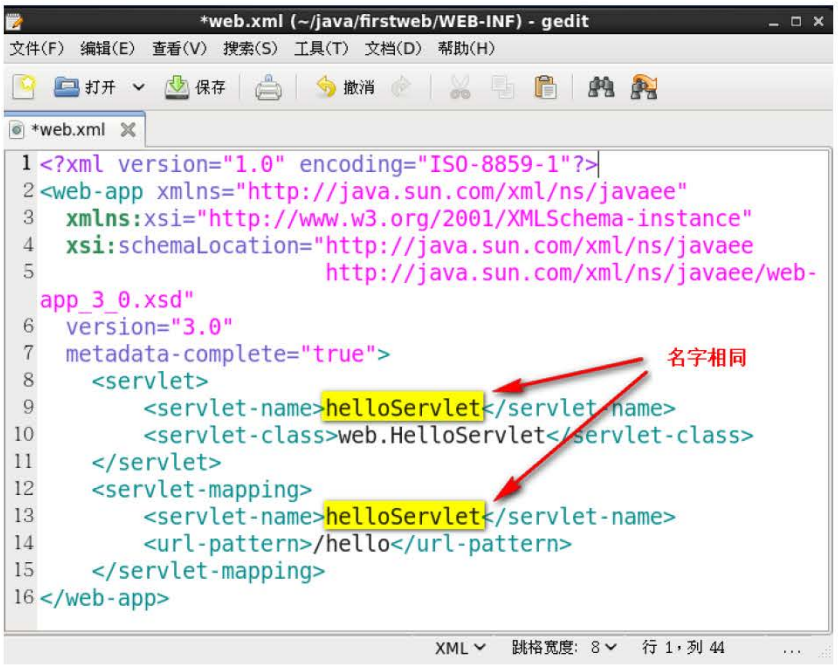


图 - 30

步骤四：部署

部署工程就是将整个应用拷贝到 tomcat 的 webapps 这个文件夹下面。拷贝完的结果如图-31所示：

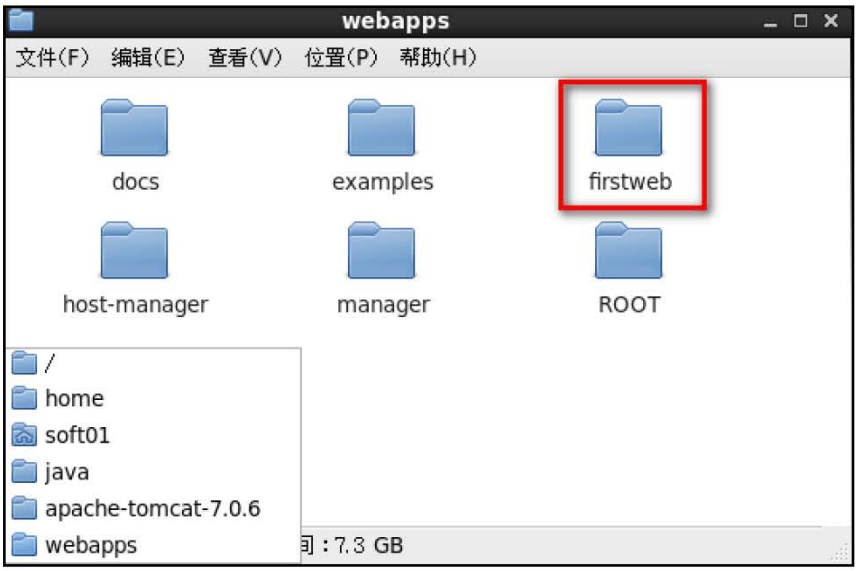


图 - 31

步骤五：启动容器，访问 Servlet 查看结果

在终端中，输入“cd /home/soft01/java/apache-tomcat-7.0.6/bin”命令，切换到 tomcat 的 bin 目录；再输入“sh startup.sh”命令，启动 Tomcat。

打开浏览器，输入“http://localhost:8080/firstweb/hello”回车，查看页面输出效果图-32 所示：

图 – 32

Windows 下开发第一个 Web 程序。

步骤一：编写源文件

新建 java 源文件，假定保存路径为“d:\java”，内容如图-33 所示：

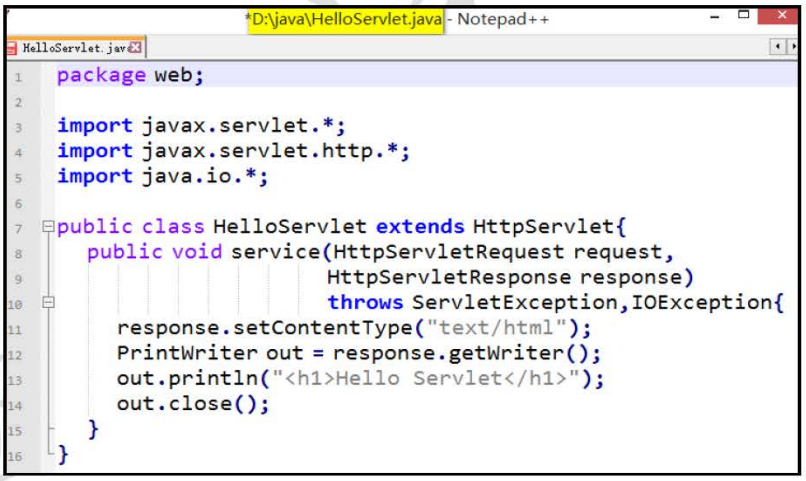


图 - 33

步骤二：编译 HelloServlet.java 文件

进入到命令行工具，输入命令“d:”切换盘符，输入命令“cd java”进入到 HelloServlet.java 源文件所在的目录，输入命令“javac -cp d:\java\tomcat7\lib\servlet-api.jar -d . HelloServlet.java”编译 java 文件到当前目录下。命令及输出结果如图-34，图-35 所示：

图 – 34

图 – 35

步骤三：打包

按照如下结构进行整理应用的目录结构。

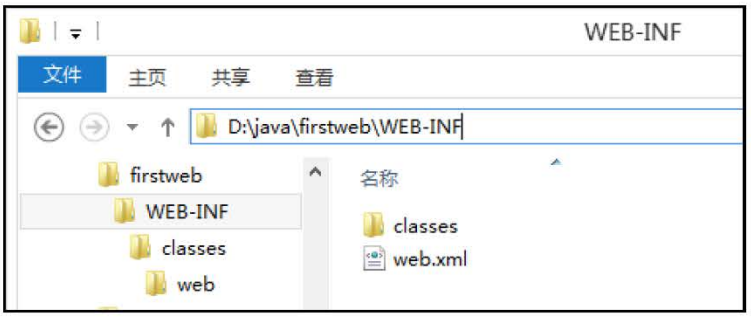


图 – 36

详细结构可以参考图-29。web.xml 文件内容参考图-30。

步骤四：部署

将整个 firstweb 文件夹拷贝到 tomcat 安装路径下的 webapps 文件夹下。如图-37 所示：

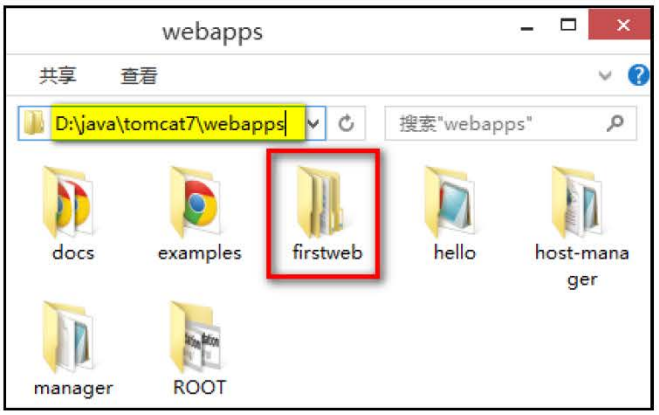


图 - 37

步骤五：启动容器，访问 Servlet 查看结果

进入到命令行工具，输入“d:”切换盘符，输入“cd java\tomcat7\bin”进入到 tomcat 的工具目录，输入“startup”回车，启动 tomcat。如图-38，图-39 所示：

```
C:\Users>d:

D:\>cd java\tomcat7\bin

D:\java\tomcat7\bin>startup
Using CATALINA_BASE:   "D:\java\tomcat7"
Using CATALINA_HOME:   "D:\java\tomcat7"
Using CATALINA_TMPDIR: "D:\java\tomcat7\temp"
Using JRE_HOME:        "D:\java\jdk1.7"
Using CLASSPATH:       "D:\java\tomcat7\bin\bootstr
mcat-juli.jar"
```

图 - 38

```
五月 08, 2014 12:29:56 下午 org.apache.coyote.AbstractProt
信息: Starting ProtocolHandler ["http-apr-8081"]
五月 08, 2014 12:29:56 下午 org.apache.coyote.AbstractProt
信息: Starting ProtocolHandler ["ajp-apr-8009"]
五月 08, 2014 12:29:56 下午 org.apache.catalina.startup.Ca
信息: Server startup in 3517 ms
```

图 - 39

打开浏览器，输入“http://localhost:8080/firstweb/hello” 查看运行结果。

图 - 40

- **完整代码**

本案例的完整代码如下所示：

HelloServlet.java 代码：

```
package web;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HelloServlet extends HttpServlet{
 public void service(HttpServletRequest request,
                                HttpServletResponse response)
                                throws ServletException,IOException{
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     out.println("<h1>Hello Servlet</h1>");
```

```
        out.close();
    }
}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app 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-app 3 0.xsd"
  version="3.0"
  metadata-complete="true">
    <servlet>
      <servlet-name>helloServlet</servlet-name>
      <servlet-class>web.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>helloServlet</servlet-name>
      <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```

3. 使用工具开发 Web 项目

- **问题**

使用 MyEclipse 进行 Web 应用的开发。

- **步骤**

在第一次使用 MyEclipse 进行 Web 应用的开发之前，需要将 MyEclipse 与解压后的 Tomcat 进行关联，也就是完成服务器的配置。

步骤一：为 MyEclipse 添加服务器配置

依据图-41 至图-45 所示步骤完成服务器配置。

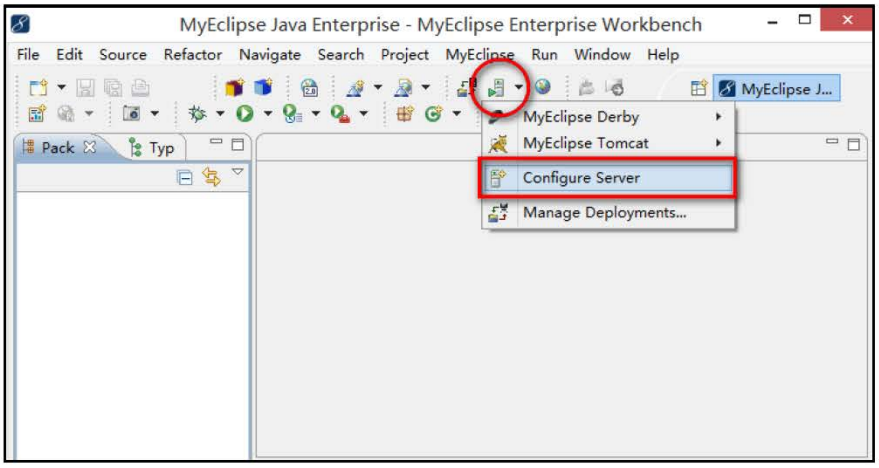


图 - 41

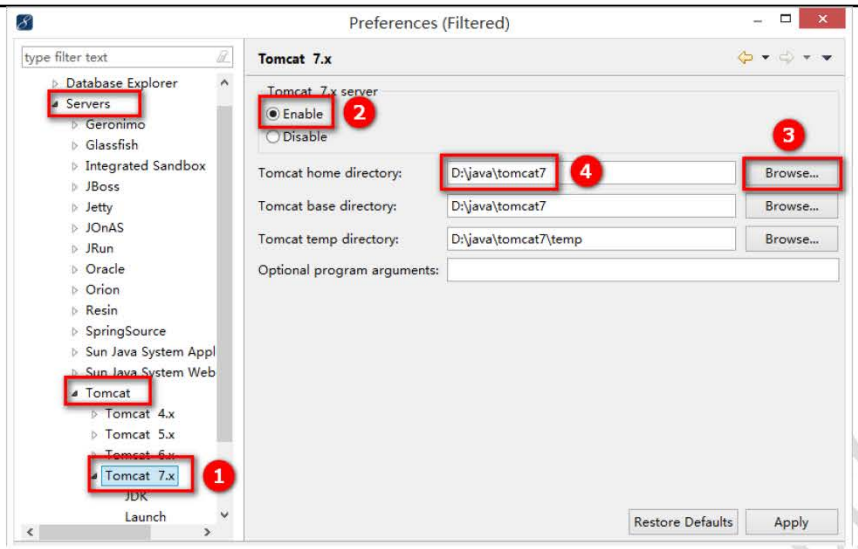

Preferences (Filtered)
type filter text
Database Explorer
Servers
Geronimo
Glassfish
Integrated Sandbox
JBoss
Jetty
JOnAS
JRun
Oracle
Orion
Resin
SpringSource
Sun Java System Appl
Sun Java System Web
Tomcat
Tomcat 4.x
Tomcat 5.x
Tomcat 7.x
JDK
Launch
Tomcat 7.x
Tomcat 7.x server
Enable
Disable
Tomcat home directory:
D:\java\tomcat7
Browse...
Tomcat base directory:
D:\java\tomcat7
Tomcat temp directory:
D:\java\tomcat7\temp
Optional program arguments:
Restore Defaults
Apply
1
2
3
4

图 – 42

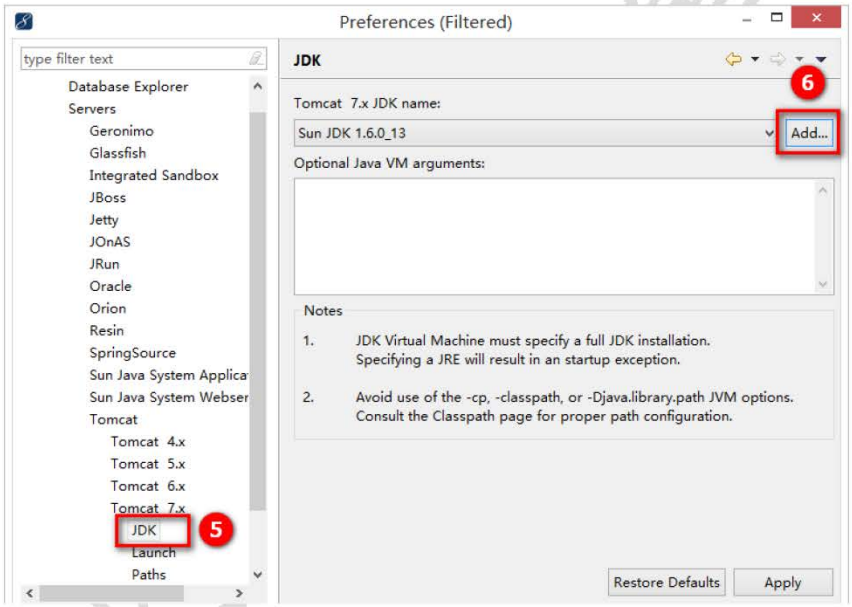

Preferences (Filtered)
type filter text
Database Explorer
Servers
Geronimo
Glassfish
Integrated Sandbox
JBoss
Jetty
JOnAS
JRun
Oracle
Orion
Resin
SpringSource
Sun Java System Applica
Sun Java System Webser
Tomcat
Tomcat 4.x
Tomcat 5.x
Tomcat 6.x
Tomcat 7.x
JDK
Launch
Paths
JDK
Tomcat 7.x JDK name:
Sun JDK 1.6.0_13
Add...
Optional Java VM arguments:
Notes
1. JDK Virtual Machine must specify a full JDK installation.
Specifying a JRE will result in an startup exception.
2. Avoid use of the -cp, -classpath, or -Djava.library.path JVM options.
Consult the Classpath page for proper path configuration.
Restore Defaults
Apply
5
6

图 – 43

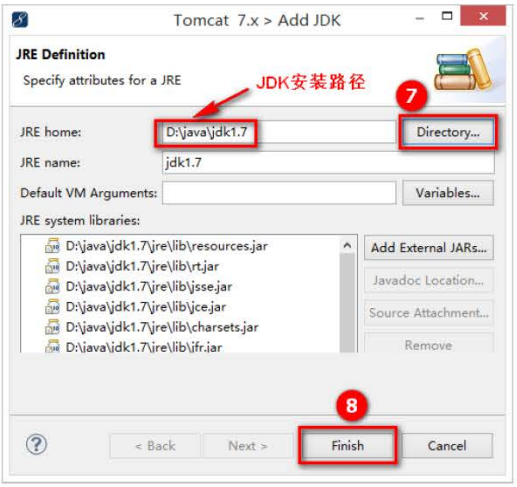

Tomcat 7.x > Add JDK
JRE Definition
Specify attributes for a JRE
JDK安装路径
JRE home:
D:\java\jdk1.7
Directory...
JRE name:
jdk1.7
Default VM Arguments:
Variables...
JRE system libraries:
D:\java\jdk1.7\jre\lib\resources.jar
D:\java\jdk1.7\jre\lib\rt.jar
D:\java\jdk1.7\jre\lib\jsse.jar
D:\java\jdk1.7\jre\lib\jce.jar
D:\java\jdk1.7\jre\lib\charsets.jar
D:\java\jdk1.7\jre\lib\jfr.jar
Add External JARs...
Javadoc Location...
Source Attachment...
Remove
< Back
Next >
Finish
Cancel
7
8

图 – 44

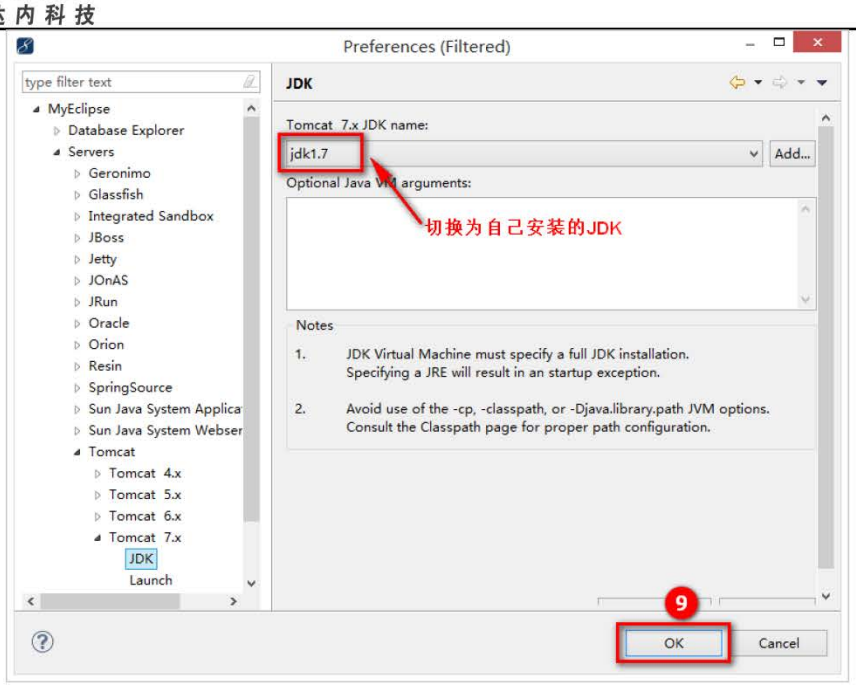


图 – 45

配置成功后结果如图-46 所示：

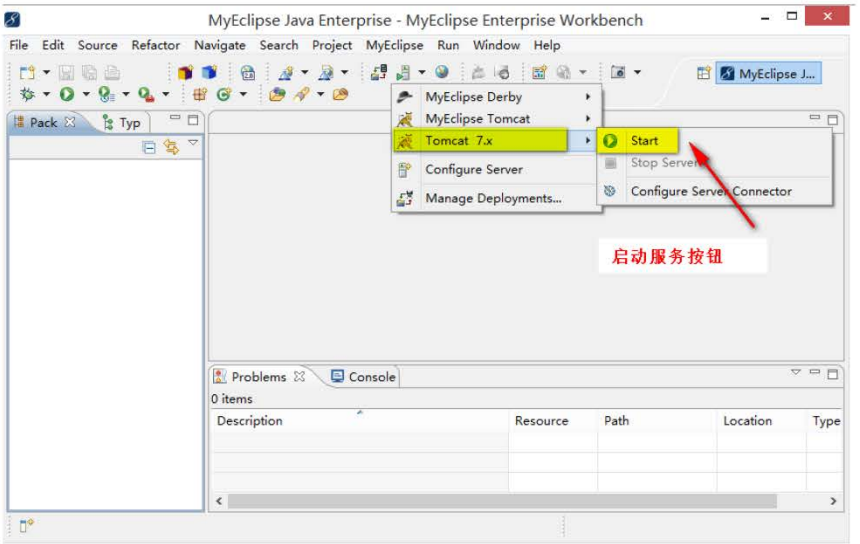


图 – 46

可以通过点击“Start”按钮启动关联到的 Tomcat，观察 Console 输出窗口会出现“Server startup in XXXX ms”的提示信息，代表服务器配置成功并能够正确启动。通过点击“Stop Server”按钮可以停止服务。

步骤二：新建 Web 工程

通过点击菜单 “File” → “New” → “Web Project”新建一个 Web 工程，填写工程名信息如图-47 后点击确定，生成工程。

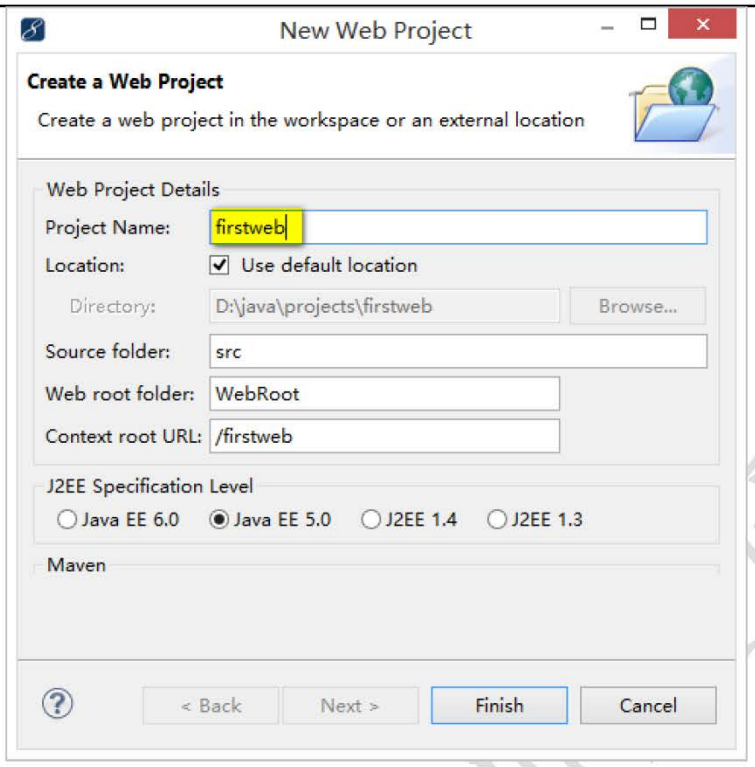


图 - 47

步骤三：创建 java 类

在工程文件的“src”上右键，new 一个 class 文件，填写包名“web”，填写类名“HelloServlet”后点击“Finish”，如图-48 所示：

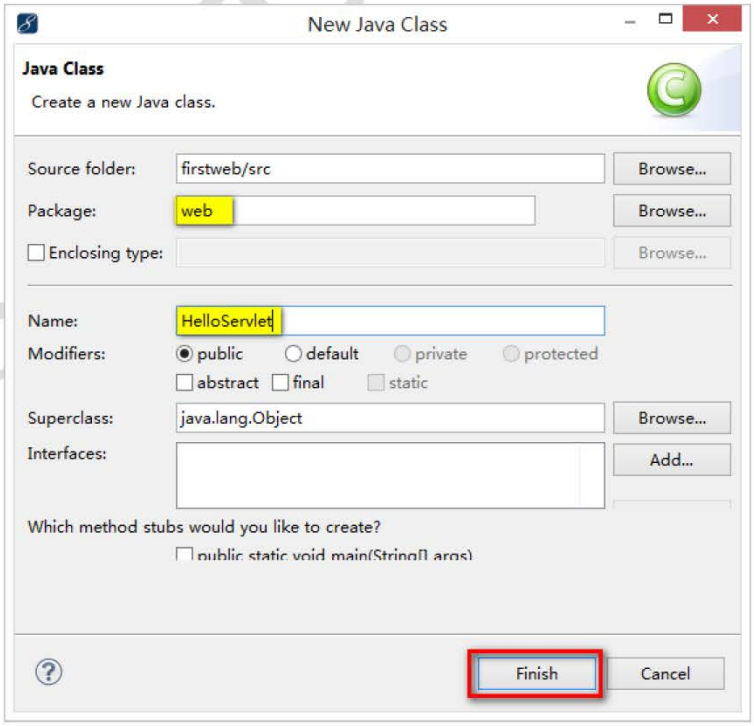


图 – 48

类文件内容参考图-33。

步骤四：配置 web.xml 文件

点击工程中“WEB-INF”节点下的 web.xml 文件，点击屏幕下方的“Source”按钮切换编辑视图，改写文件内容如图-49 所示：

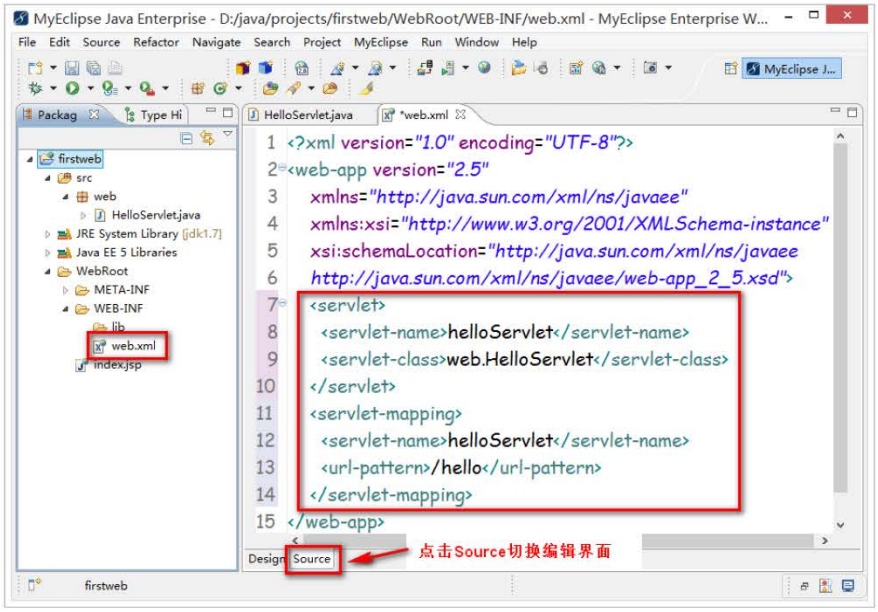


图 – 49

步骤五：部署工程到 Tomcat

点击工具栏中的 图标进入到部署工程的界面，按照图-50 至图-52 所示顺序依次选择要部署的项目、部署到哪个服务器之后点击 OK 按钮完成部署。

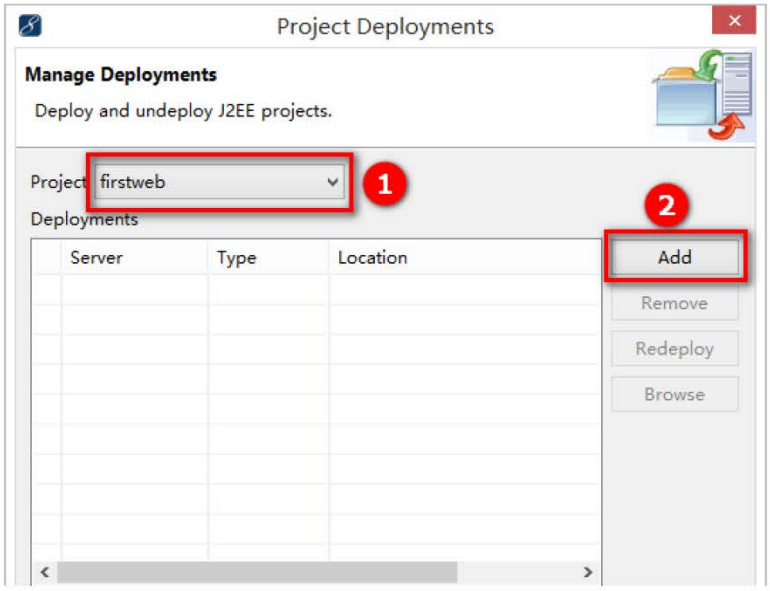


图 – 50

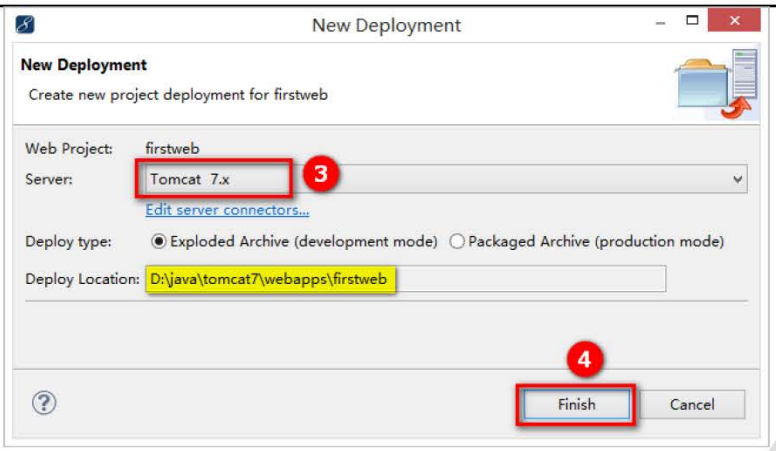


图 – 51

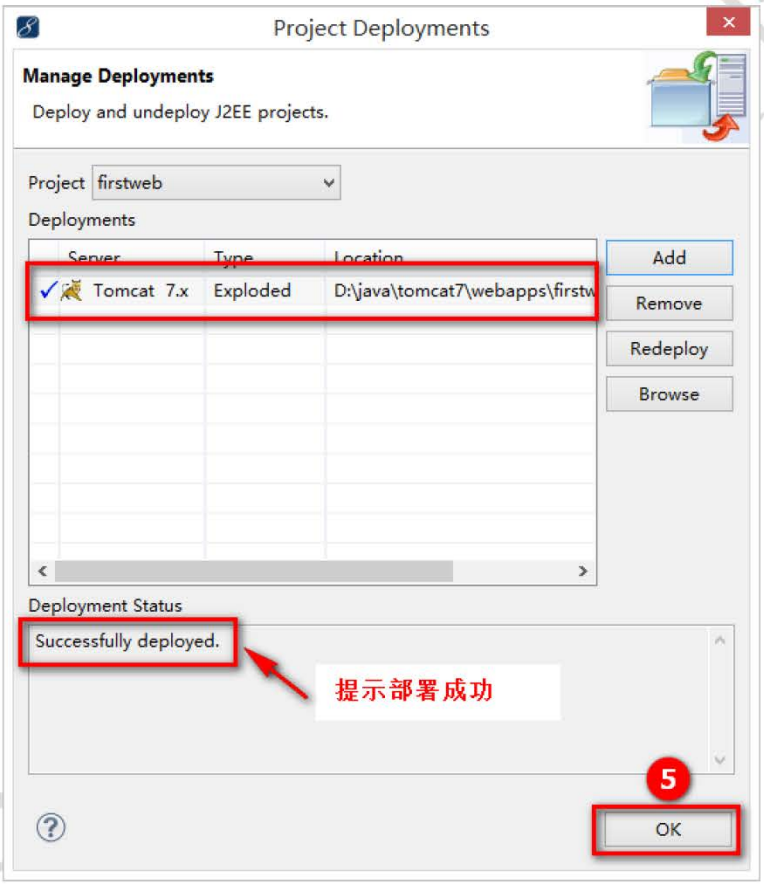


图 – 52

步骤六：启动服务器查看运行结果

点击工具栏中的 服务器图标下面“Tomcat7.X”对应的“Start”按钮启动服务器，观察控制台输出窗口中显示成功启动的信息后，打开浏览器，输入“http://localhost:8080/firstweb/hello “，查看运行结果。同图-40 显示一样。

- **完整代码**

与“手动开发第一个 Web 项目“的完整代码相同。

使用 MyEclipse 开发 Web 项目的过程与手动编写完成的步骤是一样的，只是将一些步骤变成了自动完成，如编译、打包。

- 编写 java 源文件　　手动
- 编译　　**自动**
- 打包　　**自动**
- 部署　　手动
- 启动服务，查看　　手动

4. 输出请求数据包中的内容

- **问题**

获取请求数据包中的消息头中各键值对的名称和值，提交方式，协议，请求资源路径等信息，并输出到控制台中。

- **方案**

客户端发来的请求数据包到达服务器端以后会被解释为 HttpServletRequest 类型的对象，数据包中的数据会被存储到该类型的对象之中，所以借助于该对象提供的 getHeaderNames(),getHeader(),getMethod(),getRequestURI()等方法即可获取这些信息。

- **步骤**

实现此案例按如下步骤进行。

步骤一：新建 web 工程

点击菜单“文件”→ “新建”→ “Web Project”，填写信息如图-53 所示，一般填写完工程名以后其他信息默认不用修改，部署的应用名会与工程名保持一致，但如果修改了部署名称以后，那么部署之后访问时注意资源路径要与修改后的名称保持一致。

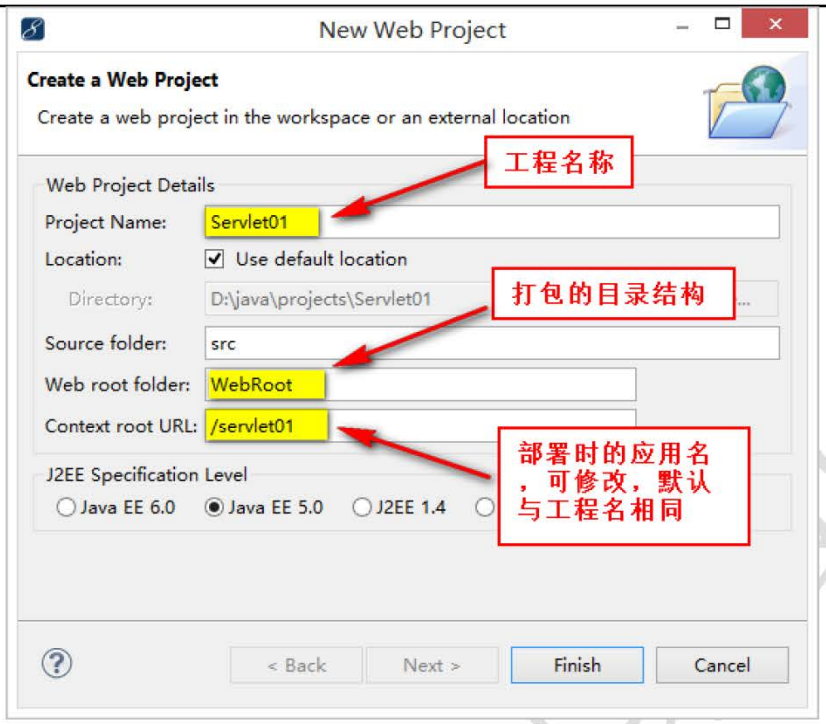


图 - 53

步骤二：新建类 RequestInfo.java

新建一个位于 web 包下面的 RequestInfo.java 的类，使得该类继承自 HttpServlet 抽象类，重写 service（）方法，代码如下。

