第2章

Servlet技术模型

本章内容

- 2.1 Servlet AP
- 2.2 Servlet生命周期
- 2.3 分析请求
- 2.4 发送响应
- 2.5 Web应用程序及结构
- 2.6 部署描述文件
- 2.7 @WebServlet和@WebInitParam注解
- 2.8 ServletConfig接口

2.1 Servlet API

- Servlet是Java Web应用开发的基础，Servlet API定义了若干接口和类。
- Servlet规范提供了一个标准的，平台独立的框架实现在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。

2.1 Servlet API

- Servlet 3.0 API由下面4个包组成：
- javax.servlet包，定义了开发独立于协议的服务器小程序的接口和类。
- javax.servlet.http包，定义了开发采用HTTP协议通信的服务器小程序的接口和类。
- javax.servlet.annotation包，定义9个注解类型和2个枚举类型。
- javax.servlet.descriptor包，定义了访问Web应用程序配置信息的类型。

2.1.1 javax.servlet包

接口名	说　明
Filter	在请求和响应之间执行过滤任务的过滤器对象
FilterChain	Servlet容器向开发人员提供的一个过滤器链对象
FilterConfig	Servlet容器使用的过滤器配置对象
RequestDispatcher	将请求转发到其他资源的对象
Servlet	所有Servlet的根接口
ServletConfig	Servlet容器使用的Servlet配置对象，用来向Servlet传递信息
ServletContext	该接口定义了一些方法，Servlet可以与Servlet容器通信
ServletRequest	提供客户请求的对象
ServletResponse	提供服务器响应的对象
ServletContextListener	用于监听Web应用程序的监听器接口
ServletContextAttributeListener	用于监听Web应用程序属性的监听器接口
ServletRequestListener	用于监听请求对象的监听器接口
ServletRequestAttributeListener	用于监听请求对象属性的监听器接口
SingleThreadModel	实现单线程的接口，已不推荐使用

2.1.1 javax.servlet包

类 名	说 明
GenericServlet	定义了一般的、独立于协议的Servlet
ServletContextAttributeEvent	Servlet环境属性的事件类
ServletContextEvent	Servlet环境的事件类
ServletInputStream	从客户请求读取二进制数据的类
ServletOutputStream	向客户发送二进制数据的类
ServletRequestAttributeEvent	请求属性事件类
ServletRequestEvent	请求事件类
ServletRequestWrapper	请求对象包装类
ServletResponseWrapper	响应对象包装类
ServletException	当Servlet遇到一般错误时抛出该异常
UnavailableException	Servlet或过滤器在其永久或临时不可用时抛出的异常

1. Servlet接口

- 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。该接口定义了如下5个方法。
- public void init(ServletConfig config)
- public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
- public ServletConfig getServletConfig()
- public String getServletInfo()
- public void destroy()

2. ServletConfig接口

- ServletConfig接口为用户提供了有关Servlet配置信息。
- Servlet配置包括Servlet名称、Servlet上下文对象、Servlet初始化参数等。

3. GenericServlet类

- GenericServlet抽象类实现了Servlet接口和ServletConfig接口，提供了Servlet接口中除了service()方法外的所有方法的实现，同时增加了几个支持日志的方法。可以扩展该类并实现service()方法来创建任何类型的Servlet。

4. ServletRequest接口

- ServletRequest接口是独立于任何协议的请求对象，定义了获取客户请求信息的方法，如getParameter()、getProtocol()、getRemoteHost()等。

5. ServletResponse接口

- ServletResponse接口是独立于任何协议的响应对象，定义了向客户发送响应的方法，如setContentType()方法、sendRedirect()方法、getWriter()方法等。

2.1.2 javax.servlet.http包

- 该包提供创建使用HTTP协议的Servlet所需要的接口和类。
- 该包共定义8个接口和7个类，其中某些接口和类扩展了javax.servlet包中对应的接口和类来实现对HTTP协议的支持。

