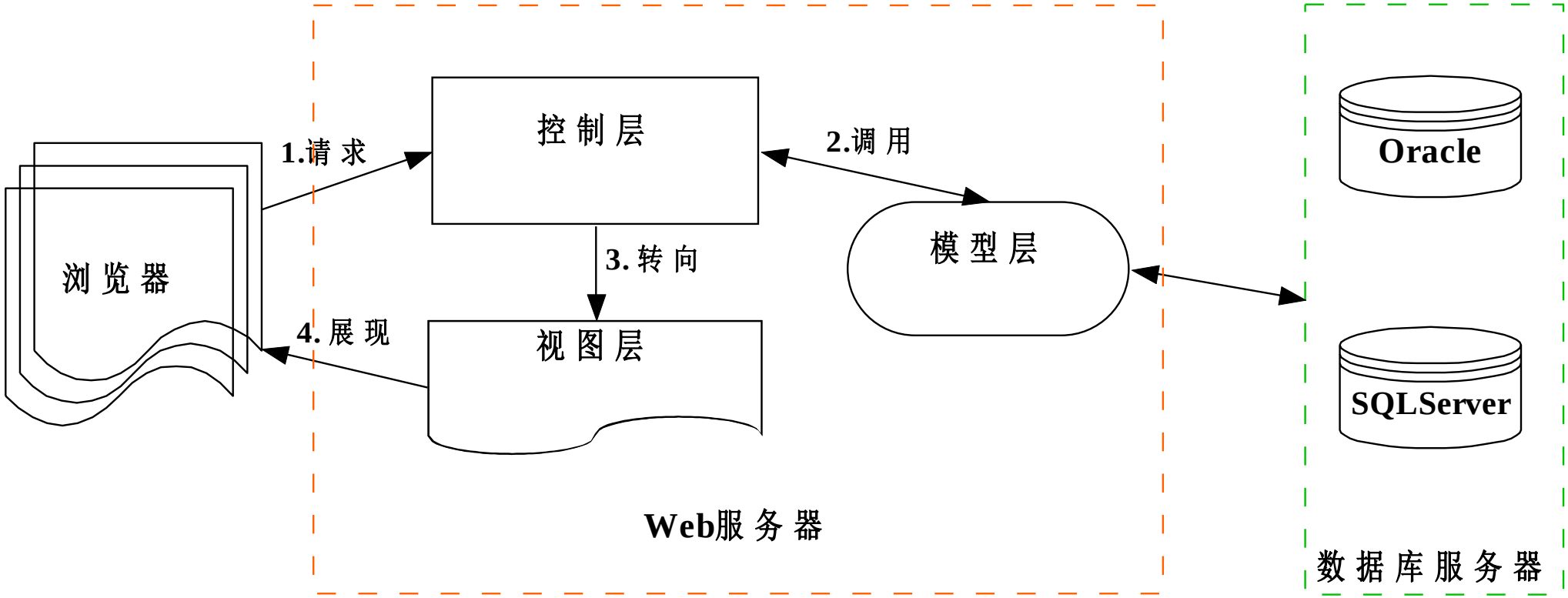
Servlet基础

王新阳
wxyyuppie@bjfu.edu.cn

主要内容

- Java Web设计模式
- 创建项目
- Servlet简介和示例
- Servlet工作原理

一、Java Web 设计模式——经典MVC架构



- 从系统外看，B/S架构
  - 浏览器——Chrome，IE，搜狗，QQ，Opera，Firefox等
  - Web服务器——Tomcat，JBOSS，WebLogic，classfish等
  - 数据库服务器——MySQL，SQLSserver，Oracle，HBase等

一、Java Web 设计模式——三层MVC架构

- 从服务器内部看：
- 控制层（Controller）
  - 根据业务逻辑流程，描述文件之间的跳转
- 视图层（Viewer）
  - 在浏览器端表现请求表单或展现处理结果
- 模型层（Model）
  - 作用
    - ✓处理具体的业务逻辑——业务逻辑组件
    - ✓存取数据库——数据访问组件