```
package web;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RequestInfo extends HttpServlet{
    public  void  service(HttpServletRequest  request,HttpServletResponse
response)
        throws ServletException,IOException{


    }
}
```

步骤三：调用 request 对象的方法获取消息头信息

```
package web;

import java.io.IOException;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
public class ReqInfoServlet extends HttpServlet{
 public   void   service(HttpServletRequest   request,HttpServletResponse
response)
        throws ServletException,IOException{
    //获取所有消息头的名称
    Enumeration  e = request.getHeaderNames();
    while(e.hasMoreElements()){
        //遍历 Enumeration 获取每一个消息头的名称
        String headerName  = e.nextElement().toString();
        //输出消息头的名-值对儿信息
        System.out.println(headerName+":"+request.getHeader(headerName));
    }
    System.out.println("请求方式："+request.getMethod());
    System.out.println("请求的协议种类："+request.getProtocol());
    System.out.println("请求资源路径："+request.getRequestURI());
    System.out.println("请求的路径信息："+request.getRequestURL());
    System.out.println("请求的 Servlet 路径："+request.getServletPath());
 }
}
```

步骤四：修改 web.xml 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
 <servlet>
    <servlet-name>reqInfoServlet</servlet-name>
    <!-- 类名要写完整，包含包名，注意大小写 -->
        <servlet-class>web.ReqInfoServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>reqInfoServlet</servlet-name>
        <url-pattern>/reqInfo</url-pattern>
    </servlet-mapping>
</web-app>
```

步骤五：部署

点击工具栏上的“部署”图标，选择要部署的工程及“Add”按钮，选择要部署到的目标服务器后点击 Finish。如图-54：

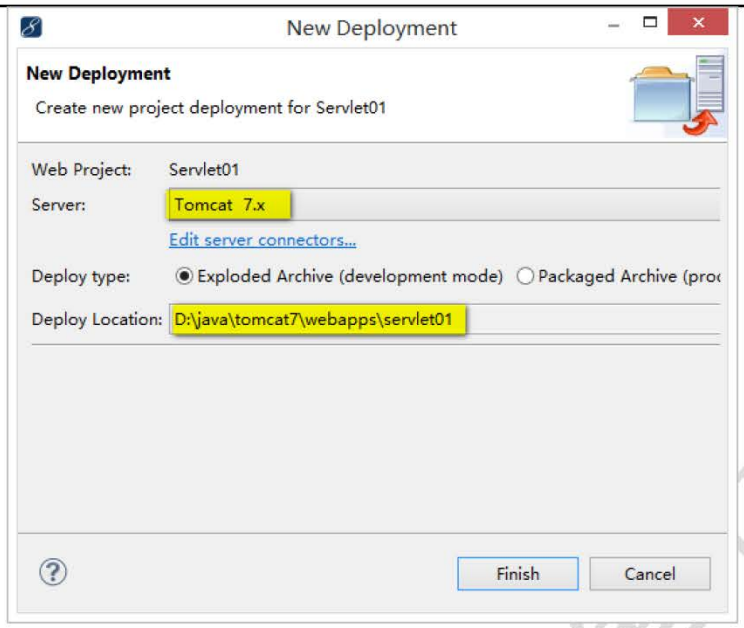


图 – 54

提示部署成功后点击 OK，如图-55：

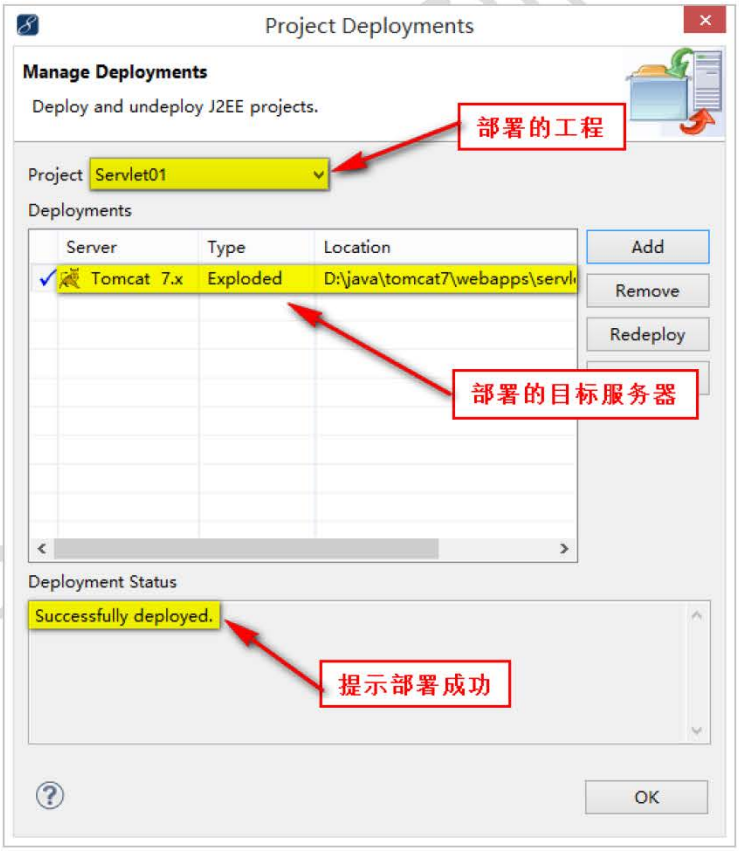


图 – 55

步骤六：启动服务，访问应用

通过点击工具栏中服务器图标，选择 Tomcat7.X，点击 Start 启动服务器。打开浏览器在地址栏中输入“http://localhost:8080/servlet01/reqInfo”查看结果。如图-56：

```
Console
tomcat7Server [Remote Java Application] D:\java\jdk1.7\bin\javaw.exe (2014-5-12 上午11:31:04)
host:localhost:8081
connection:keep-alive
cache-control:max-age=0
accept:text/html,application/xhtml+xml,application/xml;
user-agent:Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWeb
accept-encoding:gzip,deflate,sdch
accept-language:zh-CN,zh;q=0.8,en;q=0.6
请求方式：GET
请求的协议种类：HTTP/1.1
请求资源路径：/servlet01/reqInfo
请求的路径信息：http://localhost:8081/servlet01/reqInfo
请求的Servlet路径：/reqInfo
```

图 - 56

- **完整代码**

ReqInfoServlet.java 完整代码如下：

```
package web;

import java.io.IOException;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ReqInfoServlet extends HttpServlet{
 public   void   service(HttpServletRequest   request,HttpServletResponse
response)
        throws ServletException,IOException{

    //获取所有消息头的名称
    Enumeration  e = request.getHeaderNames();
    while(e.hasMoreElements()){
        //遍历 Enumeration 获取每一个消息头的名称
        String headerName  = e.nextElement().toString();
        //输出消息头的名-值对儿信息
        System.out.println(headerName+":"+request.getHeader(headerName));
    }

    System.out.println("请求方式："+request.getMethod());
    System.out.println("请求的协议种类："+request.getProtocol());
    System.out.println("请求资源路径："+request.getRequestURI());
    System.out.println("请求的路径信息："+request.getRequestURL());
    System.out.println("请求的 Servlet 路径："+request.getServletPath());
 }
}
```

web.xml 文件完成代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app_2_5.xsd">
    <servlet>
        <servlet-name>reqInfoServlet</servlet-name>
        <!-- 类名要写完整，包含包名，注意大小写 -->
            <servlet-class>web.ReqInfoServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>reqInfoServlet</servlet-name>
            <url-pattern>/reqInfo</url-pattern>
        </servlet-mapping>
   </web-app>
```

5. 向客户端输出中文

- **问题**

实现访问 Servlet 之后，服务器端能够返回带有中文信息的页面。

- **方案**

通过 response 对象修改响应数据包中“content-type”消息头，从而设置浏览器的解码方式为支持中文信息的方式。

- **步骤**

步骤一：新建类 ResInfoServlet.java

在 web 包下新建类 ResInfoServlet.java,在 service 方法中添加设置响应头的代码。如下所示：

```
   package web;

   import java.io.IOException;
   import java.io.PrintWriter;

   import javax.servlet.ServletException;
   import javax.servlet.http.HttpServlet;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;

   public class ResInfoServlet extends HttpServlet{
    public   void   service(HttpServletRequest   request,HttpServletResponse
response)
            throws ServletException,IOException{
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.print("能够输出中文信息的Servlet");
        out.close();
    }
   }
```

步骤二：修改 web.xml，配置 Servlet

在 web.xml 文件中增加对 ResInfoServlet 的配置。代码如下所示：

```
        <servlet>
            <servlet-name>resInfoServlet</servlet-name>
            <servlet-class>web.ResInfoServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>resInfoServlet</servlet-name>
            <url-pattern>/resInfo</url-pattern>
        </servlet-mapping>
```

步骤三：重新部署工程，访问应用

重新部署工程后，在浏览器中输入合法地址查看页面结果如下：

图 - 57

- **完整代码**

ResInfoServlet.java 文件的完整代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ResInfoServlet extends HttpServlet{
 public   void   service(HttpServletRequest   request,HttpServletResponse
response)
            throws ServletException,IOException{
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.print("能够输出中文信息的Servlet");
        out.close();
 }
}
```

web.xml 文件的完整代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
 <servlet>
     <servlet-name>reqInfoServlet</servlet-name>
     <!-- 类名要写完整，包含包名，注意大小写 -->
         <servlet-class>web.ReqInfoServlet</servlet-class>
     </servlet>
```

```
        <servlet>
        <servlet-name>resInfoServlet</servlet-name>
            <servlet-class>web.ResInfoServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>reqInfoServlet</servlet-name>
            <url-pattern>/reqInfo</url-pattern>
        </servlet-mapping>
        <servlet-mapping>
            <servlet-name>resInfoServlet</servlet-name>
            <url-pattern>/resInfo</url-pattern>
        </servlet-mapping>
    </web-app>
```

达内IT培训集团

课后作业

1. 简述什么是 Servlet。

2. 简述 Web 工程的目录结构。

3. 简述什么是 HTTP 协议。

4. 简述常用的请求头和响应头的键值对。

5. 下列关于 HTTP 响应代码说法正确的是（ ）。

A．404 错误是因为 service 方法运行时有错误。

B．405 错误有可能是访问时路径拼写有误。

C．500 错误有可能是 web.xml 文件中配置错误。

D．工程没有部署一定会出现 404 错误。

6. 阅读下面的代码，说明序号处代码的含义

```
public class HelloServlet extends HttpServlet{

 public HelloServlet() {
     System.out.println(
             "HelloServlet 的构造器正在执行...");
 }


 public void service(HttpServletRequest request,
                        HttpServletResponse response)//-------------1
     throws ServletException,IOException{
     System.out.println("service 方法正在执行...");
     String name = request.getParameter("name");
     String rs =
         "<span style='color:red;font-size:30px;'>" +
         "hello " + name +
         "</span>";
     response.setContentType("text/html;charset=utf-8");

     PrintWriter out = response.getWriter();//------------------------2
     out.println(rs);//----------------------------------------------3

     out.close();
 }
}
```

7. 编写 Servlet 在网页上显示当前时间。

本题的详细要求如下：

1. 新建一个名为 web01_exec 的 web 工程。
2. 在 web01_exec 工程下，创建一个名为 DateServlet 的 Servlet。
3. 访问该 DateServlet 时，输出浏览器上显示当前系统时间。

达内IT培训集团

Servlet和JSP(上)

Unit02

知识体系.....Page 48

Servlet 工作原理	Servlet 如何获取请求参数	获取请求参数值的方法
		getParameter 方法
		getParameterValues 方法
	请求方式	为什么区分请求方式
		请求方式的种类
		GET 请求方式
		GET 请求方式的特点
		POST 请求方式
		POST 请求方式的特点
	如何处理中文参数	为什么表单提交中文会出现乱码
		解决 POST 方式时的乱码问题
		解决 GET 方式时的乱码问题
	Servlet 如何输出中文	为什么返回的页面会有乱码
		如何解决输出内容的乱码
	Servlet 如何访问数据库	使用 JDBC 技术访问数据库
	Servlet 如何运行	Servlet 运行的详细步骤

经典案例.....Page 54

Servlet 获取请求参数值	获取请求参数值的方法
	getParameter 方法
	getParameterValues 方法
请求方式 GET 和 POST 的区别	为什么区分请求方式
	请求方式的种类
	GET 请求方式
	GET 请求方式的特点
	POST 请求方式
	POST 请求方式的特点
处理 POST 请求中的中文参数值	解决 POST 方式时的乱码问题
处理 GET 请求中的中文参数值	解决 GET 方式时的乱码问题

员工管理——添加员工信息	如何解决输出内容的乱码
员工管理——使用 JDBC 添加员工信息	使用 JDBC 技术访问数据库
员工管理——使用 JDBC 查询所有员工信息	Servlet 运行的步骤

课后作业.....Page 80

达内IT培训集团

1. Servlet 工作原理

1.1. Servlet 如何获取请求参数

1.1.1. 【Servlet 如何获取请求参数】获取请求参数值的方法

获取请求参数值的方法

- 获取提交的 Name-Value 数值
  - getParameter(name)
- 获取提交的 Name-Values 数值
  - getParameterValues(name)

解说内容

1.1.2. 【Servlet 如何获取请求参数】getParameter 方法

getParameter方法

- 常用于传入的参数中，一个名字对应一个值的形式
- String request.getParameter(String paramName)
- 如果参数名写错，会产生null

http://localhost:8080/appName/ hi ? **uname=Kitty**

```
//执行后value的值为" Kitty"
String value = request.getParameter( "uname" );
```

解说内容

1.1.3. 【Servlet 如何获取请求参数】getParameterValues 方法

getParameterValues方法

- String[ ]request.getParameterValues(String paramName)
- 当需要获取参数名相同的多个参数值时使用该方法
- 多用于获取提交的表单中复选框的值。
- 如果该参数名不存在，则返回null

http://localhost:8080/appName/ hi ? city=Beijing& city=Shanghai

```
//执行后values的值为[" Beijing"," Shanghi" ]
String[ ] values = request.getParameter( "city" );
```

解说内容

1.2. 请求方式

1.2.1. 【请求方式】为什么区分请求方式

为什么区分请求方式

- 请求方式是客户端对话服务器时的意向说明，是区分请求种类的关键。
- 不同的请求方式不仅仅在数据传输时会有所不同，在表单提交及服务器端处理时都会采用不同的方式，而区分不同种类的请求方式也会使得浏览器采用不同的缓存方式处理后续请求，从而提升响应速度。

解详识知

1.2.2. 【请求方式】请求方式的种类

请求方式的种类

请求方式	作用
GET	请求指定的资源
POST	向指定的资源提交需要处理的数据
HEAD	要求响应与相应的GET一样，但没有响应体
PUT	上传指定资源
DELETE	删除指定资源
……	

解详识知

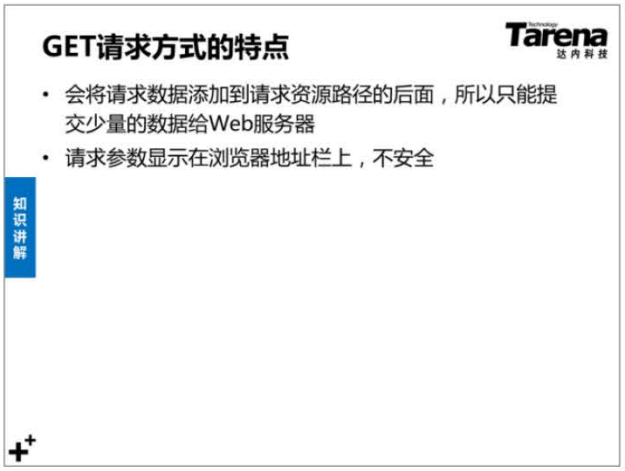
1.2.3. 【请求方式】GET 请求方式

GET请求方式

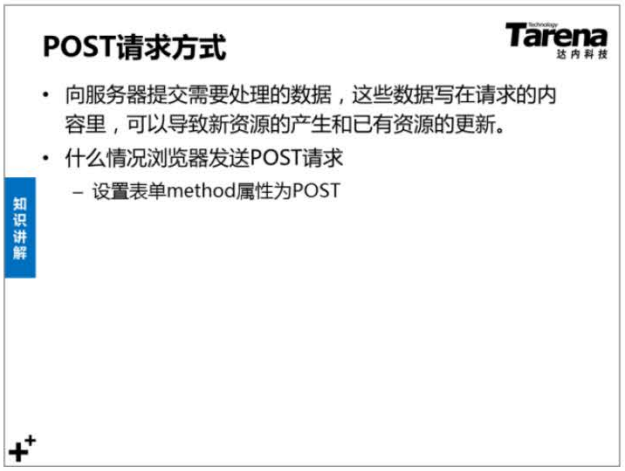
- 当需要向服务器请求指定的资源时使用的方法
- 它不应该用于一些会造成副作用的操作中（在网络应用中用它来提交请求是一种常见的错误用法）
- 什么情况浏览器发送Get请求
  - 在地址栏输入一个地址
  - 点击链接
  - 表单默认提交

解详识知

1.2.4. 【请求方式】GET 请求方式的特点



1.2.5. 【请求方式】POST 请求方式



1.2.6. 【请求方式】POST 请求方式的特点

1.3. 如何处理中文参数

1.3.1. 【如何处理中文参数】为什么表单提交中文会出现乱码

为什么表单提交中文会出现乱码

- 为什么会产生乱码
  - 当表单提交时，浏览器会对中文参数值进行编码（会使用打开表单所在的页面时的字符集进行编码）
  - Web服务器在默认情况下会使用iso-8859-1去解码
  - 编码与解码方式不一致时,就会出现乱码

解详识知

1.3.2. 【如何处理中文参数】解决 POST 方式时的乱码问题

解决POST方式时的乱码问题

- step1：确保表单所在的页面按照指定的字符集打开

```
<meta http-equiv = "content-type"
        content = "text/html;charset=utf-8" >
```

- step2：在服务器端按照这个编码格式解码即可
  - request.setCharacterEncoding("utf-8")
  - 添加在读取参数的前面
  - 此方法只针对post请求有效

解详识知

1.3.3. 【如何处理中文参数】解决 GET 方式时的乱码问题

解决GET方式时的乱码问题

- step1：使用meta确保表单所在页面按指定字符集打开
- step2：在服务器端使用如下方式获取参数值

```
String username = request.getParameter( "" );
username = new String(
    username.getBytes( "iso-8859-1" ), "utf-8" );
```

解详识知

1.4. Servlet 如何输出中文

1.4.1. 【Servlet 如何输出中文】为什么返回的页面会有乱码

为什么返回的页面会有乱码

- 编码：将Unicode字符集对应的字节数组转换成某种本地字符集（如UTF-8）对应的字节数组
- 解码：将某种本地字符集对应的字节数组转换为Unicode字符集对应的字节数组
- 编码和解码使用的字符集不一致就产生了乱码问题

解详识拓

1.4.2. 【Servlet 如何输出中文】如何解决输出内容的乱码

如何解决输出内容的乱码

- 在获取WriteOut对象及调用out方法之前调用setContentType方法
- response.setContentType("text/html;charset=utf-8")
- 作用：
  - 通知容器，在调用out.println方法输出时，使用指定的字符集
  - 生成消息头中content-type的值，通知浏览器，服务器返回的数据类型和字符集

解详识拓

1.5. Servlet 如何访问数据库

1.5.1. 【Servlet 如何访问数据库】使用 JDBC 技术访问数据库

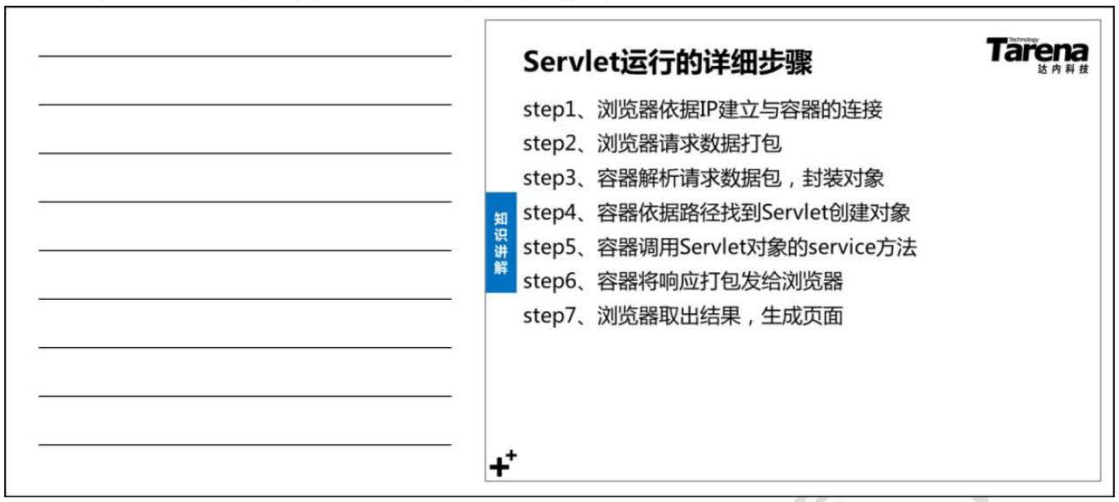
使用JDBC技术访问数据库

- 将JDBC驱动（.jar文件）放到WEB-INF\lib下。原因是：ClassLoader找到字节码文件，然后加载到JVM的方法区中，变成一个对象。Tomcat都有自己的类加载器，会去WEB-INF下面lib中找字节码文件。因为jar包中都是字节码文件
- 在Servlet中编写JDBC代码，实现对数据库的访问

解详识拓

1.6. Servlet 如何运行

1.6.1. 【Servlet 如何运行】Servlet 运行的详细步骤



达内IT培训集

经典案例

1. Servlet 获取请求参数值

- **问题**

在 Servlet 中获取客户端通过表单等形式提交的请求参数值，并输出显示。

- **方案**

使用 request 对象的 getParameter( )方法获取请求中的 **name-value**，使用 request 对象的 getParameterValues ( ) 方法获取请求中的 **name-values**。

- **步骤**

步骤一： 新建 hello1.html 页面

新建 hello1.html 页面，内容如图-1 所示：

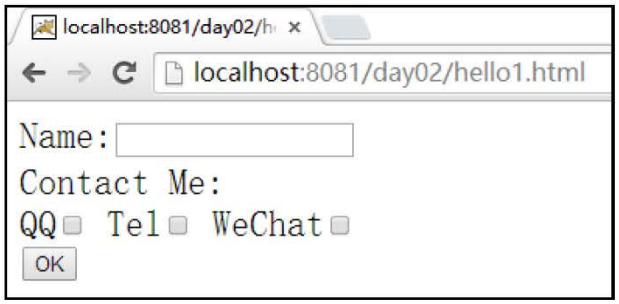


图 - 1

该页面表单内容如图-2 所示：

```
<form action="hello1" method="get">
    Name:<input name="name"><br>
    Contact Me:<Br>
    QQ<input type="checkbox" name="contact"
       value="qq"/>
    Tel<input type="checkbox" name="contact"
       value="tel"/>
    WeChat<input type="checkbox" name="contact"
       value="wechat"/>
    <br>
    <input type="submit" value="OK">
</form>
```

可修改为post

图 - 2

步骤二： 新建 Hello1Servlet 类

新建 Hello1Servlet.java 文件，用于获取参数值及输出。

```
//获取提交的name的值
String name = req.getParameter("name");
out.println("<h1>Hello," + name + "</h1>");
//获取提交的contact的值
String[] contacts = req.getParameterValues("contact");
if(contacts!=null){
    out.print("<h1>Contact Information:</h1>");
    for(String info :contacts){
        out.print("<h1>"+info+"</h1>");
    }
}
```

图 - 3

步骤三： 修改 web.xml 文件

在 web.xml 文件中添加 Hello1Servlet 的映射。

```
<servlet>
  <servlet-name>hello1Servlet</servlet-name>
  <servlet-class>web.Hello1Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>hello1Servlet</servlet-name>
  <url-pattern>/hello1</url-pattern>
</servlet-mapping>
```

图 - 4

步骤四： 部署并访问工程

部署工程后启动 Tomcat，在浏览器中输入地址访问 hello1.html 页面，注意，本案例暂时只输入英文，查看服务器端获取参数值的输出结果。

- **完整代码**

helo1.html 文件的代码如下：

```
<html>
  <head>
  </head>
  <body style="font-size:24px">
    <form action="hello1" method="get">
     Name:<input name="name"><br>
     Contact Me:<Br>
     QQ<input type="checkbox" name="contact"
        value="qq"/>
     Tel<input type="checkbox" name="contact"
        value="tel"/>
     WeChat<input type="checkbox" name="contact"
        value="wechat"/>
     <br>
     <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Hello1Servlet.java 文件的代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Hello1Servlet extends HttpServlet {
 /**
  * 获取请求中的 name、contact 两组参数值
  * 分别使用 getParameter 和 getParameterValues 方法获取
  */
 protected void service(HttpServletRequest req, HttpServletResponse resp)
         throws ServletException, IOException {
     PrintWriter out = resp.getWriter();
     //获取提交的 name 的值
     String name = req.getParameter("name");
     out.println("<h1>Hello," + name + "</h1>");
     //获取提交的 contact 的值
     String[] contacts = req.getParameterValues("contact");
     if(contacts!=null){
         out.print("<h1>Contact Information:</h1>");
         for(String info :contacts){
             out.print("<h1>"+info+"</h1>");
         }
     }
     out.close();
 }
}
```

web.xml 文件的代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
  <servlet>
     <servlet-name>hello1Servlet</servlet-name>
     <servlet-class>web.Hello1Servlet</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>hello1Servlet</servlet-name>
     <url-pattern>/hello1</url-pattern>
  </servlet-mapping>
</web-app>
```

2. 请求方式 GET 和 POST 的区别

- **问题**

查看客户端在提交请求参数值时，GET 请求方式和 POST 请求方式的根本区别，即参数

值的传递方式的区别。

- **方案**

使用 MyEclipse 自带的 TCP/IP Monitor，以代理服务器的方式解析不同请求数据包中的请求行、消息头、主体的内容。

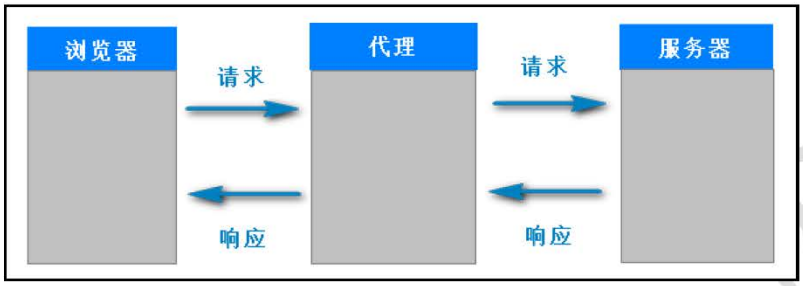


图 - 5

- **步骤**

步骤一： 新建代理服务器并启动 TCP/IP Monitor

在 MyEclipse 中，按照如下步骤打开 TCP/IP Monitor 视图。

菜单 “Windows” → “Show View” → “Other…” 在搜索框中输入 “TCP” 就可看到该视图，选中后点击 “OK” 即可打开。如图-6 所示：

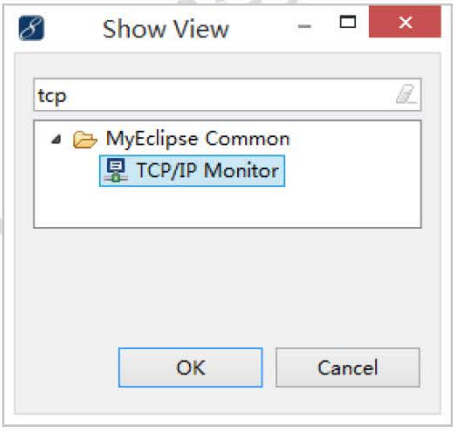


图 – 6

在该视图的空白处右键，点击 “Properties” 如图-7 所示:

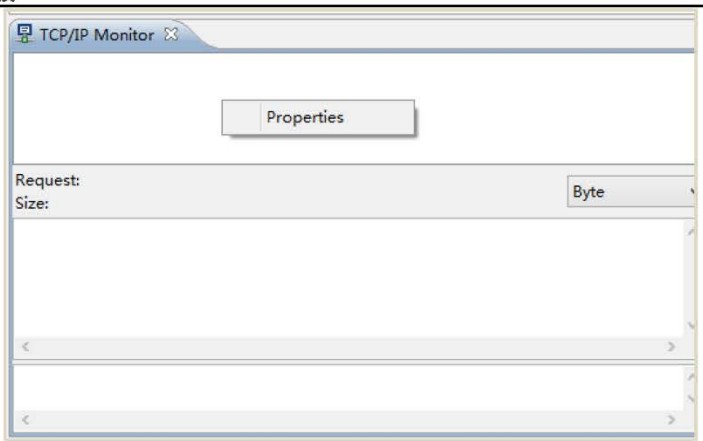


图 – 7

在弹出的窗口中，按照如下图-8 至图-10 所示的顺序依次操作，完成 Monitor 的创建及启动。

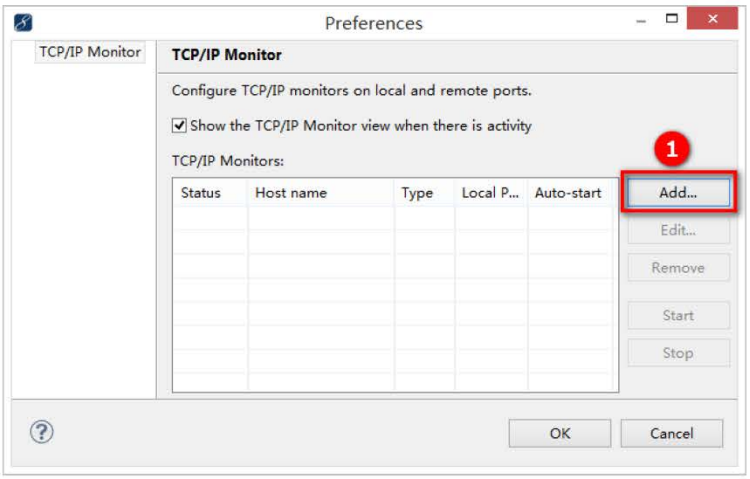


图- 8

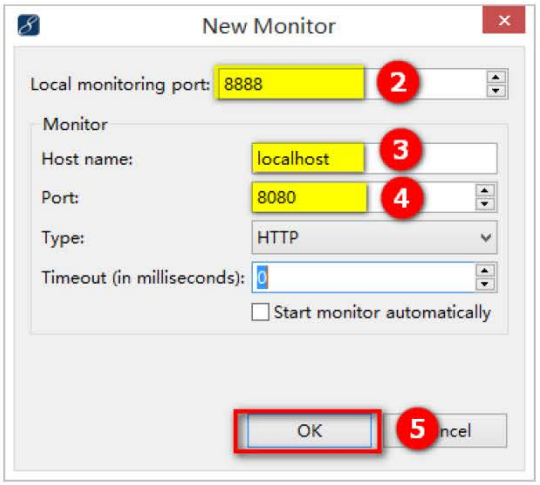


图- 9

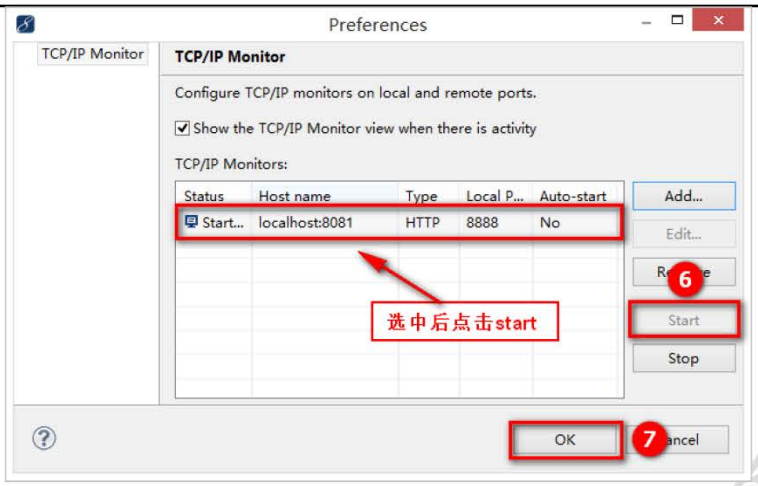


图- 10

步骤二： 访问代理服务器，以 GET 方式提交表单数据

使用上一个案例中的表单来模拟提交数据即可。本次注意表单的 method 值为 get。

```
<form action= "hello1" method= "get ">
```

启动 Tomcat 后，输入如下地址 **http://localshot:8888/day02/hello1.html**

图 – 11

填写表单数据后点击 OK，查看 Monitor 中的数据包如图-12。

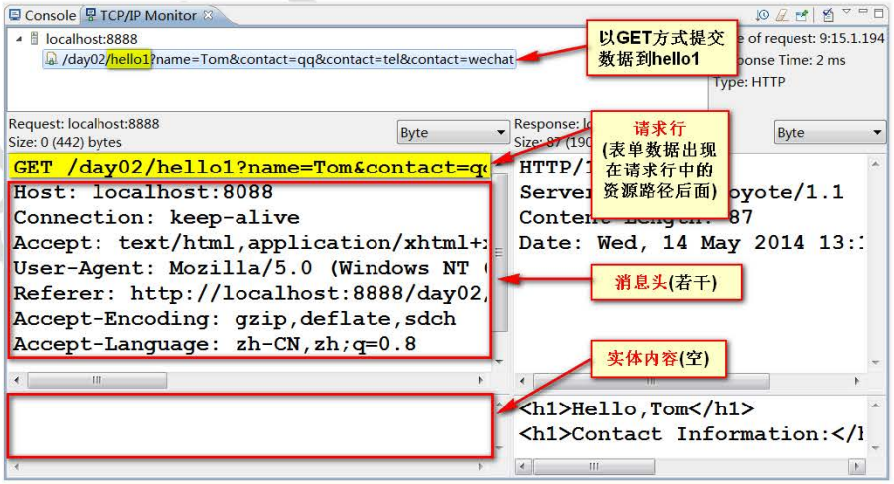


图 - 12

步骤三： 访问代理服务器，以 POST 方式提交表单数据

修改表单的 method 值为 post。

```
<form action= "hello1" method= "post ">
```

重新部署后，输入代理服务器地址 **http://localshot:8888/day02/hello1.html** 填写表单数据后点击 OK 后查看 Monitor 中的数据包如图-13 所示：

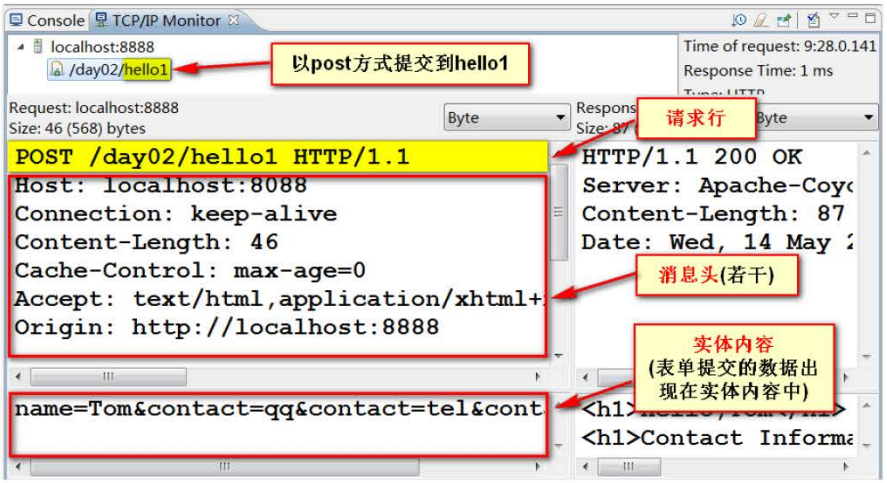


图- 13

步骤四： 对比代理服务器中的数据包数据

对比图-12 和图-13 抓取的请求数据包可以看出，get 方式和 post 方式在提交表单数据时的根本区别在于数据所处的位置。get 方式参数值在请求资源路径的后面，从请求行汇总传递过来。post 方式参数值出现在实体内容中，请求资源路径上不会出现任何多余信息。

