服务器端编程技术

- 在服务器端要实现Web应用的动态功能，需要服务器端编程技术。
- 目前，在服务器端技术有多种，包括CGI技术、Servlet技术以及动态页面技术。

静态资源和动态资源

- 如果资源本身没有任何处理功能它就是静态的，如果资源有自己的处理能力，它就是动态的。
- Web应用程序通常是静态资源和动态资源的混合。

  http://www.myserver.com/myfile.html

  http://www.myserver.com/product-report

静态文档和动态文档

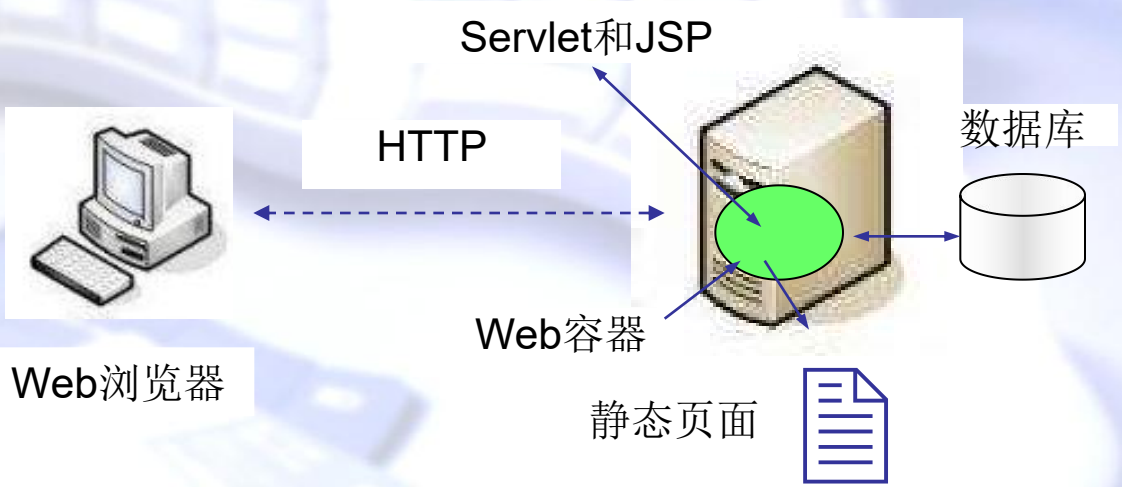
- Web文档是一种重要的Web资源，它通常是使用某种语言（如HTML，JSP等）编写的页面文件，因此也称为Web页面。Web文档又分为静态文档和动态文档。
- 静态文档创建完后存放在Web服务器中，在被用户浏览的过程中，其内容不会改变。
- 动态文档（dynamic document）是指文档的内容可根据需要动态生成。

CGI技术

- 公共网关接口（Common Gateway Interface，CGI）技术是在服务器端实现动态功能的传统方法。
- CGI是一种标准化的接口，允许Web服务器与后台程序和脚本通信，这些后台程序和脚本能够接受输入信息（例如，来自表单），访问数据库，最后动态生成HTML响应。
- 通常用Perl脚本语言来编写CGI程序。
- 使用CGI方法的主要问题是效率低。

Java解决方案-Servlet技术

- 在Java平台上，服务器扩展是使用Servlet API编写的，服务器扩展模块叫做Servlet容器（container），或称Web容器。



动态Web页面技术

- ASP（Active Server Page）称为活动服务器页面，是Microsoft公司推出的一种开发动态Web文档的技术。
- PHP（PHP：Hypertext Preprocessor）称为超文本预处理器，它是一种HTML内嵌式的语言。
- JSP是JavaServer Pages的缩写，含义是Java服务器页面，它与PHP非常相似，只不过页面中的动态部分是用Java语言编写的。

Java Web 编程技术

第2章 Servlet核心技术

主要内容

1. Servlet API
2. Servlet生命周期
3. 处理请求
4. 表单数据处理
5. 发送响应
6. 部署描述文件
7. @WebServlet和@WebInitParam注解
8. ServletConfig
9. ServletContext