一、Java Web 设计模式——三层MVC架构

- Servlet
- JSP
- JavaBeans
- JDBC

} 初级的开发框架

- 控制层——servlet
- 视图层——HTML和JSP
- 模型层——JavaBean
- 数据库组件——JDBC

二、新建一个JavaWeb项目——MyEclipse

- 打开Myeclipse，新建一个Web项目
  - File→new→Web project→next→next→勾选 Generate web.xml部署描述文件→Finish,在Project Explorer中可以看到该新建的项目
- 项目结构
  - src存放java源代码
  - WebRoot存放要部署到服务器上的文件
    - ✓WEB-INF中存放web.xml文件和项目所需类库
    - ✓META-INF中存放项目的元信息

二、新建一个JavaWeb项目——Eclipse

- 打开eclipse，新建一个动态Web项目
  - File→new→Dynamic Web project→next→next→ 勾选Generate web.xml部署描述文件→Finish,在Project Explorer中可以看到该新建的项目
- 项目结构
  - Java Resource\src存放java源代码
  - WebContent存放要部署到服务器上的文件
    - ✓WEB-INF中存放web.xml文件和项目所需类库
    - ✓META-INF中存放项目的元信息

二、新建一个JavaWeb项目——web.xml

- **Web.xml是服务器端的基本配置文件**
  - **Web服务器启动项目时首先读取该文件，并根据内容自动配置该项目**
- **Web.xml的基本内容**
  - **指定Web项目发布时的首页**
  - **指定先期加载的其它类库**
  - **将java文件映射为URL访问地址**
  - **… …**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
  <display-name>ServletLogin</display-name>

  <servlet>
    <description></description>
    <servlet-name>login</servlet-name>
    <servlet-class>servlet.LoginServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>login</servlet-name>
    <url-pattern>/UserLogin</url-pattern>
  </servlet-mapping>

</web-app>
```

二、新建一个JavaWeb项目——项目部署

如何用IDE部署并运行JavaWeb项目

参见文档《**Java EE运行环境配置过程**》

三、Servlet简介和示例

- **Servlet –Java服务器小程序**
  - 采用java语言编写的java类
  - 在Web服务器端调用和执行
  - 符合Servlet编写规范
- **主要作用：**
  处理客服端的Http请求并予以响应
  - 读取客户程序发送来的显式数据(表单数据)
  - 读取客户程序发送来的隐式数据(请求报头)
  - 生成相应的结果
  - 发送显式的数据给客户程序（HTML）
  - 发送隐式的数据给客户程序(状态代码和响应报头)

三、Servlet简介和示例——新建一个Servlet

- 选择项目Java Resource\src
  - 单击右键→new→Servlet→在class name中填写Servlet的名字
  - 也可以通过创建新类的方式，但必须要继承HttpServlet类，并实现doGet方法。
  - 注意java类的命名规则
- 创建Servlet的过程
  - 本质上是接受客户端传来的数据或者向客户端发送数据的过程,需要重写doGet方法：
    - ✓ public void doGet(HttpServletRequest request,HttpServletResponse response)

三、Servlet简介和示例——新建一个Servlet

例1—在浏览器端访问一个Servlet(输出java代码中的纯文本)

- 第一步
  - 在服务器端建立并编译**Servlet——HelloWorld1.java**文件
- 第二步
  - 配置服务器配置文件**web.xml**
- 第三步
  - 将工程部署到**Tomcat**容器中，并启动服务器
- 第四步
  - 在浏览器端请求该**Servlet**
- 示例 **ServletTest.war**
  - **bjfu.SimpleServlet.HelloWorld1.java**

三、Servlet简介和示例——新建一个Servlet

例1——第一步:实现HelloWorld1.java

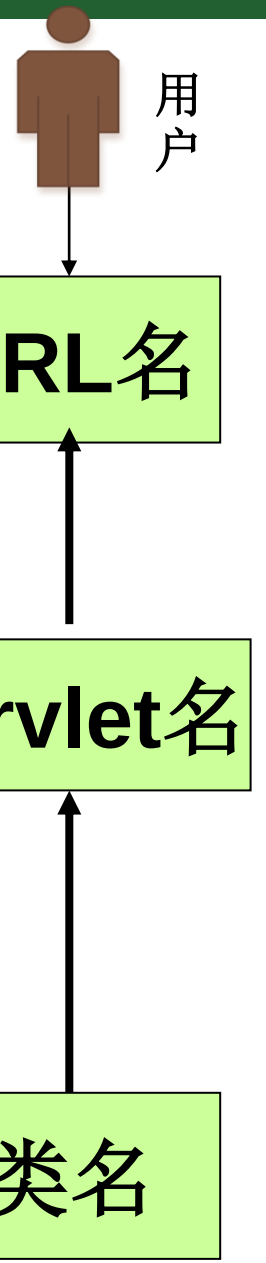
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
//需要继承 javax.servlet.http.HttpServlet
public class HelloWorld1 extends HttpServlet {

//重写doGet方法，处理浏览器的请求
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
    //指定输出的编码格式，输出中文
    response.setCharacterEncoding(“gbk"); PrintWriter
    out = response.getWriter();
    out.println("Hello World1，你好");
}}
```