3. 处理 POST 请求中的中文参数值

- **问题**

表单以 post 方式提交中文数据到服务器端会出现乱码的现象，增加对乱码的处理，使得 POST 方式能够正确提交数据并显示。

- **方案**

为 html 页面增加 meta 标记，保证浏览器以支持中文编码方式打开页面。在服务器端的 Servlet 获取请求参数值时，按照与客户端相同的编码方式来解码，就可以实现正确获取，在输出流中同样设置与该编码方式相同格式来控制中文的输出显示。

- **步骤**

步骤一： 新建 hello2.html 页面并增加 meta 标记

新建 hello2.html 页面如下所示：

姓名：
联系方式：
QQ 电话 微信
OK

图 - 14

表单为 post 提交方式，文件源代码如下：

```
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
</head>
<body style="font-size:24px">
  <form action="hello2" method="post">
      姓名：<input name="name"><br>
      联系方式:<Br>
      QQ<input type="checkbox" name="contact" value="qq"/>
      电话<input type="checkbox" name="contact" value="tel"/>
      微信<input type="checkbox" name="contact" value="wechat"/><br>
      <input type="submit" value="OK">
  </form>
</body>
```

图 - 15

步骤二： 新建 Hello2Servlet.java 文件

新建 Hello2Servlet.java 文件，添加 request.setCharacterEncoding 来设置取参数值之前的解码方式，为了保证输出中文，使用 response.setContentType 来设置 content-type 的消息头，从而控制浏览器的解码方式。

```
//保证正确读取post提交来的中文
req.setCharacterEncoding("utf-8");
//保证正确输出中文
resp.setContentType("text/html;charset=utf-8");
PrintWriter out = resp.getWriter();
//获取提交的name的值
String name = req.getParameter("name");
out.println("<h1>Hello," + name + "</h1>");
//获取提交的contact的值
String[] contacts = req.getParameterValues("contact");
if(contacts!=null){
    out.print("<h1>联系方式:</h1>");
    for(String contact :contacts){
        out.print("<h1>"+contact+"</h1>");
    }
}
```

图 - 16

步骤三： 修改 web.xml 文件

添加 Hello2Servlet 的映射，如图-17 所示：

```
<servlet>
  <servlet-name>hello2Servlet</servlet-name>
  <servlet-class>web.Hello2Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>hello2Servlet</servlet-name>
  <url-pattern>/hello2</url-pattern>
```

图 - 17

步骤四： 部署并访问工程

部署工程,启动服务，输入地址，并输入中文表单数据后查看结果。

图 - 18

- **完整代码**

hello2.html 完整代码如下：

```
<html>
  <head>
  <meta http-equiv="content-type" content="text/html;charset=utf-8">
  </head>
  <body style="font-size:24px">
    <form action="hello2" method="post">
      姓名：<input name="name"><br>
      联系方式:<Br>
      QQ<input type="checkbox" name="contact" value="qq"/>
      电话<input type="checkbox" name="contact" value="tel"/>
      微信<input type="checkbox" name="contact" value="wechat"/><br>
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Hello2servlet.java 完整代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Hello2Servlet extends HttpServlet {

 protected void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
     //保证正确读取post提交来的中文
     req.setCharacterEncoding("utf-8");
     //保证正确输出中文
     resp.setContentType("text/html;charset=utf-8");
     PrintWriter out = resp.getWriter();
     //获取提交的name的值
     String name = req.getParameter("name");
     out.println("<h1>Hello," + name + "</h1>");
     //获取提交的contact的值
     String[] contacts = req.getParameterValues("contact");
     if(contacts!=null){
         out.print("<h1>联系方式:</h1>");
         for(String contact :contacts){
             out.print("<h1>"+contact+"</h1>");
         }
     }
     out.close();
 }
}
```

web.xml 文件完整代码如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
  <servlet>
     <servlet-name>hello2Servlet</servlet-name>
     <servlet-class>web.Hello2Servlet</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>hello2Servlet</servlet-name>
     <url-pattern>/hello2</url-pattern>
  </servlet-mapping>
</web-app>
```

4. 处理 GET 请求中的中文参数值

- **问题**

表单以 get 方式提交中文数据到服务器端会出现乱码的现象，增加对乱码的处理，使得 get 方式能够正确提交数据并显示。

- **方案**

为 html 页面增加 meta 标记，保证浏览器以支持中文编码方式打开页面。在服务器端的 Servlet 获取请求参数值时，按照 iso-8859-1 的方式获取字节码，再使用与客户端相

同的编码方式来解码。

- **步骤**

步骤一：新建 hello2.html 页面并增加 meta 标记

页面中的表单提交方式为 get。

```
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
</head>
<body style="font-size:24px">
  <form action="hello2" method="get">
      姓名：<input name="name"><br>
      联系方式:<Br>
      QQ<input type="checkbox" name="contact" value="qq"/>
      电话<input type="checkbox" name="contact" value="tel"/>
      微信<input type="checkbox" name="contact" value="wechat"/><br>
      <input type="submit" value="OK">
  </form>
</body>
```

图 - 19

步骤二：新建 Hello2Servlet.java 文件

表单提交到的 Servlet 中需要按照 ios-8859-1 的方式获取字节码，再将此字节码按照 UTF-8 方式的解码。

```
//保证正确输出中文
resp.setContentType("text/html;charset=utf-8");
PrintWriter out = resp.getWriter();
//获取提交的name的值
String name = req.getParameter("name");
//保证正确读取get提交来的中文
name = new String(name.getBytes("iso-8859-1"),"UTF-8");
out.println("<h1>Hello," + name + "</h1>");
//获取提交的contact的值
String[] contacts = req.getParameterValues("contact");
if(contacts!=null){
    out.print("<h1>联系方式:</h1>");
    for(String contact :contacts){
        out.print("<h1>"+contact+"</h1>");
    }
}
```

图 - 20

步骤三：修改 web.xml 文件

与图-17 一致。

步骤四：部署并访问工程

部署并访问工程，输入中文姓名查看结果及地址栏。

图 - 21

- **完整代码**

hello2.html 文件完整代码：

```
<html>
  <head>
  <meta http-equiv="content-type" content="text/html;charset=utf-8">
  </head>
  <body style="font-size:24px">
    <form action="hello2" method="get">
      姓名：<input name="name"><br>
      联系方式:<Br>
      QQ<input type="checkbox" name="contact" value="qq"/>
      电话<input type="checkbox" name="contact" value="tel"/>
      微信<input type="checkbox" name="contact" value="wechat"/><br>
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Hello2Servlet.java 文件代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Hello2Servlet extends HttpServlet {

 protected void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
     //保证正确输出中文
     resp.setContentType("text/html;charset=utf-8");
     PrintWriter out = resp.getWriter();
     //获取提交的 name 的值
     String name = req.getParameter("name");
     //保证正确读取 get 提交来的中文
```

```
        name = new String(name.getBytes("iso-8859-1"),"UTF-8");
        out.println("<h1>Hello," + name + "</h1>");
        //获取提交的 contact 的值
        String[] contacts = req.getParameterValues("contact");
        if(contacts!=null){
            out.print("<h1>联系方式:</h1>");
            for(String contact :contacts){
                out.print("<h1>"+contact+"</h1>");
            }
        }
        out.close();
    }
}
```

web.xml 文件代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
  <servlet>
     <servlet-name>hello2Servlet</servlet-name>
     <servlet-class>web.Hello2Servlet</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>hello2Servlet</servlet-name>
     <url-pattern>/hello2</url-pattern>
  </servlet-mapping>
</web-app>
```

5. 员工管理——添加员工信息

- **问题**

编写工程实现收集员工姓名、薪水、年龄信息，将数据提交到服务器后，服务器能够正确获取参数值并输出显示，要求能够支持中文。

- **方案**

制作表单用于获取用户输入，选择 post 方式提交数据，添加 meta 标签控制浏览器的编码方式。服务器端增加 Servlet 用于获取请求参数值，添加用于处理中文乱码的 setCharaterEncoding，实现正确解码。使用 setContentType 实现中文的正确输出。

- **步骤**

步骤一： 新建 addEmp.html 页面

新建 addEmp.html 页面效果如图-22 所示：

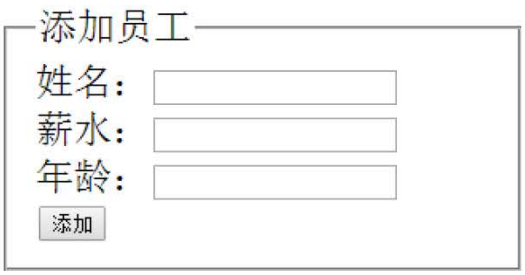


图 – 22

addEmp.html 页面的表单如下：

```
<html>
    <head>
        <meta http-equiv="content-type"
        content="text/html;charset=utf-8">
    </head>
    <body style="font-size:24px">
        <form action="add" method="post">
            <fieldset>
                <legend>添加员工</legend>
                姓名：<input name="name"/><br>
                薪水：<input name="salary"/><br>
                年龄：<input name="age"/><br>
                <input type="submit" value="添加"/>
            </fieldset>
        </form>
    </body>
</html>
```

图 - 23

步骤二： 新建 AddEmpServlet 类

新建 AddEmpServlet 类，使用 request.setCharacterEncoding("UTF-8")及 response.setContentType("text/html;charset=UTF-8")来保证正确获取中文，正确输出中文。

```
// 设置中文的输入和输出
request.setCharacterEncoding("UTF-8");
response.setContentType("text/html;charset=UTF-8");
// 获取输出流对象，并输出信息
PrintWriter out = response.getWriter();
// 获取表单提交的数据
String name = request.getParameter("name");
double salary = Double.valueOf(request.getParameter("salary"));
int age = Integer.valueOf(request.getParameter("age"));

out.print("<h1>员工信息</h1>");
out.print("<h1>姓名：" + name + "</h1>");
out.print("<h1>薪水：" + salary + "</h1>");
out.print("<h1>年龄：" + age + "</h1>");
out.close();
```

图- 24

步骤三： 修改 web.xml 文件

修改 web.xml 文件，为 AddEmpServlet 添加映射，如图-25。

```
<servlet>
  <servlet-name>addEmpServlet</servlet-name>
  <servlet-class>web.AddEmpServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>addEmpServlet</servlet-name>
  <url-pattern>/add</url-pattern>
```

图 - 25

步骤四： 部署及访问应用

部署应用，启动 Tomcat，输入访问 addEmp.html 的页面地址，输入中文，点击提交，查看运行结果。

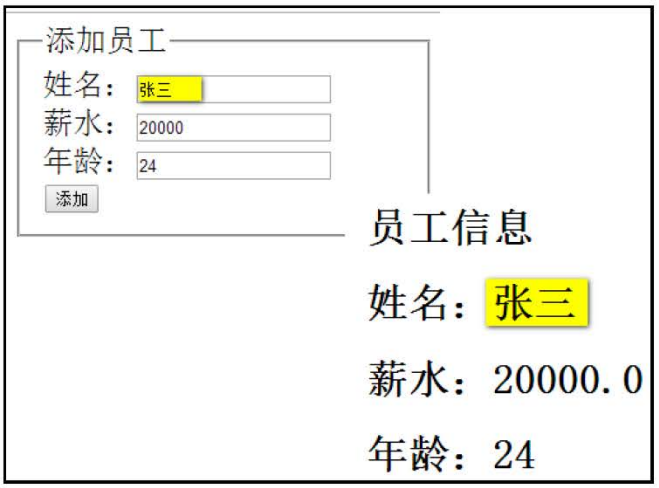


图 - 26

- **完整代码**

addEmp.html 文件完整代码：

```
<html>
 <head>
     <meta http-equiv="content-type" content="text/html;charset=utf-8">
 </head>
 <body style="font-size:24px">
     <form action="add" method="post">
         <fieldset>
             <legend>添加员工</legend>
             姓名：<input name="name"/><br>
             薪水：<input name="salary"/><br>
             年龄：<input name="age"/><br>
             <input type="submit" value="添加"/>
         </fieldset>
     </form>
 </body>
</html>
```

AddEmpServlet.java 文件完整代码：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class AddEmpServlet extends HttpServlet {

 public  void  service(HttpServletRequest  request,  HttpServletResponse
response)
          throws ServletException, IOException {
      // 设置中文的输入和输出
      request.setCharacterEncoding("UTF-8");
      response.setContentType("text/html;charset=UTF-8");
      // 获取输出流对象，并输出信息
      PrintWriter out = response.getWriter();
      // 获取表单提交的数据
      String name = request.getParameter("name");
      double salary = Double.valueOf(request.getParameter("salary"));
      int age = Integer.valueOf(request.getParameter("age"));
      out.print("<h1>员工信息</h1>");
      out.print("<h1>姓名：" + name + "</h1>");
      out.print("<h1>薪水：" + salary + "</h1>");
      out.print("<h1>年龄：" + age + "</h1>");
      out.close();
 }
}
```

web.xml 文件完整代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
  <servlet>
     <servlet-name>addEmpServlet</servlet-name>
     <servlet-class>web.AddEmpServlet</servlet-class>
  </servlet>
   <servlet-mapping>
     <servlet-name>addEmpServlet</servlet-name>
     <url-pattern>/add</url-pattern>
  </servlet-mapping>
</web-app>
```

6. 员工管理——使用 JDBC 添加员工信息

- **问题**

将客户端提交的员工信息，使用 JDBC 的技术插入到 t_emp 表中。

- **方案**

首先在 Oracle 中建立一个名字叫 t_emp 的表，包括 id，name，salary，age 四个字段，id 为主键字段，并实现自增。在处理表单信息的 Servlet 中，使用 JDBC 技术完成连接数据库，并使用获取的请求参数值构建 insert 语句，执行后完成员工信息的增加操作。操作成功或失败均返回给用户响应的提示信息。

- **步骤**

步骤一： 使用 MyEclipse 的 DB Browser 视图连接 Oracle 数据库

使用如下步骤打开 DB Browser 视图。

菜单 “window” → “Show View” → “Other…” 在搜索栏输入 DB，选择 “DB Browser “后点击 ok 即可。在视图中右键选择 new，创建一个新的连接，配置如图-27，图-28。

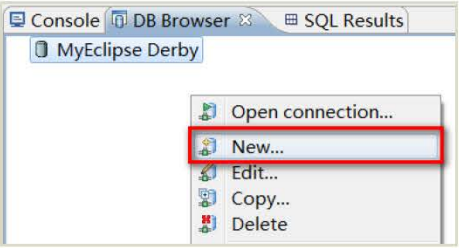


图 - 27

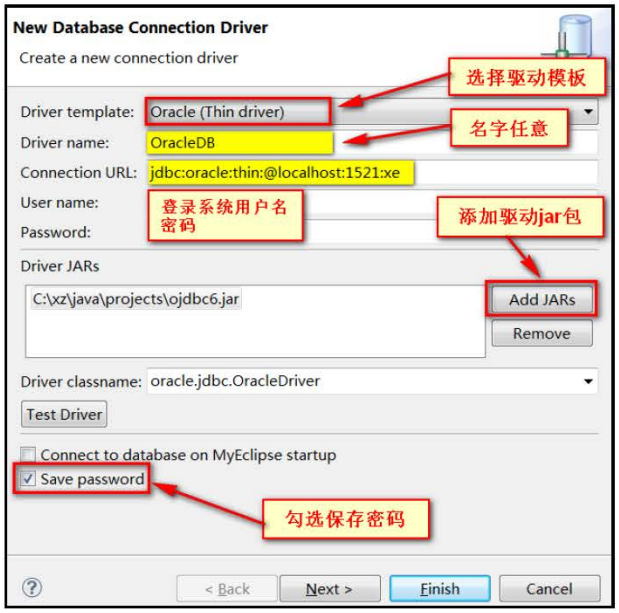


图 - 28

连接数据库正确后，点击新创建的这个连接，右键 Open Connection 即可查看连接情况如图-29。

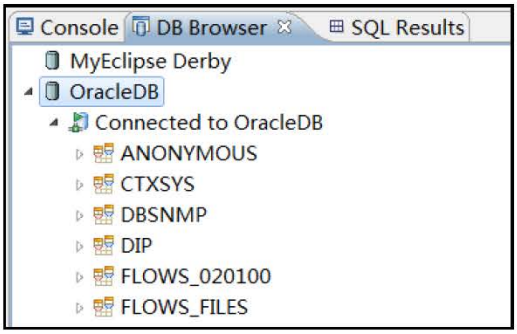


图 - 29

步骤二： 执行 SQL 语句创建 t_emp 表

在工程节点下，右键新建一个名字叫做 day02.sql 的文件，在该文件中编写建表脚本。如图-30，图-31 所示：

图 – 30

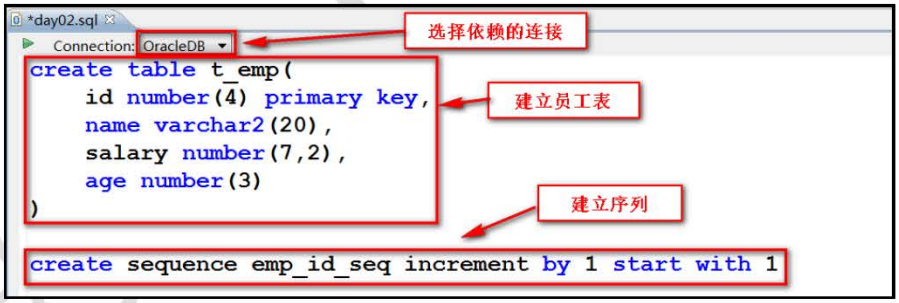


图 - 31

在 sql 文件中，编写完 sql 语句后，为了执行，可以选中，然后点击连接左边的绿色按钮，即代表执行。为了查看表是否创建，可以在执行完以后到 DB Browser 视图中，打开对应的连接，查看下面的 table 节点下是否有更新。

表创建成功之后，继续在 sql 文件中编写 insert 语句，选中后执行。为了查看表内的数据，再编写 select * from t_emp 语句。结果的查看可以通过 菜单 "Window" → "show View" → "Other…"，在搜索栏中填写 SQL 后，选中 SQL Result 视图，点击确定，即可打开视图看见 select 语句的执行结果。过程如图-32，图-33 所示：

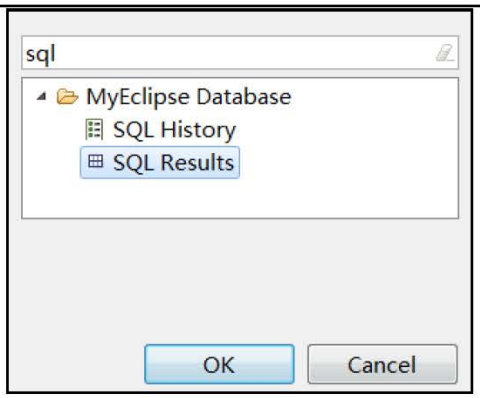


图 – 32

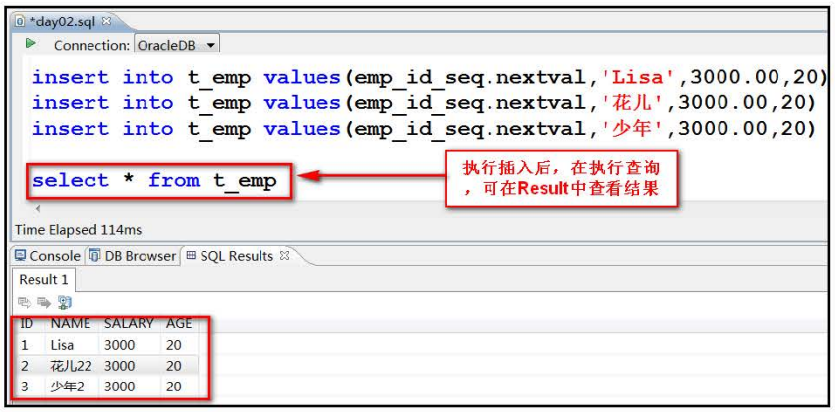


图 – 33

步骤三： 新建 addEmp.html 页面

建立 addEmp.html 页面如下。

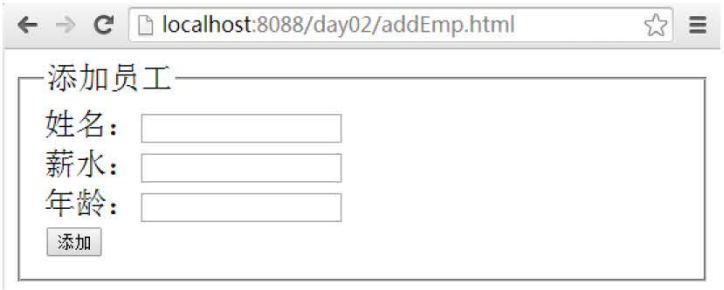


图 – 34

步骤四： 新建 AddEmpServlet.java 类

编写有关数据库操作的代码，如图-35 所示：

```
// 获取表单提交的数据
String name = request.getParameter("name");
double salary = Double.valueOf(request.getParameter("salary"));
int age = Integer.valueOf(request.getParameter("age"));
//将数据插入到数据库t_emp表中
Connection conn = null;
PreparedStatement stat = null;
try {
    Class.forName("oracle.jdbc.OracleDriver");
    conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521:xe", "xxxx", "xxxx");
    stat = conn.prepareStatement("insert into t_emp values" +
            "(emp_id_seq.nextval,?,?,?)");
    stat.setString(1, name);
    stat.setDouble(2, salary);
    stat.setInt(3, age);
    stat.executeUpdate();
    out.println("添加成功");
```

加载驱动

获取连接

命令对象

执行

图 - 35

在编写 JDBC 代码时，需要加载驱动 jar 包，否则 Class.forName 会报错。在 web Project 中添加 jar 包，只需要将文件拷贝，在 lib 文件夹下右键粘贴即可，工程会自动增加对 jar 包的构建。

如图-36 为添加了驱动 jar 包后效果：

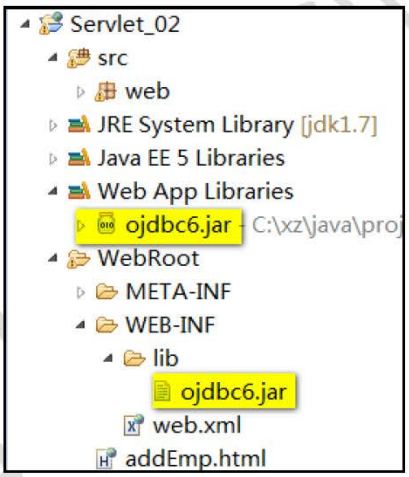


图 - 36

步骤五： 添加 AddEmpServlet 的映射

```
<servlet>
  <servlet-name>addEmpServlet</servlet-name>
  <servlet-class>web.AddEmpServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>addEmpServlet</servlet-name>
  <url-pattern>/add</url-pattern>
```

图 - 37

步骤六： 部署应用，访问 addEmp.html 页面，查看结果

部署应用，访问 addEmp.html 页面填写中文数据后点击提交。可以到 SQL Result 界面中通过执行查询语句确认增加操作是否执行成功。结果参考图-38，图-39 所示：

图 - 38

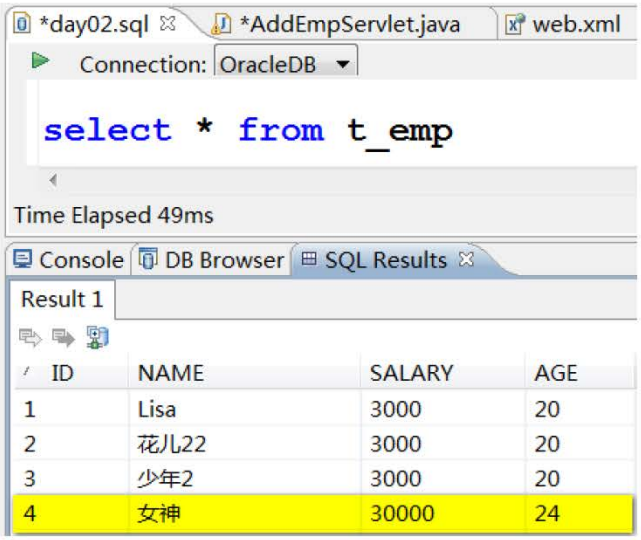


图 - 39

- **完整代码**

addEmp.html 文件完整代码：

```
<form action="add" method="post">
    <fieldset>
        <legend>添加员工</legend>
        姓名：<input name="name"/><br>
        薪水：<input name="salary"/><br>
        年龄：<input name="age"/><br>
        <input type="submit" value="添加"/>
    </fieldset>
</form>
```

AddEmpServlet.java 文件完整代码:

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class AddEmpServlet extends HttpServlet {

 public  void  service(HttpServletRequest  request,  HttpServletResponse
response)
        throws ServletException, IOException {
    // 设置中文的输入和输出
    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    // 获取输出流对象，并输出信息
    PrintWriter out = response.getWriter();
    // 获取表单提交的数据
    String name = request.getParameter("name");
    double salary = Double.valueOf(request.getParameter("salary"));
    int age = Integer.valueOf(request.getParameter("age"));
    //将数据插入到数据库 t_emp 表中
    Connection conn = null;
    PreparedStatement stat = null;
    try {
        Class.forName("oracle.jdbc.OracleDriver");
        conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "xxxx", "xxxx");
        stat = conn.prepareStatement("insert into t emp values" +
                "(emp id seq.nextval,?,?,?)");
        stat.setString(1, name);
        stat.setDouble(2, salary);
        stat.setInt(3, age);
        stat.executeUpdate();
        out.println("添加成功");
    } catch (Exception e) {
        e.printStackTrace();
        out.print("系统繁忙，稍后重试");
    } finally{
        if(stat!=null){
            try {
                stat.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(conn!=null){
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
 }
}
```

web.xml 文件完整代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
  <servlet>
```

```
      <servlet-name>addEmpServlet</servlet-name>
      <servlet-class>web.AddEmpServlet</servlet-class>
  </servlet>
  <servlet-mapping>
      <servlet-name>addEmpServlet</servlet-name>
      <url-pattern>/add</url-pattern>
  </servlet-mapping>
</web-app>
```

day02.sql 文件完整代码：

```
create table t_emp(
 id number(4) primary key,
 name varchar2(20),
 salary number(7,2),
 age number(3)
)

create sequence emp_id_seq increment by 1 start with 1

insert into t_emp values(emp_id_seq.nextval,'Lisa',3000.00,20)
insert into t_emp values(emp_id_seq.nextval,'花儿',3000.00,20)
insert into t_emp values(emp_id_seq.nextval,'少年',3000.00,20)

select * from t_emp
```

7. 员工管理——使用 JDBC 查询所有员工信息

- **问题**

客户端访问服务器后可以返回 t_emp 表中所有数据的表格展示，效果如图-40 所示：

图 - 40

- **方案**

在处理请求的 Servlet 中，使用 JDBC 技术实现数据库的访问，并且在遍历结果集时，将结果与 html 的表格行、列标签进行连接，生成符合表格样式的标记后输出到客户端。

- **步骤**

步骤一： 新建 ListEmpServlet.java 文件

在 ListEmpServlet 类中，使用 JDBC 技术连接数据库，并执行查询语句，对查询结果集遍历后，以表格形式输出。如图-41 所示：

```
Class.forName("oracle.jdbc.OracleDriver");
conn = DriverManager.getConnection(
        "jdbc:oracle:thin:@localhost:1521:xe",
        "xxx",
        "xxx");
stat = conn.prepareStatement("select * from t_emp");
rs = stat.executeQuery();
//2.输出表格之前的页面标记
out.println("<HTML>");
out.println("<HEAD></HEAD>");
out.println("<BODY>");
out.println("<table border='1' cellpadding='0' " +
        "cellspacing='0' width='60%'>");
out.println("<caption>员工信息列表</caption>");
out.println("<tr><td>ID</td><td>姓名</td><td>薪水</td>" +
        "<td>年龄</td></tr>");
//3.遍历结果集，以表格形式输出数据
while(rs.next()){
    int id = rs.getInt("id");
    String name = rs.getString("name");
    double salary = rs.getDouble("salary");
    int age = rs.getInt("age");
    out.println("<tr><td>"+
            id+"</td><td>"+
            name+"</td><td>"+
            salary+"</td><td>"+
            age+"</td></tr>");
}
out.print("</table>");
out.print("<a href='addEmp.html'>增加员工信息</a>");
out.println("</BODY>");
out.println("</HTML>");
```

得到结果集

输出页面标记

遍历结果集，并连接tr标记生成表格

图 - 41

步骤二： 添加 ListEmpServlet 映射

修改 web.xml 文件，添加映射，如图-42 所示：

```
<servlet>
  <servlet-name>listEmpServlet</servlet-name>
  <servlet-class>web.ListEmpServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>listEmpServlet</servlet-name>
  <url-pattern>/list</url-pattern>
```

图 - 42

步骤三： 部署及访问应用

部署后，在浏览器地址栏输入地址 http://localhost:8080/应用名/list,查看结果是否与图-40 一致。

- **完整代码**

ListEmpServlet.java 文件完整代码：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ListEmpServlet extends HttpServlet {

 public  void  service(HttpServletRequest  request,  HttpServletResponse
response)
        throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    //1.连接数据库获取数据
    Connection conn = null;
    PreparedStatement stat = null;
    ResultSet rs = null;
    try{
        Class.forName("oracle.jdbc.OracleDriver");
        conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe",
                "xxxx",
                "xxxx");
        stat = conn.prepareStatement("select * from t emp");
        rs = stat.executeQuery();
        //2.输出表格之前的页面标记
        out.println("<HTML>");
        out.println("<HEAD></HEAD>");
        out.println("<BODY>");
        out.println("<table  border='1'  cellpadding='0'  cellspacing='0'
width='60%'>");
        out.println("<caption>员工信息列表</caption>");
        out.println("<tr><td>ID</td><td> 姓 名 </td><td> 薪 水 </td><td> 年 龄
</td></tr>");
        //3.遍历结果集，以表格形式输出数据
        while(rs.next()){
            int id = rs.getInt("id");
            String name = rs.getString("name");
            double salary = rs.getDouble("salary");
            int age = rs.getInt("age");
            out.println("<tr><td>"+
                    id+"</td><td>"+
                    name+"</td><td>"+
                    salary+"</td><td>"+
                    age+"</td></tr>");
        }
        out.print("</table>");
        out.print("<a href='addEmp.html'>增加员工信息</a>");
        out.println("</BODY>");
```

```
            out.println("</HTML>");
            out.flush();
            out.close();
        }catch(Exception e){
            e.printStackTrace();
            out.println("系统繁忙，请稍后重试");
        }finally{
            if(conn!=null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }

    }

}
```

web.xml 文件完整代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
  <servlet>
     <servlet-name>addEmpServlet</servlet-name>
     <servlet-class>web.AddEmpServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>listEmpServlet</servlet-name>
    <servlet-class>web.ListEmpServlet</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>addEmpServlet</servlet-name>
     <url-pattern>/add</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>listEmpServlet</servlet-name>
    <url-pattern>/list</url-pattern>
  </servlet-mapping>
</web-app>
```

课后作业

1. 简述 GET 和 POST 的区别。

2. 简述如何处理表单提交的中文。

3. 阅读下面的代码，说明序号处代码的含义。

```
public class AddEmpServlet extends HttpServlet{
 public void service(HttpServletRequest request,
        HttpServletResponse response)
 throws ServletException,IOException{

     request.setCharacterEncoding("utf-8");
     String name = request.getParameter("name");//------------------------1
     double salary = Double.parseDouble(
             request.getParameter("salary"));
     int age = Integer.parseInt(
             request.getParameter("age"));
     System.out.println("name:" + name);
     System.out.println("salary:" + salary);
     System.out.println("age:" + age);
     response.setContentType("text/html;charset=utf-8");//---------------2
     PrintWriter out = response.getWriter();
     out.println("<h1>"+name+"</h1>");

 }
}
```

4. 编写一个 Servlet，用户通过表单提交，新建一个帐务帐号信息。

将客户端提交的帐务帐号信息，使用 JDBC 的技术插入到 account 表中。帐务帐号信息表的结构如表 - 1 所示：

表 - 1 帐务帐号信息表（ACCOUNT）

字段名称	类型	备注	字段描述
ACCOUNT_ID	NUMBER(9)	PRIMARY KEY NOT NULL	帐务帐号 ID
LOGIN_NAME	VARCHAR2(30)	UNIQUE NOT NULL	登录 NetCTOSS 系统的名称，用于用户自服务
STATUS	CHAR（1）	NOT NULL CHECK (0,1,2)	0：开通；1：暂停；2：删除
REAL_NAME	VARCHAR2(20)	NOT NULL	客户姓名
IDCARD_NO	CHAR(18)	NOT NULL UNIQUE	身份证号码
TELEPHONE	VARCHAR2(15)	NOT NULL	联系电话（座机或手机）

添加界面如图-1 所示：

图 – 1

在文本框中填入信息，点击“添加”按钮将数据添加到数据的 account 表中。

5. 编写一个 Servlet 查询数据库，显示所有账务账号信息。

客户端访问服务器后可以返回 Account 表中所有数据的表格展示，效果如图-2 所示：

图- 2

Servlet和JSP(上)

Unit03

知识体系.....Page 84

容器对路径的处理	重定向	什么是重定向
		重定向原理
		重定向原理（续）
		如何重定向
		重定向特点
	Servlet 容器如何处理请求资源路径	什么是请求资源路径
		Web 服务器对请求地址的处理过程
		匹配 Servlet 规则-精确匹配
		匹配 Servlet 规则-通配符匹配
		匹配 Servlet 规则-后缀匹配
		无匹配的 Servlet 的处理
	一个 Servlet 实现多请求	为什么要将多 Servlet 合并
		使用后缀匹配模式修改 web.xml
		分析请求资源后分发
Servlet 特性	Servlet 的生命周期	什么是 Servlet 的生命周期
		生命周期的四个阶段
		Servlet 生命周期原理图
		Servlet 核心接口和类
		Servlet 接口
		Servlet 核心类
	ServletContext	什么是 Servlet 上下文
		如何获得 Servlet 上下文
		Servlet 上下文的作用及特点
	Servlet 线程安全问题	为什么会有线程安全问题
		如何保证 Servlet 的线程安全

经典案例.....Page 94

员工管理——使用 JDBC 删除员工信息	什么是重定向
	重定向原理

员工管理——使用 JDBC 修改员工信息	重定向原理（续）
	如何重定向
容器对 URI 的处理	匹配 Servlet 的规则-精确匹配
	匹配 Servlet 的规则-通配符匹配
	匹配 Servlet 的规则-后缀匹配
员工管理——单一控制器实现员工信息管理	为什么要将多 Servlet 合并
	使用后缀模式完成路径匹配
	分析请求资源后分发
Servlet 的生命周期	什么是 Servlet 的生命周期
	生命周期的四个阶段
ServletContext 对象（Servlet 上下文对象）	什么是 Servlet 上下文
	如何获得 Servlet 上下文
	Servlet 上下文的作用及特点
Servlet 线程安全	为什么会有线程安全问题
	如何保证 Servlet 线程安全

