第2章

Servlet基础

Servlet基础

2.1 Servlet接口与HttpServlet类

2.2 Web应用程序与DD文件

2.3 处理HTTP请求

2.1 Servlet接口与HttpServlet类

2.1.1 Servlet接口

2.1.2 HttpServlet类

2.1.1 Servlet接口

- 什么是Servlet？

– Servlet(Server Applet)是Java Servlet的简称，称为小服务程序，是用Java编写的服务器端程序，具有独立于平台和协议的特性，主要功能在于交互式地浏览和生成数据，生成动态Web内容。

– 广义的Servlet是指任何实现了这个Servlet接口的类，狭义的Servlet是指Java语言实现的一个接口。Servlet运行于支持Java的Web容器中。

2.1.1 Servlet接口

- **Servlet API**

–Servlet规范提供了一个标准的，平台独立的框架，实现了Servlet和容器之间的通信。该框架是由一组Java接口和类组成的，它们称为Servlet API。

- **Servlet 3.0 API由4个包组成：**

–javax.servlet包

–javax.servlet.http包

–javax.servlet.annotation包

–javax.servlet.descriptor包

2.1.1 Servlet接口

- **Servlet接口**

–javax.servlet.Servlet接口是Servlet API中的核心接口，每个Servlet必须直接或间接实现该接口。

- **Servlet接口定义了5个方法：**

– void init(ServletConfig config)

– void service(ServletRequest req, ServletResponse res)

– void destroy()

– ServletConfig getServletConfig()

– String getServletInfo()