三、Servlet简介和示例——新建一个Servlet

例1——第二步：配置web.xml

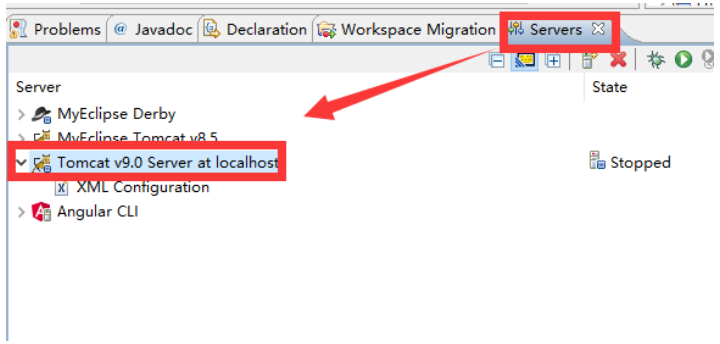
- Web.xml是服务器端的基本配置文件
  - 服务器启动时会自动读取该文件
- Web文件的配置
  - 设置服务器启动时的默认打开页面
    - ✓ <welcome-file-list>
    - ✓ <welcome-file>index.html</welcome-file>
    - ✓ <welcome-file>default.jsp</welcome-file>
    - ✓ </welcome-file-list>
  - 注册Servlet，命名为HelloWorld，建立注册名和java类的对应关系
    - ✓ <servlet>
    - ✓ <description></description>
    - ✓ <servlet-name>HelloWorld1</servlet-name>
    - ✓ <servlet-class>test.HelloWorld1</servlet-class>
    - ✓ //此处应为完整类路径
    - ✓ </servlet>
  - 建立URL名称和Servlet的对应关系
    - ✓ <servlet-mapping>
    - ✓ <servlet-name>HelloWorld1</servlet-name>
    - ✓ <url-pattern>/HelloWorld1</url-pattern>
    - ✓ </servlet-mapping>