课后作业.....Page 130

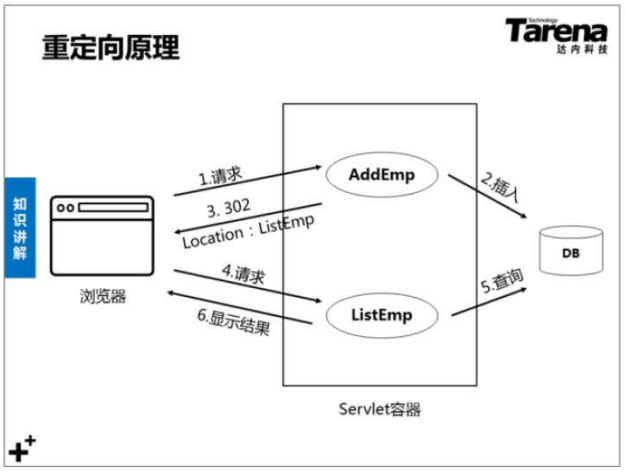
达内IT培训集团

1. 容器对路径的处理

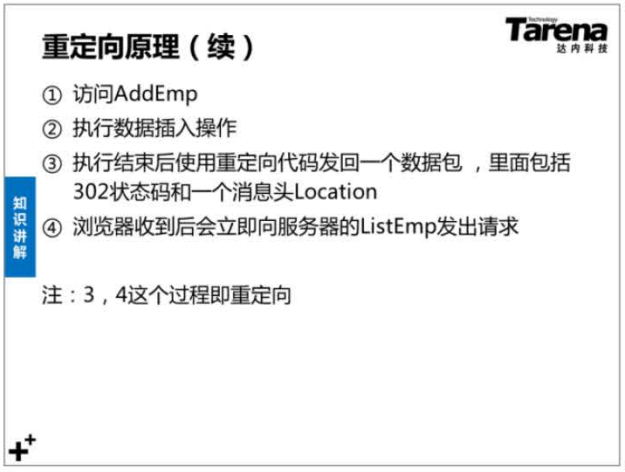
1.1. 重定向

1.1.1. 【重定向】什么是重定向

1.1.2. 【重定向】重定向原理



1.1.3. 【重定向】重定向原理（续）



1.1.4. 【重定向】如何重定向

1.1.5. 【重定向】重定向特点

1.2. Servlet 容器如何处理请求资源路径

1.2.1. 【Servlet 容器如何处理请求资源路径】什么是请求资源路径

1.2.2. 【Servlet 容器如何处理请求资源路径】Web 服务器对请求地址的处理过程

1.2.3. 【Servlet 容器如何处理请求资源路径】匹配 Servlet 规则-精确匹配

1.2.4. 【Servlet 容器如何处理请求资源路径】匹配 Servlet 规则-通配符匹配

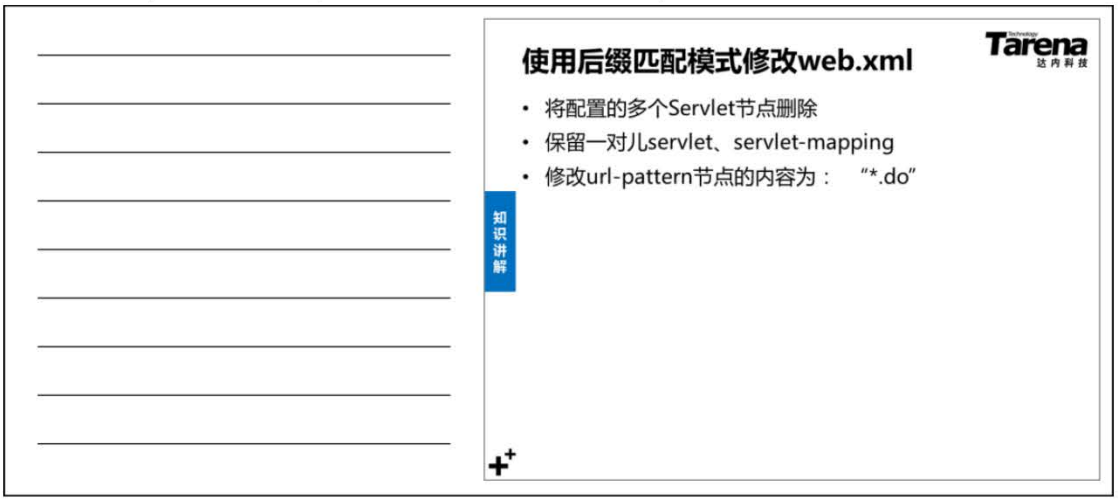
1.2.5. 【Servlet 容器如何处理请求资源路径】匹配 Servlet 规则-后缀匹配

1.2.6. 【Servlet 容器如何处理请求资源路径】无匹配的 Servlet 的处理

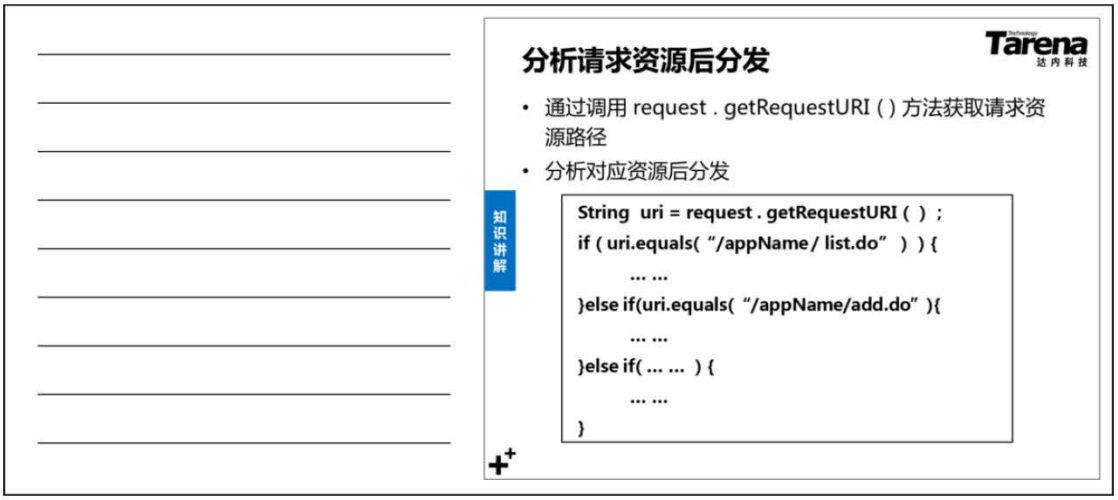
1.3. 一个 Servlet 实现多请求

1.3.1. 【一个 Servlet 实现多请求】为什么要将多 Servlet 合并

1.3.2. 【一个 Servlet 实现多请求】使用后缀匹配模式修改 web.xml



1.3.3. 【一个 Servlet 实现多请求】分析请求资源后分发



2. Servlet 特性

2.1. Servlet 的生命周期

2.1.1. 【Servlet 的生命周期】什么是 Servlet 的生命周期

2.1.2. 【Servlet 的生命周期】生命周期的四个阶段

生命周期的四个阶段

解详识知

- 阶段一：实例化
- 什么是实例化？
  - 容器调用Servlet的构造器，创建一个Servlet对象。
- 什么时候实例化？
  - 情形1，开始容器里面没有Servlet对象，收到请求后创建Servlet对象。
  - 情形2，容器启动之后就立即创建相应的实例。

生命周期的四个阶段（续1）

解详识知

- 阶段二：初始化
- 什么是初始化？
  - 容器在创建好Servlet对象之后，会立即调用该对象的init方法。
  - 一般情况下，我们不用写init方法，因为GenericServlet已经提供了init方法的实现(将容器传递过来的ServletConfig对象保存来下，并且，提供了getServletConfig方法来获得ServletConfig对象)。
- init方法只会执行一次

生命周期的四个阶段（续2）

解详识知

Servlet的初始化参数如何配置

```
<init-param>
    <param-name>company</param-name>
    <param-value>北京达内</param-value>
</init-param>
```

如何读取Servlet的初始化参数？

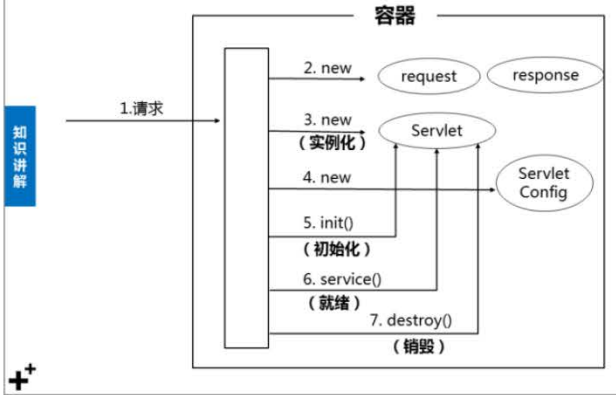
```
String ServletConfig.getInitParameter("company")
```

生命周期的四个阶段（续3）

- 阶段三：就绪
  - 容器收到请求之后调用Servlet对象的service（）来处理请求。
- 阶段四：销毁
  - 容器依据自身的算法删除Servlet对象，删除前会调用destroy（）
  - 只会执行一次
  - 可以override destroy方法来实现自己的处理逻辑
  - 应用程序卸载时一定会调用destroy方法。

2.1.3. 【Servlet 的生命周期】Servlet 生命周期原理图

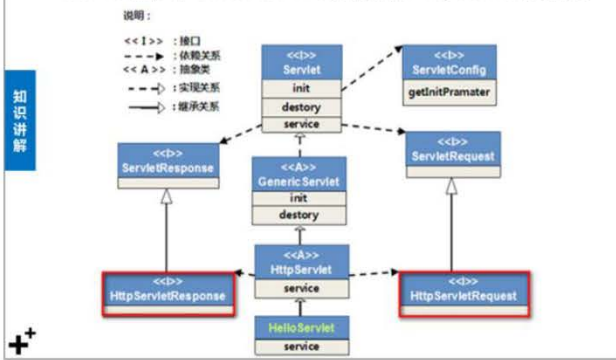
Servlet生命周期原理图



2.1.4. 【Servlet 的生命周期】Servlet 核心接口和类

Servlet核心接口与类

- Servlet接口，GenericServlet抽象类，HttpServlet抽象类



2.1.5. 【Servlet 的生命周期】Servlet 接口

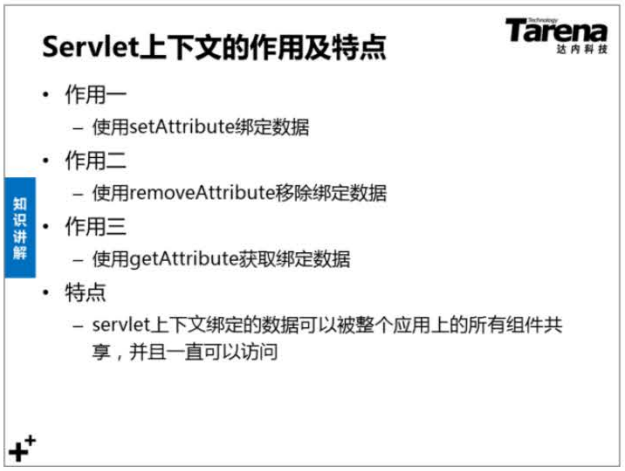
2.1.6. 【Servlet 的生命周期】Servlet 核心类

2.2. ServletContext

2.2.1. 【ServletContext】什么是 Servlet 上下文

2.2.2. 【ServletContext】如何获得 Servlet 上下文

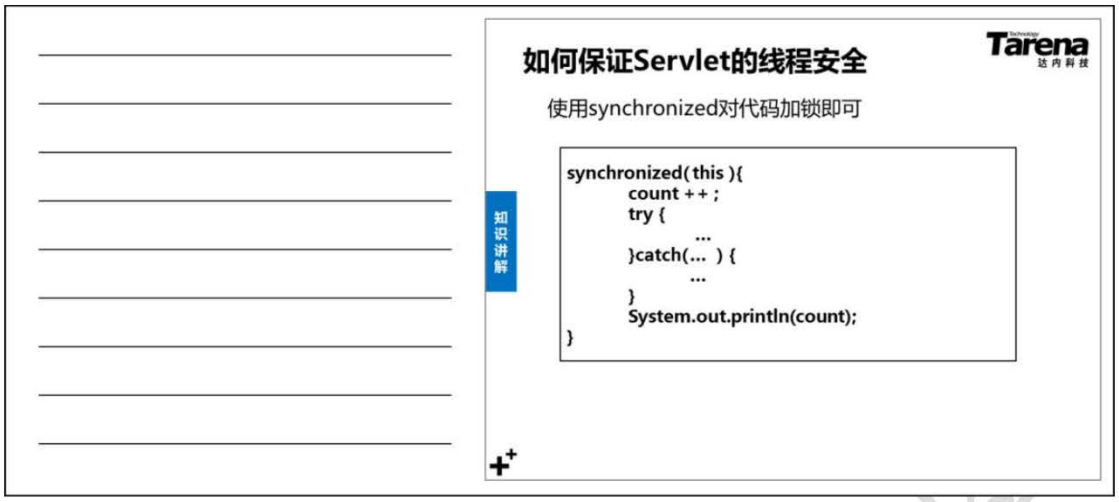
2.2.3. 【ServletContext】Servlet 上下文的作用及特点



2.3. Servlet 线程安全问题

2.3.1. 【Servlet 线程安全问题】为什么会有线程安全问题

2.3.2. 【Servlet 线程安全问题】如何保证 Servlet 的线程安全



达内IT培训集团

经典案例

1. 员工管理——使用 JDBC 删除员工信息

- **问题**

为员工管理添加删除的功能，即在查询结果页面中提供删除的操作链接，并实现在删除后回到查询页面的效果。如图-1 所示：

图- 1

- **方案**

在查询的 Servlet 中，输出每一行员工信息的同时，多输出一个列，内容为超链接，文字显示为“删除”字样，隐含的超链接地址为删除对应动作的 Servlet，并且为了区分每一个员工信息，在向删除地址发出请求的同时，传递一个当前选中的员工的 id 号，以 get 的方式提交给服务器，服务器获取该参数，依据 id 构建删除语句并执行，执行之后使用重定向技术请求列表页面，重新查询，生成新的查询结果页面。

- **步骤**

步骤一： 修改 ListEmpServlet，添加删除链接

如图-2 所示，输出“删除”超链接。

```
out.println("<tr><td>编号</td><td>姓名</td><td>薪水</td>" +
        "<td>年龄</td><td>操作</td></tr>");
while (rs.next()) {
    int id = rs.getInt("id");
    String name = rs.getString("name");
    double salary = rs.getDouble("salary");
    int age = rs.getInt("age");
    out.println("<tr>");
    out.println("<td>" + id + "</td>");
    out.println("<td>" + name + "</td>");
    out.println("<td>"+ salary + "</td>");
    out.println("<td>" + age + "</td>");
    out.println("<td><a href='delete?id="+id+"' " +
            "onclick=\"return confirm('是否确定删除"+name+"');\">删除</a></td>");
    out.println("</tr>");
}
```

图 - 2

步骤二： 新建 DeleteEmpServlet 类，处理删除动作

如图-3 所示，获取请求参数值 id，构建删除语句并执行。

```
//获取请求参数值中的id信息
response.setContentType("text/html;charset=UTF-8");
PrintWriter out =response.getWriter();
int id = Integer.parseInt(request.getParameter("id"));
//连接数据库，执行删除操作
Connection conn = null;
PreparedStatement stmt = null;
try {
    Class.forName("oracle.jdbc.OracleDriver");
    conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521:xe", "x", "x");
    stmt = conn.prepareStatement("delete from t_emp where id=?");
    stmt.setInt(1, id);
    stmt.executeUpdate();
    response.sendRedirect("list");
} catch (Exception e) {
```

获取链接里面的id

删除的sql语句

重定向到查询页面

图 - 3

步骤三： 修改 web.xml，配置 DeleteEmpServlet

```
<!-- 删除员工 -->
<servlet>
    <servlet-name>deleteEmpServlet</servlet-name>
    <servlet-class>emp.DeleteEmpServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>deleteEmpServlet</servlet-name>
    <url-pattern>/delete</url-pattern>
</servlet-mapping>
```

- **完整代码**

ListEmpServlet.java 文件代码：

```
package emp;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ListEmpServlet extends HttpServlet {
 /**
  * 查询员工信息列表
  */
 protected void service(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException {
     // 设置输出流编码
     response.setContentType("text/html;charset=UTF-8");
     PrintWriter out = response.getWriter();
     // JDBC 查询员工信息
     Connection conn = null;
     PreparedStatement stmt = null;
     ResultSet rs = null;
     try {
         Class.forName("oracle.jdbc.OracleDriver");
         conn = DriverManager.getConnection(
                 "jdbc:oracle:thin:@localhost:1521:xe", "xxx", "xxx");
         stmt = conn.prepareStatement("select * from t emp");
         rs = stmt.executeQuery();
         out.println("<html><head></head><body style='font-size:30px'>");
         out.println("<table  border='1'  cellspacing='0'  cellpadding='0'
width='600px'>");
         out.println("<caption>员工信息</caption>");
         out.println("<tr><td>编号</td><td>姓名</td><td>薪水</td><td>年龄
</td><td>操作</td></tr>");
         while (rs.next()) {
             int id = rs.getInt("id");
             String name = rs.getString("name");
             double salary = rs.getDouble("salary");
             int age = rs.getInt("age");
             out.println("<tr>");
             out.println("<td>" + id + "</td>");
             out.println("<td>" + name + "</td>");
             out.println("<td>"+ salary + "</td>");
             out.println("<td>" + age + "</td>");
             out.println("<td><a href='delete?id="+id+"' " +
     "onclick=\"return confirm('是否确定删除"+name+"');\">删除</a></td>");
             out.println("</tr>");
         }
         out.println("</table></body></html>");
     } catch (Exception e) {
         e.printStackTrace();
         out.println("系统异常，请重试");
     } finally {
         if (conn != null) {
             try {
                 conn.close();
             } catch (SQLException e) {
                 e.printStackTrace();
             }
         }
     }
 }
}
```

DeleteEmpServlet.java 文件代码：

```
package emp;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DeleteEmpServlet extends HttpServlet{
 /**
  * 实现删除员工信息的操作
  */
 protected void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //获取请求参数值中的 id 信息
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out =response.getWriter();
        int id = Integer.parseInt(request.getParameter("id"));
        //连接数据库，执行删除操作
        Connection conn = null;
        PreparedStatement stmt = null;
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            conn = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe", "x", "x");
            stmt = conn.prepareStatement("delete from t emp where id=?");
            stmt.setInt(1, id);
            stmt.executeUpdate();
            response.sendRedirect("list");
        } catch (Exception e) {
            e.printStackTrace();
            out.println("Error");
        } finally{
            if(conn != null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
 }
}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
 <!-- 增加员工 -->
 <servlet>
     <servlet-name>addEmpServlet</servlet-name>
```

```
        <servlet-class>emp.AddEmpServlet</servlet-class>
    </servlet>
    <!-- 查询员工信息 -->
    <servlet>
        <servlet-name>listEmpServlet</servlet-name>
        <servlet-class>emp.ListEmpServlet</servlet-class>
    </servlet>
    <!-- 删除员工 -->
    <servlet>
        <servlet-name>deleteEmpServlet</servlet-name>
        <servlet-class>emp.DeleteEmpServlet</servlet-class>
    </servlet>
    <!-- add -->
    <servlet-mapping>
        <servlet-name>addEmpServlet</servlet-name>
        <url-pattern>/add</url-pattern>
    </servlet-mapping>
    <!-- list -->
    <servlet-mapping>
        <servlet-name>listEmpServlet</servlet-name>
        <url-pattern>/list</url-pattern>
    </servlet-mapping>
    <!-- delete -->
    <servlet-mapping>
        <servlet-name>deleteEmpServlet</servlet-name>
        <url-pattern>/delete</url-pattern>
    </servlet-mapping>
   </web-app>
```

2. 员工管理——使用 JDBC 修改员工信息

- **问题**

为员工管理实现修改功能。查询结果中具备修改的链接，点击后将选中的员工信息加载到表单中，修改完毕后点击“修改”按钮，回到查询页面，查看修改后的结果。如图-4 所示：

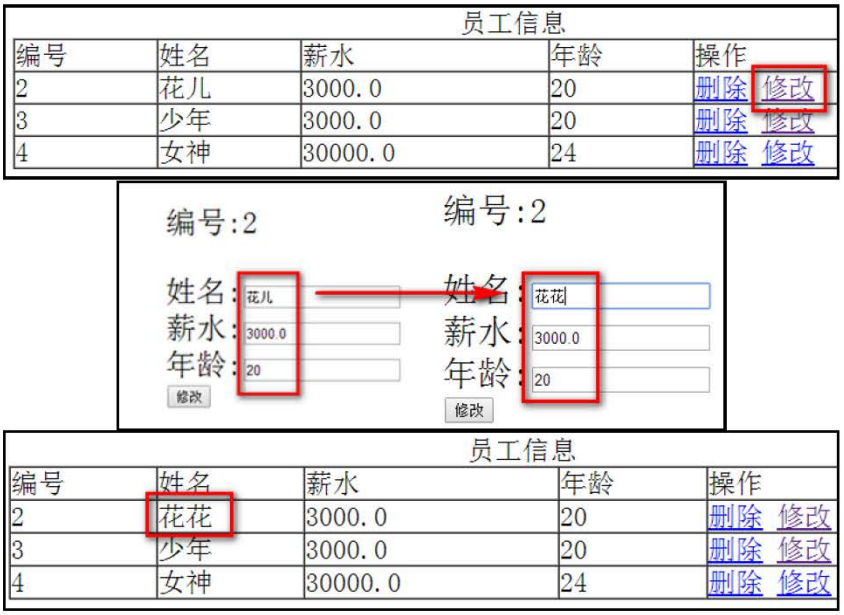


图 - 4

- **方案**

在查询页面增加“修改”的超链接，并按照所选员工的 id 查询出一条数据，加载到表单中。当修改结束后，点击表单的“修改”那妞时，不仅仅提交表单中的修改数据，同时使用隐藏表单域 将存储的该员工 id 信息一同提交 ,服务器端收到数据后构建更新语句并执行，执行结束后使用重定向技术重新请求查询页面，生成新的查询结果，确定更新操作已成功。

- **步骤**

步骤一： 修改 ListEmpServlet，增加“修改”链接

如图-5 所示为增加的修改链接。

```
while (rs.next()) {
    int id = rs.getInt("id");
    String name = rs.getString("name");
    double salary = rs.getDouble("salary");
    int age = rs.getInt("age");
    out.println("<tr>");
    out.println("<td>" + id + "</td>");
    out.println("<td>" + name + "</td>");
    out.println("<td>"+ salary + "</td>");
    out.println("<td>" + age + "</td>");
    out.println("<td><a href='delete?id="+id+"' " +
            "onclick=\"return confirm('是否确定删除"+name+"');\">删除</a>");
    out.println("<a href='load?id="+id+"'>修改</a></td>");
    out.println("</tr>");
}
out.println("</table></body></html>");
```

图 - 5

步骤二： 新建 LoadEmpServlet.java 文件

获取请求参数值 id，查询数据并构建表单显示数据，如图-6 所示：

```
// 获取请求参数值id
int id = Integer.parseInt(request.getParameter("id"));
// 根据Id查询数据库获取员工信息
Connection conn = null;
PreparedStatement stmt = null;
ResultSet rs = null;
try {
    Class.forName("oracle.jdbc.OracleDriver");
    conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521:xe", "xxx", "xx");
    stmt = conn.prepareStatement("select * from t_emp where id=?");
    stmt.setInt(1, id);
    rs = stmt.executeQuery();
    out.println("<html><head></head><body style='font-size:30px'>");
    if (rs.next()) {
        String name = rs.getString("name");
        double salary = rs.getDouble("salary");
        int age = rs.getInt("age");
        out.println("<form action='modify' method='post'>");
        out.println("编号:" + id + "<br>");
        out.println("<input type='hidden' name='id' value='"+id+"'/><br>")
        out.println("姓名:<input name='name' value='"+name+"'/><br>");
        out.println("薪水:<input name='salary' value='"+salary+"'/><br>");
        out.println("年龄:<input name='age' value='"+age+"'/><br>");
        out.println("<input type='submit' value='修改'/>");
        out.println("</form>");
```

获取参数id

构建查询语句

输出表单

输出隐藏表单域

图 - 6

步骤三： 新建 ModifyEmpServlet.java 文件

```
//1.获取请求参数值
int id = Integer.parseInt(request.getParameter("id"));
String name = request.getParameter("name");
double salary = Double.parseDouble(request.getParameter("salary"))
int age = Integer.parseInt(request.getParameter("age"));
//2.数据库操作
Connection conn = null;
PreparedStatement stmt = null;
try {
    Class.forName("oracle.jdbc.OracleDriver");
    conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521:xe", "xxx", "xxx");
    stmt = conn.prepareStatement("update t_emp set name=?," +
            "salary=?,age=? where id=?");
    stmt.setString(1, name);
    stmt.setDouble(2, salary);
    stmt.setInt(3, age);
    stmt.setInt(4, id);
    stmt.executeUpdate();
    response.sendRedirect("list");
```

获取参数包括id

构建sql语句

重定向到查询

图 - 7

步骤四： 修改 web.xml 文件实现配置

```
<!-- 加载员工 -->
<servlet>
    <servlet-name>loadEmpServlet</servlet-name>
    <servlet-class>emp.LoadEmpServlet</servlet-class>
</servlet>
<!-- 修改员工 -->
<servlet>
    <servlet-name>modifyEmpServlet</servlet-name>
    <servlet-class>emp.ModifyEmpServlet</servlet-class>
</servlet>
```

```
<!-- load -->
<servlet-mapping>
    <servlet-name>loadEmpServlet</servlet-name>
    <url-pattern>/load</url-pattern>
</servlet-mapping>
<!-- modify -->
<servlet-mapping>
    <servlet-name>modifyEmpServlet</servlet-name>
    <url-pattern>/modify</url-pattern>
</servlet-mapping>
```

图 - 8

- **完整代码**

ListEmpServlet.java 文件：

```
package emp;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ListEmpServlet extends HttpServlet {
 /**
  * 查询员工信息列表
  */
 protected void service(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException {
     // 设置输出流编码
     response.setContentType("text/html;charset=UTF-8");
     PrintWriter out = response.getWriter();
     // JDBC 查询员工信息
     Connection conn = null;
     PreparedStatement stmt = null;
     ResultSet rs = null;
     try {
         Class.forName("oracle.jdbc.OracleDriver");
         conn = DriverManager.getConnection(
                 "jdbc:oracle:thin:@localhost:1521:xe", "xxx", "xxx");
         stmt = conn.prepareStatement("select * from t_emp");
         rs = stmt.executeQuery();
         out.println("<html><head></head><body style='font-size:30px'>");
         out.println("<table  border='1'  cellspacing='0'  cellpadding='0'
width='600px'>");
         out.println("<caption>员工信息</caption>");
         out.println("<tr><td>编号</td><td>姓名</td><td>薪水</td><td>年龄
</td><td>操作</td></tr>");
         while (rs.next()) {
             int id = rs.getInt("id");
             String name = rs.getString("name");
             double salary = rs.getDouble("salary");
             int age = rs.getInt("age");
             out.println("<tr>");
             out.println("<td>" + id + "</td>");
             out.println("<td>" + name + "</td>");
             out.println("<td>"+ salary + "</td>");
             out.println("<td>" + age + "</td>");
             out.println("<td><a href='delete?id="+id+"' " +
             "onclick=\"return confirm('是否确定删除"+name+"');\">删除</a>");
             out.println("<a href='load?id="+id+"'>修改</a></td>");
             out.println("</tr>");
         }
         out.println("</table></body></html>");
     } catch (Exception e) {
         e.printStackTrace();
         out.println("系统异常，请重试");
     } finally {
         if (conn != null) {
             try {
                 conn.close();
             } catch (SQLException e) {
                 e.printStackTrace();
             }
         }
     }
 }
}
```

LoadEmpServlet.java 文件：

```
package emp;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoadEmpServlet extends HttpServlet {
 protected void service(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException {
     response.setContentType("text/html;charset=UTF-8");
     PrintWriter out = response.getWriter();
     // 获取请求参数值 id
     int id = Integer.parseInt(request.getParameter("id"));
     // 根据 Id 查询数据库获取员工信息
     Connection conn = null;
     PreparedStatement stmt = null;
     ResultSet rs = null;
     try {
         Class.forName("oracle.jdbc.OracleDriver");
         conn = DriverManager.getConnection(
                 "jdbc:oracle:thin:@localhost:1521:xe", "xxx", "xxx");
         stmt = conn.prepareStatement("select * from t emp where id=?");
         stmt.setInt(1, id);
         rs = stmt.executeQuery();
         out.println("<html><head></head><body style='font-size:30px'>");
         if (rs.next()) {
             String name = rs.getString("name");
             double salary = rs.getDouble("salary");
             int age = rs.getInt("age");
             out.println("<form action='modify' method='post'>");
             out.println("编号:" + id + "<br>");
             out.println("<input          type='hidden'          name='id'
value='"+id+"'/><br>");
             out.println("姓名:<input name='name' value='"+name+"'/><br>");
             out.println("      薪      水      :<input       name='salary'
value='"+salary+"'/><br>");
             out.println("年龄:<input name='age' value='"+age+"'/><br>");
             out.println("<input type='submit' value='修改'/>");
             out.println("</form>");
         }
         out.println("</body></html>");
     } catch (Exception e) {
         e.printStackTrace();
         out.println("系统异常，请重试");
     } finally {
         if (conn != null) {
             try {
                 conn.close();
             } catch (SQLException e) {
                 e.printStackTrace();
             }
```

```
            }
        }
    }
}
```

ModifyEmpServlet.java 文件

```
package emp;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ModifyEmpServlet extends HttpServlet{
 /**
  * 增加员工信息
  */
 protected void service(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        //0.解决中文乱码问题
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //1.获取请求参数值
        int id = Integer.parseInt(request.getParameter("id"));
        String name = request.getParameter("name");
        double salary = Double.parseDouble(request.getParameter("salary"));
        int age = Integer.parseInt(request.getParameter("age"));
        //2.数据库操作
        Connection conn = null;
        PreparedStatement stmt = null;
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            conn = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xxx", "xuze", "xxx");
            stmt = conn.prepareStatement("update t_emp set name=?,salary=?,age=?
where id=?");
            stmt.setString(1, name);
            stmt.setDouble(2, salary);
            stmt.setInt(3, age);
            stmt.setInt(4, id);
            stmt.executeUpdate();
            response.sendRedirect("list");
        } catch (Exception e) {
            e.printStackTrace();
            out.print("系统出现问题，稍后重试<br><p><a href='list.do'>员工信息列表
</a></p>");
        } finally{
            if(stmt!=null){
                try {
                    stmt.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
```

```
            }
            if(conn!=null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

web.xml 文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">

 <!-- 增加员工 -->
 <servlet>
     <servlet-name>addEmpServlet</servlet-name>
     <servlet-class>emp.AddEmpServlet</servlet-class>
 </servlet>
 <!-- 查询员工信息 -->
 <servlet>
     <servlet-name>listEmpServlet</servlet-name>
     <servlet-class>emp.ListEmpServlet</servlet-class>
 </servlet>
 <!-- 删除员工 -->
 <servlet>
     <servlet-name>deleteEmpServlet</servlet-name>
     <servlet-class>emp.DeleteEmpServlet</servlet-class>
 </servlet>
 <!-- 加载员工 -->
 <servlet>
     <servlet-name>loadEmpServlet</servlet-name>
     <servlet-class>emp.LoadEmpServlet</servlet-class>
 </servlet>
 <!-- 修改员工 -->
 <servlet>
     <servlet-name>modifyEmpServlet</servlet-name>
     <servlet-class>emp.ModifyEmpServlet</servlet-class>
 </servlet>

 <servlet-mapping>
     <servlet-name>addEmpServlet</servlet-name>
     <url-pattern>/add</url-pattern>
 </servlet-mapping>
 <!-- list -->
 <servlet-mapping>
     <servlet-name>listEmpServlet</servlet-name>
     <url-pattern>/list</url-pattern>
 </servlet-mapping>
 <!-- delete -->
 <servlet-mapping>
     <servlet-name>deleteEmpServlet</servlet-name>
     <url-pattern>/delete</url-pattern>
 </servlet-mapping>
 <!-- load -->
 <servlet-mapping>
```

```
        <servlet-name>loadEmpServlet</servlet-name>
        <url-pattern>/load</url-pattern>
    </servlet-mapping>
    <!-- modify -->
    <servlet-mapping>
        <servlet-name>modifyEmpServlet</servlet-name>
        <url-pattern>/modify</url-pattern>
    </servlet-mapping>
   </web-app>
```

3. 容器对 URI 的处理

- **问题**

请求到达服务器以后，服务器是如何处理 URI 请求资源路径的，在与 web.xml 文件中的映射进行比对时的原则是什么。

- **方案**

针对精确匹配、通配符匹配、后缀匹配三种模式修改 web.xml 文件中的配置，查看不同比对原则的访问结果。

- **步骤**

步骤一： 新建 URIServlet.java 文件

新建 URIServlet.java 文件，用于添加简单的输出，标识访问成功。

```
protected void service(HttpServletRequest req, HttpServ
        throws ServletException, IOException {
    System.out.println("这里是URIServlet的service方法");
}
```

图 - 9

步骤二： 在 web.xml 文件中添加精确匹配

为 URIServlet.java 文件添加映射，精确匹配即路径需要严格匹配，如图-10 所示：

```
<servlet>
    <servlet-name>uri1</servlet-name>
    <servlet-class>web.URIServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>uri1</servlet-name>
    <url-pattern>/abc/abc.html</url-pattern>
</servlet-mapping>
```

图 - 10

步骤三： 新建 uri.html 文件，测试精确匹配

新建 uri.html 文件，添加两个超链接，地址分别为 “abc.html”和“abc/abc.html”。

点击不同的连接查看是否能和 web.xml 文件的映射规则相对应。经过测试后，如果配置模式为"/abc/abc.html" 则路径中应用名称后面的地址需要严格相等 如果只写"abc.html"的话，是不符合匹配原则，无法实现找到 URIServlet 并执行的。

uri.html 文件如下图-11 所示：

图 - 11

步骤四： 修改 web.xml 文件为通配符匹配

修改 web.xml 中的匹配模式为通配符模式，使用*号代替任意字符，如图-12 所示：

```
<servlet-mapping>
    <servlet-name>uri1</servlet-name>
    <url-pattern>/abc/*</url-pattern>
</servlet-mapping>
```

图 - 12

步骤五： 修改 uri.html 文件，测试通配符匹配

修改 uri.html 文件，再次添加两个超链接，一个地址为"abc.html"，由于该通配符前面有 abc，则需要先精确匹配 abc 后再任意长度，所以地址为"abc.html"时，是不能够匹配成功的。第二个地址为"abc/abc/def.html"，第一个 abc 匹配成功后，不管后面写了任意长度任意字符都代表匹配成功，所以第二个地址可以成功访问到 URIServlet。如图-13 所示：