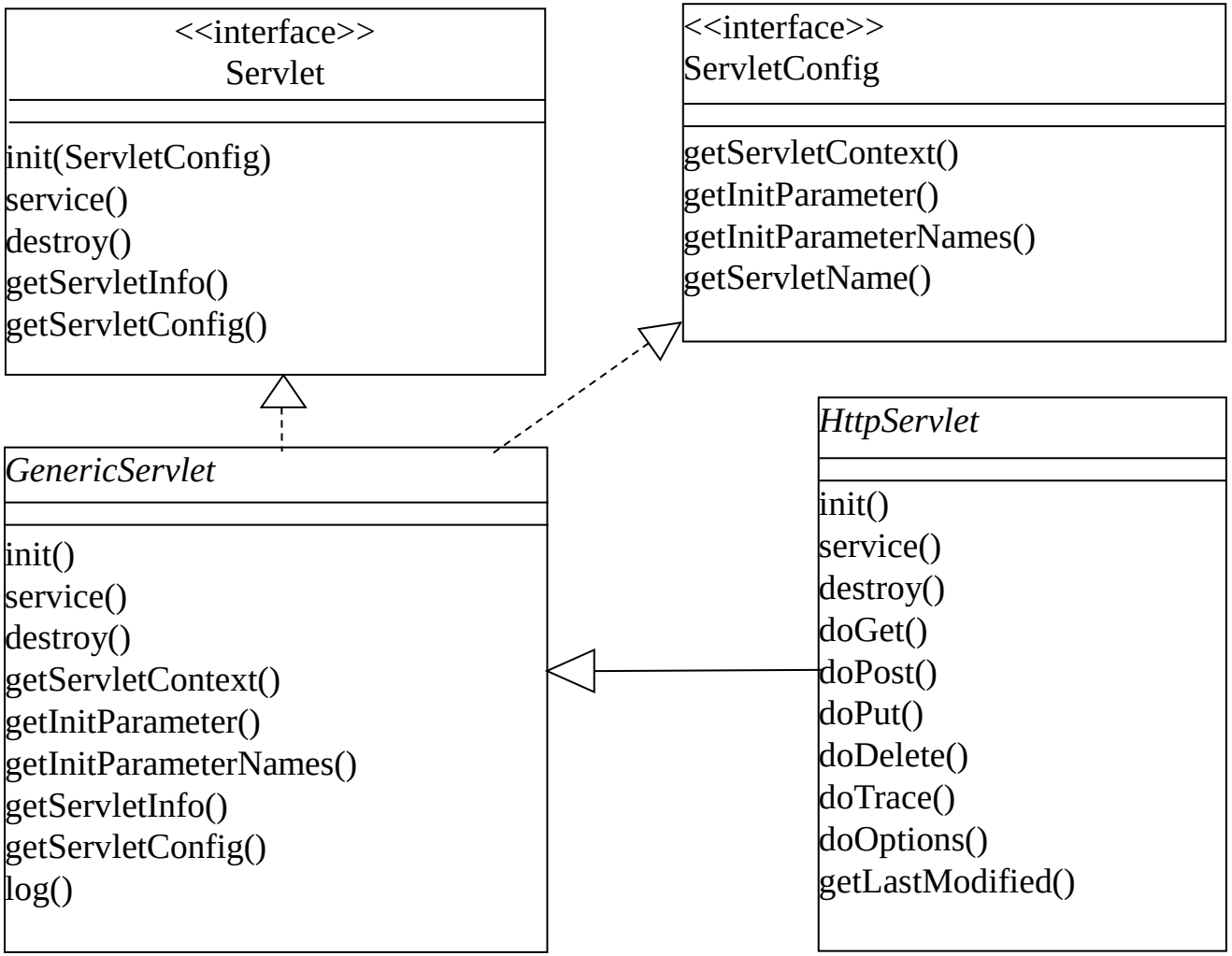
2.1.2 HttpServlet类

- **HttpServlet类**

  –HttpServlet抽象类用来实现针对HTTP协议的Servlet，它扩展了GenericServlet类，该类实现了Servlet接口和ServletConfig接口。

- **HttpServlet类与其他接口和类的层次关系如图所示**

2.1.2 HttpServlet类



2.1.2 HttpServlet类

Servlet例子

```
package ***;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorldServlet extends HttpServlet {
        protected void doGet(HttpServletRequest request,
                HttpServletResponse response)
                throws ServletException, IOException {
                PrintWriter out=response.getWriter();
                out.println("Hello,world");
                out.close();
        }
}
```

2.1.2 HttpServlet类

- **doXxx()方法**

–在HttpServlet中针对不同的HTTP请求方法定义了不同的处理方法，如处理GET请求的doGet()方法格式如下：

```
void doGet (HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, IOException
```

–我们编写的Servlet通常覆盖doGet()方法或doPost()方法。

2.2 Web应用程序与DD文件

2.2.1 Web应用程序

2.2.2 应用服务器

2.2.3 Web应用程序结构

2.2.4 部署描述文件

2.2.1 Web应用程序

- **Web应用程序**
  - Web应用程序是一种可以通过Web访问的应用程序。
  - 一个Web应用程序是由完成特定任务的各种Web组件构成。
  - 在实际应用中，Web应用程序是由多个Servlet、JSP页面、HTML文件以及图像文件等组成。