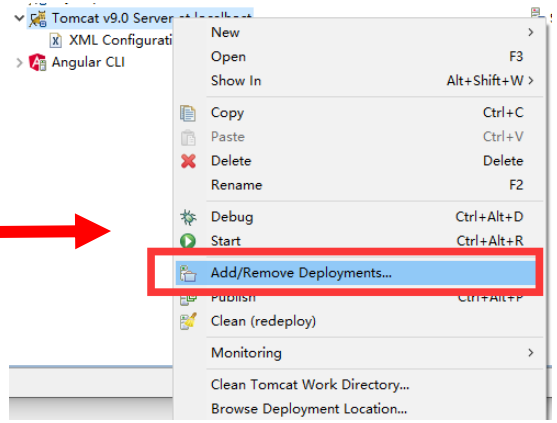


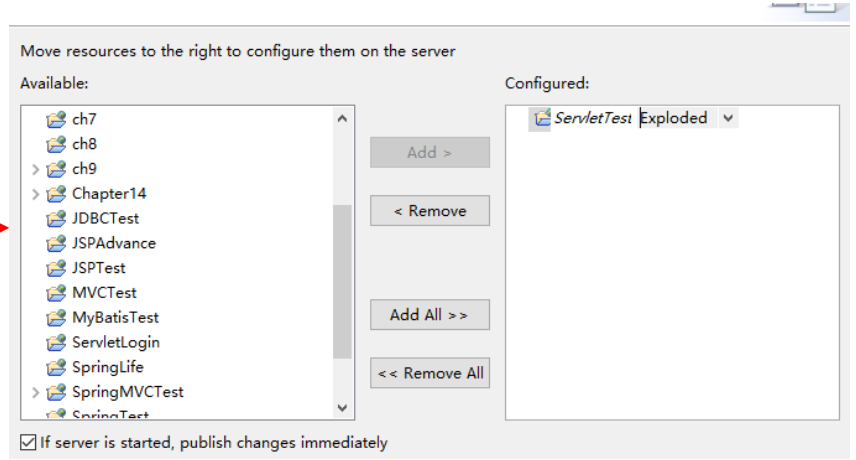
三、Servlet简介和示例——新建一个Servlet

例1——第三步：部署项目并启动

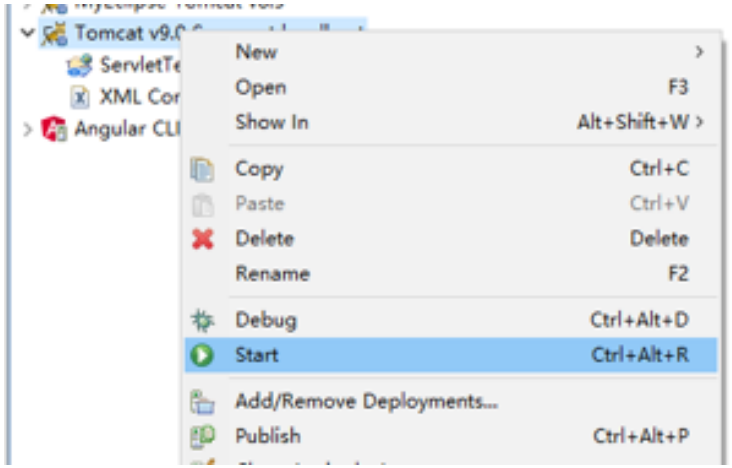
1、部署项目







2、启动服务器




```
信息: 开始协议处理句柄[ nttp-nio-8080 ]
九月 07, 2022 7:26:59 下午 org.apache.coyote.AbstractProtocol start
信息: 开始协议处理句柄["ajp-nio-8009"]
九月 07, 2022 7:26:59 下午 org.apache.catalina.startup.Catalina sta
信息: Server startup in [12,411] milliseconds
```