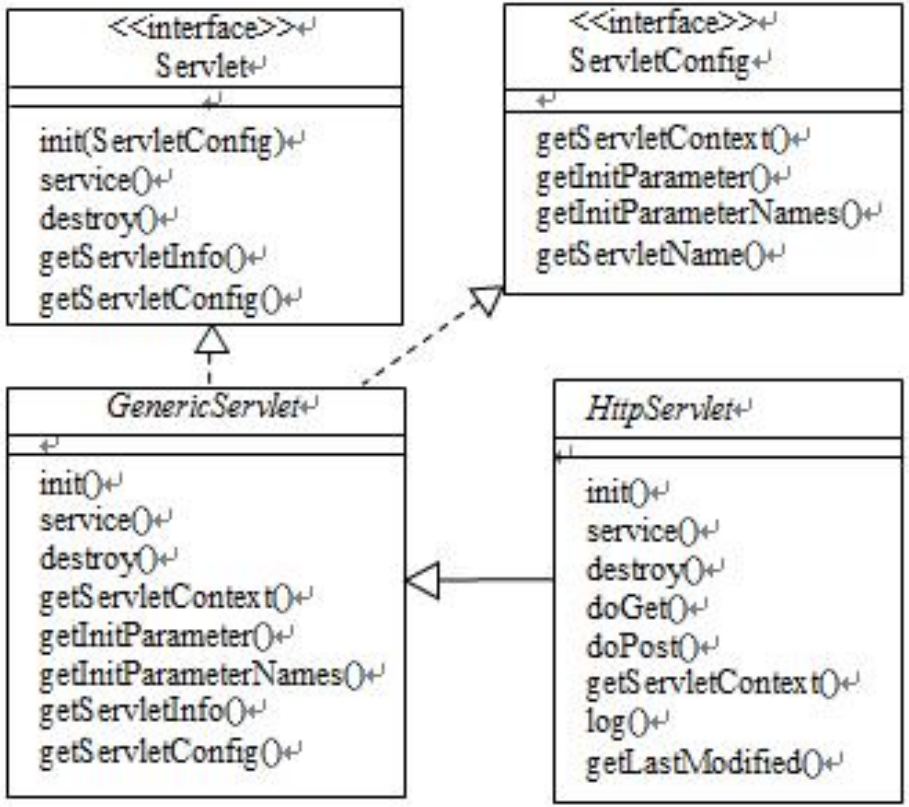
2.1 Servlet概述

Servlet API

- Servlet API定义了若干接口和类，它由下面4个包组成。
  - javax.servlet包。
  - javax.servlet.http包。
  - javax.servlet.annotation包。
  - javax.servlet.descriptor包。
- Apache Tomcat网站提供了Servlet API文档，地址为：

  http://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html

Servlet API



ServletConfig

ServletRequest

HttpServletRequest

ServletResponse

HttpServletResponse

Servlet接口

- Servlet接口是Servlet API中的基本接口，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。

```
void init(ServletConfig config)
void service(ServletRequest request,
                        ServletResponse response)
void destroy()
ServletConfig getServletConfig()
String getServletInfo()
```

GenericServlet抽象类

- GenericServlet实现了Servlet接口和ServletConfig接口。
- 它提供了Servlet接口中除了service()外的所有方法的实现，同时增加了支持日志的方法。
- 可以继承GenericServlet类并实现service()方法来创建任何类型的Servlet。

HttpServlet类

- HttpServlet抽象类继承了GenericServlet类，它用来实现针对HTTP协议的Servlet，在HttpServlet类中增加了一个新的service()方法，格式如下：