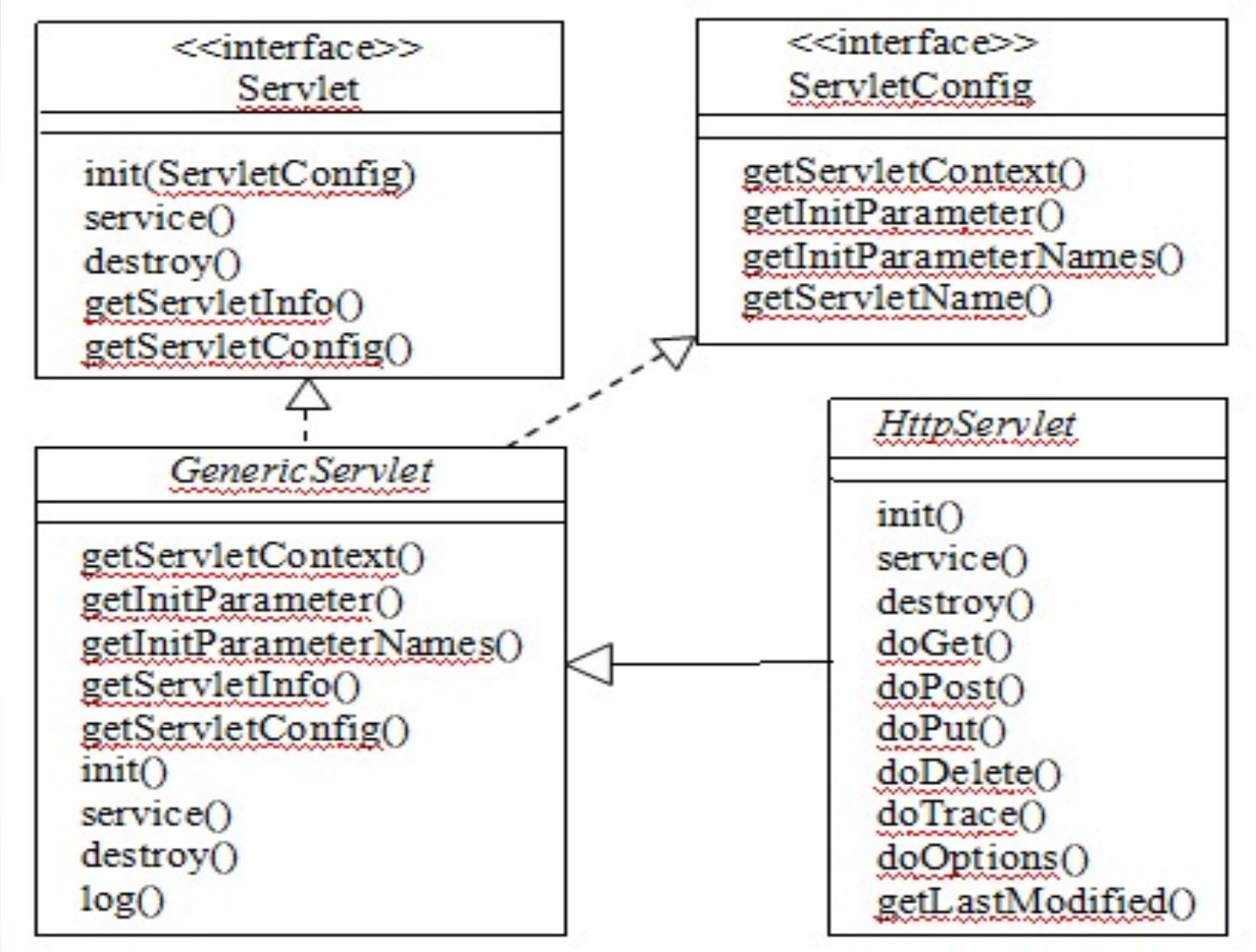
2.1.2 javax.servlet.http包

接口名	说　明
HttpServletRequest	该接口提供了有关HTTP请求的信息
HttpServletResponse	该接口提供了有关HTTP响应的信息
HttpSession	实现会话管理的接口，也用来存储用户信息
HttpSessionActivationListener	HTTP会话启动监听器接口
HttpSessionAttributeListener	HTTP会话属性监听器接口
HttpSessionBindingListener	HTTP会话绑定监听器接口
HttpSessionListener	HTTP会话监听器接口
HttpSessionContext	该接口已不推荐使用

2.1.2 javax.servlet.http包

类　名	说　明
HttpServlet	用于创建HTTP Servlet的抽象类
Cookie	创建Cookie对象的一个实现类
HttpServletRequestWrapper	HttpServletRequest接口的实现类
HttpServletResponseWrapper	HttpServletResponse接口的实现类
HttpSessionEvent	会话事件类
HttpSessionBindingEvent	会话绑定事件或会话属性事件类
HttpUtils	一个工具类，已不推荐使用

Servlet API的层次结构



1. HttpServlet类

- HttpServlet抽象类用来实现针对HTTP协议的Servlet，它扩展了GenericServlet类。
- 在HttpServlet类中增加了一新的service()方法，格式如下：

  protected void service (HttpServletRequest,
  HttpServletResponse)
  throws ServletException, IOException
- 是Servlet向客户提供服务的一个方法，我们编写的Servlet可以覆盖该方法。

1. HttpServlet类

- 此外，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：

```
protected void doGet(HttpServletRequest,
                     HttpServletResponse)
          throws ServletException,IOException
```

- 通常，我们编写的Servlet覆盖doGet()方法或doPost()方法。

2. HttpServletRequest接口

- HttpServletRequest接口扩展了ServletRequest接口并提供了针对HTTP请求操作方法，如定义了从请求对象中获取HTTP请求头、Cookie等信息的方法。