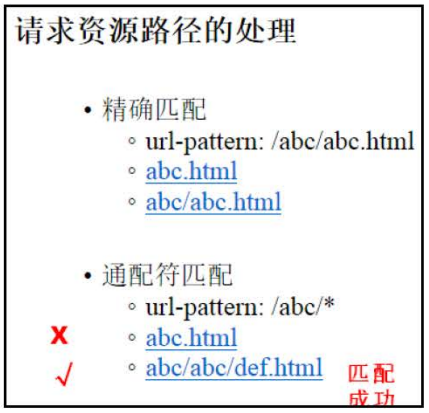


图 - 13

步骤六： 修改 web.xml 文件为后缀匹配

修改 web.xml 中的匹配模式为后缀匹配模式 ,使用*.do 来代替以 do 结尾的任意字符，如图-14 所示：

```
<servlet-mapping>
    <servlet-name>uri1</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

图 - 14

步骤七： 修改 uri.html 文件，测试后缀匹配

修改 uri.html 文件，再次添加两个超链接，一个地址为"abc.do"，另一个地址为"abc/abc.do"，不管 do 前面有多长的字符串统统由*匹配掉了，只要结尾是 do 就代表匹配成功，所以两个地址都能够访问到 URIServlet。如图-15 所示：

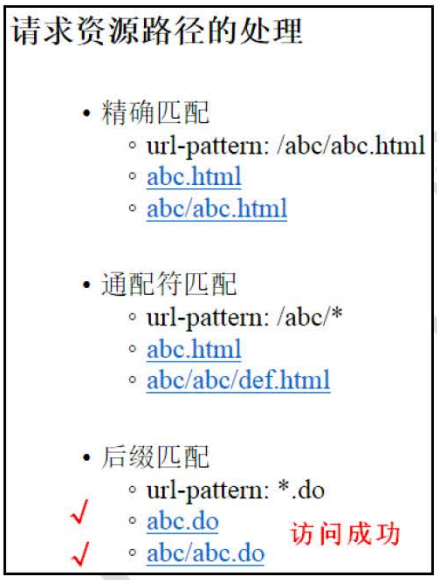


图 - 15

- **完整代码**

URIServlet.java 文件代码：

```
package web;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class URIServlet extends HttpServlet{

 protected void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
     System.out.println("这里是URIServlet的service方法");
 }
}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
 <!-- URI 匹配 -->
 <servlet>
     <servlet-name>uri1</servlet-name>
     <servlet-class>web.URIServlet</servlet-class>
 </servlet>
 <!-- URI 匹配 -->
  <!--
  <servlet-mapping>
     <servlet-name>uri1</servlet-name>
     <url-pattern>/abc/abc.html</url-pattern>
 </servlet-mapping>
<servlet-mapping>
     <servlet-name>uri1</servlet-name>
     <url-pattern>/abc/*</url-pattern>
 </servlet-mapping>
  -->
 <servlet-mapping>
     <servlet-name>uri1</servlet-name>
     <url-pattern>*.do</url-pattern>
 </servlet-mapping>
</web-app>
```

uri.html 文件代码：

```
<html>
  <head>
  <body style="font-size:24">
     <h3>请求资源路径的处理</h3>
    <ul>
     <li style="margin:40px">精确匹配
         <ul>
             <li>url-pattern: /abc/abc.html
             <li><a href="abc.html">abc.html</a>
             <li><a href="abc/abc.html">abc/abc.html</a>
         </ul>
     <li style="margin:40px">通配符匹配
         <ul>
             <li>url-pattern: /abc/*
             <li><a href="abc.html">abc.html</a>
             <li><a href="abc/abc/def.html">abc/abc/def.html</a>
         </ul>
     <li style="margin:40px">后缀匹配
         <ul>
             <li>url-pattern: *.do
             <li><a href="abc.do">abc.do</a>
             <li><a href="abc/abc.do">abc/abc.do</a>
         </ul>
    </ul>
  </body>
</html>
```

4. 员工管理——单一控制器实现员工信息管理

- **问题**

将员工管理的所有操作请求处理合并到一个 Servlet 中，实现请求的分发。

- **方案**

使用后缀匹配模式，将以 do 结尾的请求都提交到 ActionServlet 中，分析 do 前的操作请求种类，分发到不同的分支执行相应的动作。同时将 JDBC 以 dao 和实体的形式来实现，减少分支内的重复代码。

- **步骤**

步骤一： 新建 entity.Employee 类

新建与 t_emp 表对应的实体类，属性与表字段一一对应，添加 get/set 方法及构造、toString 方法。结构如图-16 所示：

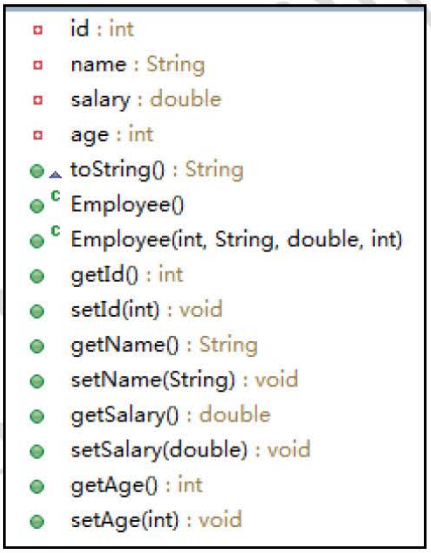


图 - 16

步骤二： 新建 dao.DBUtil 公共类

用于实现数据库的连接和断开的工具类，结构如图-17 所示：

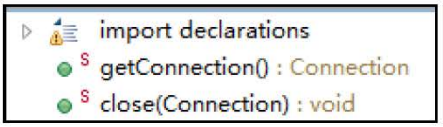


图 - 17

步骤三： 新建 dao.EmployeeDAO 类

建立针对 Employee 实体的数据操作类 dao。结构如图-18 所示：

```
import declarations
findAll() : List<Employee>
delete(int) : void
save(Employee) : void
findById(int) : Employee
modify(Employee) : void
```

图 - 18

步骤四： 新建 emp.ActionServlet 类

创建用于进行动作分发的 ActionServlet 类，通过 getRequestURI 获取请求资源路径，分析 do 前面的动作种类，执行不同的分支。

```
// 获取请求资源路径
String uri = request.getRequestURI();
// 获取请求资源路径中除应用名以外的部分
String action = uri.substring(uri.lastIndexOf("/") + 1,
        uri.lastIndexOf("."));

if (action.equals("list")) {
```

图 - 19

步骤五： 新建 addEmp.html 文件

新建用于完成增加员工的表单页面。表单的 action 地址为 ActionServlet 对应的 add.do，method 方式为 post。

```
<html>
  <head>
    <title>addEmp.html</title>
        <meta http-equiv="content-type" content
  </head>
  <body style="font-size:24px">
    <form action="add.do" method="post">
        <fieldset>
            <legend>员工管理</legend>
            姓名:<input name="name"/><br>
            薪水:<input name="salary"/><br>
            年龄:<input name="age"/><Br>
            <input type="submit" value="增加"/>
        </fieldset>
    </form>
  </body>
</html>
```

图 - 20

步骤六： 修改 web.xml 文件

使用后缀匹配模式配置 ActionServlet 的映射原则。

```
<!-- 合并员工管理多请求-->
<servlet>
    <servlet-name>actionServlet</servlet-name>
    <servlet-class>emp.ActionServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>actionServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

图 - 21

步骤七： 部署应用，访问

部署并访问工程，访问地址为 http://localhost:8080/day03/list.do。

- **完整代码**

entity.Employee.java 文件代码：

```
package entity;

/**
 * 实体类：员工 属性与 t_emp 表中的字段匹配
 */
public class Employee {
 private int id;
 private String name;
 private double salary;
 private int age;

 @Override
 public String toString() {
     return id + "  " + name + "  " + salary + "  " + age;
 }

 public Employee() {
     super();
 }

 public Employee(int id, String name, double salary, int age) {
     super();
     this.id = id;
     this.name = name;
     this.salary = salary;
     this.age = age;
 }

 public int getId() {
     return id;
 }

 public void setId(int id) {
     this.id = id;
 }

 public String getName() {
     return name;
 }

 public void setName(String name) {
```

```
        this.name = name;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

}
```

dao.DBUtil.java 文件代码：

```
package dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.taglibs.standard.tag.common.core.OutSupport;

/**
 * JDBC 管理连接的工具类，可以获取连接和关闭连接
 */
public class DBUtil {
 /**
  * 获取连接对象
  */
 public static Connection getConnection()throws Exception{
     Connection conn = null;
     try {
         Class.forName("oracle.jdbc.OracleDriver");
         conn = DriverManager.getConnection(
                 "jdbc:oracle:thin:@localhost:1521:xe", "xxx", "xxx");
     } catch (Exception e) {
         e.printStackTrace();
         throw e;
     }
     return conn;
 }
 /**
  * 关闭连接对象
  */
 public static void close(Connection conn) throws Exception{
     if(conn!=null){
         try {
             conn.close();
         } catch (SQLException e) {
             e.printStackTrace();
             throw e;
         }
     }
 }
```

```
    public static void main(String[] args) throws Exception{
        System.out.println(getConnection());
    }
}
```

dao.EmployeeDAO 文件代码：

```
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import entity.Employee;

/**
 * Employee 的数据操作对象，负责 Employee 的增删改查
 */
public class EmployeeDAO {

    /**
     * 查询所有员工
     */
    public List<Employee> findAll() throws Exception{
        List<Employee> emps = new ArrayList<Employee>();
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try{
            conn = DBUtil.getConnection();
            stmt = conn.prepareStatement("select * from t_emp");
            rs = stmt.executeQuery();
            while(rs.next()){
                Employee emp = new Employee(
                            rs.getInt("id"),
                            rs.getString("name"),
                            rs.getDouble("salary"),
                            rs.getInt("age")
                        );
                emps.add(emp);
            }
        }catch(Exception e){
            e.printStackTrace();
            throw e;
        }finally{
            DBUtil.close(conn);
        }
        return emps;
    }

    /**
     * 删除员工信息
     */
    public void delete(int id) throws Exception{
        Connection conn = null;
        PreparedStatement stmt = null;
        try{
            conn = DBUtil.getConnection();
            stmt = conn.prepareStatement("delete from t_emp where id=?");
            stmt.setInt(1, id);
            stmt.executeUpdate();
        }catch(Exception e){
```

```
            e.printStackTrace();
            throw e;
        }finally{
            DBUtil.close(conn);
        }
    }

    /**
     * 增加员工信息
     */
    public void save(Employee emp) throws Exception{
        Connection conn = null;
        PreparedStatement stmt = null;
        try{
            conn = DBUtil.getConnection();
            stmt     =     conn.prepareStatement("insert     into     t_emp
values(emp_id_seq.nextval,?,?,?)");
            stmt.setString(1, emp.getName());
            stmt.setDouble(2, emp.getSalary());
            stmt.setInt(3, emp.getAge());
            stmt.executeUpdate();
        }catch(Exception e){
            e.printStackTrace();
            throw e;
        }finally{
            DBUtil.close(conn);
        }
    }

    /**
     * 根据id查询员工信息
     */
    public Employee findById(int id) throws Exception{
        Employee emp = null;
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try{
            conn = DBUtil.getConnection();
            stmt = conn.prepareStatement("select * from t_emp where id=?");
            stmt.setInt(1, id);
            rs = stmt.executeQuery();
            if(rs.next()){
                emp = new Employee(
                            rs.getInt("id"),
                            rs.getString("name"),
                            rs.getDouble("salary"),
                            rs.getInt("age")
                        );
            }
        }catch(Exception e){
            e.printStackTrace();
            throw e;
        }finally{
            DBUtil.close(conn);
        }
        return emp;
    }

    /**
     * 保存修改员工信息
     */
    public void modify(Employee emp)throws Exception{
        Connection conn = null;
        PreparedStatement stmt = null;
        try{
```

```
            conn = DBUtil.getConnection();
            stmt = conn.prepareStatement("update t_emp set name=?,salary=?,age=?
where id=?");
            stmt.setString(1, emp.getName());
            stmt.setDouble(2, emp.getSalary());
            stmt.setInt(3, emp.getAge());
            stmt.setInt(4, emp.getId());
            stmt.executeUpdate();
        }catch(Exception e){
            e.printStackTrace();
            throw e;
        }finally{
            DBUtil.close(conn);
        }
    }
}
```

emp.ActionServlet 文件代码：

```
package emp;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import dao.EmployeeDAO;
import entity.Employee;

public class ActionServlet extends HttpServlet {
 @Override
 protected void service(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException
{

    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    // 获取请求资源路径
    String uri = request.getRequestURI();
    // 获取请求资源路径中除应用名以外的部分
    String action = uri.substring(uri.lastIndexOf("/") + 1,
            uri.lastIndexOf("."));

    if (action.equals("list")) {
        try {
            EmployeeDAO dao = new EmployeeDAO();
            List<Employee> emps = dao.findAll();
            out.println("<table>");
            out.println("<caption>员工信息列表</caption>");
            out.println("<tr><td>编号</td><td>姓名</td><td>薪水</td><td>年龄
</td><td>操作</td></tr>");
            for (Employee emp : emps) {
                out.println("<tr>");
                out.println("<td>" + emp.getId() + "</td>");
                out.println("<td>" + emp.getName() + "</td>");
                out.println("<td>" + emp.getSalary() + "</td>");
                out.println("<td>" + emp.getAge() + "</td>");
```

```
                out.println("<td><a href='#'>删除</a>");
                out.println("<a href='#'>修改</a></td>");
                out.println("</tr>");
            }
            out.println("</table>");
            out.println("<a href='addEmp.html'>增加新员工</a>");
        } catch (Exception e) {
            e.printStackTrace();
            out.print("系统繁忙");
        }
    } else if (action.equals("add")) {
        String name = request.getParameter("name");
        double salary = Double.parseDouble(request.getParameter("salary"));
        int age = Integer.parseInt(request.getParameter("age"));
        try {
            Employee emp = new Employee();
            emp.setName(name);
            emp.setSalary(salary);
            emp.setAge(age);
            EmployeeDAO dao = new EmployeeDAO();
            dao.save(emp);
            response.sendRedirect("list.do");
        } catch (Exception e) {
            e.printStackTrace();
            out.print("系统繁忙");
        }
    }
}

}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">

 <!-- 合并员工管理多请求-->
 <servlet>
     <servlet-name>actionServlet</servlet-name>
     <servlet-class>emp.ActionServlet</servlet-class>
 </servlet>
 <servlet-mapping>
     <servlet-name>actionServlet</servlet-name>
     <url-pattern>*.do</url-pattern>
 </servlet-mapping>

</web-app>
```

addEmp.html 文件代码：

```
<html>
  <head>
    <title>addEmp.html</title>
     <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  </head>
  <body style="font-size:24px">
    <form action="add.do" method="post">
```

```
        <fieldset>
            <legend>员工管理</legend>
            姓名:<input name="name"/><br>
            薪水:<input name="salary"/><br>
            年龄:<input name="age"/><Br>
            <input type="submit" value="增加"/>
        </fieldset>
      </form>
    </body>
  </html>
```

5. Servlet 的生命周期

- **问题**

Servlet 生命周期的四个阶段的执行时机。

- **方案**

创建一个 LifeServlet 类，添加构造方法、init 方法的重写、destroy 方法的重写及 service 方法，通过部署动作、访问动作、停止服务器动作观察方法的执行情况及执行次数。

- **步骤**

步骤一： 创建 LifeServlet 类，实现四个阶段对应的四个方法

```
public class LifeServlet extends HttpServlet {
    /**
     * 阶段1：实例化
     */
    public LifeServlet() {
        System.out.println("1.Constructor is running ...");
    }
    /**
     * 阶段2：初始化
     */
    @Override
    public void init() throws ServletException {
        System.out.println("2.Init is running ...");
    }
    /**
     * 阶段3：就绪
     */
    protected void service(HttpServletRequest req, HttpServle

    /**
     * 阶段4：销毁
     */
    @Override
    public void destroy() {
        super.destroy();
        System.out.println("4.Destroy is running...");
    }
```

图 - 22

步骤二： 修改 web.xml 完成配置

```
<servlet>
    <servlet-name>lifeServlet</servlet-name>
    <servlet-class>web.LifeServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>lifeServlet</servlet-name>
    <url-pattern>/life</url-pattern>
</servlet-mapping>
```

图 - 23

步骤三： 访问 Servlet 查看输出

部署应用，启动服务，此时输出中没有四个方法的任何输出，在地址栏中输入 url 后，在控制台输出如图-24 内容。构造和初始化及 service 都被指定，但构造和初始化只会执行这一次，再次刷新页面，会发现只有 service 方法被执行。如图-25 所示：

图 – 24

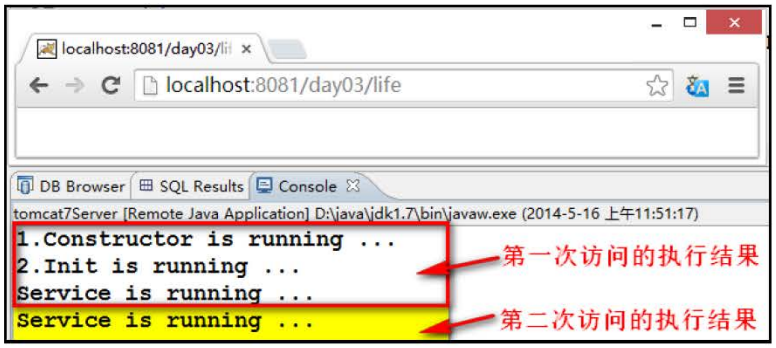


图 - 25

步骤四： 修改 web.xml 文件，添加<load-on-startup>配置

添加配置说明，使得 Servlet 的实例化初始化发生在服务器启动时或部署时。

```
<servlet>
    <servlet-name>lifeServlet</servlet-name>
    <servlet-class>web.LifeServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

图 - 26

步骤五： 重新部署，不访问查看输出

重新部署程序，但不在地址栏中访问应用，会发现实例化和初始化阶段已经被执行了。

```
DB Browser  SQL Results  Console
tomcat7Server [Remote Java Application] D:\java\jdk1.7\bin\javaw.exe (2014-5-16 上午11:59:23)
1.Constructor is running ...
2.Init is running ...
五月 16, 2014 11:59:49 上午 org.apache.catalina.core.Stand
信息: Reloading Context with name [/day03] is completed
```

部署后立即执行

图 - 27

步骤六： 停止服务查看输出

停止服务器，第 4 个阶段销毁阶段执行。如图-28 所示：

```
DB Browser  SQL Results  Console
<terminated> tomcat7Server [Remote Java Application] D:\java\jdk1.7\bin\javaw.exe (2014-5-16 T
4.Destroy is running...
```

图 - 28

步骤七： 修改 web.xml 文件，添加<init-param>配置

增加 init-param 配置，即增加 servlet 的初始参数，可以在初始化阶段读取该值。

```
<servlet>
    <servlet-name>lifeServlet</servlet-name>
    <servlet-class>web.LifeServlet</servlet-class>
    <init-param>
        <param-name>company</param-name>
        <param-value>Tarena</param-value>
    </init-param>
    <init-param>
        <param-name>address</param-name>
        <param-value>Beijing</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

图 - 29

步骤八： 修改 service 方法获取 init-param 参数并输出

初始参数的获取，可以使用 ServletConfig 对象的 getInitParameter 方法，而 ServletConfig 对象使用 getServletConfig（）方法获取。

```
/**
 * 阶段3：就绪
 */
protected void service(HttpServletRequest req, HttpServletRe
        throws ServletException, IOException {
    //读取初始化参数
    ServletConfig config = getServletConfig();
    String name = config.getInitParameter("company");
    String address = config.getInitParameter("address");
    System.out.println("Service is running ..." );
    System.out.println("初始参数为：" + name+"    " + address);
}
```

图 - 30

步骤九： 访问 service，查看参数的输出情况

图 - 31

- **完整代码**

LifeServlet.java 文件代码：

```
package web;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LifeServlet extends HttpServlet {
 /**
  * 阶段 1：实例化
  */
 public LifeServlet() {
     System.out.println("1.Constructor is running ...");
 }
 /**
  * 阶段 2：初始化
  */
 @Override
 public void init() throws ServletException {
     System.out.println("2.Init is running ...");
 }
 /**
  * 阶段 3：就绪
  */
 protected void service(HttpServletRequest req, HttpServletResponse resp)
         throws ServletException, IOException {
     //读取初始化参数
     ServletConfig config = getServletConfig();
     String name = config.getInitParameter("company");
     String address = config.getInitParameter("address");
     System.out.println("Service is running ..." );
     System.out.println("初始参数为：" + name+"   " + address);
 }

 /**
  * 阶段 4：销毁
  */
 @Override
 public void destroy() {
```

```
        super.destroy();
        System.out.println("4.Destroy is running...");
    }

}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app_2_5.xsd">
 <!-- 生命周期 -->
 <servlet>
     <servlet-name>lifeServlet</servlet-name>
     <servlet-class>web.LifeServlet</servlet-class>
     <init-param>
         <param-name>company</param-name>
         <param-value>Tarena</param-value>
     </init-param>
     <init-param>
         <param-name>address</param-name>
         <param-value>Beijing</param-value>
     </init-param>
     <load-on-startup>1</load-on-startup>
 </servlet>
 <!-- 生命周期 -->
 <servlet-mapping>
     <servlet-name>lifeServlet</servlet-name>
     <url-pattern>/life</url-pattern>
 </servlet-mapping>
</web-app>
```

6. ServletContext 对象（Servlet 上下文对象）

- **问题**

实现访问网站总人数的记录，以及基于某一特定起点的访问记录。

- **方案**

由于网站中的资源较多，要想保留每一次的访问计数则需要一个从应用一启动就存在的空间，并且应用中的所有资源都能访问到这个存储空间，所以使用 ServletContext 及 Servlet 上下文对象保存每一次的访问计数。当计数需要基于某一个特定的数字开始时，可以使用上下文配置参数，存储在 web.xml 文件中，当应用启动时就可以读取到这个初始值，再实现计数。

- **步骤**

步骤一： 新建 Context01Servlet.java 和 Context02Servlet.java 文件

两个文件内容一致，获取上下文对象后判断是否是第一次访问，是第一次则初始值为 1，

不是第一次就将上次的值增 1，如图-32 所示：

```
//获取全局的上下文对象
ServletContext context = getServletContext();
Object count = context.getAttribute("count");
if(count==null){
    context.setAttribute("count", 1);
}else{
    context.setAttribute("count", Integer.parseInt(count.toString())+1);
}
out.print("总浏览量为："+context.getAttribute("count"));
```

图 - 32

步骤二：配置 web.xml 文件

为两个 ContextServlet 添加配置：

```
<servlet>
    <servlet-name>context01Servlet</servlet-name>
    <servlet-class>web.Context01Servlet</servlet-class>
</servlet>
<servlet>
    <servlet-name>context02Servlet</servlet-name>
    <servlet-class>web.Context02Servlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>context01Servlet</servlet-name>
    <url-pattern>/context01</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>context02Servlet</servlet-name>
    <url-pattern>/context02</url-pattern>
</servlet-mapping>
```

图 - 33

步骤三：访问不同的 Servlet 查看计数效果

访问两个 Servlet，刷新后计数递增，并且递增后在另一个 servlet 中访问会基于这个递增后的数字，如图-34 所示：

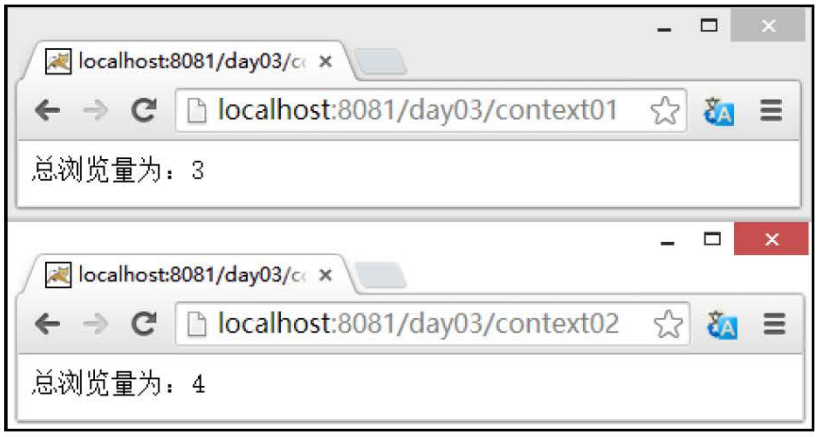


图 - 34

步骤四：修改 web.xml 文件，增加 context-param 配置

为该应用增加上下文初始参数如图-35 所示：一个 count-1000 的键值对。通过上下文对象的 getInitParameter 方法可以获取该值。

```
<context-param>
    <param-name>count</param-name>
    <param-value>1000</param-value>
</context-param>
<servlet>
    <servlet-name>context01Servlet</servlet-name>
    <servlet-class>web.Context01Servlet</servlet-class>
</servlet>
<servlet>
    <servlet-name>context02Servlet</servlet-name>
    <servlet-class>web.Context02Servlet</servlet-class>
</servlet>
```

图 - 35

步骤五：修改 Context01 和 Context02,使计数从设定的初值开始

修改两个文件的代码，从初始参数开始计数，如图-36 所示：

```
//获取全局的上下文对象
ServletContext context = getServletContext();
Object count = context.getAttribute("count");
if(count==null){
    context.setAttribute("count", context.getInitParameter("count"));
    //context.setAttribute("count", 1);
}else{
    context.setAttribute("count", Integer.parseInt(count.toString())+1);
}
```

图 - 36

步骤六：访问不同的 Servlet 查看计数效果

重新部署后，交替访问两个 ContextServlet，可以看到计数从初始参数开始。

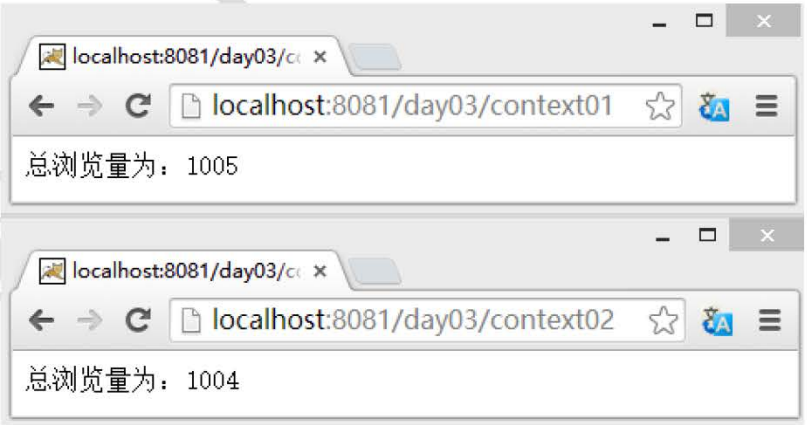


图 - 37

- **完整代码**

Context01Servlet.java 和 Context02Servlet.java 文件代码如下：

```
package web;
```

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Context01Servlet extends HttpServlet{
 @Override
 protected void service(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    //获取全局的上下文对象
    ServletContext context = getServletContext();
    Object count = context.getAttribute("count");
    if(count==null){
        context.setAttribute("count", context.getInitParameter("count"));
        //context.setAttribute("count", 1);
    }else{
        context.setAttribute("count",
Integer.parseInt(count.toString())+1);
    }
    out.print("总浏览量为："+context.getAttribute("count"));
 }
}
```

web.xml 文件代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">
 <!-- Servlet 上下文 -->
 <context-param>
     <param-name>count</param-name>
     <param-value>1000</param-value>
 </context-param>
 <servlet>
     <servlet-name>context01Servlet</servlet-name>
     <servlet-class>web.Context01Servlet</servlet-class>
 </servlet>
 <servlet>
     <servlet-name>context02Servlet</servlet-name>
     <servlet-class>web.Context02Servlet</servlet-class>
 </servlet>

 <servlet-mapping>
     <servlet-name>context01Servlet</servlet-name>
     <url-pattern>/context01</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
     <servlet-name>context02Servlet</servlet-name>
     <url-pattern>/context02</url-pattern>
 </servlet-mapping>
</web-app>
```

7. Servlet 线程安全

- **问题**

每个 Servlet 只会创建一个对象实例，多个请求同时访问这个实例时，如果有修改属性的操作，那么就会有线程安全的隐患，也就是说 Servlet 是线程不安全的。如何避免线程不安全带来的后果？

- **方案**

使用 Apache 组织开发 Apache Jmeter 进行压力测试，模拟 5 个请求，观察线程在同时修改私有属性时产生的后果。修改 Servlet 中的代码，增加同步的功能以避免线程不安全带来的后果。

- **步骤**

步骤一： 新建 ThreadSafeServlet 类

在这个 Servlet 中增加一个私有属性，service 方法中修改属性。模拟网络延迟，线程休眠 1 秒后再输出这个属性。如图-38 所示：

```
/** 私有属性count */
private int count = 0;          ← 增加私有属性

@Override
protected void service(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOExcepti

    count++;                    ← 修改属性值
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println(Thread.currentThread().getName() + ":" + count);
                                                          输出查看
}
```

图 - 38

步骤二： 修改 web.xml 配置映射

```
<!-- 线程安全 -->
<servlet>
    <servlet-name>threadSafeServlet</servlet-name>
    <servlet-class>web.ThreadSafeServlet</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>threadSafeServlet</servlet-name>
    <url-pattern>/safe</url-pattern>
</servlet-mapping>
```

图 - 39

步骤三： 使用 Jmeter 模拟 5 个 HTTP 请求

下载 Jmeter 之后，解压缩，使用命令行工具进入到解压后的 bin 目录下，输入“jmeter”代表启动 Jmeter，然后新建测试计划→线程组。如图-40 所示：

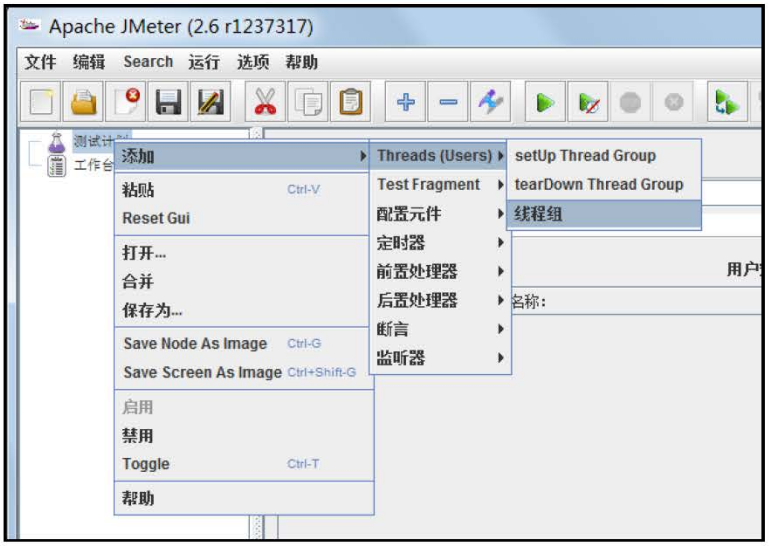


图 - 40

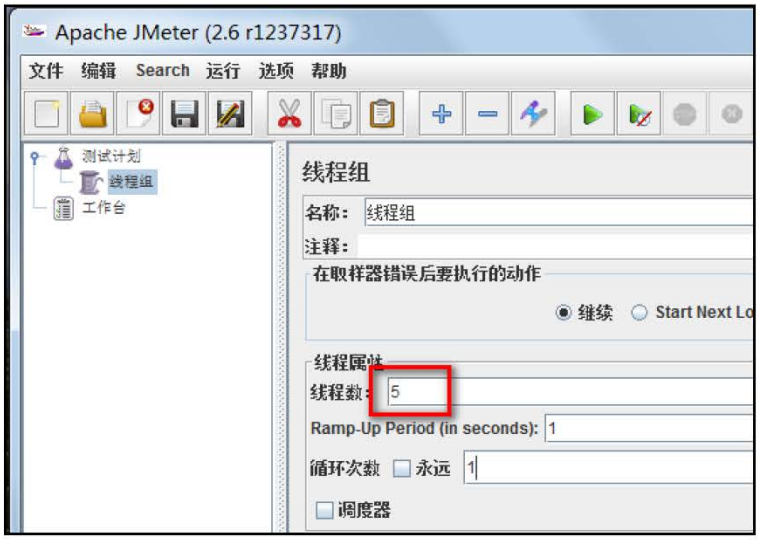


图 - 41

改写完线程数为 5 以后，在线程组上右键，新建 HTTP 请求，配置请求参数成功后，即可启动。

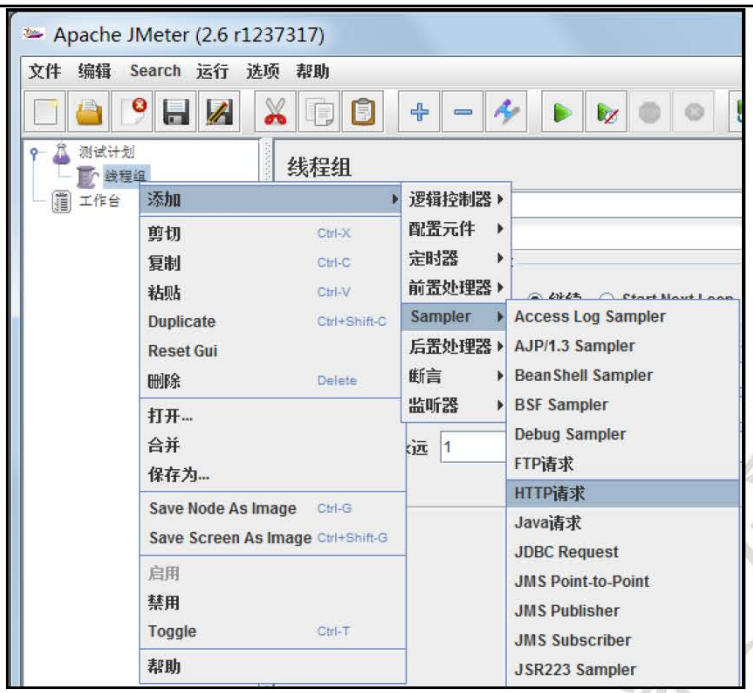


图 – 42

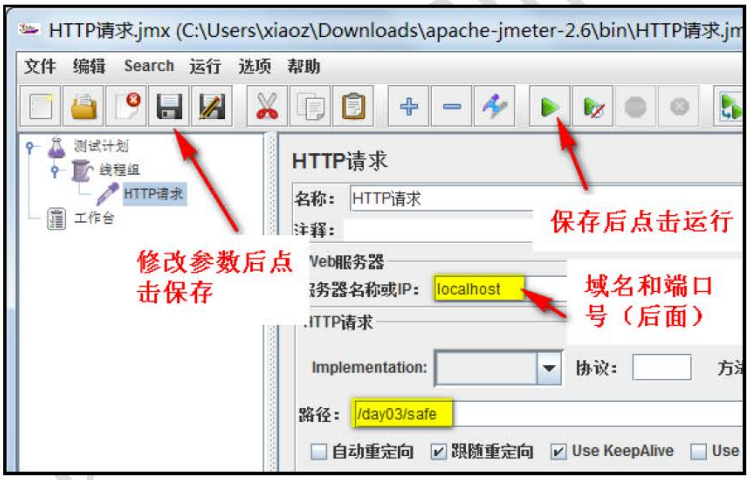


图 - 43

步骤四： 查看控制台输出结果

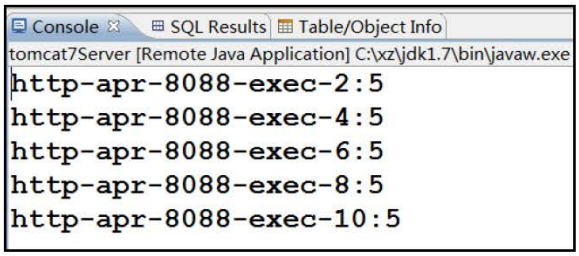


图 – 44

从结果中可以看到五个线程产生了修改值的冲突问题，也就是线程不安全现象。

步骤五： 修改 ThreadSafeServlet，增加同步功能

```
synchronized (this) {
    count++;
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println(Thread.currentThrea
}
```

图 - 45

步骤六： 运行 Jmeter 再次发送 5 个请求

步骤七： 查看控制台输出结果

```
Console  SQL Results  Table/Object Info
tomcat7Server [Remote Java Application] C:\xz\jdk1.7\bin\javaw.exe
http-apr-8088-exec-2:1
http-apr-8088-exec-10:2
http-apr-8088-exec-8:3
http-apr-8088-exec-6:4
http-apr-8088-exec-4:5
```

图 – 46

根据图-46 可以看到，使用同步后保证了线程安全，不会产生访问冲突的现象。

- **完整代码**

ThreadSafeServlet.java 文件代码：

```
package web;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ThreadSafeServlet extends HttpServlet {

 /** 私有属性count*/
 private int count = 0;

 @Override
 protected void service(HttpServletRequest request,
         HttpServletResponse response) throws ServletException, IOException
{
     synchronized (this) {
         count++;
         try {
             Thread.sleep(1000);
         } catch (InterruptedException e) {
             e.printStackTrace();
         }
         System.out.println(Thread.currentThread().getName() + ":" + count);
     }
```

```
    }
}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app 2 5.xsd">

 <!-- 线程安全 -->
 <servlet>
     <servlet-name>threadSafeServlet</servlet-name>
     <servlet-class>web.ThreadSafeServlet</servlet-class>
 </servlet>
 <servlet-mapping>
     <servlet-name>threadSafeServlet</servlet-name>
     <url-pattern>/safe</url-pattern>
 </servlet-mapping>
</web-app>
```

达内IT培训

课后作业

1. 简述什么是重定向。

2. 实现 NetCTOSS 系统中的帐务帐号的增删改查功能。

3. 简述 URI 和 URL 的区别， 如何获取 ？

4. Servlet 生命周期分哪几个阶段?什么时候执行?

5. 如何配置 Servlet 的初始参数,以及如何获取该初始参数?