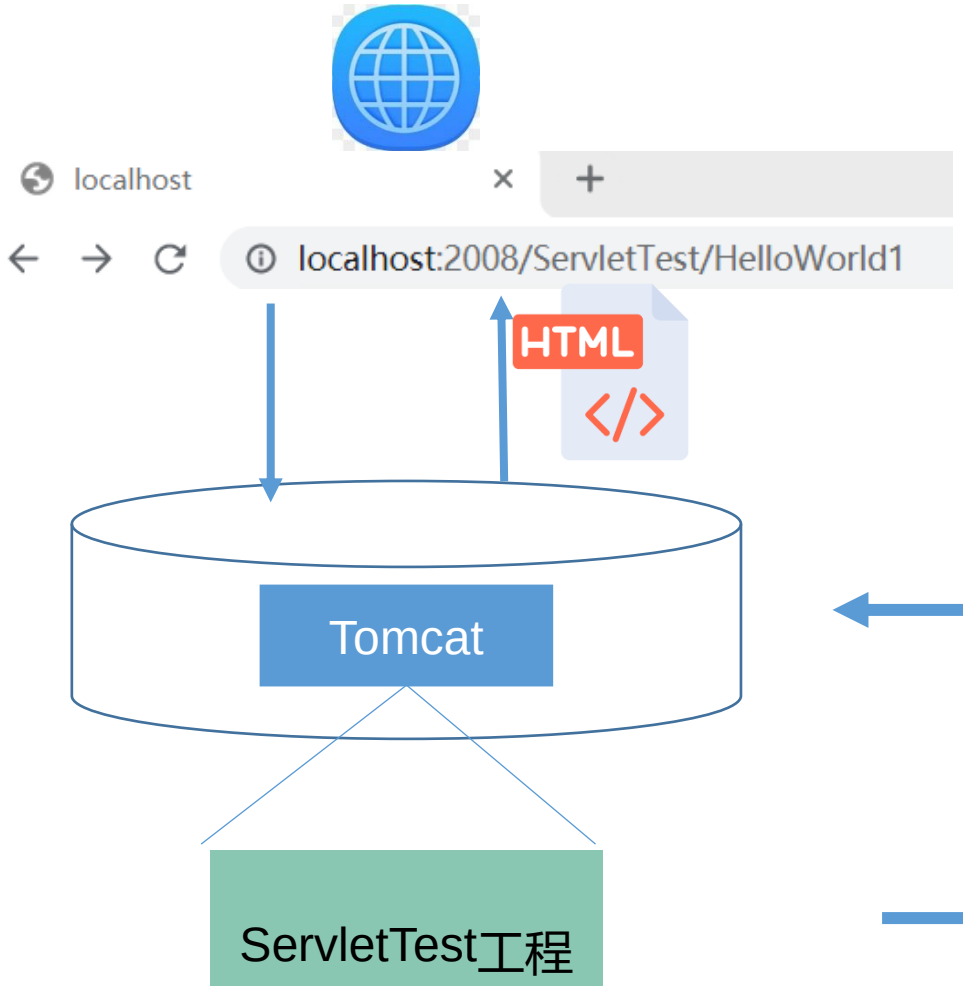
三、Servlet简介和示例——新建一个Servlet

例1——第四步:在浏览器请求该Servlet

- Java源文件ServletTest→ URL地址HelloWorld
- 在浏览器地址栏中输入
  - http://localhost:8080/ServletTest/HelloWorld1
- 注意:
  - HelloWorld的大小写需与<url-pattern>中指定的URL名称一致
  - 项目名称和URL地址区分大小写!

三、Servlet简介和示例——新建一个Servlet

例1——这个例子做了什么



HelloWorld.java


```
<servlet>
  <description></description>
  <servlet-name>HelloWorld1</servlet-name>
  <servlet-class>bjfu.SimpleServelet.HelloWorld1</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>HelloWorld1</servlet-name>
  <url-pattern>/HelloWorld1</url-pattern>
</servlet-mapping>
```

Web.xml

三、Servlet简介和示例——新建一个Servlet

例2—在浏览器端显示HTML

- 第一步，需要修改HelloWorld2.java
  - 增加输出代码格式的指定
    - ✓response.setContentType("text/html");
  - 在输出语句中增加HTML标签
- 第二步，增加对HelloWorld2这个Servlet的配置
- 第三步，在浏览器中查看

- 示例 ServletTest.war
  - HelloWorld2.java

三、Servlet简介和示例——新建一个Servlet

第一步——HelloWorld2.java

```
//重写doGet方法，处理浏览器的请求
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html");
    response.setCharacterEncoding("gb2312");
    //response.setContentType("text/html;charset=gb2312");//等价于上面两个语句的设置
    PrintWriter out = response.getWriter();
    out.println(
    "<HTML>\n" +
    "<HEAD><TITLE>HelloWorld2</TITLE></HEAD>\n"+
    "<BODY BGCOLOR=\"#FDF5E6\">\n" +
    "<H1>HelloWorld2,你好！！ </H1>\n" +
    "</BODY></HTML>");
    }
```

三、Servlet简介和示例——新建一个Servlet

第二步——web.xml

- Web文件的配置
  - 注册Servlet，命名为HelloWorld，建立注册名和java类的对应关系
    - ✓ <servlet>
    - ✓ <description></description>
    - ✓ <servlet-name>HelloWorld2</servlet-name>
    - ✓ <servlet-class>test.HelloWorld2</servlet-class>
    - ✓ </servlet>
  - 建立URL名称和Servlet的对应关系
    - ✓ <servlet-mapping>
    - ✓ <servlet-name>HelloWorld2</servlet-name>
    - ✓ <url-pattern>/HelloWorld2</url-pattern>
    - ✓ </servlet-mapping>