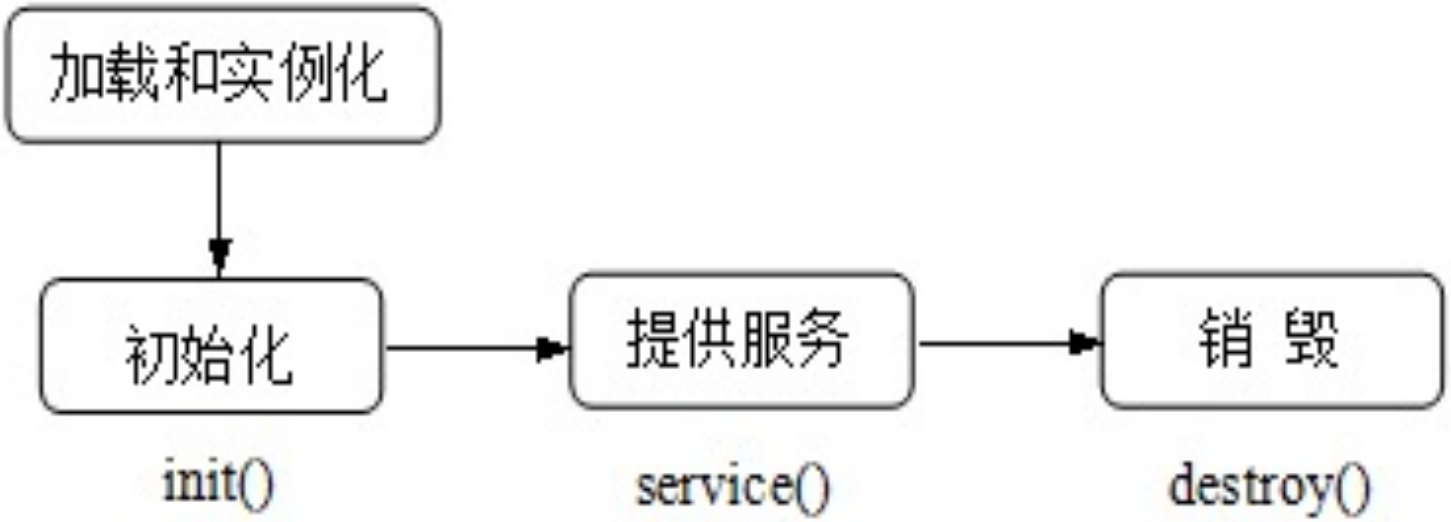
3. HttpServletResponse接口

- HttpServletResponse接口扩展了ServletResponse接口并提供了针对HTTP的发送响应的方法。它定义了为响应设置如HTTP头、Cookie信息的方法。

2.2 Servlet生命周期

- Servlet作为一种在容器中运行的组件，有一个从创建到销毁的过程，这个过程被称为Servlet生命周期。
- Servlet生命周期包括以下几个阶段：
  - 加载和实例化Servlet类，
  - 调用init()方法初始化Servlet实例，
  - 一旦初始化完成，容器从客户收到请求时就将调用它的service()方法，
  - 最后容器在Servlet实例上调用destroy()方法使它进入销毁状态。

2.2 Servlet生命周期



2.2.1 加载和实例化Servlet

- 对一个Servlet，可能在Web容器启动时或第一次被访问时加载到容器中。对每个Servlet，容器使用Class.forName()方法对其加载并实例化。
- 容器创建了Servlet实例后就进入生命周期阶段，Servlet生命周期方法包括
  - init()方法
  - service()方法
  - destroy()方法

2.2.2 初始化Servlet

- 容器创建Servlet实例后，将调用init(ServletConfig)方法初始化Servlet。
- 调用init(ServletConfig)方法后，容器将调用无参数的init()方法，之后Servlet就完成初始化。在Servlet生命周期中init()方法仅被调用一次。

2.2.3 为客户提供服务

- 在Servlet实例初始化后，它就准备为客户提供服务。
- 当容器接收到对Servlet的请求时，容器根据请求中的URL找到正确的Servlet，首先创建两个对象(请求和响应)，然后创建一个新的线程，在该线程中调用service()方法，同时将请求对象和响应对象作为参数传递给该方法。

2.2.3 为客户提供服务

- Servlet使用响应对象（response）获得输出流对象，调用有关方法将响应发送给客户浏览器。
- 之后，线程将被销毁或者返回到容器管理的线程池。请求和响应对象已经离开其作用域，也将被销毁。最后客户得到响应。

2.2.4 销毁和卸载Servlet

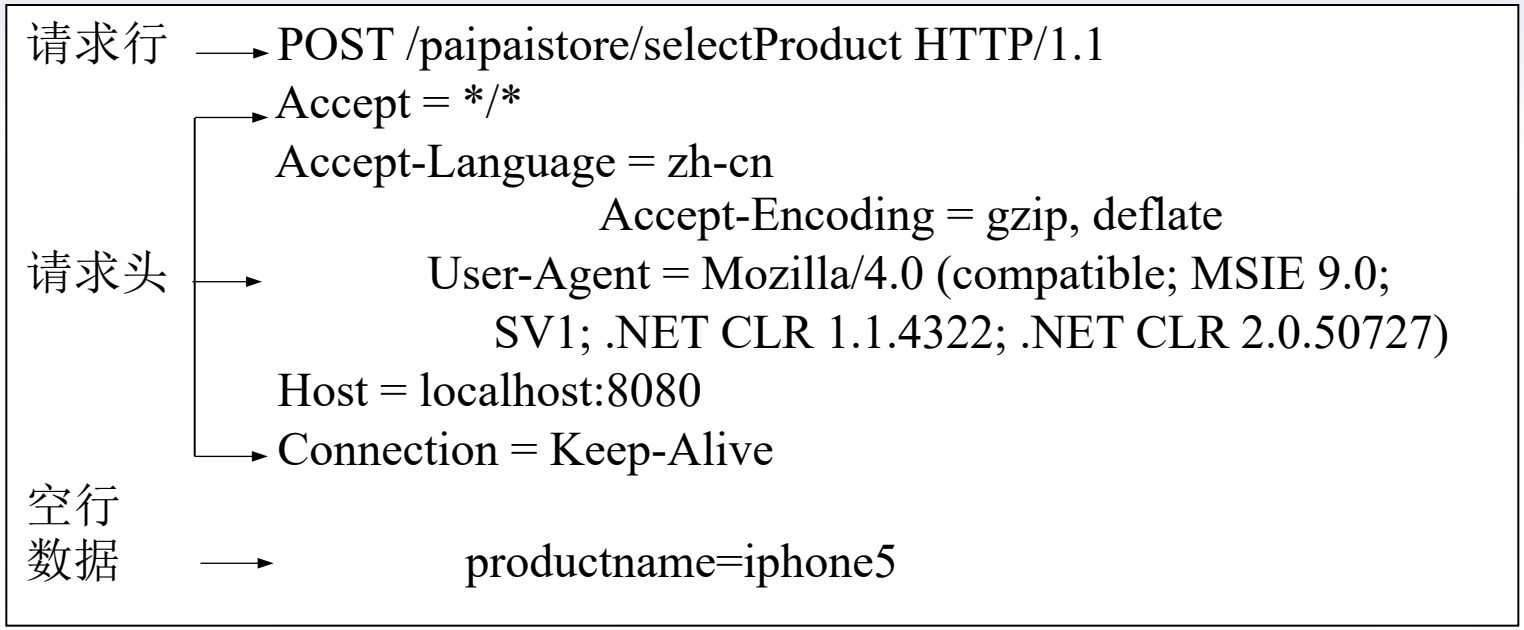
- 当容器决定不再需要Servlet实例时，它将在Servlet实例上调用destroy()方法，Servlet在该方法中释放资源，如它在init()方法中获得的数据库连接。一旦该方法被调用，Servlet实例不能再提供服务。
- 一旦Servlet实例被销毁，它将作为垃圾被回收。如果Web容器关闭，Servlet也将被销毁和卸载。