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

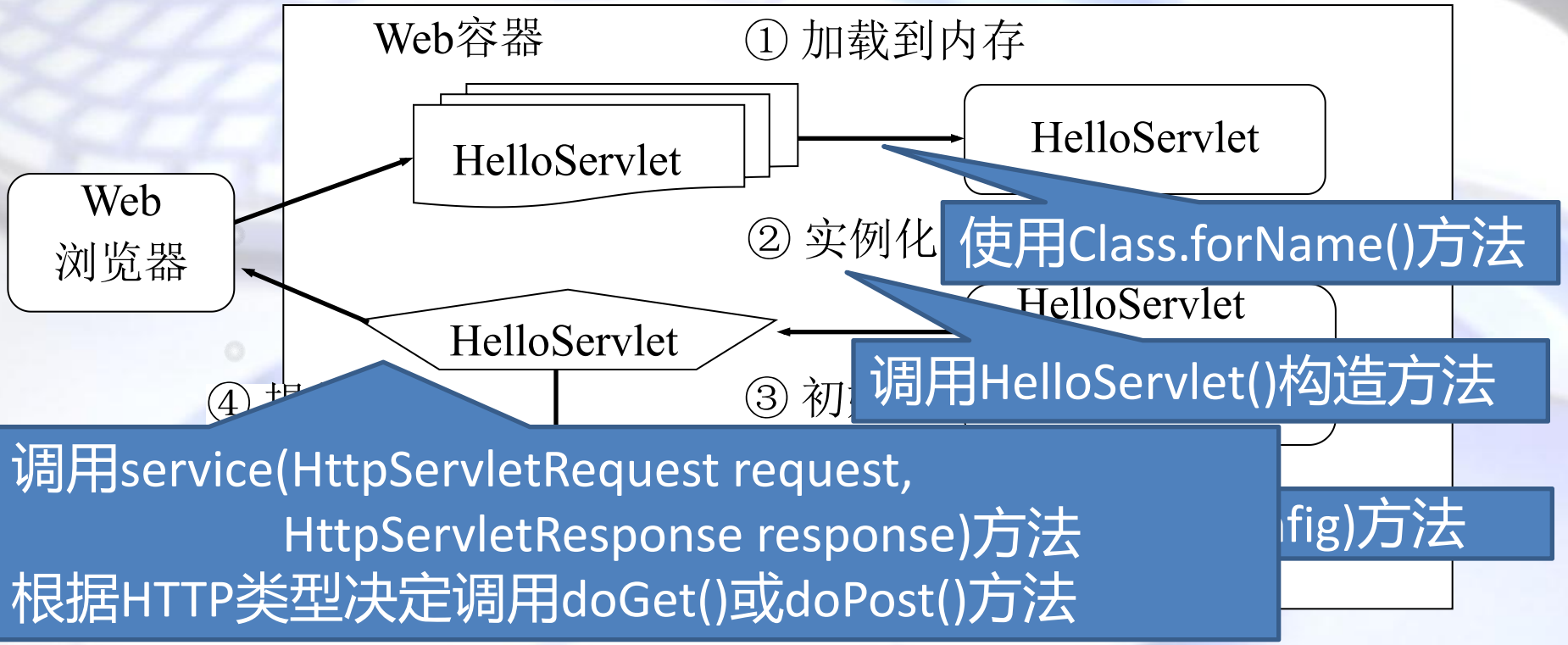
HttpServlet类

- 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()格式如下：

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

- 还定义了处理其他请求的方法，如处理POST请求的方法是**doPost()**。

Servlet生命周期



Java Web 编程技术

2.3 处理请求

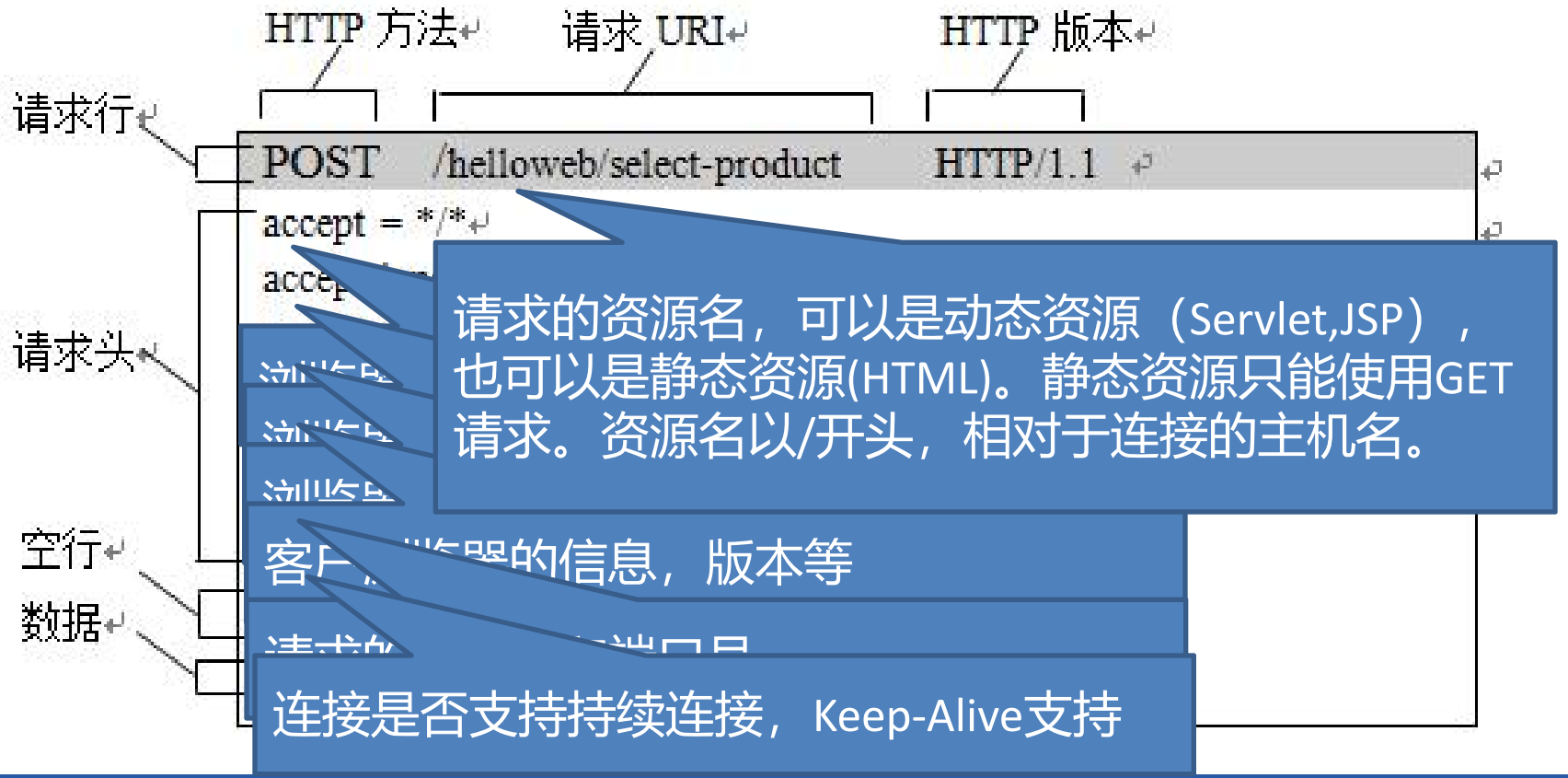
主要内容

1. HTTP请求结构
2. 发送HTTP请求
3. 处理请求
4. 检索请求参数
5. 请求转发
6. 使用请求存储数据
7. 检索客户端信息
8. 检索请求头

HTTP请求结构

特征	GET方法	POST方法
资源类型	静态的或动态的	动态的
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

HTTP请求结构


HTTP 方法
请求 URI
HTTP 版本
请求行
POST /helloweb/select-product HTTP/1.1
accept = */*
请求头
空行
数据
请求的资源名，可以是动态资源（Servlet,JSP），也可以是静态资源(HTML)。静态资源只能使用GET请求。资源名以/开头，相对于连接的主机名。
客户
的信息，版本等
连接是否支持持续连接，Keep-Alive支持

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。

① 用户在浏览器的地址栏中输入URL并按回车键。

② 用户点击了HTML页面中的超链接。

③ 用户在HTML页面中添写一个表单并提交。

- 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。

发送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”。

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

处理HTTP请求