2.2.2 应用服务器

应用服务器

– Web应用程序驻留在应用服务器上。应用服务器为 Web 应用程序提供一种简单的和可管理的对系统资源的访问机制。

– 常用的有Tomcat 、Jetty、Resin、JRun、JBoss、Oracle的WebLogic和IBM 的WebSphere等。

2.2.3 Web应用程序结构

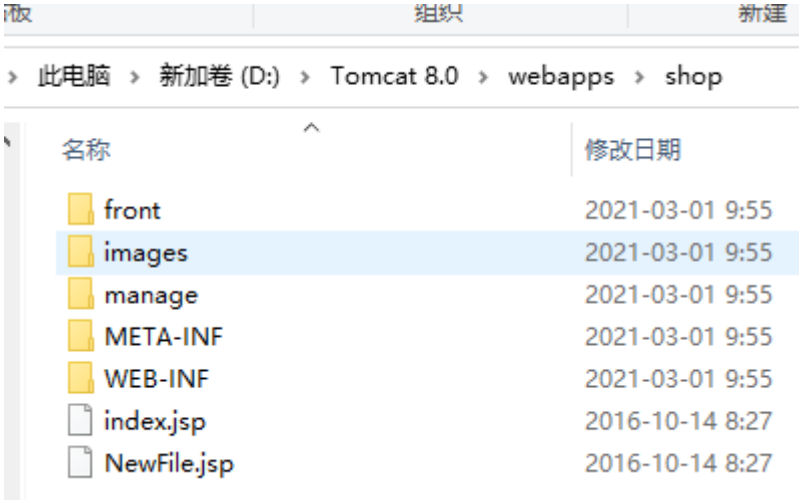
Web应用程序结构

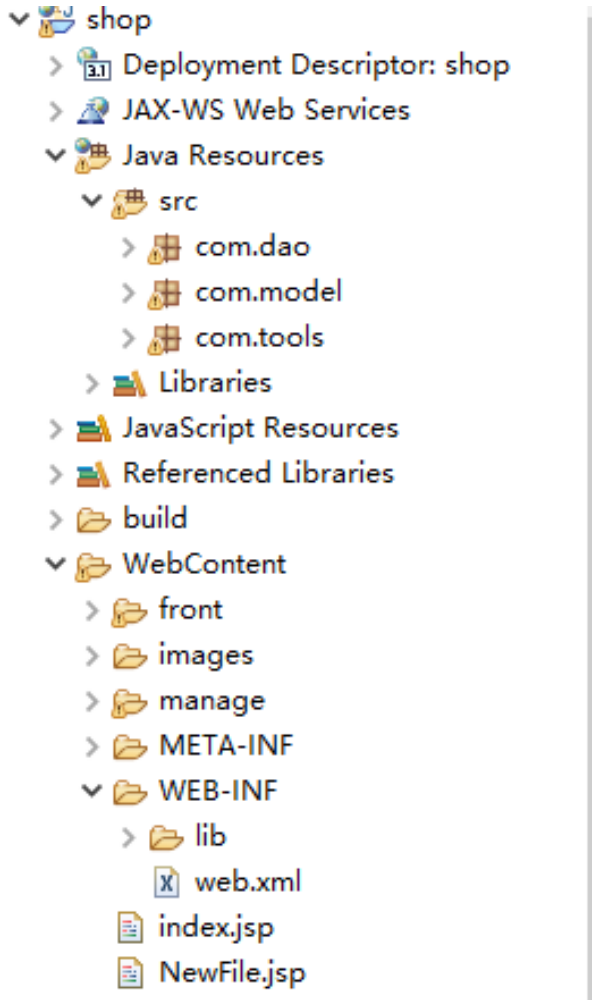
–Web应用程序具有严格定义的目录结构。一个Web应用程序的所有资源被保存在一个结构化的目录中，目录结构是按照资源和文件的位置严格定义的。

–Tomcat服务器的webapps目录是所有Web应用程序的根目录。

–假如有一个名为helloweb的Web应用程序，在webapps中就应建立一个helloweb目录。

2.2.3 Web应用程序结构





2.2.3 Web应用程序结构

1. 文档根目录

–每个Web应用程序都有一个文档根目录（document root），它是应用程序所在的目录。

–应用程序所有可以被公开访问的文件都应该放在该目录或其子目录中。

–通常把该目录中的文件组织在多个子目录中。例如，HTML文件存放在html目录中，JSP页面存放在jsp目录中，而图像文件存放在images目录中，这样方便对Web应用程序中的文件进行管理。

2.2.3 Web应用程序结构

- 要访问helloweb应用程序根目录下的index.html 文件，应该使用下面的URL。

  http://www.myserver.com/helloweb/index.html

- 如果要访问html目录中的/hello.html 文件，应该使用下面的URL。

  http://www.myserver.com/helloweb/**html**/hello.html 。

2.2.3 Web应用程序结构

URL与URI

–URL（Uniform Resource Locator）称为**统一资源定位器**，指向Internet上位于某个位置的某个资源。资源包括HTML文件、图像文件、Servlet和JSP页面等。