2.3 分析请求

- HTTP消息是客户向服务器的请求或者服务器向客户的响应。
- HTTP消息的各部分

消息部分	说明
请求行或状态行	指定请求或响应消息的目的
请求头或响应头	指定元信息，如关于消息内容的大小、类型、编码方式
空行	
可选的消息体	请求或响应消息的主要内容

2.3.1 HTTP请求结构



2.3.1 HTTP请求结构

- 由客户向服务器发出的消息叫做HTTP请求。

1. 请求行

- HTTP的请求行由三部分组成：方法名、请求资源的URI和HTTP版本。这三部分由空格分隔。

2. 请求头

- 请求行之后的内容称为请求头（request header），它可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

- 接下来是一个空行。
- 空行的后面是请求的数据。

3. HTTP的请求方法

- 请求行中的方法名指定了客户请求服务器完成的动作。

方　法	说　明	方　法	说　明
GET	请求读取一个Web页面	DELETE	移除Web页面
POST	请求向服务器发送数据	TRACE	返回收到的请求
PUT	请求存储一个Web页面	OPTIONS	查询特定选项
HEAD	请求读取一个Web页面的头部	CONNECT	保留作将来使用

4. GET方法和POST方法

- 在所有的HTTP请求方法中，GET方法和POST方法是两种最常用的方法。
- 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”。

GET和POST方法的比较

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页面中添写一个表单并提交。

2.3.3 处理HTTP请求

- 在HttpServlet类中，除定义了service()方法为客户提供服务外，还针对每个HTTP方法定义了相应的doXxx()方法，一般格式如下：

```
protected void doXxx (HttpServletRequest,
                      HttpServletResponse)
throws ServletException, IOException;
```

HTTP方法	HttpServlet方法	HTTP方法	HttpServlet方法
GET	doGet()	DELETE	doDelete()
POST	doPost()	OPTIONS	doOptions()
HEAD	doHead()	TRACE	doTrace()
PUT	doPut()		

2.3.4 分析请求

- 客户发送给服务器的请求信息被封装在`HttpServletRequest`对象中，其中包含了由浏览器发送给服务器的数据，这些数据包括请求参数、客户端有关信息等。
-

1. 检索请求参数

- **请求参数**是随请求一起发送到服务器的数据，它是以名/值对的形式发送的。可以使用ServletRequest接口中定义的方法检索由客户发送的参数
- `public String getParameter(String name)`

  返回由name指定的请求参数值，如果指的参数不存在，则返回null值。使用该方必须确信指定的参数只有一个值。

1. 检索请求参数

- public String[] getParameterValues(String name)：返回指定参数name所包含的所有值，返回值是一个String数组。如果指定的参数不存在，则返回null值。

1. 检索请求参数

- public Enumeration getParameterNames()：返回一个Enumeration对象，它包含请求中所有的请求参数名，元素是String类型的。如果没有请求参数，则返回一个空的Enumeration对象。
- public Map getParameterMap()：返回一个包含所有请求参数的Map对象，该对象以参数名作为键、以参数值作为值。

请求参数传递的方法

（1）通过表单指定请求参数，每个表单域可以传递一个请求参数，这种方法适用于GET请求和POST请求。

（2）通过查询串指定请求参数，将参数名和值附加在请求的URL后面，这种方法只适用于GET请求。

- 程序2.1 login.jsp
- 程序2.2 LoginServlet.java

2. 检索客户端有关信息

- 在HttpServletRequest接口中还定义了下面常用的方法用来检索客户端有关信息：

  public String getMethod()

  public String getRemoteHost()

  public String getRemoteAddr()

  public int getRemotePort()

2. 检索客户端有关信息

public String getProtocol()

public String getRequestURI()

public String getQueryString()

public String getContentType()

public String getCharacterEncoding()