- 在HttpServlet中对每种HTTP方法定义了相应的方法，处理GET请求使用doGet()方法：

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

处理HTTP请求

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()		

请求参数传递和获取

- 请求参数是随请求一起发送到服务器的数据，它以“名/值”对的形式发送。POST请求，参数在数据区；GET请求，参数附加在URI后面。
- 从客户端向服务器端传递请求参数有下面两种方法。

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URL中的查询串指定请求参数，将参数名和值附加在请求URI后面，这种方法只适用于GET请求。

检索请求参数

- 使用HttpServletRequest中定义的方法检索这些参数。

  String getParameter(String name)

  String[] getParameterValues(String name)

- 返回name指定的请求参数值，如果参数不存在，则返回null值。若指定的参数存在，用户没有提供值，则返回空字符串。

  Enumeration getParameterNames()

  Map getParameterMap()：

检索请求参数

- 程序2.3 login.html
- 程序2.4 LoginServlet.java

检索请求参数

- 向服务器发送GET请求，还可以将请求参数附加在请求URL的后面。例如，可以直接使用下面的URL访问LoginServet，而不需要通过表单提供参数。

  http://localhost:8080/chapter02/user-login**?username=admin&password=admin**

- 这里，问号后面内容为请求参数名和参数值对，若有多个参数，中间用“&”符号分隔，参数名和参数值之间用等号（=）分隔。问号后面内容称为**查询串**（query string）。

请求转发

- 可能需要将请求转发（forward）到其他资源。通过请求对象的getRequestDispatcher()得到RequestDispatcher对象，该对象称为请求转发器对象，格式如下。

```
RequestDispatcher getRequestDispatcher(String path)
```