三、Servlet简介和示例——新建一个Servlet

第三步——在浏览器请求该Servlet

- **Java源文件ServletTest→ URL地址 HelloWorld2**

- 在浏览器地址栏中输入
  - **http://localhost:2008/ServletTest/HelloWorld2**

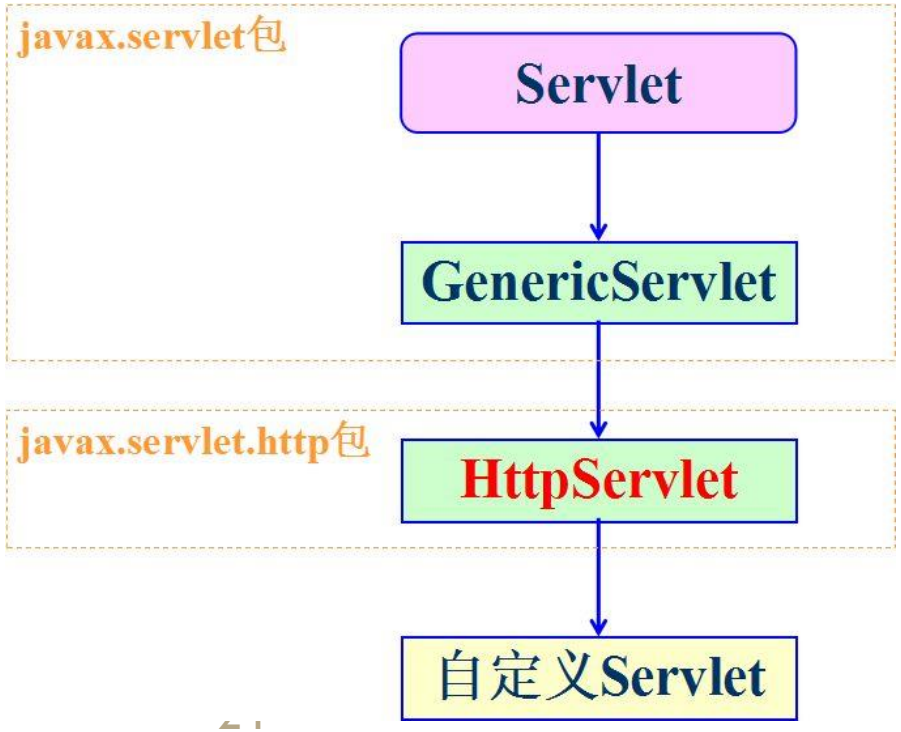
三、Servlet简介和示例——新建一个Servlet

- **web.xml修改的注意事项**
  - 区分大小写
  - 先写<servlet>… </servlet>，再写<servlet-mapping>…</servlet-mapping>
  - <servlet>表示把一个Servlet类注册为服务器能找到的Servlet
  - <servle> 后面紧跟<servlet-mapping>
  - <servlet-mapping>表示访问服务中Servlet时采用的url模式
  - 例如：
    - ✓<servlet>
    - ✓<servlet-name>bb</servlet-name>
    - ✓ <servlet-class>servlettest.HelloWorld3</servlet-class>
    - ✓ </servlet>
    - ✓ <servlet-mapping>
    - ✓ <servlet-name>bb</servlet-name>
    - ✓ <url-pattern>/aa</url-pattern>
    - ✓ </servlet-mapping>
- 第三步，在浏览器地址栏中输入以下地址就可以调用HelloWorld3
  - http://localhost：2008/ServletTest/aa

Servlet工作原理

Servlet工作原理——概述

- **Server + let**
  - 意为：运行在服务器端的小程序。
  - Servlet本质上是一个接口。
  - 广义上，凡是实现Servlet接口的类，都称为Servlet
- **Servlet的作用：**
  - 1.接收用户发送的请求
  - 2.调用其他的java程序来处理请求
  - 3.根据处理结果，返回给用户一个页面