- 程序2.3 ClientInfoServlet.java

3. 检索HTTP请求头

- HTTP请求头是随请求一起发送到服务器息，它是以“名/值”对的形式发送。

请求头	内　容
User-Agent	关于浏览器和它的平台的信息
Accept	客户能接受并处理的MIME类型
Accept-Charset	客户可以接受的字符集
Accept-Encoding	客户能处理的页面编码的方法
Accept–Language	客户能处理的语言
Host	服务器的DNS名字
Authorization	访问密码保护的Web页面时，客户用这个请求头来标识自己的身份
Cookie	将一个以前设置的Cookie送回服务器
Date	消息被发送的日期和时间
Connection	指示连接是否支持持续连接，值Keep-Alive表示支持持续连接

3. 检索HTTP请求头

- public String getHeader(String name)：返回指定名称的请求头的值。
- public Enumeration getHeaders(String name)：返回指定名称的请求头的Enumeration对象。
- public Enumeration getHeaderNames()：返回一个Enumeration对象，它包含所有请求头名。
- public int getIntHeader(String name)：返回指定名称的请求头的整数值。
- public long getDateHeader(String name)：返回指定名称的请求头的日期值。
- 程序2.4 ShowHeadersServlet.java

2.3.5 请求转发

- 在实际应用中可能需要将请求转发(forward)到其他资源。
- 使用ServletRequest接口中定义的方法，格式如下：

RequestDispatcher getRequestDispatcher(String path)

RequestDispatcher接口定义了下面两个方法

- public void **forward**(ServletRequest request, ServletResponse response)：将请求转发到服务器上的另一个动态或静态资源（如Servlet、JSP页面或HTML页面）。
- public void **include**(ServletRequest request, ServletResponse response)：将控制转发到指定的资源，并将其输出包含到当前输出中。

2.3.6 使用请求对象存储数据

```
void setAttribute(String name,Object obj)
Object getAttribute(String name)
void removeAttribute(String name)
```

- 程序2.5 LoginServlet.java
- 程序2.6 welcome.jsp

2.3.7　实例：一个简单的考试系统

- 开发一个简单的考试系统，在JSP页面中建立一个表单，通过POST方法传递参数。
- 程序2.7　questions.jsp
- 程序2.8　SimpleTestServlet.java

2.3.8 文件上传

- 文件上传是将客户端的一个或多个文件传输到服务器上保存。
- 实现文件上传首先需要在客户端的HTML页面中通过一个表单打开一个文件，然后提交给服务器。
- 上传文件表单的<form>标签中应该指定enctype属性，它的值应该为“multipart/form-data”，<form>标签的method属性应该指定为“post”，同时表单应该提供一个<input type="file">的输入域用于指定上传的文件。

2.3.8 文件上传

- 在服务器端，可以使用请求对象的getInputStream()返回ServletInputStream输入流对象，文件内容就包含在该对象中，另外其中还包含表单域的名称和值、上传的文件名、内容类型等信息。例如，假设上传一个Java源文件，返回的输入流的内容可能如下。

2.3.8 文件上传

```
----------------------------7d81a5209008a
Content-Disposition: form-data; name="mnumber"
223344
----------------------------7d81a5209008a
Content-Disposition: form-data; name="fileName";
filename="C:\study\HelloWorld.java"
Content-Type: application/octet-stream
public class HelloWorld {
   public static void main(String ars[]){
      System.out.println("Hello,World!");
  }
}
----------------------------7d81a5209008a
Content-Disposition: form-data; name="submit"
提交
----------------------------7d81a5209008a--
```

2.3.8　文件上传

- 程序2.9　fileUpload.jsp
- 当表单提交时，浏览器将表单各部分的数据发送到服务器端，每个部分之间使用分隔符分隔开。通过请求对象的下面两个方法来处理上传的文件。

2.3.8 文件上传

- public Part getPart(String name)：返回用name指定名称的Part对象。
- public Collection<Part> getParts()：返回所有Part对象的一个集合。
- Part是Servlet 3.0 API新增的一个接口，定义在javax.servlet.http包中。它提供了下面的常用方法：
- public InputStream getInputStream() throws IOException：返回Part对象的输入流对象。

2.3.8 文件上传

- public String getContentType()：返回Part对象的内容类型。
- public String getName()：返回Part对象的名称。
- public long getSize()：返回Part对象的大小。
- public String getHeader(String name)：返回Part对象指定的MIME头的值。
- public Collection<String> getHeaders(String name)：返回name指定的头值的集合。

2.3.8 文件上传

- public Collection<String> getHeaderNames()：返回Part对象头名称的集合。
- public void delete() throws IOExceeption：删除临时文件。
- public void write(String fileName) throws IOException：将Part对象写到指定的文件中。
- 程序2.10 FileUploadServlet.java

2.3.8 文件上传

- 对实现文件上传的Servlet类必须使用@MultipartConfig注解，使用该注解告诉容器该Servlet能够处理multipart/form-data的请求。使用该注解，HttpServletRequest对象才可以得到表单数据的各部分。
- 使用该注解可以配置容器存储临时文件的位置，文件和请求数据的大小限制以及阈值大小。该注解定义了如表2-10所示的元素。

2.3.8 文件上传

表2-10 @MultipartConfig注解的常用元素

元素名	类 型	说 明
location	String	指定容器临时存储文件的目录位置
maxFileSize	long	指定允许上传文件的最大字节数
maxRequestSize	long	指定允许整个请求的multipart/form-data数据的最大字节数
fileSizeShreshold	int	指定文件写到磁盘后阈值的大小