- 调用RequestDispatcher对象的forward()方法。

```
void forward(ServletRequest request,
             ServletResponse response)
```

使用请求对象存储数据

- 可使用请求对象存储数据。请求对象是一个作用域对象，可以在其上存储属性实现数据共享。
- 属性（attribute）包括属性名和属性值。属性名是一个字符串，属性值是一个对象。
- 通常，在一个组件（Servlet）中将数据存储到请求作用域对象上，在转发到的另一个组件（Servlet或JSP）中取出这些数据使用。

使用请求对象存储数据

- 有关属性存储的方法有4个，定义在HttpServletRequest接口中。格式如下。

public void setAttribute(**String name,Object obj**)

public **Object** getAttribute(**String name**)　:要类型转换

public void removeAttribute(String name)

public Enumeration getAttributeNames()

检索客户端信息

- 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()
public String getProtocol()
public String getRequestURI()
public String getQueryString()
```

检索请求头

- HTTP请求头是随请求一起发送到服务器的信息，它是以“名/值”对的形式发送。例如，关于浏览器的信息就是通过User-Agent请求头发送的。

String getHeader(String name)

Enumeration getHeaders(String name)

int getIntHeader(String name)

long getDateHeader(String name)

2.4 表单数据处理

主要内容

1. 常用表单控件
2. 表单数据处理
3. 解决乱码问题

常用表单控件

- 表单使用<form>元素创建，一般格式如下：

```
<form action=“user-register" method=“post">
...
</form>
```

常用表单控件

- 常用文本输入控件有四种：文本框、密码框、文本区和隐藏域。
- 每种类型的控件都有一个给定的名字，名字对应的值取自控件的内容。在表单提交时，名字和值一同发送到服务器。

```
<input type="text" name="…" size="" >
<input type="password" name="…" size="" >
<input type=“hidden" name="…" value="…" >
```

常用表单控件

```
<textarea name="…"  rows="…" cols="…"  >
    …
</textarea>
```

- 上述四种文本控件的输入值在服务器端的Servlet中使用request.getParameter(name)方法取得，如果值为空将返回空字符串。

常用表单控件

- 按钮控件包括提交和重置按钮以及普通的按钮控件。
- 提交和重置按钮控件的一般格式为：

```
<input type="submit" name="…" value="…">
<input type="reset" name="…" value="…">
<input type="button" name="…" value="…">
```

常用表单控件

- **单选按钮**在给定的一组值中只能选择一个，格式如下：

```
<input type="radio" name ="sex" value="male">男
<input type="radio" name ="sex" value="female">女
```

- 只有当name属性值相同的情况下，它们才属于一个组，在一组中只能有一个按钮被选中。
- 获取单选按钮被选中的值用request.getParameter(name)方法。

常用表单控件

- **复选框**通常用于多选的情况，它的一般格式如下：

```
<input type="checkbox" name="hobby" value="read">文学
<input type="checkbox" name="hobby" value="sport">体育
<input type="checkbox" name="hobby" value="computer">电脑
```

- 获取复选框被选中的值用request.getParameterValues(name)，它返回字符串数组，对数组的迭代可知用户选中了哪些选项。

常用表单控件

- 组合框和列表框可为用户提供一系列选项，它通过下拉列表框为用户列出各个选项供用户选择。