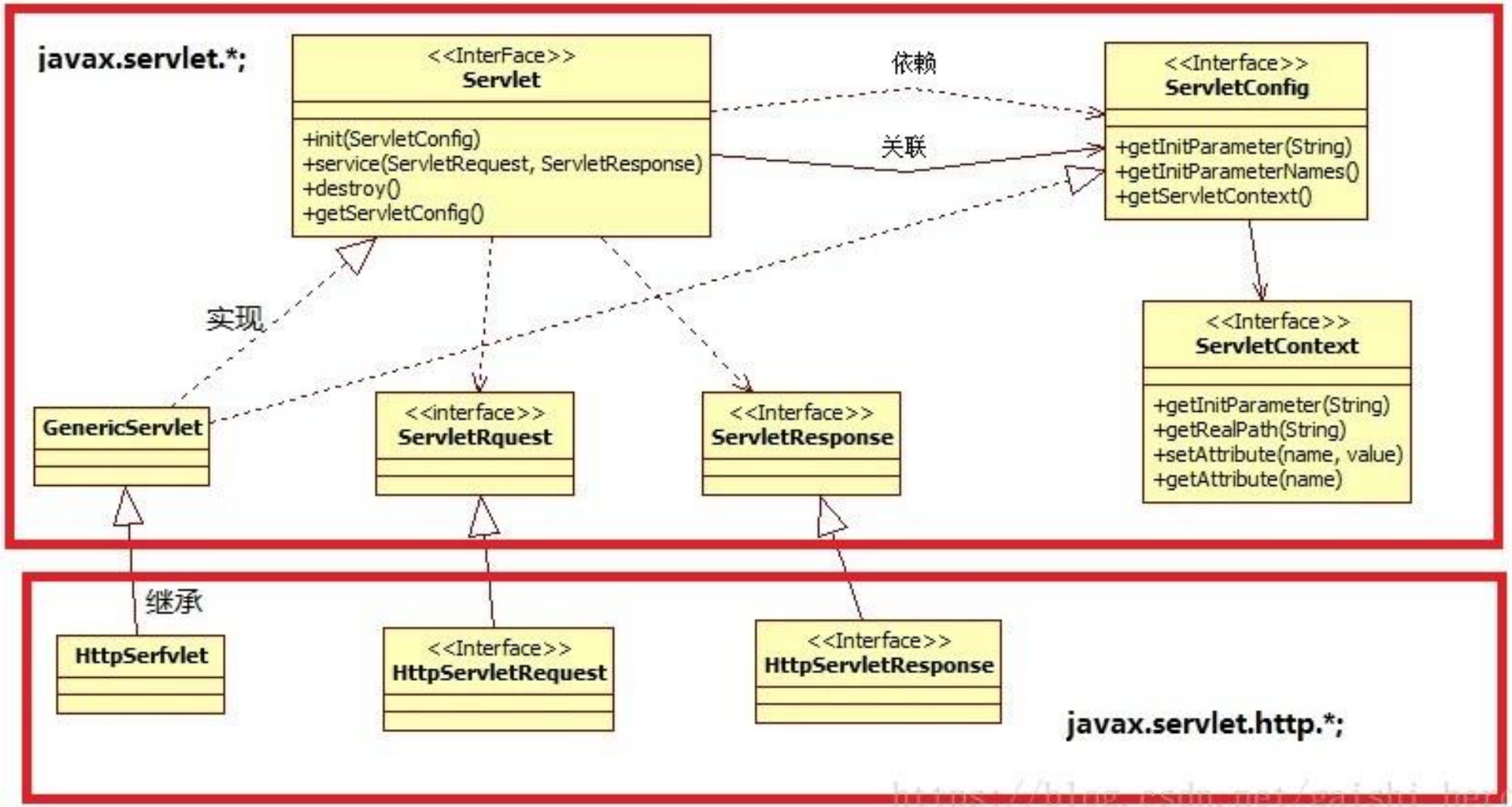


- 参考
  - API 文档： https://docs.oracle.com/javaee/7/api/toc.htm

Servlet工作原理——概述

- Servlet是运行于服务器端即Web 服务器(如Tomcat)中的Java类。Servlet对象由web服务器创建。
- Servlet拥有web上下文，即Servlet类中可以读取页面一些对象。如Response,Request等。
- 当客户端发送请求至服务器时，服务器可以将请求信息（request）发送给 Servlet，并让 Servlet 建立起服务器返回给客户机的响应（response）。
- 当启动 Web 服务器或客户机第一次请求服务时，可以自动装入Servlet。装入后， Servlet 继续运行直到其它客户机发出请求。