2.3.8 文件上传

- 除了在注解中指定文件的限制外，还可以在web.xml文件中使用<servlet>的子元素<multipart-config>指定这些限制，该元素包括4个子元素，分别为：<location>、<max-file-size>、<max-request-size>和<file-size-threshold>。
- 在带有multipart/form-data的表单中还可以包含一般的文本域，这些域的值仍然可以使用请求对象的getParameter()得到。

2.4 发送响应

- 2.4.1 HTTP响应结构
- 2.4.2 理解ServletResponse
- 2.4.3 理解HttpServletResponse
- 2.4.4 发送状态码和错误消息

2.4.1 HTTP响应结构

- 由服务器向客户发送的HTTP消息称为HTTP响应（HTTP response）。
- 一个典型的HTTP响应消息

状态行	HTTP/1.1 200 OK
响应头	Date: Tue, 01 Sep 2004 23:59:59 GMT Content-Type: text/html Content-Length: 52
空行	
响应数据	<html> <body> Hello, John! </body></html>

1. 状态行与状态码

- 状态行由三部分组成，各部分由空格分隔：
  - HTTP版本
  - 说明请求结果的响应状态码
  - 描述状态码的短语
- `HTTP/1.1 404 Not Found //` 表示没有找到与给定的URI匹配的资源
- `HTTP/1.1 500 Internal Error //` 表示服务器检测到一个内部错误

2. 响应头

- 响应头是服务器向客户端发送的消息。

  Date响应头表示消息发送的日期。

  Content-Type响应头指定响应的内容类。

  Content-Length指示响应内容的长度。

3. 响应数据

- 空行的后面是响应的数据。

```
<html><body>
    <h1>Hello, World!</h1>
</body></html>
```

2.4.2 输出流与内容类型

- Servlet使用输出流向客户发送响应。
- 通常，在发送响应数据之前还需通过响应对象的setContentType()方法设置响应的内容类型。
- public PrintWriter getWriter()
- public ServletOutputStream getOutputStream() throws IOException
- public void setContentType(String type)

1. 使用PrintWriter

- PrintWriter对象被Servlet用来动态产生页面。调用响应对象的getWriter()方法返回PrintWriter类的对象，它可以向客户发送文本数据。

```
PrintWriter out = response.getWriter();
```

2. 使用ServletOutputStream

- 如果要向客户发送二进制数据（如JAR文件），应该使用OutputStream对象。

```
ServletOutputStream sos =
        response.getOutputStream();
```

3. 设置内容类型

- 在向客户发送数据之前，一般应该设置发送数据的MIME（Multipurpose Internet Mail Extensions）内容类型。MIME是描述消息内容类型的因特网标准。

```
response.setContentType("text/html;charset=UTF-8");
```

3. 设置内容类型

表2-11 常见的MIME内容类型

类型名	含义
application/msword	Microsoft Word文档
application/pdf	Acrobat 的pdf文件
application/vnd.ms-excel	Excel 电子表格
application/vnd.ms-powerpoint	PowerPoint演示文稿
application/jar	JAR文件
application/zip	ZIP压缩文件
audio/midi	MIDI音频文件
image/gif	GIF图像
image/jpeg	JPEG图像
text/html	HTML文档
text/plain	纯文本
video/mpeg	MPEG视频片段

3. 设置内容类型

- 通过将响应内容类型设置为“application/vnd.ms-excel”可将输出以Excel电子表格的形式发送给客户浏览器，这样客户可将结果保存到电子表格中。
- 输出内容可以是用制表符分隔的数据或HTML表格数据等，并且还可以使用Excel内建的公式。下面的Servlet使用制表符分隔数据生成Excel电子表格。程序2.11 ExcelServlet.java

2.4.3 设置响应头

- 响应头是随响应数据一起发送到浏览器的附加信息。
- public void setHeader(String name, String value)
- public void setIntHeader(String name, int value)
- public void setDateHeader(String name, long date)
- public void addIntHeader(String name, int value)
- public void addDateHeader(String name, long date)

典型的响应头名及其用途

响应头名称	说明
Date	指定服务器的当前时间
Expires	指定内容被认为过时的时间
Last-Modified	指定文档被最后修改的时间
Refresh	告诉浏览器重新装载页面
Content-Type	指定响应的内容类型
Content-Length	指定响应的内容的长度
Content-Disposition	为客户指定将响应的内容保存到磁盘上的名称
Content-Encoding	指定页面在传输过程中使用的编码方式

2.4.3 设置响应头

- 下面的ShowTimeServlet通过设置Refresh响应头实现每5秒钟刷新一次页面。
- 程序2.12 ShowTimeServlet.java
- 要告诉浏览器在5秒钟后跳转到http://host/path页面，可以使用下面语句。

```
response.setHeader("Refresh","5;URL=http://host/path/");
```

2.4.3 设置响应头

- 实际上，在HTML页面中通过在<head>标签内添加下面代码也可以实现这个功能。

```
<meta http-equiv="Refresh"
  content="5;URL=
  http://host/path/">
```

2.4.4 响应重定向

- Servlet可能决定不直接向浏览器发送响应，而是将响应重定向到其他资源。

```
public void sendRedirect(
                    String location)
```

- location为指定的新的资源的URL，该URL可以是绝对URL（如http://www.microsoft.com），也可以是相对URL。若路径以“/”开头，则相对于服务器根目录（如，/helloweb/login.html），若不以“/”开头，则相对于Web应用程序的文档根目录（如，login.jsp）。
- 程序2.13 RedirectServlet.java