```
<select name="education">
    <option value="bachelor">学士</option>
    <option value="master">硕士</option>
    <option value="doctor">博士</option>
</select>
```

常用表单控件

- 如果省略可选属性multiple，则控件为组合框且只允许选择一项，在Servlet中使用request.getParameter(name)返回选中值。
- 若指定multiple属性，则控件为列表框且允许选择多项，在Servlet中用request.getParameterValues(name)返回选中值的数组。

常用表单控件

- **文件上传控件**用于向服务器上传文件，一般格式为：

  <input type="file" name="…"size="…">

- 生成一个文本框和一个“浏览”按钮。用户可以直接在文本框输入文件名，或单击按钮打开文件对话框选择文件。

创建表单页面

- 创建一个名为register.html页面，其中包含多种表单控件，访问该页面运行结果如图2.9所示。
- 在服务器端的Servlet中通常使用请求对象的getParameter()方法和getParameterValues()方法获取表单数据。
- 中文乱码解决办法：

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

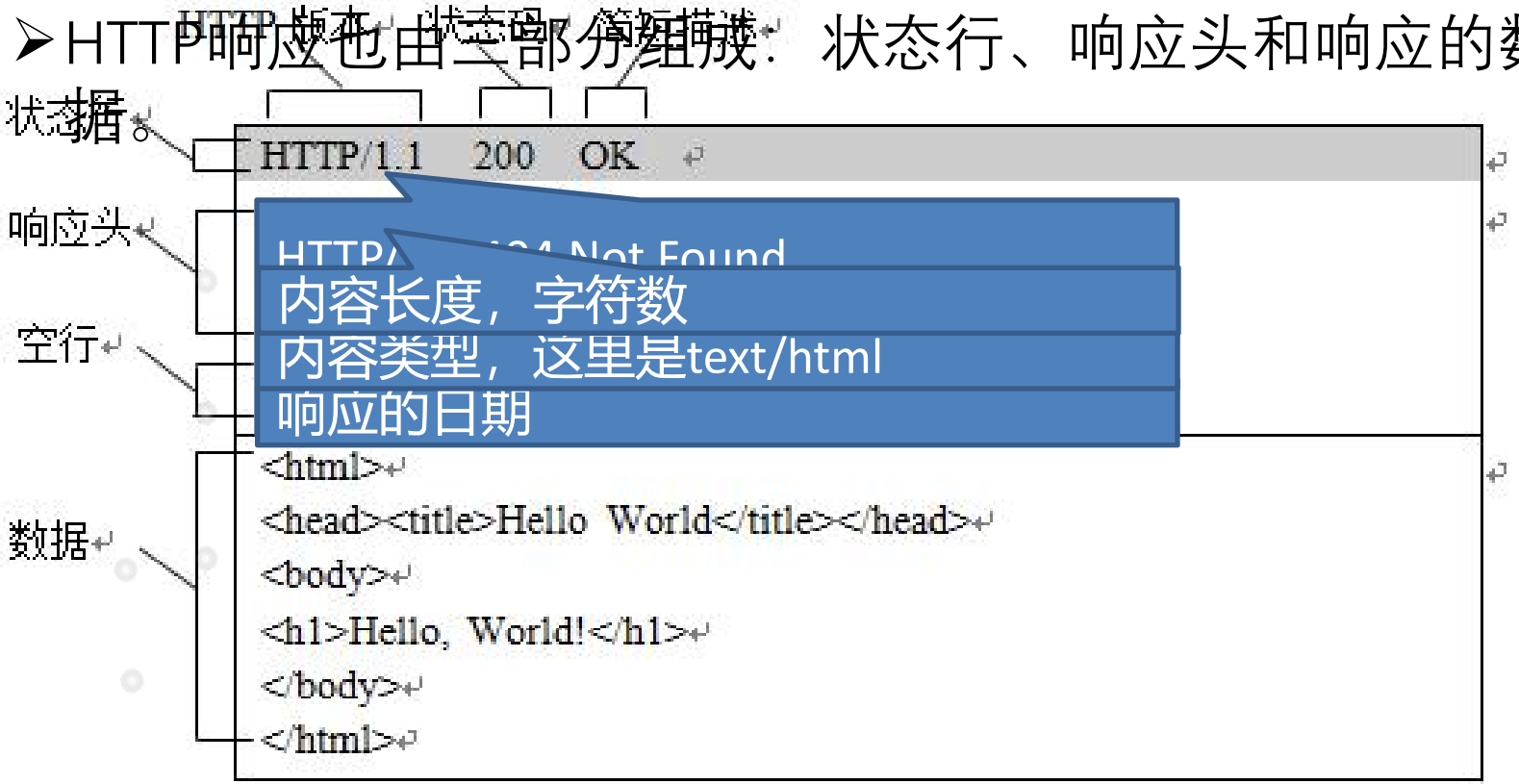
2.5 发送响应

主要内容

1. 响应结构
2. 输出流和内容类型
3. 响应重定向
4. 设置响应头
3. 发送状态码

HTTP响应结构

- HTTP响应也由三部分组成：状态行、响应头和响应的数据



输出流与内容类型

- Servlet使用输出流向客户发送响应。返回一个PrintWriter对象用于向客户发送文本数据。

  PrintWriter getWriter()

- 返回一个输出流对象，它用来向客户发送二进制数据。

  ServletOutputStream getOutputStream()

- 在发送响应数据之前还需通过响应对象的setContentType()设置响应数据的MIME内容类型。

  void setContentType(String type)

输出流与内容类型

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/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

响应重定向

- Servlet在对请求进行分析后，可能不直接向浏览器发送响应，而是向浏览器发送一个Location响应头，告诉浏览器访问其他资源，这称为响应重定向。
- 响应重定向是通过响应对象的sendRedirect()实现，格式如下：

  public void sendRedirect(String location)

- URL可以是绝对URL（如https://www.baidu.com），也可以是相对URL。

响应重定向与请求转发

图 2.13 请求转发示意图

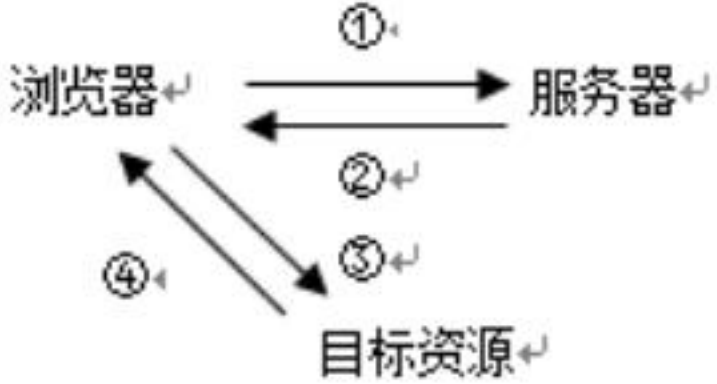


图 2.14 响应重定向示意图

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。每个响应头通过“名/值”对的形式发送到客户端。