http://www.baidu.com/index.html

http://www.mydomain.com/files/sales/report.html

http://localhost:8080/helloweb/hello.do。

–URL通常由4部分组成：协议名称、所在主机的DNS名、可选的端口号和资源的路径及名称。端口号和资源名称可以省略。

2.2.3 Web应用程序结构

URI

–URI（Uniform Resource Identifier）称为**统一资源标识符**，是以特定语法标识一个资源的字符串。

–URI由模式和模式特有的部分组成，在Web应用中可以使用三种类型的URI：

1）绝对URI。例如，http://www.mydomain.com/sample和http://localhost:8080/taglibs都是绝对URI。

2.2.3 Web应用程序结构

2）根相对URI。以“/”开头且不带协议、主机名或端口号的URI。它被解释为相对于Web应用程序文档根目录。/mytaglib和/taglib1/helloLib是根相对URI。

3）非根相对URI。不以“/”开头也不带协议、主机名或端口号的URI。它被解释为相对于当前页面或相对于WEB-INF目录，这要看它在哪使用的。HelloLib和taglib2/helloLib是非根相对URI。

2.2.3 Web应用程序结构

- **2. WEB-INF目录**
- 每个Web应用程序在它的根目录中都必须有一个WEB-INF目录。该目录中主要存放供服务器访问的资源，客户端无权访问。
- 该目录主要包含三个内容。
- classes目录:存放支持该Web应用程序的类文件，如Servlet类文件、JavaBeans类文件等。

2.2.3 Web应用程序结构

- **2. WEB-INF目录**
- lib目录:存放Web应用程序使用的全部JAR文件，包括第三方的JAR文件。JDBC驱动程序JAR文件应该存放在这里。
- web.xml文件:通常每个Web应用程序可以建立一个web.xml文件。它包含Web容器运行Web应用程序所需要的信息，如Servlet声明、映射、属性、授权及安全限制等。

2.2.3 Web应用程序结构

3. 默认的Web应用程序

–Tomcat服务器还维护一个默认的Web应用程序。

–Tomcat安装目录的\webapps\ROOT目录被设置为默认的文档根目录。

–它与其他的Web应用程序类似，只不过访问它的资源不需要指定应用程序的名称或上下文路径。访问默认Web应用程序的URL为：

http://localhost:8080/

2.2.4 部署描述文件

▪部署描述文件

–Web应用程序中包含多种组件，有些组件可使用注解配置，有些组件需使用部署描述文件配置。

–部署描述文件（Deployment Descriptor，简称DD）可用来初始化Web应用程序的组件。

–Web容器在启动时读取该文件，对应用程序配置，所以有时也将该文件称为配置文件。

–【例2-1】web.xml文件。是一个简单的部署描述文件

2.2.4 部署描述文件 （例2-1）

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1"  metadata-complete="true">
  <description>Servlet and JSP Examples.</description>
  <display-name>Servlet and JSP Examples</display-name>

  <servlet>
     <servlet-name>helloServlet</servlet-name>
     <servlet-class>com.demo.HelloServlet</servlet-class>
     <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>helloServlet</servlet-name>
    <url-pattern>/hs</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

2.2.4 部署描述文件

1.DD文件的DTD定义

–文档类型定义（Document Type Definition，DTD）的标准规定了文档的语法和标签的规则，这些规则包括一系列的元素和实体声明。

–<!ELEMENT web-app (description?,display-name?, icon?,distributable?,context-param*, filter*, filter-mapping*, listener*, **servlet*,servlet-mapping*,** session-config?, **welcome-file-list?,** error-page*, jsp-config*, security-constraint*,login-config?, security-role*)>

2.2.4 部署描述文件

2.<Servlet>元素

–为Web应用程序定义一个Servlet，下面代码展示了<servlet>元素的一个典型的使用。

```
<servlet>
    <servlet-name>helloServlet</servlet-name>
    <servlet-class>com.demo.HelloServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

2.2.4 部署描述文件

- 说明

–<load-on-startup>指定是否在Web应用程序启动时载入该Servlet，一般情况下，Servlet是在被请求时由容器装入内存的，也可以使用Servlet在Web容器启动时就装入内存。

–如果该数值为负数，则容器根据需要决定何时装入Servlet；如果为正数，则在Web应用程序启动时载入该Servlet。

–对不同的Servlet，可以指定不同的值，值小的先装入。

2.2.4 部署描述文件

3.<servlet-mapping>元素

–为Servlet定义一个映射，它指定哪个URL模式被该Servlet处理。