6. 简述什么是 Servlet 上下文。

7. Servlet 是否是线程安全的,如何解决?

Servlet和JSP(上)

Unit04

知识体系.....Page 133

JSP 基本语法	JSP 的由来	为什么有 JSP 规范
		什么是 JSP
	JSP 编写规范	如何编写 JSP
		JSP 页面中的 HTML 代码
		JSP 页面中的注释
		JSP 页面中的 Java 代码
		JSP 表达式
		JSP 小脚本
		JSP 声明
		JSP 页面中的指令
		Page 指令
		Include 指令
		JSP 页面中的隐含对象
JSP 运行原理	JSP 运行原理	JSP 是如何运行的
		JSP 如何转换为 Java
		如何将静态页面转化为动态页面

经典案例.....Page 140

JSP 基本元素练习	如何编写 jsp
	JSP 页面中的 HTML 代码
	JSP 页面中的注释
	JSP 页面中的 Java 代码
	JSP 表达式
	JSP 小脚本
	JSP 声明
JSP 指令练习	JSP 页面中的指令
	page 指令
	include 指令
员工管理——使用 JSP 实现员工信息列表 1	JSP 页面中的隐含对象

达内IT培训集团

员工管理——使用 JSP 实现员工信息列表 2	JSP 是如何运行的
	JSP 如何转换为 Java
	如何将静态页面转化为动态页面

课后作业.....Page 152

1. JSP 基本语法

1.1. JSP 的由来

1.1.1. 【JSP 的由来】为什么有 JSP 规范

为什么有JSP规范

- Servlet技术产生以后，在使用时最麻烦的是使用大量的out.print语句输出页面。这样的形式在系统变更、维护、预览效果时都不能方便快捷的完成任务，于是推出JSP这种技术，用来将Servlet中负责显示的语句抽取出来

```
class   xxxServlet{
   . . . service ( ) {
      . . . . . . .
      out.println( "<html>" );
      . . . . . .
      out.println( "</html>" );
      . . . . . . .
```

1.1.2. 【JSP 的由来】什么是 JSP

什么是JSP

- Sun公司制定的一种服务器端动态页面技术的组件规范。JSP是一个以".jsp"为后缀的文件，在该文件中，主要是HTML和少量的Java代码。JSP文件会被容器转换成一个Servlet类，然后执行。

```
<html>
  <head>
  </head>
   <body>
     <table>
        java代码 ...  ...
     </table>
        java代码...  ...
  </body>
</html>
```

1.2. JSP 编写规范

1.2.1. 【JSP 编写规范】如何编写 JSP

如何编写JSP

- step1，写一个以".jsp"为后缀的文件
- step2，在该文件中，可以包含如下的内容：
  - HTML（CSS，JavaScript）
  - 注释
  - Java代码
  - 指令
  - 隐含对象

1.2.2. 【JSP 编写规范】JSP 页面中的 HTML 代码

JSP页面中的HTML代码

解说识知

- JSP页面中的HTML包括：
  - HTML标记
  - CSS
  - JavaScript
- 像编写HTML页面一样编写即可
- 作用：控制页面在浏览器中显示的效果
- 转译成Servlet时的规则
  - 成为Servlet中service（ ）方法的out.write 语句

1.2.3. 【JSP 编写规范】JSP 页面中的注释

JSP页面中的注释

解说识知

- 语法：
  ① <! -- 注释内容 -- >
  HTML注释，注释中的内容如果包含Java代码，这些Java代码会被执行
  ② <%-- 注释内容 -- %>
  JSP特有的注释，如果注释的内容中出现Java代码，会被忽略

1.2.4. 【JSP 编写规范】JSP 页面中的 Java 代码

JSP页面中的Java代码

解说识知

- JSP页面中的Java代码，包含以下三种：
  - JSP 表达式
  - JSP 小脚本
  - JSP 声明
- 编写位置
  - 页面的任意位置
- 作用
  - 控制页面中可变内容的产生

1.2.5. 【JSP 编写规范】JSP 表达式

JSP表达式

- 语法规则： <%= … … %>
- 合法内容：变量、变量加运算符组合的表达式、有返回值的方法
- 转译成Servlet时的规则：在service（）方法中用out.print语句输出该变量、表达式、方法的值

解说识知

JSP表达式（续）

- 例如：

```
<p>The square root of 5 is <%=Math.sqrt(5)%></p>
```

转换成

```
out . write( "<p>The square root of 5 is " ) ;
out . print( Math.sqrt( 5 ) ) ;
out . write( "</p>" );
```

解说识知

1.2.6. 【JSP 编写规范】JSP 小脚本

JSP 小脚本

- 语法规则： <% … … %>
- 合法内容：能够写在方法里的Java代码片段都可以作为小脚本
- 转译成Servlet时的规则：原封不动成为Servlet类的service（）方法里面的一段代码

解说识知

JSP小脚本（续）

解详识知

```
<%
    String  name = request.getParameter( "name" );
    if( name!=null && !name.equals( "" )){
%>
    <p>Your  name  is <%= name %> </p>
<%
    }
%>
```

转换成如下代码插入到service方法中

```
String  name = request.getParameter( "name" );
if( name!=null && !name.equals( "" )){
    out.write( "<p>Your  name  is " );
    out.print( name );
    out.write( " </p>" );
}
```

1.2.7. 【JSP 编写规范】JSP 声明

JSP声明

解详识知

- 语法规则： <%！ … … %>
- 合法内容：成员属性或成员方法的声明
- 转译成Servlet时的规则：成为JSP页面转译成的Servlet类中成员属性或成员方法

JSP声明（续）

解详识知

```
<%！
    public  String  getResult ( ) {
        //… …
    }
%>
```

将代码整体插入到Servlet类中

```
public  class  Index_JSP extends JSPBase{
    public  String  getResult ( ) {
        //… …
    }
    public void  service( ) . . .
}
```

1.2.8. 【JSP 编写规范】JSP 页面中的指令

JSP页面中的指令

- 语法规则：<%@指令名　属性=值 %>
- 常用指令：
  - page指令
  - include指令
  - taglib指令
- 作用
  - 控制JSP在转译成Servlet类时生成的内容

解说识知

1.2.9. 【JSP 编写规范】Page 指令

page指令

- 作用：用于导包、设置页面属性
- 例如：

```
<%-- 导包 -- >
<%@ page import = "java.util.*" %>
<%@ page import = "java.util.* , java.sql.*" %>

<%--设置response.setContentType ( ) 方法的参数值-- >
<%@ page contentType = "image/gif" %>

<%--设置容器读取该文件时的解码-- >
<%@ page pageEncoding= "UTF-8" %>
```

解说识知

1.2.10. 【JSP 编写规范】Include 指令

include指令

- 作用：在JSP页面转换成Servlet时，能够将其他文件包含进来。可以包含JSP文件也可以是静态的HTML文件。
- 通过该语句能方便的在每个JSP页面中包含导航栏、版权声明、logo等。
- 语法：

```
<%@ include file = "url" %>
```

- 例如

```
<%@include file = "header.html" %>
<%@include file = "footer.html" %>
```

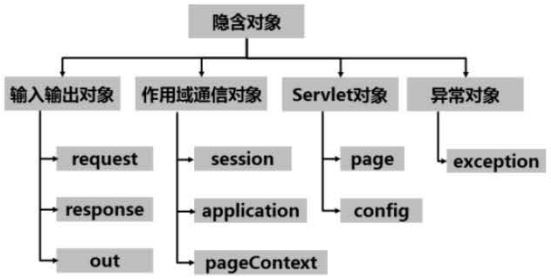
解说识知

1.2.11. 【JSP 编写规范】JSP 页面中的隐含对象

JSP页面中的隐含对象

- 什么是隐含对象？
  - 容器自动创建，在JSP文件中可以直接使用的对象
- 作用：
  - JSP预先创建的这些对象可以简化对HTTP请求、响应信息的访问

JSP页面中的隐含对象（续1）



JSP页面中的隐含对象（续2）

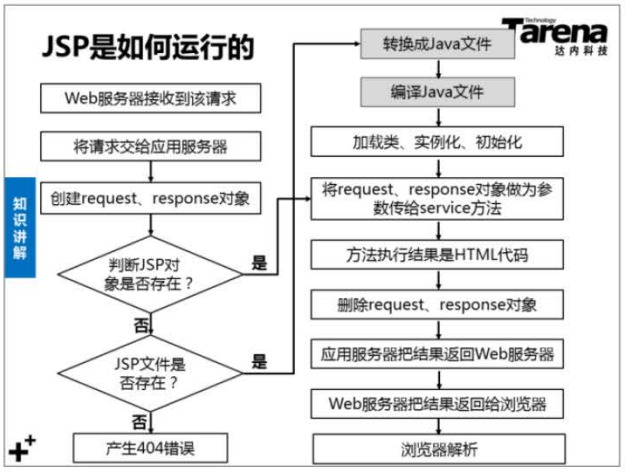
- JSP页面中可使用的隐含对象如下：

隐含对象	类型	说明
request	HttpServletRequest	请求信息
response	HttpServletResponse	响应信息
out	JSPWriter	输出的数据流
session	HttpSession	会话
application	ServletContext	全局的上下文对象
pageContext	PageContext	JSP页面上下文
page	Object	JSP页面本身
config	ServletConfig	Servlet配置对象
exception	Throwable	捕获网页异常

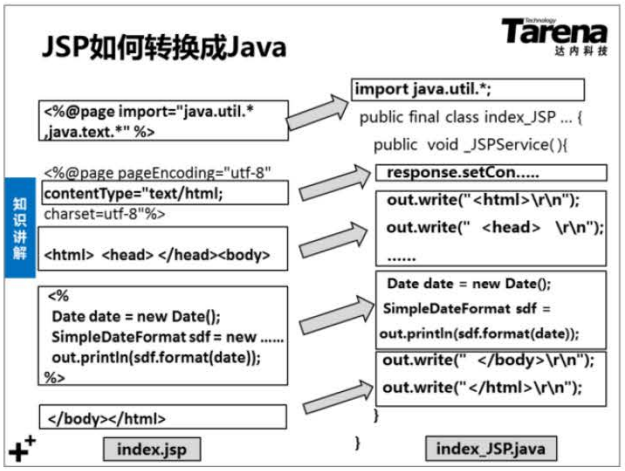
2. JSP 运行原理

2.1. JSP 运行原理

2.1.1. 【JSP 运行原理】JSP 是如何运行的



2.1.2. 【JSP 运行原理】JSP 如何转换为 Java



2.1.3. 【JSP 运行原理】如何将静态页面转化为动态页面

经典案例

1. JSP 基本元素练习

- **问题**

在 JSP 页面中输出 20 行“Hello JSP”。

- **方案**

使用表达式、小脚本、注释配合 HTML 标记的方式实现动态页面的生成。

- **步骤**

步骤一： 新建 hello.jsp 页面

新建 web Project ,在 webRoot 下右键 ,新建 File ,名称为 hello.jsp。点击 Finish。如图-1 所示：

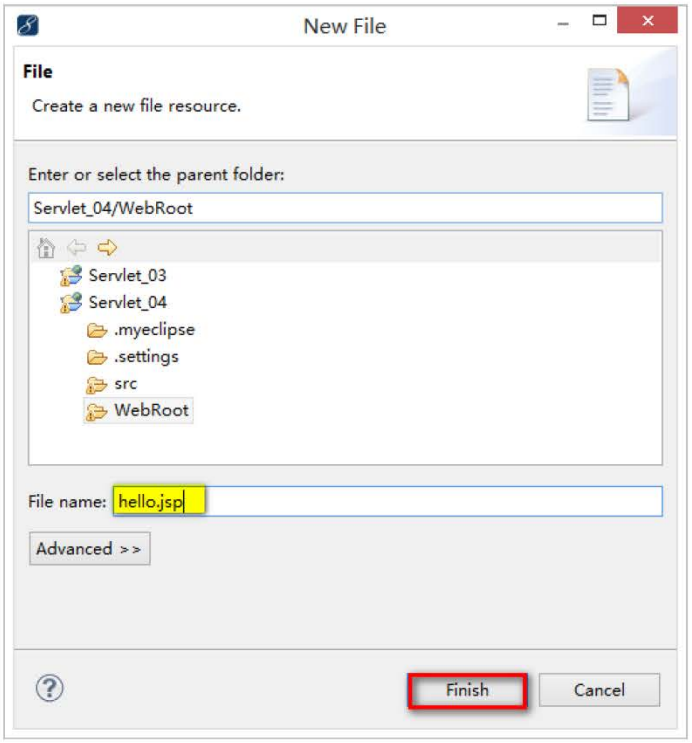
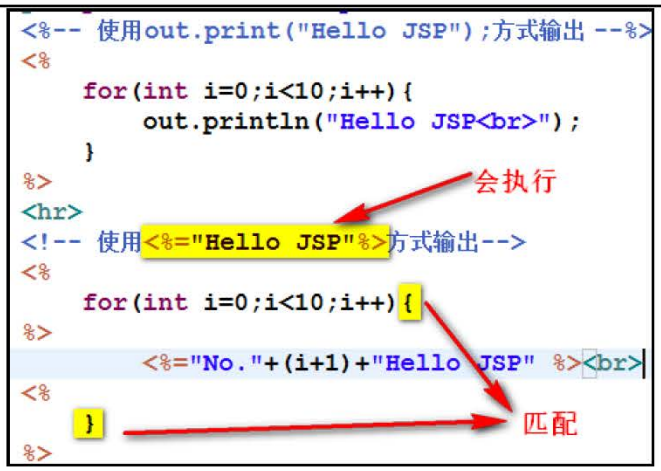


图 - 1

步骤二： 添加页面代码

在 hello.jsp 页面内编写如下代码，实现循环输出 10 行 Hello JSP。再使用表达式输出 10 行 Hello JSP。并且为两种输出添加注释。如图-2 所示：




```
<%-- 使用out.print("Hello JSP");方式输出 --%>
<%
    for(int i=0;i<10;i++){
        out.println("Hello JSP<br>");
    }
%>
<hr>
<!-- 使用<%="Hello JSP"%>方式输出-->
<%
    for(int i=0;i<10;i++){
%>
        <%="No."+(i+1)+"Hello JSP" %><br>
<%
    }
%>
```

图 - 2

步骤三： 部署，访问应用

部署应用，在地址栏中输入地址 “http://localhost:8080/部署应用名/hello.jsp” 查看页面效果以及页面源码，会发现第二种注释中的 java 代码会执行也会出现在源代码中，但第一种注释既不会运行也不会出现在结果的源代码中。

- **完整代码**

hello.jsp 文件代码如下：

```
<%@ page pageEncoding="UTF-8" %>
<html>
 <head></head>
 <body style="font-size:24px">
     <%-- 使用out.print("Hello JSP");方式输出 --%>
     <%
         for(int i=0;i<10;i++){
             out.println("Hello JSP<br>");
         }
     %>
     <hr>
     <!-- 使用<%="Hello JSP"%>方式输出-->
     <%
         for(int i=0;i<10;i++){
     %>
             <%="No."+(i+1)+"Hello JSP" %><br>
     <%
         }
    %>
 </body>
</html>
```

2. JSP 指令练习

- **问题**

在页面中输出当前系统时间，并且该时间能够嵌入到其他页面中。

- **方案**

使用 page 指令的 import 导包，contentType 控制编码，pageEncoding 控制显示中文，使用 include 指令实现页面内包含其他页面的效果。

- **步骤**

步骤一： 新建 date.jsp 页面

在 WebRoot 节点上右键→New→File,文件名称为 date.jsp。在页面内添加 page 指令及小脚本，如图-3 所示：

```
<%@page import="java.util.*,java.text.SimpleDateFormat"
    pageEncoding="UTF-8"
    contentType="text/html;charset=UTF-8"%>
<div style="font-size:20px;border:1px solid red
        width:350px;padding:20px">
    <%
        Date date = new Date();
        SimpleDateFormat sdf =
        new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    %>
    当前系统时间：<%=sdf.format(date)%>
</div>
```

导包
客户端解码
页面编码

图 - 3

步骤二： 新建 includeDemo.jsp 页面

webRoot 节点上右键，创建 New File 为 includeDemo.jsp 页面，用于包含其他页面。使用 include 指令添加，file 属性说明文件的位置。如图-4 所示：

```
<%@ page    pageEncoding="UTF-8"%>
<html>
    <head>
        <title>Insert title here</title>
    </head>
    <body>
    <%@include file="date.jsp" %><br>
    时间来自于date.jsp页面！
    </body>
</html>
```

添加其他文件到页面中

图 - 4

步骤三： 部署应用，查看结果

部署后输入地址访问页面，查看结果如图-5 所示：

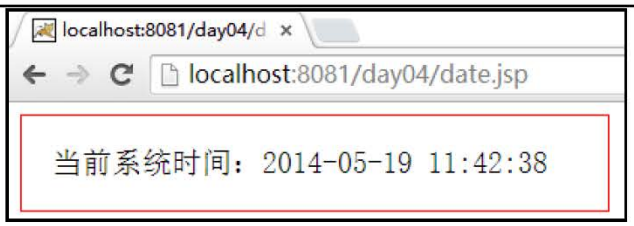


图 - 5

步骤四： 小技巧，创建自定义的 JSP 模板

在随便哪一个 JSP 页面上，右键，选择 Preferences，选择 JSP Template，点击 New，为模板命名并填写模板内容，步骤如图-6 所示：

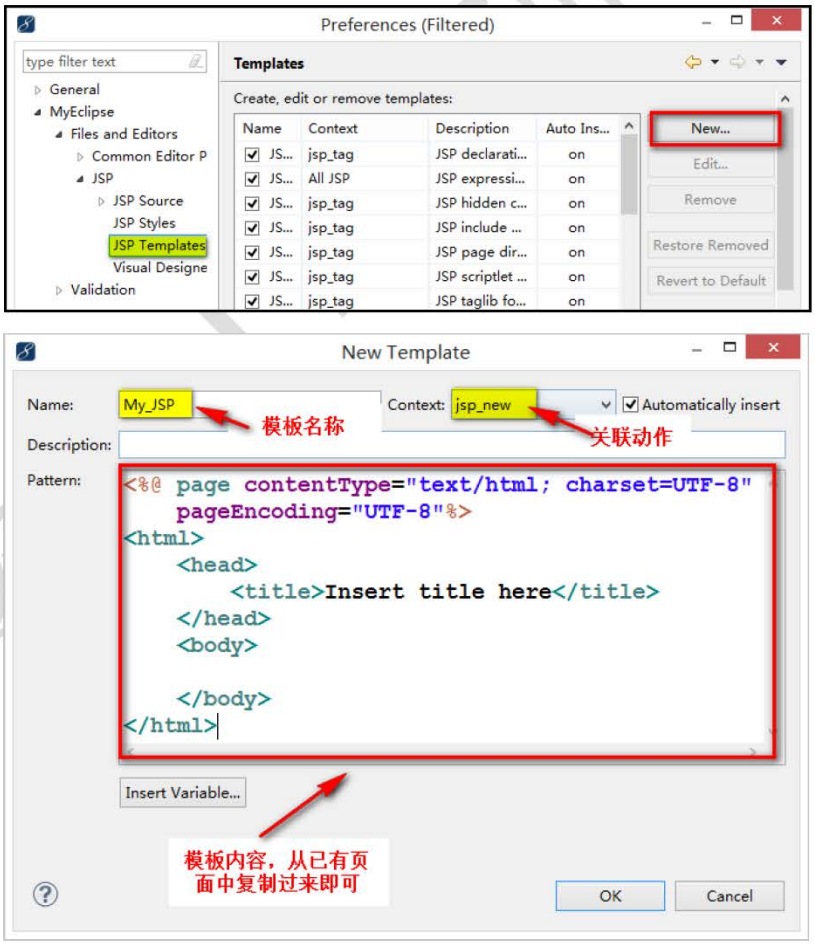
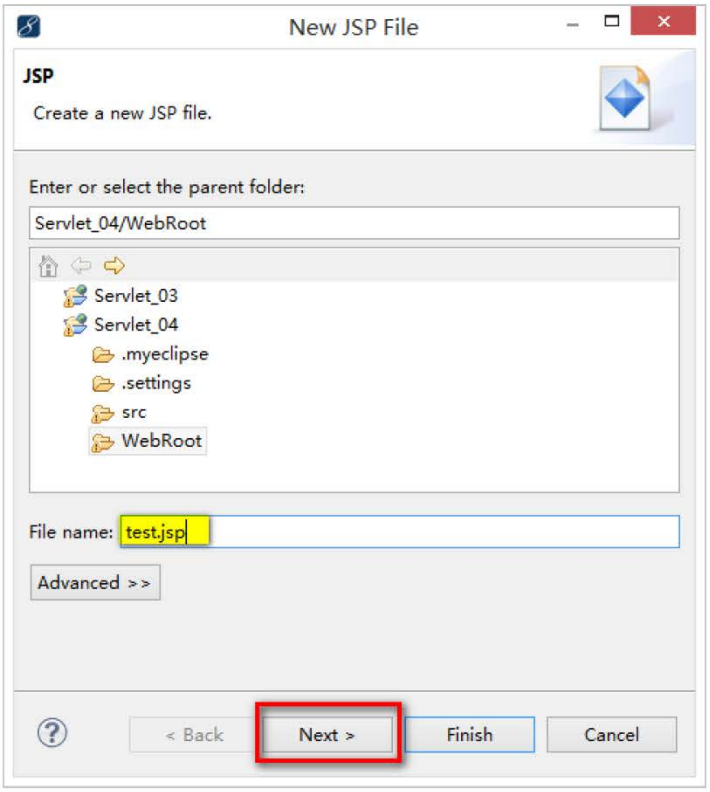


图 - 6

步骤五： 使用自定义的模板创建 JSP 页面

在 webRoot 节点上右键选择 New→JSP，填写文件名称后，点击 Next，如图-7 所示：



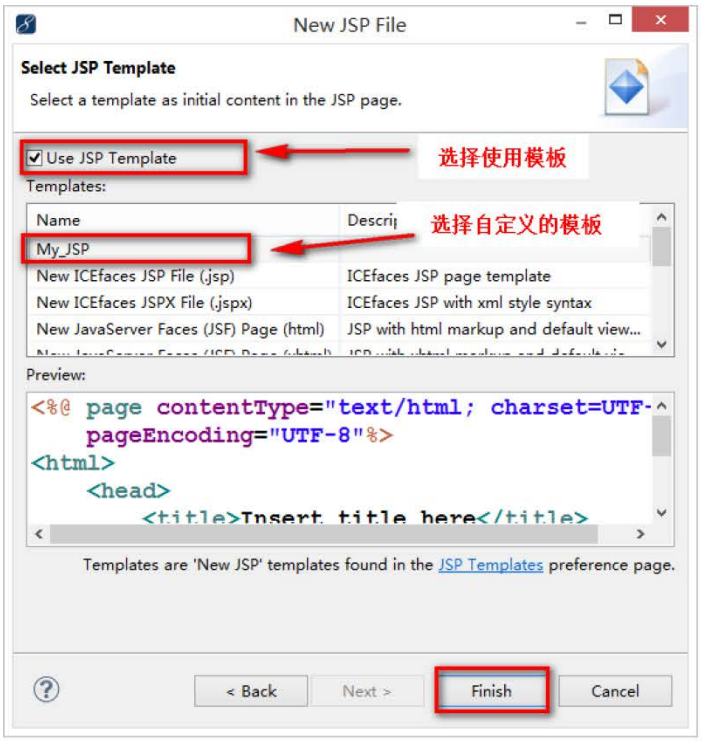


图 - 7

- **完整代码**

date.jsp 文件代码：

```
<%@page import="java.util.*,java.text.SimpleDateFormat"
 pageEncoding="UTF-8"
 contentType="text/html;charset=UTF-8"%>

<div style="font-size:20px;border:1px solid red;
        width:350px;padding:20px">
    <%
        Date date = new Date();
        SimpleDateFormat sdf =
        new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    %>
    当前系统时间：<%=sdf.format(date)%>
</div>
```

includeDemo.jsp 文件代码：

```
<%@ page   pageEncoding="UTF-8"%>
<html>
 <head>
    <title>Insert title here</title>
 </head>
 <body>
 <%@include file="date.jsp" %><br>
 时间来自于 date.jsp 页面！
 </body>
</html>
```

3. 员工管理——使用 JSP 实现员工信息列表 1

- **问题**

在页面中显示员工信息列表，并且能够实现奇偶行背景交错变换的效果。

- **方案**

在 JSP 页面中，通过创建 EmployeeDAO 的实例来获取所有雇员信息，再使用循环语句以表格的形式输出雇员信息。交错行变色为不同的背景样式决定的，所以在输出行标记时根据行坐标输出不同的样式即可。

- **步骤**

步骤一： 新建 listEmp1.jsp 页面

```
*listEmp1.jsp
1 <%@ page contentType="text/html; charset=UTF-8"
2          pageEncoding="UTF-8"%>
3 <%@ page import="dao.*,entity.*,java.util.*"%>
```

图 - 8

步骤二： 添加循环输出表格

```
<table border="1" cellpadding="0" cellspacing="0
    <caption>员工信息</caption>
    <tr>
        <td>编号</td>
        <td>姓名</td>
        <td>薪水</td>
        <td>年龄</td>
    </tr>
    <%
        EmployeeDAO dao = new EmployeeDAO();
        List<Employee> emps = dao.findAll();
        for (int i = 0; i < emps.size(); i++) {
            Employee emp = emps.get(i);
    %>
    <tr>
        <td><%=emp.getId()%></td>
        <td><%=emp.getName()%></td>
        <td><%=emp.getSalary()%></td>
        <td><%=emp.getAge()%></td>
    </tr>
    <%
        }
    %>
</table>
```

图 - 9

步骤三： 部署查看结果

运行结果如图-10 所示：

localhost:8081/day04/listEmp1.jsp

员工信息

编号	姓名	薪水	年龄
28	刘备	30000.0	28
23	张飞	18000.0	26
22	关羽	20000.0	28
24	赵云云	26000.0	22

图 - 10

步骤四： 添加两个 CSS 类样式

```
<style type="text/css">
.row1 { background-color: #E4E4F1}
.row2 { background-color: #FBD10A}
</style>
```

图 - 11

步骤五：为表格添加类样式

```
<tr class="row<%=i % 2 + 1%>">
    <td><%=emp.getId()%></td>
    <td><%=emp.getName()%></td>
    <td><%=emp.getSalary()%></td>
    <td><%=emp.getAge()%></td>
</tr>
```

图 - 12

- **完整代码**

dao 包中的 DBUtil.java、EmployeeDAO.java 和 entity 包中的 Employee.java 文件源码参考第三天代码。

listEmp1.jsp 文件代码：

```
<%@ page contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<%@ page import="dao.*,entity.*,java.util.*"%>

<html>
<head>
<style type="text/css">
.row1 {
 background-color: #E4E4F1
}

.row2 {
 background-color: #FBD10A
}
</style>
</head>
<body style="font-size:24px">
 <table border="1" cellpadding="0" cellspacing="0" width="500px">
     <caption>员工信息</caption>
     <tr>
         <td>编号</td>
         <td>姓名</td>
         <td>薪水</td>
         <td>年龄</td>
     </tr>
     <%
         EmployeeDAO dao = new EmployeeDAO();
         List<Employee> emps = dao.findAll();
         for (int i = 0; i < emps.size(); i++) {
             Employee emp = emps.get(i);
     %>
     <tr class="row<%=i % 2 + 1%>">
         <td><%=emp.getId()%></td>
         <td><%=emp.getName()%></td>
         <td><%=emp.getSalary()%></td>
         <td><%=emp.getAge()%></td>
     </tr>
     <%
         }
     %>
 </table>
</body>
</html>
```

4. 员工管理——使用 JSP 实现员工信息列表 2

- **问题**

使用美工提供的静态页面实现员工管理的信息列表展示，如图-13 所示：

图 - 13

- **方案**

将静态页面代码复制到 jsp 页面中，添加 page 指令，修改表格的输出，使用循环控制集合的展示，修改行标题，修改行的背景样式。

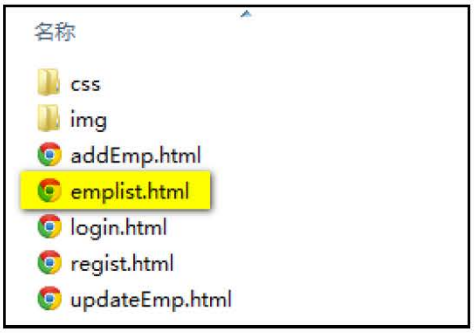
- **步骤**

步骤一： 新建 listEmp2.jsp 页面

依据自定义模板，新建 listEmp2.jsp 页面。

步骤二： 复制静态页面代码到 jsp 页面中

打开准备好的静态页面，选择 emplist.html 页面，将页面的全部代码拷贝到 listEmp2.jsp 页面中，并且 css 和 img 两个文件夹也要拷贝到工程的 WebRoot 文件夹下。



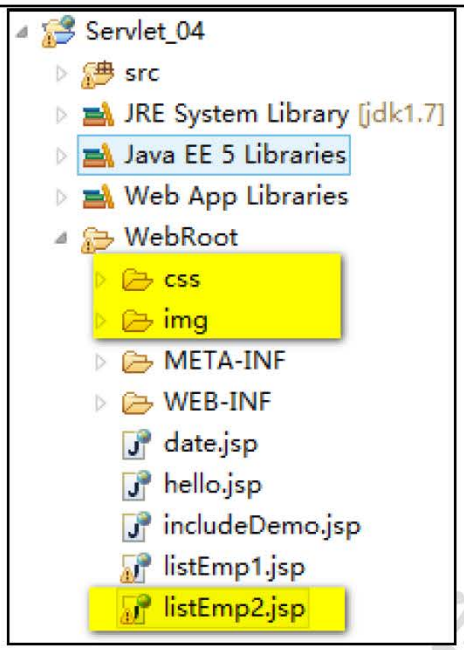


图 - 14

步骤三： 修改 page 指令，添加 java 代码

修改 page 指令，添加 import 及 content-Type 属性，保证导入 dao 和实体的包，以及正确设定浏览器对内容的解码方式。

```
*listEmp2.jsp
1 <%@ page contentType="text/html; charset=UTF-8"
2          pageEncoding="UTF-8"%>
3 <%@ page import="dao.*,entity.*,java.util.*"%>
```

图- 15

步骤四： 修改行的背景样式

```
<%
    EmployeeDAO dao = new EmployeeDAO();
    List<Employee> emps = dao.findAll();
    for (int i = 0; i < emps.size(); i++) {
        Employee emp = emps.get(i);
%>
<tr class="row<%=i % 2 + 1%>">
    <td><%=emp.getId()%></td>
    <td><%=emp.getName()%></td>
    <td><%=emp.getSalary()%></td>
    <td><%=emp.getAge()%></td>
    <td>
        <a href="emplist.html">删除</a> <
    </td>
</tr>
```

图 - 16

步骤五：部署，访问应用

运行结果如图-17 所示：

图 - 17

- **完整代码**

listEmp2.jsp 文件代码如下：

```
<%@ page contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<%@ page import="dao.*,entity.*,java.util.*"%>
<html>
 <head>
     <title>员工信息</title>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <link rel="stylesheet" type="text/css" href="css/style.css" />
 </head>
 <body>
     <div id="wrap">
         <div id="top content">
             <div id="header">
                 <div id="rightheader">
                     <p>
                         2009/11/20
                         <br />
                     </p>
                 </div>
                 <div id="topheader">
                     <h1 id="title">
                         <a href="#">main</a>
                     </h1>
                 </div>
                 <div id="navigation">
                 </div>
             </div>
             <div id="content">
                 <p id="whereami">
                 </p>
                 <h1>
                     员工信息列表
                 </h1>
                 <table class="table">
                     <tr class="table header">
                         <td>
                             编号
```

```
                        </td>
                        <td>
                            姓名
                        </td>
                        <td>
                            薪水
                        </td>
                        <td>
                            年龄
                        </td>
                        <td>
                            操作
                        </td>
                    </tr>
                    <%
                        EmployeeDAO dao = new EmployeeDAO();
                        List<Employee> emps = dao.findAll();
                        for (int i = 0; i < emps.size(); i++) {
                            Employee emp = emps.get(i);
                    %>
                    <tr class="row<%=i % 2 + 1%>">
                        <td><%=emp.getId()%></td>
                        <td><%=emp.getName()%></td>
                        <td><%=emp.getSalary()%></td>
                        <td>><%=emp.getAge()%></td>
                        <td>
                            <a  href="emplist.html#">删 除 </a> <a
href="#">更新</a>
                        </td>
                    </tr>
                    <%
                        }
                    %>

                </table>
                <p>
                    <input   type="button"   class="button"   value="Add
Employee" onclick="location='#'"/>
                </p>
            </div>
        </div>
        <div id="footer">
            <div id="footer bg">
            ABC@126.com
            </div>
        </div>
    </div>
  </body>
 </html>
```

课后作业

1. 简述在 JSP 页面中能够包含哪些内容？各有什么作用？

2. 阅读下面的代码，说明序号处代码的含义。

```
<%@page pageEncoding="utf-8"   contentType="text/html;charset=utf-8"%>
  //------1
<table class="table">
    <tr class="table header">
        <td>ID</td>
        <td>姓名</td>
        <td>薪水</td>
        <td>年龄</td>
        <td>操作</td>
    </tr>

    <%                                                          //------2
        EmployeeDAO dao =
            new EmployeeDAO();
        List<Employee> employees = dao.findAll();

        for(int i=0;i<employees.size();i++){
        Employee e = employees.get(i);
    %>
    <tr class="row1">
        <td>
            <%=e.getId()%>                                      //------3
        </td>
        <td>
            <%=e.getName()%>
        </td>
        <td>
            <%=e.getSalary()%>
        </td>
        <td>
            <%=e.getAge()%>
        </td>
        <td>
            <a href="emplist.html">删除</a> 
            <a href="updateEmp.html">更新</a>
        </td>
    </tr>
    <%
        }
    %>
</table>
```