```
void setHeader(String name, String value)

void setIntHeader(String name, int value)

void setDateHeader(String name, long date)
```

- 示例代码：

```
response.setHeader(“Refresh”,“5”);      // 每5秒刷新页面
```

设置响应头

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

设置响应头

- 除了让浏览器重新载入当前页面外，使用该方法还可以载入指定的页面。
- 例如，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/");
```

- 在HTML页面中用<meta>标签也可以实现同样功能。

发送状态码

- 下是一个典型的状态行：

  HTTP/1.1 200 OK

- 状态码200是系统自动设置的，Servlet一般不需要指定该状态码。对于其他状态码，可以由系统自动设置，也可以用响应对象的setStatus()设置，格式为：

  public void setStaus (int sc)

- 状态码通常使用HttpServletResponse接口定义的常量。如404状态，使用HttpServletResponse.SC_NOT_FOUND。

发送状态码

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

2.6 部署描述文件web.xml

部署描述文件

- 在Servlet 3.0之前，每个Web应用程序都必须有一个部署描述文件（web.xml），部署描述文件（Deployment Descriptor，简称DD文件）。它用来部署Web应用中所包含的组件，如Servlet等。
- 从Servlet 3.0开始，有些组件可使用注解配置，但还有些内容（安全配置等）需要使用部署描述文件配置。
- Web应用启动时，容器读取该文件，对应用程序配置，所以有时也将该文件称为配置文件。

部署描述文件

```
<?xml version="1.0" encoding="UTF-8"?>

<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_4_0.xsd"
version="4.0"   metadata-complete="true">
    ...
</web-app>
```

@WebServlet和@WebInitParam注解

@WebServlet和@WebInitParam注解

- 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;
```

- 下面一行是为HelloServlet添加的注解。

```
@WebServlet(name="helloServlet",
                    urlPatterns={"/hello-servlet"})
```

- 注解在应用程序启动时被Web容器处理，容器根据具体的属性配置将相应的类部署为Servlet。

@WebServlet和@WebInitParam注解

- @WebServlet注解包含多个属性，它们与web.xml中的对应元素等价。
  - name属性指定Servlet名称，等价于web.xml中的<servlet-name>元素。
  - urlPatterns属性指定一组URL映射模式，该元素等价于web.xml文件中的<url-pattern>元素
  - loadOnStartup属性指定该Servlet的加载顺序，等价于web.xml文件中的<load-on-startup>元素

@WebServlet和@WebInitParam注解

- @WebServlet注解包含多个属性，它们与web.xml中的对应元素等价。
  - initParams属性，它的类型是WebInitParam[]，指定Servlet的一组初始化参数，等价于<init-param>元素。

@WebServlet和@WebInitParam注解

- @WebInitParam注解主要作用是为Servlet或Filter指定初始化参数，它等价于web.xml文件中<servlet>和<filter>元素的<init-param>子元素。
- 通常配合@WebServlet和@WebFilter使用。
  - name属性，指定初始化参数名，等价于<param-name>元素。
  - value属性，指定初始化参数值，等价于<param-value>元素。

@WebServlet和@WebInitParam注解

- 下面是一个WebServlet注解且带两个初始化参数。