```
<servlet-mapping>
    <servlet-name>helloServlet</servlet-name>
    <url-pattern>/hs</url-pattern>
</servlet-mapping>
```

2.2.4 部署描述文件

4.<welcome-file-list>元素

–定义欢迎页面列表。

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
```

–在Web服务器中，如果访问的URL是目录，它将在该目录中首先查找index.html文件 。

2.2.4 部署描述文件 –> 注解

- 为了简化Servel的配置，对于3.0以后版本，可以使用注解的方式，设置相应参数。例：

@WebServlet(name=“helloServlet”,value=“/hs”）

- 要使用需要导入包

```
import javax.servlet.annotation.WebServlet;
```

2.3 处理HTTP请求

2.3.1 HTTP请求结构

2.3.2 发送和处理HTTP请求

2.3.1 HTTP请求结构

- HTTP请求结构
  - 由客户向服务器发出的消息叫作HTTP请求（HTTP request）。
  - HTTP请求通常包括请求行、请求头、空行和请求的数据。
  - 图2是一个典型的POST请求。

2.3.1 HTTP请求结构

HTTP的请求行由三部分组成：方法名、请求资源的URI

接下来是一个空行。空行的后面是请求的数据。如果是GET请求，可能不包含请求数据。

以指定请求使用的浏览器信息、字符编码信息及客户能处理的页面类型等。

2.3.1 HTTP请求结构

- GET方法和POST方法比较
  - GET方法用来检索资源。它的含义是“获得（get）由该URI标识的资源”。
  - POST方法用来向服务器发送需要处理的数据，它的含义是“将数据发送（post）到由该URI标识的主动资源”。
  - POST方法只能请求**动态资源**，如将表单数据发给程序处理。

2.3.1 HTTP请求结构

- 下表比较了GET方法和POST方法的不同。

特征	GET方法	POST方法
资源类型	主动的或被动的	主动的
数据类型	文本	文本或二进制数据
数据量	一般不超过255个字符	没有限制
可见性	数据是URL的一部分，在浏览器的地址栏中对用户可见	数据不是URL的一部分而是作为请求的消息体发送，在浏览器的地址栏中对用户不可见
数据缓存	数据可在浏览器的URL历史中缓存	数据不能在浏览器的URL历史中缓存

2.3.2 发送和处理HTTP请求

- 在客户端如果发生下面的事件，浏览器就向Web服务器发送一个HTTP请求。
  - –用户在浏览器的地址栏中输入URL并按回车键。
  - –用户点击了HTML页面中的超链接。
  - –用户在HTML页面中填写一个表单并提交。
- 在上面的三种方法中，前两种方法向Web服务器发送的都是GET请求。如果使用HTML表单发送请求，可以通过method属性指定使用GET请求或POST请求。

2.3.2 发送和处理HTTP请求

- 默认情况下使用表单发送的请求也是GET请求，如果发送POST请求，需要将method属性值指定为“post”，例如：

```
<form action="login.do" method="post">
    用户名：<input type="text" name="username" />
    密码：<input type="password" name="password" />
    <input type="submit"  value="登录">
</form>。
```

2.3.2 发送和处理HTTP请求

- 处理HTTP请求

–在HttpServlet类中针对每个HTTP方法定义了相应的doXxx()方法，Servlet使用这些方法处理请求，一般格式如下：

```
protected void doXxx(HttpServletRequest request,
                     HttpServletResponse response)
```

–这里，doXxx()方法依赖于HTTP方法，如处理GET请求使用doGet()方法，处理POST请求使用doPost()方法。

–所有的doXxx()方法都有两个参数：HttpServletRequest对象和HttpServletResponse对象。

Servlet基础 上半部分 小结

（1）Servlet接口与HttpServlet类

（2）Web应用程序与DD文件

（2）处理HTTP请求