3. 使用 JSP 实现帐务帐号信息的展示。

使用美工提供的静态页面实现帐务帐号管理的信息列表展示，如图-1 所示：

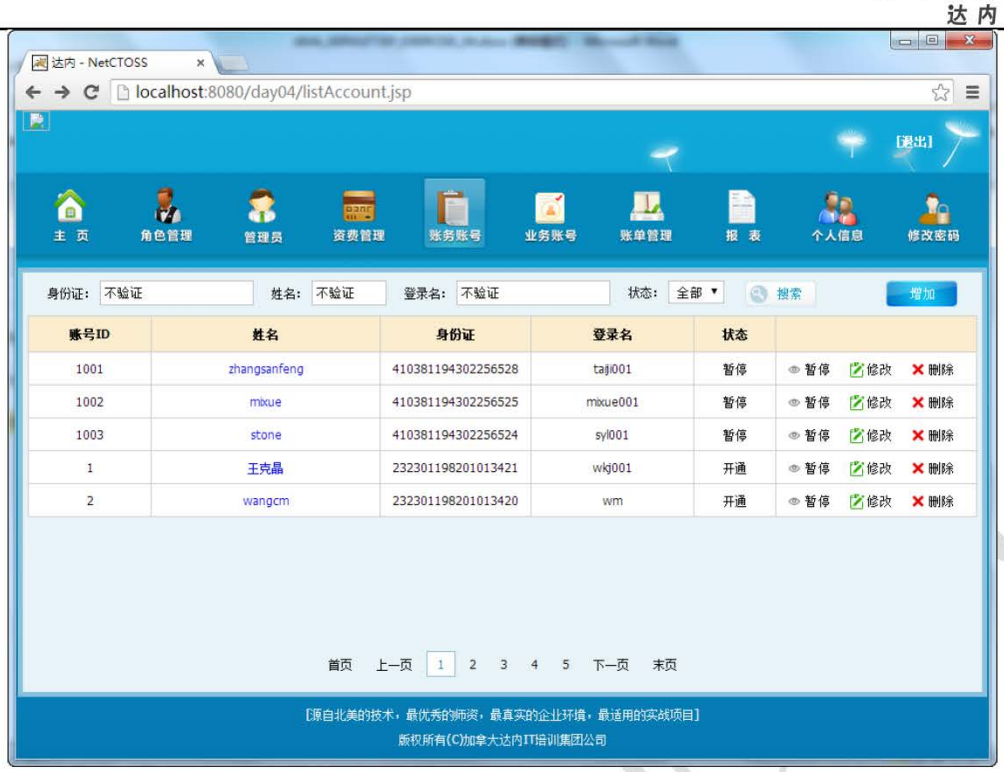

达内 - NetCTOSS
localhost:8080/day04/listAccount.jsp
[退出]
主 页
角色管理
管理员
资费管理
账务账号
业务账号
账单管理
报 表
个人信息
修改密码
身份证：不验证
姓名：不验证
登录名：不验证
状态：全部
搜索
增加
账号ID 姓名 身份证 登录名 状态
1001 zhangsanfeng 410381194302256528 tajji001 暂停 暂停 修改 删除
1002 mixue 410381194302256525 mixue001 暂停 暂停 修改 删除
1003 stone 410381194302256524 syl001 暂停 暂停 修改 删除
1 王克晶 232301198201013421 wkj001 开通 暂停 修改 删除
2 wangcm 232301198201013420 wm 开通 暂停 修改 删除
首页 上一页 1 2 3 4 5 下一页 末页
[原自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]
版权所有(C)加拿大达内IT培训集团公司

图 - 1

达内IT培训

Servlet和JSP(上)

Unit05

知识体系.....Page 156

转发	转发	什么是转发
		如何实现转发
		绑定数据到 request 对象
		获得转发器
		转发
		转发的原理
		转发的特点
		转发和重定向的区别
JSP 开发常见问题	异常处理	编程式的异常处理
		容器中的声明式处理
	路径问题	什么是路径
		什么是相对路径
		相对路径（续）
		什么是绝对路径
		绝对路径（续）
		路径的处理技巧

经典案例.....Page 162

员工管理——使用转发、JSP 实现完整的增删改查	什么是转发
	如何实现转发
	绑定数据到 request 对象
	获得转发器
	转发
	转发的原理
	转发的特点
员工管理——添加异常处理	编程式异常处理
	容器中声明式处理
	不同异常处理方式的应用场景
路径的练习	什么是路径
	什么是相对路径

	相对路径（续）
	什么是绝对路径
	绝对路径（续）
员工管理——注册	路径的处理技巧

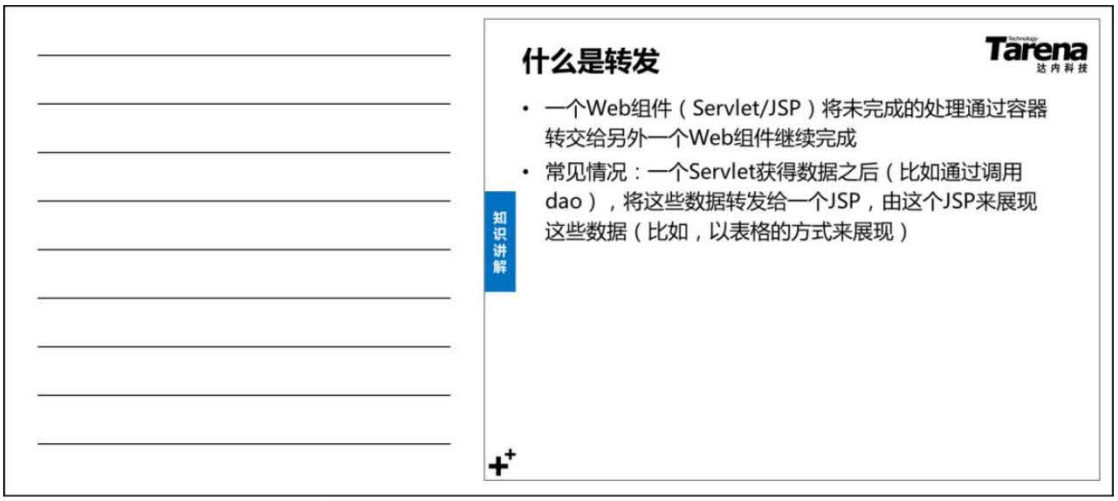
课后作业.....Page 194

达内IT培训集团

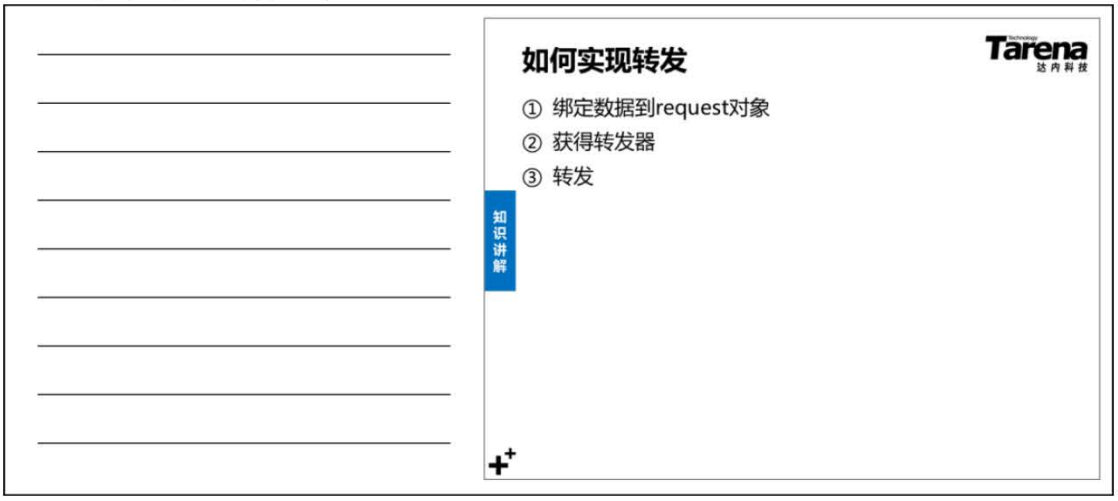
1. 转发

1.1. 转发

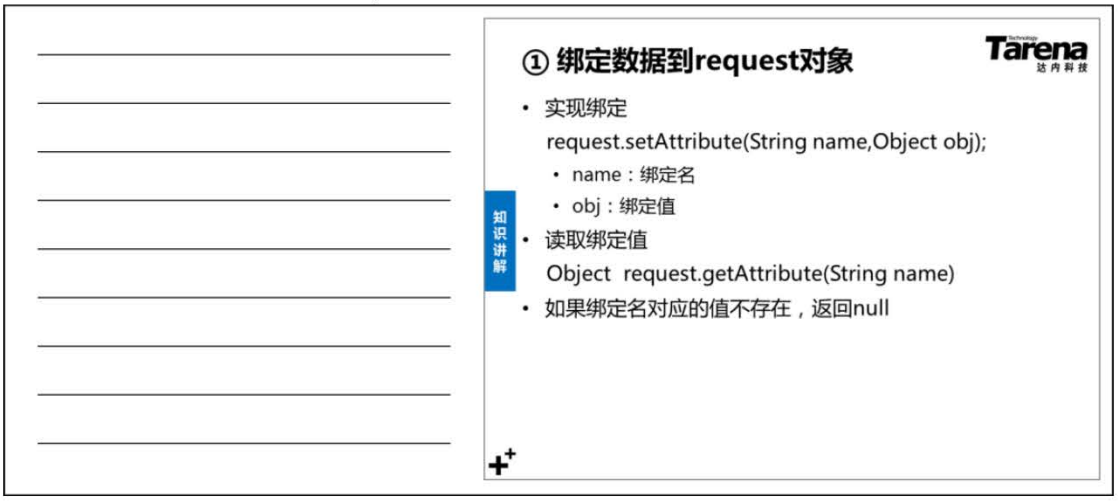
1.1.1. 【转发】什么是转发



1.1.2. 【转发】如何实现转发



1.1.3. 【转发】绑定数据到 request 对象



1.1.4. 【转发】获得转发器

② 获得转发器

- RequestDispatcher rd =
  request.getRequestDispatcher(String path);
- **path**：转发的目的地，即将未完成的处理继续下去的另一个组件，比如一个JSP文件

解说识知

1.1.5. 【转发】转发

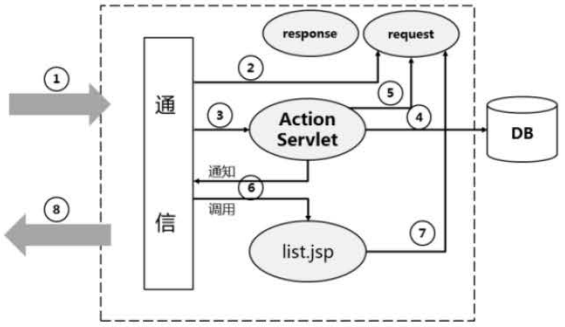
③ 转发

- 实现转发：rd . forward (request , response) ;
- 通常情况 第 2 步 和 第 3 步 合并为一行语句，如下：
  request . getRequestDispatcher(path)
  . forward(request , response) ;

解说识知

1.1.6. 【转发】转发的原理

转发的原理



解说识知

1.1.7. 【转发】转发的特点

转发的特点

解详识知

- 转发之后，地址栏地址不会发生变化。原因是转发的过程是发生在服务器内部的，浏览器并不知道。
- 转发的目的地必须是同一个应用内部的某个地址
- 转发所涉及的各个web组件会共享同一个request对象和response对象
- 注意：在forward之后的其它语句还会执行吗？
- 答：一定会执行，只要不报异常。

1.1.8. 【转发】转发和重定向的区别

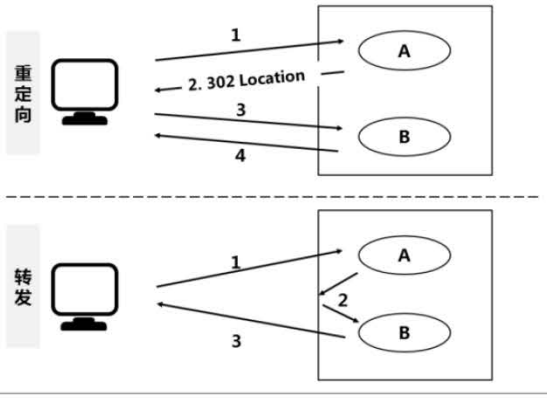
转发和重定向的区别

解详识知

- 重定向是浏览器发送请求并收到响应以后再次向一个新地址发请求，转发是服务器收到请求后为了完成响应转到一个新的地址
- 重定向中有两次请求对象，不共享数据，转发只产生一次请求对象且在组件间共享数据
- 重定向后地址栏地址改变，而转发则不会
- 重定向的新地址可以是任意地址，转发到的新地址必须是同一个应用内的某地址

转发和重定向的区别（续）

解详识知



2. JSP 开发常见问题

2.1. 异常处理

2.1.1. 【异常处理】编程式的异常处理

编程式的异常处理

- 使用转发跳转到指定页面进行提示说明

```
try{
    //... ...
} catch ( Exception  e){
    request.getRequestDispatcher(url)
            .forward(request,response);
}
```

2.1.2. 【异常处理】容器中的声明式处理

容器中声明式处理

- step1，将异常抛给容器，但底层的错误提示不要返回给用户
  - 注意：异常只允许抛service指定的异常，不能超出指定范围
  - 使用如下写法： throw new ServletException(e) ;
- step2，在web.xml文件中配置错误处理页面节点

```
<! -- 配置错误处理页面 -- >
<error-page>
    <exception-type>
            javax.servlet.ServletException
    </exception-type>
    <location> /error.jsp </location>
</error-page>
```

2.2. 路径问题

2.2.1. 【路径问题】什么是路径

什么是路径

- 链接地址<a href= “url ” > </a>
- 表单提交<form action= “ url “ >
- 重定向 response.sendRedirect (url)
- 转发 request.getRequestDispatcher(url)

2.2.2. 【路径问题】什么是相对路径

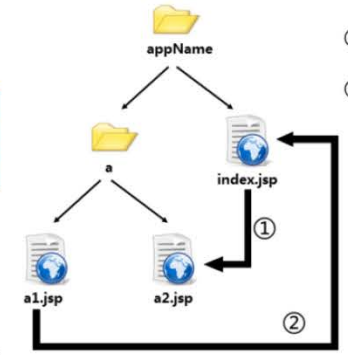
什么是相对路径

- 从当前文件出发到达目标文件所经过的路径叫做相对路径
- 书写格式不以 “/” 开头
- 退至上一级目录以 “../” 开头

解说识知

2.2.3. 【路径问题】相对路径（续）

相对路径（续）



① 路径 a / a2.jsp

② 路径 ../ index.jsp

解说识知

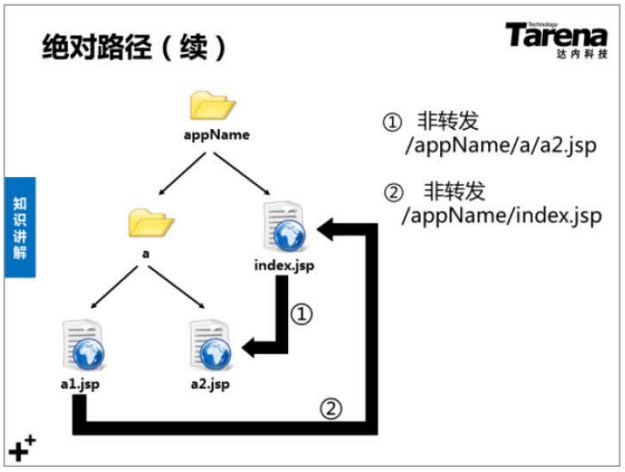
2.2.4. 【路径问题】什么是绝对路径

什么是绝对路径

- 以 ” / ” 开头的路径都是绝对路径，不以当前文件的位置作为起始，而是以一个固定位置作为起始到达目标文件所经过的路径
- 这个固定位置可能是应用名，也可能是应用名之后

解说识知

2.2.5. 【路径问题】绝对路径（续）



2.2.6. 【路径问题】路径的处理技巧

经典案例

1. 员工管理——使用转发、JSP 实现完整的增删改查

- **问题**

将美工制作的页面转变成 JSP 页面，使用转发、重定向等技术实现完整的员工管理功能。

- **方案**

功能的完成顺序为 查询→增加→删除→修改。在查询功能中，JSP 页面负责展示，数据来源使用转发的技术从 request 对象中获取。请求的地址映射规则为后缀匹配模式，由一个 ActionServlet 处理分发请求。

- **步骤**

步骤一： 使用 MyEclipse 创建 Servlet

在工程的 src 目录下，右键新建选择 Servlet，通过向导进行 Servlet 的相关配置。如图-1 所示：

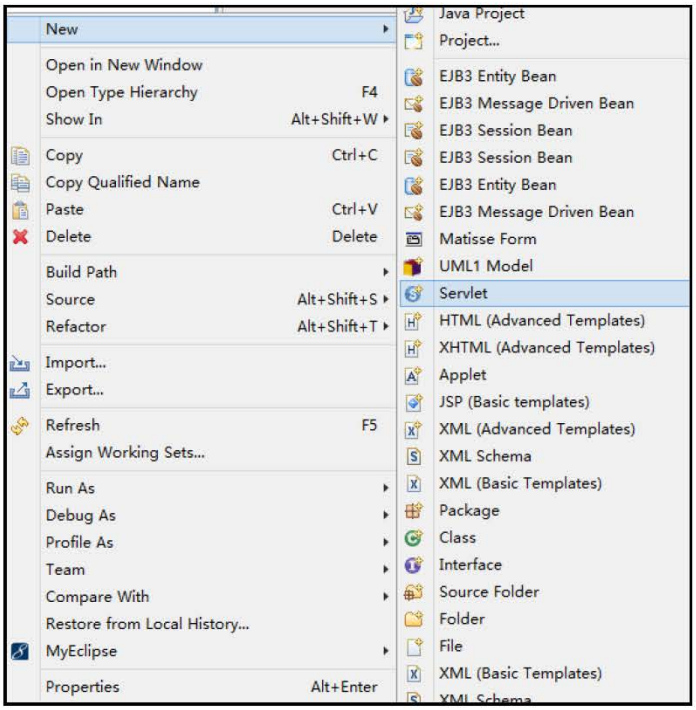


图 - 1

填写 Servlet 所在的包为 web ,名称为 ActionServlet。需要包含的方法只保留 doGet 的复选框。如图-2 所示：

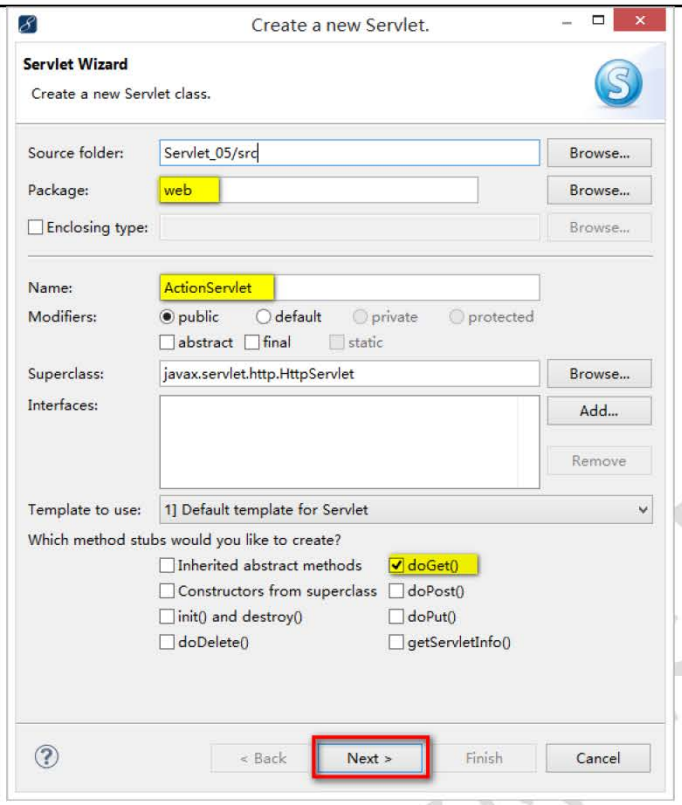


图 – 2

点击 Next 之后为配置路径的步骤。名字可以按需修改，Mapping 填写“*.do”为后缀匹配模式，Display Name 和 Description 可以不用填写。

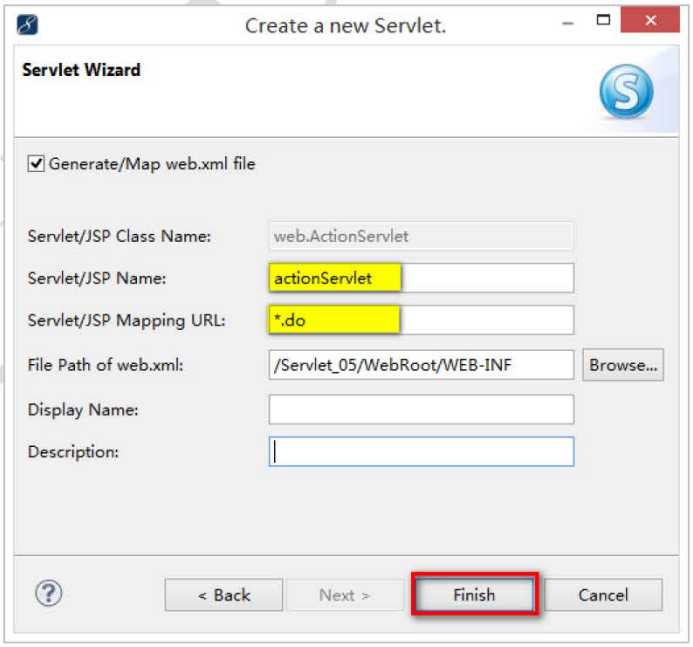


图 – 3

查看生成的文件中的代码，修改 doGet 的方法名称为 service，保留前面两行用于设置输出的中文。其余的输出语句删除。如图-4 所示：

```
public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request,
                        HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

    }

}
```

图 - 4

步骤二： 将前面工程中的 dao 包和 entity 包拷贝到 src 中。

dao 包下的 DBUtil.java 文件为获取连接关闭连接的工具类，EmployeeDAO.java 文件为针对 Employee 的数据操作对象。ActionServlet.java 文件为处理请求的控制类。如图-5 所示为完整结构。

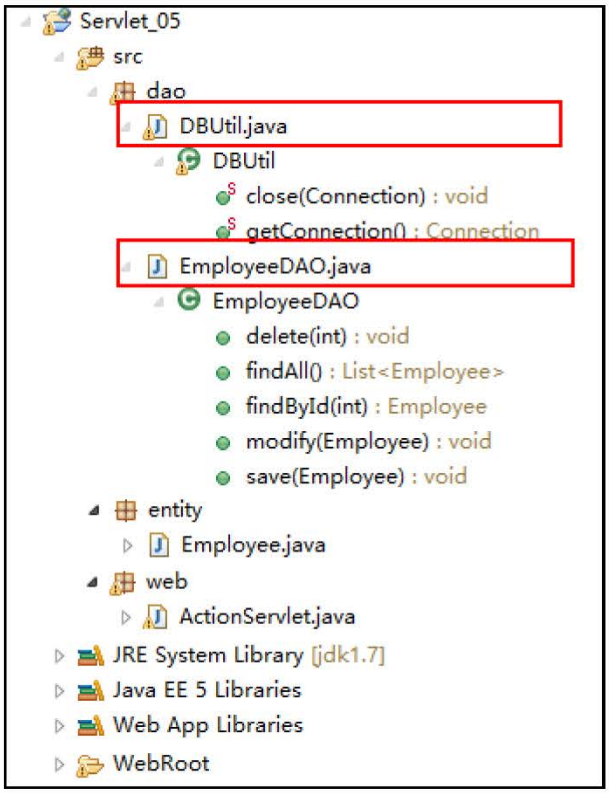


图 - 5

步骤三： 完成员工信息显示功能

修改 ActionServlet 中的 service 方法，判断动作是否是 list，查询数据后存入 request 中，并转发给 listEmp.jsp 页面。

修改 service 方法如图-6 所示：

```
//获取请求资源路径，截取具体动作
String uri = request.getRequestURI();
String action = uri.substring(uri.lastIndexOf("/")+1,
                              uri.lastIndexOf("."));

//创建员工管理操作类
EmployeeDAO dao = new EmployeeDAO();

//判断请求动作种类，分不同种类情况处理
if(action.equals("list")){//员工信息列表
    try {
        List<Employee> emps = dao.findAll();
        request.setAttribute("emps", emps);
        request.getRequestDispatcher("listEmp.jsp").
                                forward(request, response);
    } catch (Exception e) {
        e.printStackTrace();
```

区分请求种类

绑定数据到request

转发

图 - 6

WebRoot 下面新建 listEmp.jsp 页面，复制静态代码至该页面，复制图片、css，修改 page 指令。项目结构如图-7 所示，添加循环输出如图-8 所示：

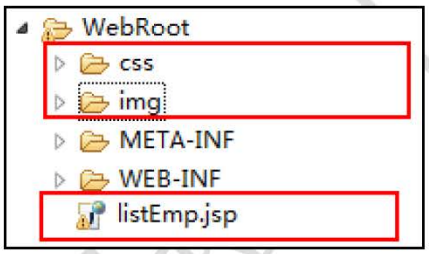


图 - 7

```
<%
    List<Employee> emps = (List<Employee>) request.
                                getAttribute("emps");
    for (int i = 0; i < emps.size(); i++) {
        Employee emp = emps.get(i);
%>
<tr class="row<%=i % 2 + 1%>">
    <td><%=emp.getId()%></td>
    <td><%=emp.getName()%></td>
    <td><%=emp.getSalary()%></td>
    <td><%=emp.getAge()%></td>
    <td><a href="delete.do?id=<%=emp.getId()%>" onclick
    <a
        href="load.do?id=<%=emp.getId()%>">修改</a></td>
</tr>
<%
    }
%>
```

图 - 8

运行效果如图-9 所示：

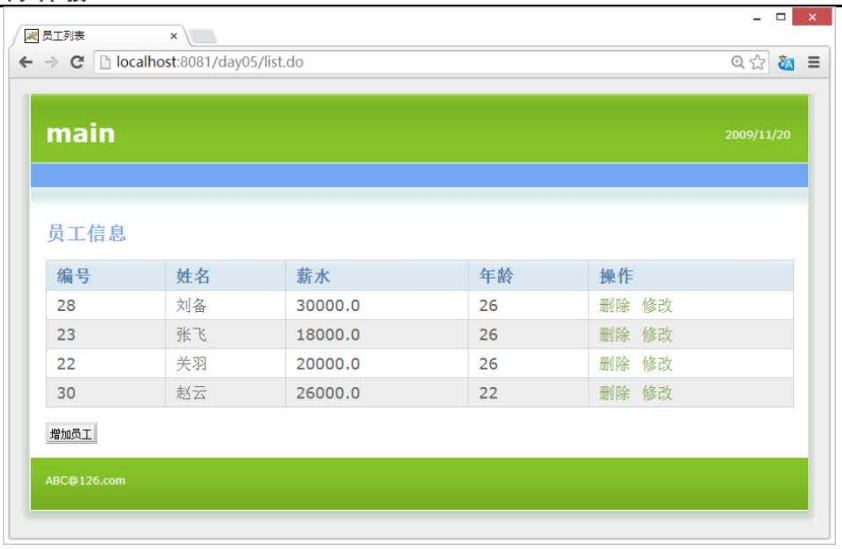


图 - 9

步骤四： 完成增加员工信息功能

新建 addEmp.jsp，表单的 action 值为 add.do。修改 ActionServlet，添加对增加动作的处理。添加成功后，重定向到 list.do，查看全部员工信息。

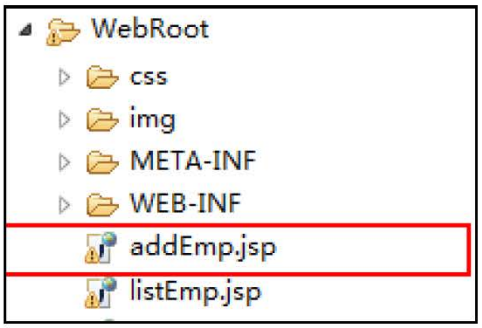


图 - 10

```
}else if(action.equals("add")){//增加员工
    String name = request.getParameter("name");
    double salary = Double.parseDouble(request.
                getParameter("salary"));
    int age = Integer.parseInt(request.getParameter("age"));
    Employee emp = new Employee();
    emp.setName(name);
    emp.setSalary(salary);
    emp.setAge(age);
    try {
        dao.save(emp);
        response.sendRedirect("list.do");
    } catch (Exception e) {
        e.printStackTrace();
```

图 – 11

运行结果如图 - 12 所示：

图 - 12

步骤五：完成删除员工信息功能

修改 listEmp.jsp 中删除操作的链接地址，修改 ActionServler 的判断分支。

```
<a href="delete.do?id=<%=emp.getId()%>"
onclick="return confirm('是否确认删除<%=emp.getName()%>信息？');">删除</a>
```

图 - 13

```
}else if(action.equals("delete")){//删除员工
    int id = Integer.parseInt(request.getParameter("id"));
    try {
        dao.delete(id);
        response.sendRedirect("list.do");
    } catch (Exception e) {
        e.printStackTrace();
    }
```

图 - 14

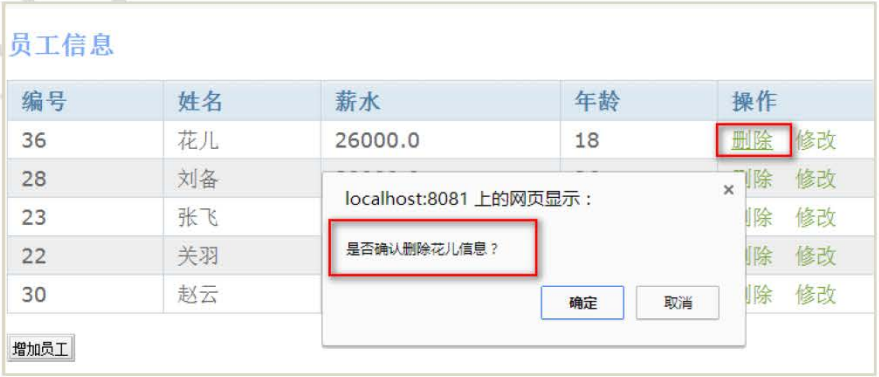


图- 15

步骤六：完成修改员工信息功能

修改 listEmp.jsp 页面的修改功能的链接，修改 ActionServlet.java 的判断，添加 update.jsp 页面，能够成功加载某一个员工信息后，再修改 update.jsp 页面的 action 地址，修改 ActionServlet 的判断，完成修改保存后重定向到 listEmp.jsp 页面。

工程结果如图-16 所示：

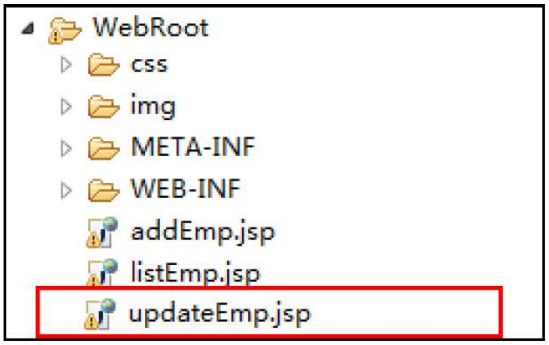


图- 16

修改 ActionServlet，添加加载的判断：

```
}else if(action.equals("load")){//加载某个员工
    int id = Integer.parseInt(request.getParameter("id"));
    try {
        Employee emp = dao.findById(id);
        request.setAttribute("emp", emp);
        request.getRequestDispatcher("updateEmp.jsp").
        forward(request, response);
    } catch (Exception e) {
        e.printStackTrace();
    }
```

图 – 17

修改 listEmp.jsp 页面的修改链接的地址

```
<a href="load.do?id=<%=emp.getId()%>">修改</a>
```

图 – 18

修改 uploadEmp.jsp 页面，读取绑定数据，将数据显示到表单中，如图-19 所示：

```
<%
    Employee emp = (Employee)request.getAttribute("emp");
%>
<form action="update.do?id=<%=emp.getId()%>" method="post">
    <table cellpadding="0" cellspacing="0" border="0"
        class="form_table">
        <tr>
            <td valign="middle" align="right">
                编号:
            </td>
            <td valign="middle" align="left">
                <%=emp.getId()%>
            </td>
```

图 – 19

运行，查看是否能够正确加载员工信息，如图-20 所示：

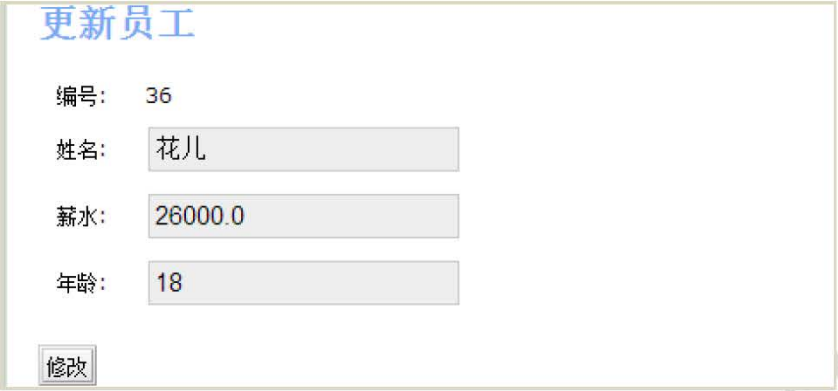


图 - 20

修改 updateEmp.jsp 页面的表单提交，修改 ActionServlet，保存成功后重定向到 list.do：

```
<form action="update.do?id=<%=emp.getId()%>" method="post">
```

图 – 21

ActionServlet.java 文件修改如图-22 所示：

```
}else if(action.equals("update")){//更新员工
    int id = Integer.parseInt(request.getParameter("id"));
    String name = request.getParameter("name");
    double salary = Double.parseDouble(
            request.getParameter("salary"));
    int age = Integer.parseInt(request.getParameter("age"));
    Employee emp = new Employee();
    emp.setName(name);
    emp.setSalary(salary);
    emp.setAge(age);
    emp.setId(id);
    try {
        dao.modify(emp);
        response.sendRedirect("list.do");
    } catch (Exception e) {
```

图 - 22

- **完整代码**

DBUtil.java 和 EmployeeDAO.java 和 Employee.java 文件代码参考前面的练习。

ActionServlet.java 代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import dao.EmployeeDAO;
import dao.UserDAO;
import entity.Employee;
import entity.User;

public class ActionServlet extends HttpServlet {
 public void service(HttpServletRequest request,
                     HttpServletResponse response)
        throws ServletException, IOException {

    //设置编码、解码格式
    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    //获取请求资源路径，截取具体动作
    String uri = request.getRequestURI();
    String action = uri.substring(uri.lastIndexOf("/")+1,
                                  uri.lastIndexOf("."));

    //创建员工管理操作类
    EmployeeDAO dao = new EmployeeDAO();