```
@WebServlet(name="ConfigDemoServlet",
            urlPatterns = {"/config-demo"},
    initParams = {
            @WebInitParam(name = "email",
                    value = “webmaster@163.com"),
            @WebInitParam(name = "telephone",
                    value = "8899123")
    })
```

2.8 ServletConfig

ServletConfig

- ServletConfig称为Servlet配置对象。
- 在Servlet初始化时，容器调用init(ServletConfig)并为其传递一个ServletConfig对象。
- 使用该对象可以获得Servlet初始化参数、Servlet名称、ServletContext对象等。

ServletConfig

- 要得到ServletConfig接口对象有两种方法：
- 覆盖Servlet的init(ServletConfig config)，然后把容器创建的ServletConfig对象保存到一个成员变量中。

```
ServletConfig config = null;
public void init(ServletConfig config){
        super.init(config);  // 调用超类的init()
        this.config = config;
}
```

ServletConfig

- 另一种方法是在Servlet中直接使用getServletConfig()获得ServletConfig对象，如下所示。

```
ServletConfig  config = getServletConfig();
```

ServletConfig

- ServletConfig接口定义了下面4个方法：

String getInitParameter(String name)

Enumeration getInitParameterNames()

ServletContext getServletContext()

String getServletName()

ServletConfig

- 在web.xml文件通过<servlet>的子元素<init-param>为Servlet指定初始化参数。

```
<servlet>
   <servlet-name>configDemoServlet</servlet-name>
   <servlet-class>demo.ConfigDemoServlet</servlet-class>
   <init-param>
     <param-name>telephone</param-name>
     <param-value>8899123</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
```

2.9 ServletContext

得到ServletContext引用

- Web容器在启动时会加载每个Web应用程序，并为每个Web应用程序创建一个唯一的ServletContext实例对象，该对象称为**Servlet上下文**对象。
- 使用ServletContext对象获得Web应用程序的初始化参数、它是重要的作用域对象，可实现数据共享、获得Web容器的版本等信息。

得到ServletContext引用

- 有两种方法得到ServletContext引用。
- 直接调用getServletContext()。

```
ServletContext context = getServletContext();
```

- 先得到ServletConfig，再调用它的getServletContext()。

```
ServletContext context =
        getServletConfig().getServletContext();
```

获取应用程序初始化参数

- 正像Servlet具有初始化参数一样，Web应用也有初始化参数，它们是应用程序范围内的信息。

  String getInitParameter(String name)

  Enumeration getInitParameterNames()

获取应用程序初始化参数

- 应用程序初始化参数应该在web.xml文件中使用<context-param>元素定义。

```
<context-param>
    <param-name>adminEmail</param-name>
    <param-value>webmaster@163.com</param-value>
</context-param>
```

- 注意，<context-param>元素是<web-app>元素的直接子元素，它针对整个应用，并不嵌套在<servlet>元素中。

获取应用程序初始化参数

- 在Servlet中使用下面代码检索adminEmail参数值。

```
ServletContext context = getServletContext();
String email =   context.getInitParameter("adminEmail");
```

使用ServletContext存储数据

- ServletContext是一个作用域对象，使用它可以存储数据，它的作用域是整个应用程序。

  void setAttribute(String **name**, Object **object**)

  Object getAttribute(String **name**)

  Enumeration getAttributeNames()

  void removeAttribute(String **name**):

实现请求转发

- 使用ServletContext接口的下列两个方法也可以获得RequestDispatcher对象，实现请求转发。

  RequestDispatcher getRequestDispatcher(String path)

  RequestDispatcher getNamedDispatcher(String name)

- 参数path表示资源路径，它必须以“/”开头，表示相对于Web应用的文档根目录。如果不能返回转发器对象，将返回null。 name是Servlet名称。

获得资源

- 通过ServletContext对象方法获得服务器上资源

URL getResource(String path)

InputStream getResourceAsStream(String path)

String getRealPath(String path)

登录日志

- ServletContext接口定义了log()方法，可以将指定的消息写到服务器的日志文件中。

  void log(String msg)

  void log(String msg, Throwable throwable)

- 日志将被写入<tomcat-install>\logs\localhost.YYYY-MM-DD.log文件中。