2.4.4 响应重定向

- 关于sendRedirect()方法，应该注意如果响应被提交，即响应头已经发送到浏览器，就不能调用该方法，否则将抛出java.lang.IllegalStateException异常。

```
PrintWriter out =
  response.getWriter();
out.println("<html><body>Hello
  World!</body></html>");
out.flush();          // 响应在这一点被提交了
response.sendRedirect("http://www.cnn
  .com");
```

2.4.5 发送状态码和错误消息

- 服务器向客户发送的响应的第一行是状态行，它由三部分组成：HTTP版本、状态码和状态码的描述信息，如下是一个典型的状态行：

```
HTTP/1.1 200 OK
```

- 由于HTTP的版本是由服务器决定的，而状态的消息与状态码有关，因此，在Servlet中一般只需要设置状态码。

2.4.5 发送状态码和错误消息

- 状态码200是系统自动设置的，Servlet不需要指定该状态码。对其他状态码，可以由系统自动设置，也可用响应对象的setStatus()方法设置，该方法的格式为：

```
public void setStaus (int sc)
```

- 可以设置任意的状态码。参数sc表示要设置的状态码
- 对于404状态码，其消息为Not Found，HttpServletResponse接口中为该状态码定义的常量名为SC_NOT_FOUND。

2.4.5 发送状态码和错误消息

- 在HTTP协议1.1版中定义了若干状态码，这些状态码由3位整数表示，一般分为5类

状态码范围	含　义	示　例
100~199	表示信息	100表示服务器同意处理客户的请求
200~299	表示请求成功	200表示请求成功，204表示内容不存在
300~399	表示重定向	301表示页面移走了，304表示缓存的页面仍然有效
400~499	表示客户的错误	403表示禁止的页面，404表示页面没有找到
500~599	表示服务器的错误	500表示服务器内部错误，503表示以后再试

2.4.5 发送状态码和错误消息

- HTTP为常见的错误状态定义了状态码，这些错误状态包括：资源没有找到、资源被永久移动以及非授权访问等。所有这些代码都在接口HttpServletResponse中作为常量定义。
- HttpServletResponse也提供了sendError()方法用来向客户发送状态码，该方法有两个重载的形式，如下所示。

```
public void sendError (int sc)
public void sendError (int sc, String msg)
```

2.4.5 发送状态码和错误消息

- 第一个方法使用一个状态码，第二个方法同时指定显示消息。服务器在默认情况下创建一个HTML格式的响应页面，其中包含指定的错误消息。
- 例如，如果Servlet发现客户不应访问其结果，它将调用 `sendError(HttpServletResponse.SC_UNAUTHORIZED)`
- 程序2.14 StatusServlet.java

2.5 Web应用程序及结构

- 2.5.1 Web应用程序
- 2.5.2 应用服务器
- 2.5.3 Web应用程序的结构

2.5.1 Web应用程序

- 所谓**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（Web Components）构成的并通过Web将服务展示给外界。

2.5.2 应用服务器

- Web应用程序驻留在应用服务器（Application Server）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。它也提供低级的服务，如 HTTP 协议的实现和数据库连接管理。Servlet 容器仅仅是应用服务器的一部分。
-

2.5.2 应用服务器

- 市场上可以得到多种应用服务器，其中包括
- Apache 的Tomcat
- Caucho Technology 的Resin
- Macromedia 的JRun
- JBoss
- Oracle的WebLogic
- IBM 的WebSphere
- 其中有些如WebLogic、WebSphere不仅仅是Servlet容器，它们也提供对EJB、JMS以及其他Java EE技术的支持。

2.5.3 Web应用程序的结构

- 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安装目录的webapps目录是所有Web应用程序的根目录。

1. 理解文档根目录

- 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。
- 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

helloweb
- css (存放级联样式表文件)
- html (存放HTML文件)
- images (存放GIF、JPEG或PNG文件)
- js (存放JavaScript脚本文件)
- jsp (存放JSP文件)
- index.html (默认的欢迎文件)
- WEB-INF
  - classes（类文件目录）
    - com.demo.LoginServlet.class
  - lib（库文件目录）
    - *.jar(jdbcdriver.jar,mytaglib.jar)
  - web.xml（部署描述文件）

2. 理解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。
- 该目录主要包含三个内容。

  1）classes目录

  2）lib目录

  3）web.xml文件

3. Web归档文件

- 一个Web应用程序包含许多文件，可以将这些文件打包成一个扩展名为.war的Web归档文件中，一般称为**WAR文件**。
- 可以直接把一个WAR文件放到Tomcat的webapps目录中，Tomcat会自动把该文件的内容释放到webapps目录中并创建一个与WAR文件同名的应用程序。

4. 默认的Web应用程序

- 除用户创建的Web应用程序外，Tomcat服务器还维护一个默认的Web应用程序。*<tomcat-install>***webapps\ROOT**目录被设置为默认的Web应用程序的文档根目录。
- 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。

2.6 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。
- 程序2.15 web.xml

2.6.1 DD文件的定义

- 为了保证跨Web容器的可移植性，部署描述文件的文档类型定义（Document Type Definition，**DTD**）的标准由Sun公司制定。DTD规定了XML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的声明。
- 下面列出了<web-app>元素的DTD定义，这里给出常用元素。