    //判断请求动作种类，分不同种类情况处理
    if(action.equals("list")){//员工信息列表
        try {
            List<Employee> emps = dao.findAll();
            request.setAttribute("emps", emps);
            request.getRequestDispatcher("listEmp.jsp").
                                forward(request, response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }else if(action.equals("add")){//增加员工
        String name = request.getParameter("name");
        double salary = Double.parseDouble(request.
                    getParameter("salary"));
        int age = Integer.parseInt(request.getParameter("age"));
        Employee emp = new Employee();
        emp.setName(name);
        emp.setSalary(salary);
        emp.setAge(age);
        try {
            dao.save(emp);
            response.sendRedirect("list.do");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }else if(action.equals("delete")){//删除员工
        int id = Integer.parseInt(request.getParameter("id"));
        try {
            dao.delete(id);
            response.sendRedirect("list.do");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }else if(action.equals("load")){//加载某个员工
        int id = Integer.parseInt(request.getParameter("id"));
        try {
            Employee emp = dao.findById(id);
            request.setAttribute("emp", emp);
```

```
                request.getRequestDispatcher("updateEmp.jsp").
                forward(request, response);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }else if(action.equals("update")){//更新员工
            int id = Integer.parseInt(request.getParameter("id"));
            String name = request.getParameter("name");
            double salary = Double.parseDouble(
                    request.getParameter("salary"));
            int age = Integer.parseInt(request.getParameter("age"));
            Employee emp = new Employee();
            emp.setName(name);
            emp.setSalary(salary);
            emp.setAge(age);
            emp.setId(id);
            try {
                dao.modify(emp);
                response.sendRedirect("list.do");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

listEmp.jsp 文件代码如下：

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="entity.*,java.util.*"%>

<html>
<head>
<title>员工列表</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="css/style.css" />
</head>
<body>
 <div id="wrap">
     <div id="top_content">
         <div id="header">
             <div id="rightheader">
                 <p>
                     2009/11/20 <br />
                 </p>
             </div>
             <div id="topheader">
                 <h1 id="title">
                     <a href="#">main</a>
                 </h1>
             </div>
             <div id="navigation"></div>
         </div>
         <div id="content">
             <p id="whereami"></p>
             <h1>员工信息</h1>
             <table class="table">
                 <tr class="table_header">
                     <td>编号</td>
                     <td>姓名</td>
                     <td>薪水</td>
                     <td>年龄</td>
```

```
                        <td>操作</td>
                    </tr>
                    <%
                        List<Employee> emps = (List<Employee>) request.
                                                getAttribute("emps");
                        for (int i = 0; i < emps.size(); i++) {
                            Employee emp = emps.get(i);
                    %>
                    <tr class="row<%=i % 2 + 1%>">
                        <td><%=emp.getId()%></td>
                        <td><%=emp.getName()%></td>
                        <td><%=emp.getSalary()%></td>
                        <td><%=emp.getAge()%></td>
                        <td>
                <a href="delete.do?id=<%=emp.getId()%>"
                onclick="return confirm('是否确认删除<%=emp.getName()%>信息？
');">删除</a>
                 
                <a href="load.do?id=<%=emp.getId()%>">修改</a></td>
                    </tr>
                    <%
                        }
                    %>
                </table>
                <p>
                    <input type="button" class="button" value="增加员工"
                        onclick="location='addEmp.jsp'" />
                </p>
            </div>
        </div>
        <div id="footer">
            <div id="footer bg">ABC@126.com</div>
        </div>
    </div>
    </body>
    </html>
```

addEmp.jsp 文件代码如下：

```
    <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
    <html>
    <head>
    <title>增加员工</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="css/style.css" />
    </head>

    <body>
     <div id="wrap">
         <div id="top content">
             <div id="header">
                 <div id="rightheader">
                     <p>
                         2009/11/20 <br />
                     </p>
                 </div>
                 <div id="topheader">
                     <h1 id="title">
                         <a href="#">Main</a>
                     </h1>
                 </div>
                 <div id="navigation"></div>
             </div>
```

```
            <div id="content">
                <p id="whereami"></p>
                <h1>增加员工:</h1>
                <form action="add.do" method="post">
                    <table cellpadding="0" cellspacing="0" border="0"
                        class="form table">
                        <tr>
                            <td valign="middle" align="right">姓名:</td>
                            <td valign="middle" align="left"><input type="text"
                                class="inputgri" name="name" /></td>
                        </tr>
                        <tr>
                            <td valign="middle" align="right">薪水:</td>
                            <td valign="middle" align="left"><input type="text"
                                class="inputgri" name="salary" /></td>
                        </tr>
                        <tr>
                            <td valign="middle" align="right">年龄:</td>
                            <td valign="middle" align="left"><input type="text"
                                class="inputgri" name="age" /></td>
                        </tr>
                    </table>
                    <p>
                        <input type="submit" class="button" value="增加" />
                    </p>
                </form>
            </div>
        </div>
        <div id="footer">
            <div id="footer bg">ABC@126.com</div>
        </div>
    </div>
    </body>
    </html>
```

updateEmp.jsp 文件代码如下：

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="entity.*" %>
<html>
 <head>
     <title>修改员工信息</title>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <link rel="stylesheet" type="text/css" href="css/style.css" />
 </head>

 <body>
         <div id="wrap">
         <div id="top content">
                 <div id="header">
                     <div id="rightheader">
                         <p>
                             2009/11/20
                             <br />
                         </p>
                     </div>
                     <div id="topheader">
                         <h1 id="title">
                             <a href="#">Main</a>
                         </h1>
                     </div>
                     <div id="navigation">
                     </div>
```

```
                    </div>
                <div id="content">
                    <p id="whereami">
                    </p>
                    <h1>
                        更新员工
                    </h1>
                    <% Employee emp = (Employee)request.getAttribute("emp"); %>
                    <form              action="update.do?id=<%=emp.getId()%>"
method="post">
                        <table cellpadding="0" cellspacing="0" border="0"
                            class="form table">
                            <tr>
                                <td valign="middle" align="right">
                                    编号:
                                </td>
                                <td valign="middle" align="left">
                                    <%=emp.getId()%>
                                </td>
                            </tr>
                            <tr>
                                <td valign="middle" align="right">
                                    姓名:
                                </td>
                                <td valign="middle" align="left">
                                    <input    type="text"    class="inputgri"
name="name" value="<%=emp.getName()%>"/>
                                </td>
                            </tr>
                            <tr>
                                <td valign="middle" align="right">
                                    薪水:
                                </td>
                                <td valign="middle" align="left">
                                    <input    type="text"    class="inputgri"
name="salary" value="<%=emp.getSalary()%>"/>
                                </td>
                            </tr>
                            <tr>
                                <td valign="middle" align="right">
                                    年龄:
                                </td>
                                <td valign="middle" align="left">
                                    <input    type="text"    class="inputgri"
name="age" value="<%=emp.getAge()%>"/>
                                </td>
                            </tr>
                        </table>
                        <p>
                            <input type="submit" class="button" value="修改" />
                        </p>
                    </form>
                </div>
            </div>
            <div id="footer">
                <div id="footer bg">
                    ABC@126.com
                </div>
            </div>
        </div>
    </body>
   </html>
```

2. 员工管理——添加异常处理

- **问题**

解决系统在出现异常时的不友好提示，避免客户端看到有关错误的任何说明。

- **方案**

两种解决方法，一种是编程式处理，即在捕获到异常时，绑定错误提示信息，转发到指定的错误页面。一种是声明式处理，即在捕获到异常时，将其封装成 ServletException 后抛给容器，在 web.xml 中添加配置，在捕获到异常后由容器控制着跳转到指定的错误页面。

- **步骤**

步骤一： 添加编程式处理的判断

```
} catch (Exception e) {
    e.printStackTrace();
    //编程式的处理异常
    request.setAttribute("err_msg", "系统出错，请重试");
    request.getRequestDispatcher("error.jsp").
                        forward(request, response);
}
```

图 - 23

步骤二： 新建 error.jsp 页面，读取绑定的错误信息

```
error.jsp
1  <%@ page contentType="text/html; charset=UTF
2  <html>
3      <head>
4          <title>错误提醒</title>
5      </head>
6      <body style="font-size:24px">
7
8      <%=request.getAttribute("err_msg")%><Br>
9      <a href="list.do">返回</a>
10     </body>
11 </html>
```

图 - 24

步骤三： 添加声明式处理的代码

```
} catch (Exception e) {
    e.printStackTrace();
    //将异常抛给容器，在web.xml中添加声明，变成声明式的处理异常
    throw new ServletException(e);
}
```

图 - 25

步骤四： 配置 web.xml 文件

```
<error-page>
  <exception-type>javax.servlet.ServletException</exception-type>
  <location>/error.jsp</location>
</error-page>
```

图 - 26

- **完整代码**

ActionServlet.java 文件代码：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import dao.EmployeeDAO;
import dao.UserDAO;
import entity.Employee;
import entity.User;

public class ActionServlet extends HttpServlet {

 public void service(HttpServletRequest request,
                     HttpServletResponse response)
        throws ServletException, IOException {

    //设置编码、解码格式
    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    //获取请求资源路径，截取具体动作
    String uri = request.getRequestURI();
    String action = uri.substring(uri.lastIndexOf("/")+1,
                                  uri.lastIndexOf("."));

    //创建员工管理操作类
    EmployeeDAO dao = new EmployeeDAO();

    //判断请求动作种类，分不同种类情况处理
    if(action.equals("list")){//员工信息列表
        try {
            List<Employee> emps = dao.findAll();
            request.setAttribute("emps", emps);
            request.getRequestDispatcher("listEmp.jsp").
                                forward(request, response);
        } catch (Exception e) {
            e.printStackTrace();
            //编程式的处理异常
            request.setAttribute("err_msg", "系统出错，请重试");
            request.getRequestDispatcher("error.jsp").
                                forward(request, response);
```

```
            }
        }else if(action.equals("add")){//增加员工
            String name = request.getParameter("name");
            double salary = Double.parseDouble(request.
                        getParameter("salary"));
            int age = Integer.parseInt(request.getParameter("age"));
            Employee emp = new Employee();
            emp.setName(name);
            emp.setSalary(salary);
            emp.setAge(age);
            try {
                dao.save(emp);
                response.sendRedirect("list.do");
            } catch (Exception e) {
                e.printStackTrace();
                //将异常抛给容器，在web.xml中添加声明，变成声明式的处理异常
                throw new ServletException(e);
            }
        }else if(action.equals("delete")){//删除员工
            int id = Integer.parseInt(request.getParameter("id"));
            try {
                dao.delete(id);
                response.sendRedirect("list.do");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }else if(action.equals("load")){//加载某个员工
            int id = Integer.parseInt(request.getParameter("id"));
            try {
                Employee emp = dao.findById(id);
                request.setAttribute("emp", emp);
                request.getRequestDispatcher("updateEmp.jsp").
                forward(request, response);
            } catch (Exception e) {
                e.printStackTrace();
            }

        }else if(action.equals("update")){//更新员工
            int id = Integer.parseInt(request.getParameter("id"));
            String name = request.getParameter("name");
            double salary = Double.parseDouble(
                    request.getParameter("salary"));
            int age = Integer.parseInt(request.getParameter("age"));
            Employee emp = new Employee();
            emp.setName(name);
            emp.setSalary(salary);
            emp.setAge(age);
            emp.setId(id);
            try {
                dao.modify(emp);
                response.sendRedirect("list.do");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
 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-app_2_5.xsd">
   <display-name></display-name>
   <servlet>
     <servlet-name>actionServlet</servlet-name>
     <servlet-class>web.ActionServlet</servlet-class>
   </servlet>

   <servlet-mapping>
     <servlet-name>actionServlet</servlet-name>
     <url-pattern>*.do</url-pattern>
   </servlet-mapping>

   <error-page>
      <exception-type>javax.servlet.ServletException</exception-type>
      <location>/error.jsp</location>
   </error-page>
 </web-app>
```

error.jsp 文件代码：

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
 <head>
     <title>错误提醒</title>
 </head>
 <body style="font-size:24px">

 <%=request.getAttribute("err msg")==null
    ?"":request.getAttribute("err msg")%><Br>

  系统异常<a href="list.do">返回</a>
 </body>
</html>
```

3. 路径的练习

- **问题**

绝对路径和相对路径的练习。

- **方案**

在工程中添加 biz01 和 biz02 两个文件夹，分别包含 a1.jsp 和 a2.jsp 两个文件，并且使用相对路径和绝对路径两种方式完成 a1 和 a2 之间的链接跳转。再增加一个 SomeServlet，实现 form.jsp 表单数据提交时到 Servlet 的链接写法。

- **步骤**

步骤一： 新建 a1.jsp 和 a2.jsp 文件

分别创建 biz01/a1.jsp 和 biz02/a2.jsp 两个文件，工程结构如图-27 所示：

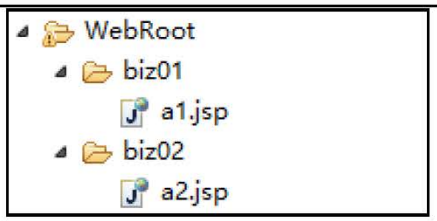


图 - 27

步骤二： 为 a1.jsp 和 a2.jsp 添加跳转链接

a1.jsp 文件中的链接写法如下：

```
a1.jsp's content<br/>
相对路径的写法：
<a href="../biz02/a2.jsp">a2.jsp</a>
绝对路径的写法：
<a href="/day05/biz02/a2.jsp">a2.jsp</a>
变量路径的写法：
<a href="<%=request.getContextPath()%>/biz02/a2.jsp">a2.jsp</a>
```

图 - 28

a2.jsp 文件中的链接写法如下：

```
a2.jsp's content<br/>
相对路径的写法：
<a href="../biz01/a1.jsp">a1.jsp</a>
绝对路径的写法：
<a href="/day05/biz01/a1.jsp">a1.jsp</a>
变量路径的写法：
<a href="<%=request.getContextPath()%>/biz01/a1.jsp">a2.jsp</a>
```

图- 29

步骤三： 添加 SomeServlet

```
response.setContentType("text/html;charset=utf-8");
PrintWriter out = response.getWriter();
System.out.println("SomeServlet's service is running...");
//重定向到biz01/a1.jsp
request.getRequestDispatcher("/biz01/a1.jsp").forward(requ
```

图 - 30

步骤四： 添加 form.jsp 文件

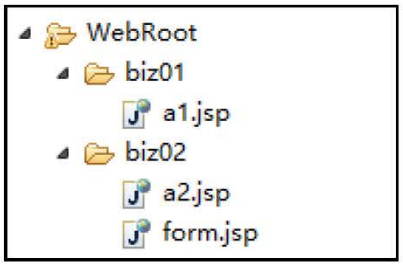


图 – 31

form.jsp 文件内容

```
<!-- <form action="../some" method="post"> -->
<form action="<%=request.getContextPath()%>/some" method="post">
    <input type="text" name="name" /><br>
    <input type="submit" value="提交"/>
</form>
```

图 - 32

- **完整代码**

a1.jsp 文件代码：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body style="font-size:24px;">
 a1.jsp's content<br/>
 相对路径的写法：
 <a href="../biz02/a2.jsp">a2.jsp</a>
 绝对路径的写法：
 <a href="/day05/biz02/a2.jsp">a2.jsp</a>
 变量路径的写法：
 <a href="<%=request.getContextPath()%>/biz02/a2.jsp">a2.jsp</a>
</body>
</html>
```

a2.jsp 文件代码：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body style="font-size:24px;">
 a2.jsp's content<br/>
 相对路径的写法：
 <a href="../biz01/a1.jsp">a1.jsp</a>
 绝对路径的写法：
 <a href="/day05/biz01/a1.jsp">a1.jsp</a>
 变量路径的写法：
 <a href="<%=request.getContextPath()%>/biz01/a1.jsp">a2.jsp</a>
</body>
</html>
```

SomeServlet.java 文件代码：

```
package path;

import java.io.IOException;
```

```
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SomeServlet extends HttpServlet {


 public void service(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    System.out.println("SomeServlet's service is running...");
    //重定向到biz01/a1.jsp
    request.getRequestDispatcher("/biz01/a1.jsp").forward(request,
response);
 }

}
```

form.jsp 文件代码：

```
<%@ 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>
 <!-- <form action="../some" method="post"> -->
 <form action="<%=request.getContextPath()%>/some" method="post">
    <input type="text" name="name" /><br>
    <input type="submit" value="提交"/>
 </form>
</body>
</html>
```

4. 员工管理——注册

- **问题**

为员工管理系统添加注册和登录功能，注册时需要进行用户名是否存在的判断，如果存在则出现提示。

- **方案**

创建 t_user 表，编写 User 实体和 UserDAO 操作类。编写注册页面，提交到服务器以后，判断用户名是否存在，存在则绑定提示信息后转发回注册页面，页面输出绑定的提示信息，如果不存在则添加用户信息后重定向到登录页面。登录页面功能相同。

- **步骤**

步骤一： 创建 t_user 表

```sql
*DB.sql
Connection: OracleDB
create table t_user(
    id number(4) primary key ,
    username varchar2(20) unique,
    pwd varchar2(30),
    name varchar2(50),
    gender char(2)

)

create sequence user_id_seq increment by 1 start with 1

insert into t_user values(user_id_seq.nextval,'cat','123456','汤姆','m')

select * from t_user
```

图 - 33

步骤二： 创建 entity.User 类

```java
*User.java
package entity;

/**
 * 实体类：用于描述用户，与数据库中的t_user表中的字段一一对应
 */
public class User {
    private int id;
    private String userName;
    private String pwd;
    private String name;
    private String gender;

    @Override
    public String toString() {
        return id + "  " + userName + "  " +
                pwd + "  " + name + "  " + gender;
    }
```

图 - 34

步骤三： 创建 dao.UserDAO 类

dao.DBUtil.java 的内容同前面的练习一致。

```
*User.java   *UserDAO.java
1  package dao;
2
3  import java.sql.Connection;
8  /**
9   *  用户的数据操作类
10  */
11 public class UserDAO {
12     /**
13      * 根据用户名称查找用户对象
14      * @param userName 用户名称
15      * @return  用户实体
16      */
17     public User findByUserName(String userName) thr
46
47     /**
48      * 保存实体信息到数据库
49      */
50     public void save(User user) throws Exception {
70 }
```

查找用户的方法

保存用户信息的方法

图- 35

步骤四： 创建 regist.jsp 页面

将美工制作的注册页面内容复制到 regist.jsp 页面中。

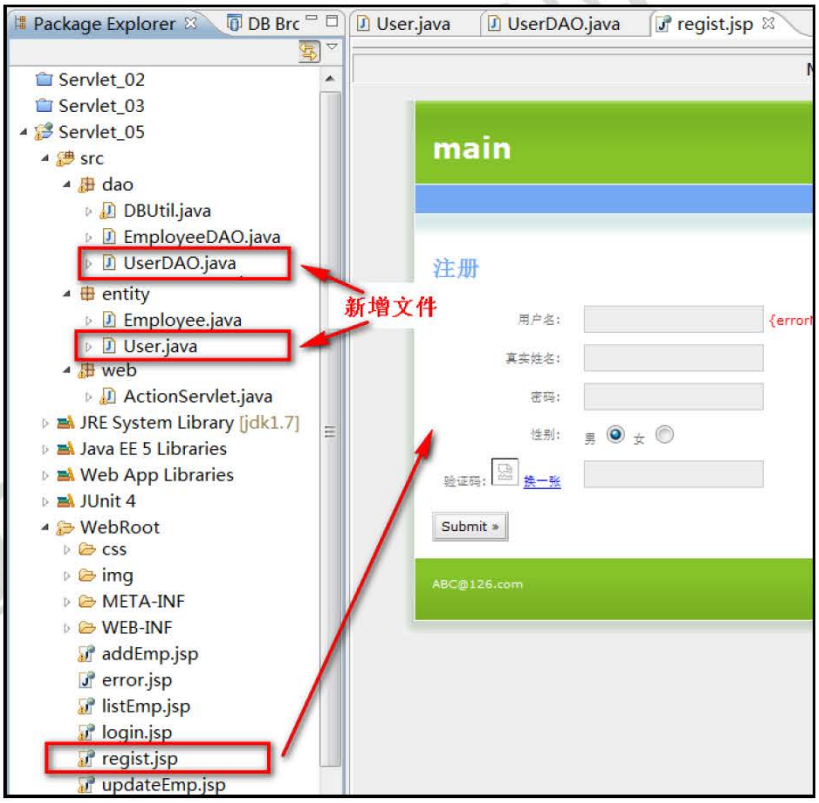


图 - 36

步骤五： 修改 ActionServlet 类

```
} else if (action.equals("regist")) {
    //1.获取表单中的用户名等信息
    //2.根据用户名查询实体
    //3.判断实体是否为null
    //4.等于null则执行插入
    //5.不等于空则绑定提示信息后跳转
} else if (action.equals("login")) {



}
```

图 - 37

修改 regist.jsp 页面，读取绑定的信息，非空则显示在文本框的后面。然后运行程序，查看效果：

```
<input type="text" class="inputgri" name="username" />
<%
    String errorMsg = (String)request.getAttribute("regis_err");
%><span style="color:red;"><%=errorMsg==null?"":errorMsg%></span>
```

图 - 38

图 - 39

步骤六： 创建 login.jsp 页面

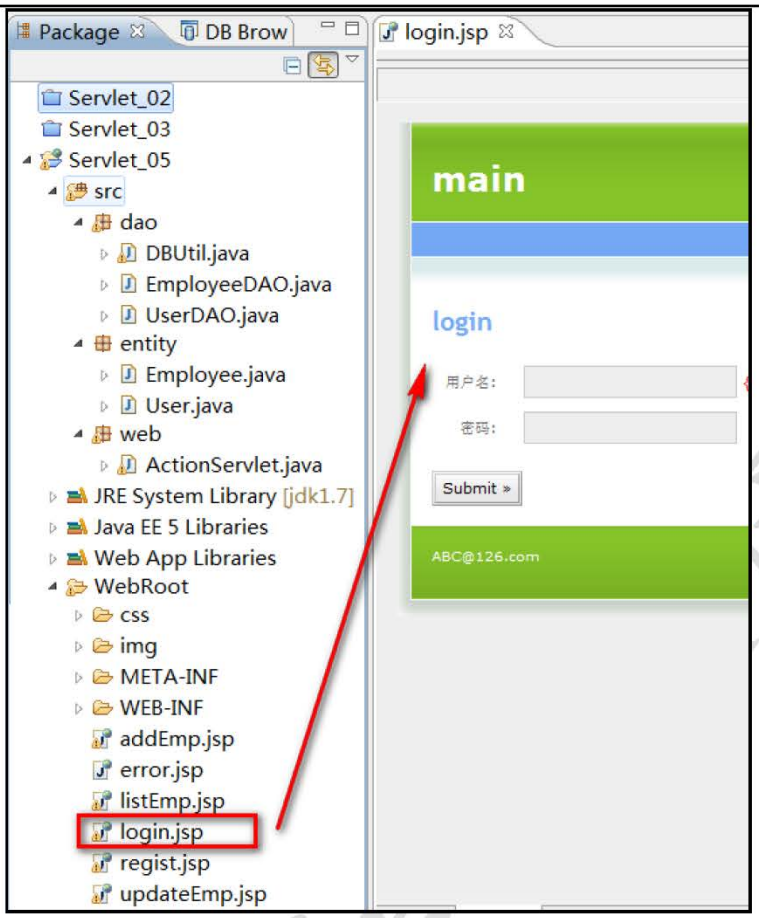


图 - 40

步骤七： 修改 ActionServlet 页面

```
} else if (action.equals("regist")) {




} else if (action.equals("login")) {
    //1.获取表单中的用户名等信息
    //2.根据用户名查询实体
    //3.如果实体等于null或者密码不相等则绑定信息后转
    发回login
    //4.非空且密码相等则登录成功，重定向到查询页面
}
```

图 – 41

修改 login.jsp 页面，读取绑定信息，非空则显示在文本框的后面

```
<input type="text"  class="inputgri" name="name" />
<%
String errorMsg = (String)request.getAttribute("login_err");
%><span style="color:red;"><%=errorMsg==null?"":errorMsg%></span>
```

图 - 42

运行查看结果。

- **完整代码**

User.java 文件代码：

```
package entity;

/**
 * 实体类：用于描述用户，与数据库中的 t_user 表中的字段一一对应
 */
public class User {
 private int id;
 private String userName;
 private String pwd;
 private String name;
 private String gender;

 @Override
 public String toString() {
     return id + "  " + userName + "  " +
             pwd + "  " + name + "  " + gender;
 }

 public int getId() {
     return id;
 }

 public void setId(int id) {
     this.id = id;
 }

 public String getUserName() {
     return userName;
 }

 public void setUserName(String userName) {
     this.userName = userName;
 }

 public String getPwd() {
     return pwd;
 }

 public void setPwd(String pwd) {
     this.pwd = pwd;
 }

 public String getName() {
     return name;
 }

 public void setName(String name) {
     this.name = name;
 }

 public String getGender() {
     return gender;
 }

 public void setGender(String gender) {
     this.gender = gender;
 }

}
```

UserDAO.java 文件的代码：

```
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import entity.User;
/**
 *  用户的数据操作类
 */
public class UserDAO {
 /**
  * 根据用户名称查找用户对象
  * @param userName 用户名称
  * @return  用户实体
  */
 public User findByUserName(String userName) throws Exception {

     User user = null;
     Connection con = null;
     PreparedStatement stmt = null;
     ResultSet rs = null;

     try {
         con = DBUtil.getConnection();
         stmt = con.prepareStatement("select * from t_user where username=?");
         stmt.setString(1, userName);
         rs = stmt.executeQuery();
         if (rs.next()) {
             user = new User();
             user.setId(rs.getInt("id"));
             user.setUserName(rs.getString("username"));
             user.setName(rs.getString("name"));
             user.setPwd(rs.getString("pwd"));
             user.setGender(rs.getString("gender"));
         }
     } catch (Exception e) {
         e.printStackTrace();
         throw e;
     } finally {
         DBUtil.close(con);
     }

     return user;
 }

 /**
  * 保存实体信息到数据库
  */
 public void save(User user) throws Exception {

     Connection con = null;
     PreparedStatement stmt = null;

     try {
         con = DBUtil.getConnection();
         stmt = con.prepareStatement("insert into t_user
values(user_id_seq.nextval,?,?,?,?)");
         stmt.setString(1, user.getUserName());
         stmt.setString(2, user.getPwd());
         stmt.setString(3, user.getName());
         stmt.setString(4, user.getGender());
```

```
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
            throw e;
        } finally {
            DBUtil.close(con);
        }
    }
}
```

ActionServlet.java 文件代码：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import dao.EmployeeDAO;
import dao.UserDAO;
import entity.Employee;
import entity.User;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        // 设置编码、解码格式
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        // 获取请求资源路径，截取具体动作
        String uri = request.getRequestURI();
        String action = uri.substring(uri.lastIndexOf("/") + 1,
                uri.lastIndexOf("."));

        // 创建员工管理操作类
        EmployeeDAO dao = new EmployeeDAO();

        // 判断请求动作种类，分不同种类情况处理
        if (action.equals("list")) {// 员工信息列表
            try {
                List<Employee> emps = dao.findAll();
                request.setAttribute("emps", emps);
                request.getRequestDispatcher("listEmp.jsp").forward(request,
                        response);
            } catch (Exception e) {
                e.printStackTrace();
                // 编程式的处理异常
                request.setAttribute("err_msg", "系统出错，请重试");
                request.getRequestDispatcher("error.jsp").forward(request,
                        response);
            }
        } else if (action.equals("add")) {// 增加员工
```

```
            String name = request.getParameter("name");
            try {
                double salary = Double.parseDouble(request
                        .getParameter("salary"));
                int age = Integer.parseInt(request.getParameter("age"));
                Employee emp = new Employee();
                emp.setName(name);
                emp.setSalary(salary);
                emp.setAge(age);

                dao.save(emp);
                response.sendRedirect("list.do");
            } catch (Exception e) {
                e.printStackTrace();
                // 将异常抛给容器，在 web.xml 中添加声明，变成声明式的处理异常
                throw new ServletException(e);
            }
        } else if (action.equals("delete")) {// 删除员工
            int id = Integer.parseInt(request.getParameter("id"));
            try {
                dao.delete(id);
                response.sendRedirect("list.do");
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else if (action.equals("load")) {// 加载某个员工
            int id = Integer.parseInt(request.getParameter("id"));
            try {
                Employee emp = dao.findById(id);
                request.setAttribute("emp", emp);

    request.getRequestDispatcher("updateEmp.jsp").forward(request,
                        response);
            } catch (Exception e) {
                e.printStackTrace();
            }

        } else if (action.equals("update")) {// 更新员工
            int id = Integer.parseInt(request.getParameter("id"));
            String name = request.getParameter("name");
            double salary = Double.parseDouble(request.getParameter("salary"));
            int age = Integer.parseInt(request.getParameter("age"));
            Employee emp = new Employee();
            emp.setName(name);
            emp.setSalary(salary);
            emp.setAge(age);
            emp.setId(id);
            try {
                dao.modify(emp);
                response.sendRedirect("list.do");
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else if (action.equals("regist")) {//注册
            UserDAO userDAO = new UserDAO();
            String userName = request.getParameter("username");
            String name = request.getParameter("name");
            String pwd = request.getParameter("pwd");
            String gender = request.getParameter("sex");
            try {
                User u = userDAO.findByUserName(userName);
                if (u != null) {
                    request.setAttribute("regis_err", "用户名已经存在");
                    request.getRequestDispatcher("regist.jsp").forward(
                            request, response);
```

```
                } else {
                    u = new User();
                    u.setUserName(userName);
                    u.setName(name);
                    u.setPwd(pwd);
                    u.setGender(gender);
                    userDAO.save(u);
                    response.sendRedirect("login.jsp");
                }
            } catch (Exception e) {
                e.printStackTrace();
                throw new ServletException(e);
            }
        } else if (action.equals("login")) {//登录

            String userName = request.getParameter("name");
            String pwd = request.getParameter("pwd");
            UserDAO userDAO = new UserDAO();
            try {
                User u = userDAO.findByUserName(userName);
                if (u == null || !u.getPwd().equals(pwd)) {
                    request.setAttribute("login_err", "用户名或密码错误");
                    request.getRequestDispatcher("login.jsp").forward(
                            request, response);
                } else {
                    response.sendRedirect("list.do");
                }
            } catch (Exception e) {
                e.printStackTrace();
                throw new ServletException(e);
            }

        }
    }
}
```

regist.jsp 文件代码：

```
<%@ page contentType="text/html; charset=UTF-8"   pageEncoding="UTF-8"%>

<html>
 <head>
     <title>注册</title>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <link rel="stylesheet" type="text/css" href="css/style.css" />
 </head>
 <body>
     <div id="wrap">
         <div id="top content">
                 <div id="header">
                     <div id="rightheader">
                         <p>
                             2009/11/20
                             <br />
                         </p>
                     </div>
                     <div id="topheader">
                         <h1 id="title">
                             <a href="#">main</a>
                         </h1>
                     </div>
                     <div id="navigation">
                     </div>
                 </div>
```

```
                <div id="content">
                    <p id="whereami">
                    </p>
                    <h1>
                        注册
                    </h1>
                    <form action="regist.do" method="post">
                        <table cellpadding="0" cellspacing="0" border="0"
                            class="form_table">
                            <tr>
                                <td valign="middle" align="right">
                                    用户名:
                                </td>
                                <td valign="middle" align="left">
                                    <input    type="text"    class="inputgri"
name="username" />
                                    <%
                                        String          errorMsg          =
(String)request.getAttribute("regis err");
                                    %><span
style="color:red;"><%=errorMsg==null?"":errorMsg%></span>
                                </td>
                            </tr>
                            <tr>
                                <td valign="middle" align="right">
                                    真实姓名:
                                </td>
                                <td valign="middle" align="left">
                                    <input    type="text"    class="inputgri"
name="name" />
                                </td>
                            </tr>
                            <tr>
                                <td valign="middle" align="right">
                                    密码:
                                </td>
                                <td valign="middle" align="left">
                                    <input  type="password"  class="inputgri"
name="pwd" />
                                </td>
                            </tr>
                            <tr>
                                <td valign="middle" align="right">
                                    性别:
                                </td>
                                <td valign="middle" align="left">
                                    男
                                    <input    type="radio"    class="inputgri"
name="sex" value="m" checked="checked"/>
                                    女
                                    <input    type="radio"    class="inputgri"
name="sex" value="f"/>
                                </td>
                            </tr>

                            <tr>
                                <td valign="middle" align="right">
                                    验证码:
                                    <img id="num" src="image" />
                                    <a                    href="javascript:;"
onclick="document.getElementById('num').src = 'image?'+(new Date()).getTime()">
换一张</a>
                                </td>
```

```
                                <td valign="middle" align="left">
                                    <input    type="text"    class="inputgri"
name="number" />
                                </td>
                            </tr>
                        </table>
                        <p>
                            <input type="submit" class="button" value="Submit
&raquo;" />
                        </p>
                    </form>
                </div>
            </div>
            <div id="footer">
                <div id="footer_bg">
                ABC@126.com
                </div>
            </div>
        </div>
    </body>
   </html>
```

login.jsp 文件代码：

```
   <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

   <html>
   <head>
   <title>login</title>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <link rel="stylesheet" type="text/css" href="css/style.css" />
   </head>

   <body>
    <div id="wrap">
        <div id="top content">
            <div id="header">
                <div id="rightheader">
                    <p>
                        2009/11/20 <br />
                    </p>
                </div>
                <div id="topheader">
                    <h1 id="title">
                        <a href="#">main</a>
                    </h1>
                </div>
                <div id="navigation"></div>
            </div>
            <div id="content">
                <p id="whereami"></p>
                <h1>login</h1>
                <form action="login.do" method="post">
                    <table cellpadding="0" cellspacing="0" border="0"
                        class="form table">
                        <tr>
                            <td valign="middle" align="right">用户名:</td>
                            <td valign="middle" align="left">
                               <input type="text"  class="inputgri"
name="name" />
                                <%
                                String               errorMsg               =
(String)request.getAttribute("login err");
                                %><span
style="color:red;"><%=errorMsg==null?"":errorMsg%></span>
```

```
                        </td>
                    </tr>
                    <tr>
                        <td valign="middle" align="right">密码:</td>
                        <td      valign="middle"      align="left"><input
type="password"
                            class="inputgri" name="pwd" />
                        </td>
                    </tr>
                </table>
                <p>
                    <input   type="submit"   class="button"   value="Submit
&raquo;" />
                </p>
            </form>
        </div>
    </div>
    <div id="footer">
        <div id="footer bg">ABC@126.com</div>
    </div>
 </div>
</body>
</html>
```

达内IT培训集

课后作业

1. 简述什么是转发?以及如何实现转发?

2. 简述转发和重定向有什么区别?

3. NETCTOSS 的登录功能和帐务帐号的 CRUD。

将美工制作的页面转变成 JSP 页面，使用转发、重定向等技术实现完整的帐务帐号管理功能。

另外，有管理员表的信息如表-1 所示：

表 - 1 管理员表 （ADMIN_INFO）

字段名称	类型	备注	字段描述
ADMIN_ID	NUMBER(4)	PRIMARY KEY NOT NULL	主键，管理员 ID
ADMIN_CODE	VARCHAR2(30)	UNIQUE NOT NULL	管理员帐号
PASSWORD	VARCHAR2(30)	NOT NULL	密码
NAME	VARCHAR2(30)	NOT NULL	姓名
TELEPHONE	VARCHAR2(15)	NULL	电话
EMAIL	VARCHAR2(50)	NULL	电子邮件
ENROLLDATE	DATE	NOT NULL DEFAULT SYSDATE	创建日期

为 NETCTOSS 项目添加登录功能。登录成功后显示所有帐务帐号信息。