<web-app>元素的DTD定义

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

在部署描述文件中定义的元素

元素名	说明
description	对应用程序的简短描述
display-name	定义应用程序的显示名称
context-param	定义应用程序的初始化参数
servlet	定义Servlet
servlet-mapping	定义Servlet映射
welcome-file-list	定义应用程序的欢迎文件
session-config	定义会话时间
listener	定义监听器类
filter	定义过滤器
filter-mapping	定义过滤器映射
error-page	定义错误处理页面
security-constraint	定义Web应用程序的安全约束
mime-mapping	定义常用文件扩展名的MIME类型

2.6.2 <servlet >元素

- <servlet>元素为Web应用程序定义一个Servlet，该元素的DTD定义如下。

```
<!ELEMENT servlet (description?,
  icon?, display-name?, servlet-
  name,
  (servlet-class | jsp-file),
  init-param*,
load-on-startup?, security-role-
  ref*)>
```

1. <servlet-name>元素

- 该元素用来定义Servlet名称，该元素是必选项。定义的名称在DD文件中应该唯一。可以通过ServletConfig的getServletName()方法检索Servlet名。

2. <servlet-class>元素

- 该元素指定Servlet类的完整名称，即需要带包的名称，例如com.demo.HelloServlet。
- 容器将使用该类创建Servlet实例。
- Servlet类以及它所依赖的所有类都应该在Web应用程序的类路径中。WEB-INF目录中的classes目录和lib目录中的JAR文件被自动添加到容器的类路径中，因此如果把类放到这两个地方就不需要设置类路径。

3. <init-param>元素

- 该元素定义向Servlet传递的初始化参数。
- 在一个<servlet>元素中可以定义任意多个<init-param>元素。每个<init-param>元素必须有且仅有一组<param-name>和<param-value>子元素。
- Servlet可以通过ServletConfig接口的getInitParameter( )方法检索初始化参数。

4. <load-on-startup>元素

- <load-on-startup>元素指定是否在Web应用程序启动时载入该Servlet。
- 该元素的值是一个整数。如果没有指定该元素或其内容为一个负数，容器将根据需要决定何时装入Servlet。如果其内容为一个正数，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，这可以控制容器装入这些Servlet的顺序，值小的先装入。

2.6.3 <servlet-mapping >元素

- <servlet-mapping>元素定义一个映射，它指定哪个URL模式被该Servlet处理。
- 容器使用这些映射根据实际的URL访问合适的Servlet。
- <servlet-mapping>元素的DTD定义：

<!ELEMENT servlet-mapping (servlet-name, url-pattern)>

2.6.4 <welcome-file-list>元素

- 通常在浏览器的地址栏中输入一个路径名称，而没有指定特定的文件，也能访问到一个页面，这个页面就是欢迎页面，文件名通常为index.html。
- 在Tomcat中，如果访问的URL是目录，并且没有特定的Servlet与这个URL模式匹配，那么它将在该目录中首先查找index.html文件，如果找不到将查找index.jsp文件，如果找到上述文件，将该文件返回给客户。如果找不到（包括目录也找不到），将向客户发送404错误信息。

2.7 @WebServlet和@WebInitParam注解

- 在Servlet 3.0中可以使用@WebServlet注解而不需要在web.xml文件中定义Servlet。该注解属于javax.servlet.annotation包，因此在定义Servlet时应使用下列语句导入。

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

@WebServlet注解的常用元素

元素名	类 型	说 明
name	String	指定Servlet名称，等价于web.xml中的<servlet-name>元素。如果没有显式指定，则使用Servlet的完全限定名作为名称
urlPatterns	String[]	指定一组Servlet的URL映射模式，该元素等价于web.xml文件中的<url-pattern>元素
value	String[]	该元素等价于urlPatterns元素。两个元素不能同时使用
loadOnStartup	int	指定该Servlet的加载顺序，等价于web.xml文件中的<load-on-startup>元素
initParams	WebInitParam[]	指定Servlet的一组初始化参数，等价于<init-param>元素
asyncSupported	boolean	声明Servlet是否支持异步操作模式，等价于web.xml文件中的<async-supported>元素
description	String	指定该Servlet的描述信息，等价于<description>元素
dispalyName	String	指定该Servlet的显示名称，等价于<display-name>元素

@WebInitParam注解的常用元素

元素名	类 型	说 明
name	String	指定初始化参数名，等价于<param-name>元素
value	String	指定初始化参数值，等价于<param-value>元素
description	String	关于初始化参数的描述，等价于<description>元素

2.8 ServletConfig接口

- 在Servlet初始化时，容器将调用init(ServletConfig)方法，并为其传递一个ServletConfig对象，该对象称为Servlet配置对象，使用该对象可以获得Servlet初始化参数、Servlet名称、ServletContext对象等。
- 要得到ServletConfig接口对象有两种方法：覆盖Servlet的init(ServletConfig config)方法，然后把容器创建的ServletConfig对象保存到一个成员变量中

2.8 ServletConfig接口

- 另一种方法是在Servlet中直接使用getServletConfig()方法获得ServletConfig对象
- 程序2.16 ConfigDemoServlet.java

2.9 小 结

- Servlet API由4个包组成，每个包中都定义了若干接口和类，它们是开发Servlet所需的全部内容。
- 本章介绍了Servlet的执行过程和生命周期，重点介绍了请求和响应模型，其中包括如何获取请求参数、如何检索请求头以及如何发送响应。