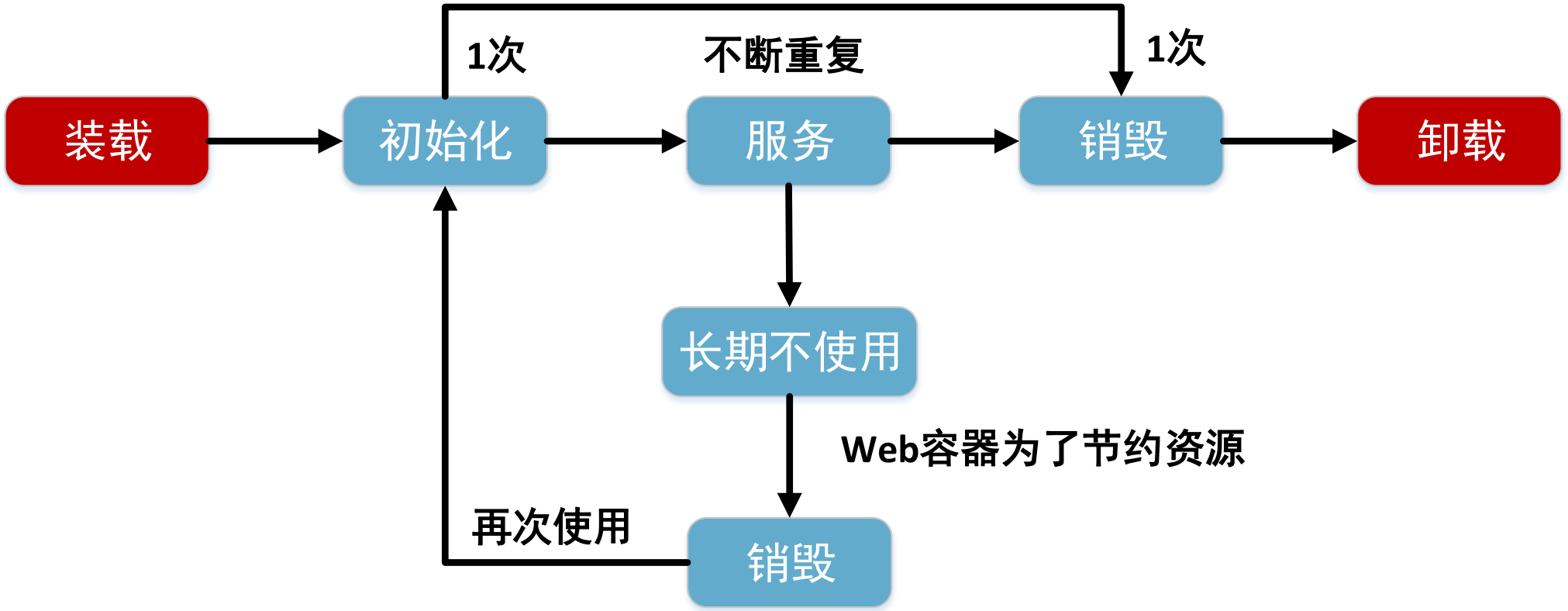
Servlet工作原理—— Servlet类分析



Servlet工作原理—— Servlet类分析

- 每个Servlet类都需要继承HttpServlet类和它的方法
  - ✓init(ServletConfig config):仅在servlet首次载入时执行一次
  - ✓service(ServletRequest req, ServletResponse res):接受浏览器端发送的请求并进行转发其它相应的函数，不能覆盖
  - ✓doGet(ServletRequest req, ServletResponse res)：处理Service转发过来的Get请求,需要重写
  - ✓doPost(ServletRequest req, ServletResponse res)：处理Service转发过来的Post请求，需要重写
  - ✓destory ()：释放该Servlet资源

Servlet工作原理—— Servlet生命周期



Servlet工作原理—— Servlet生命周期

- 1. Servlet第一次被请求时，由Servlet容器建立其实例，并调用其init()方法。Servlet初始化仅此一次。
- 2.当一个客户端请求来临时，Servlet容器会将请求自动组装为一个ServletRequest对象，并自动产生一个ServletResponse对象，这两个对象一并传递给Servlet的service(req,res)方法。
  - ServletRequest:负责接受用户请求数据
  - ServletResponse:负责响应用户，展现结果

Servlet工作原理—— Servlet生命周期

- 3.调用service()方法处理业务逻辑，访问其他资源，获得需要的信息 。Service方法根据请求的方法类型执行doGet和doPost方法.
- 4. Servlet不再使用时调用destory方法释放(关闭Web服务器时)

注意：对于多个客户端请求，Server创建新的请求和响应对象，仍然激活此Servlet的service()方法，将这两个对象作为参数传递给它。如此重复以上的循环，但无需再次调用init()方法。一般Servlet只初始化一次 ,当Server不再需要Servlet时(一般当Server关闭时)，Server调用Servlet的Destroy()方法。

Servlet工作原理—— Servlet单例模式

1. web.xml声明一次：

单例模式，创建一个实例，通过多线程响应多个客户端请求

```
public class Test1{
        ...
        public void fun1(){
                String s =  "";
                System.out.print(s);
    }
}
```

不同的用户访问s变量，每一个用户执行的fun1方法都由自身的线程单独开辟

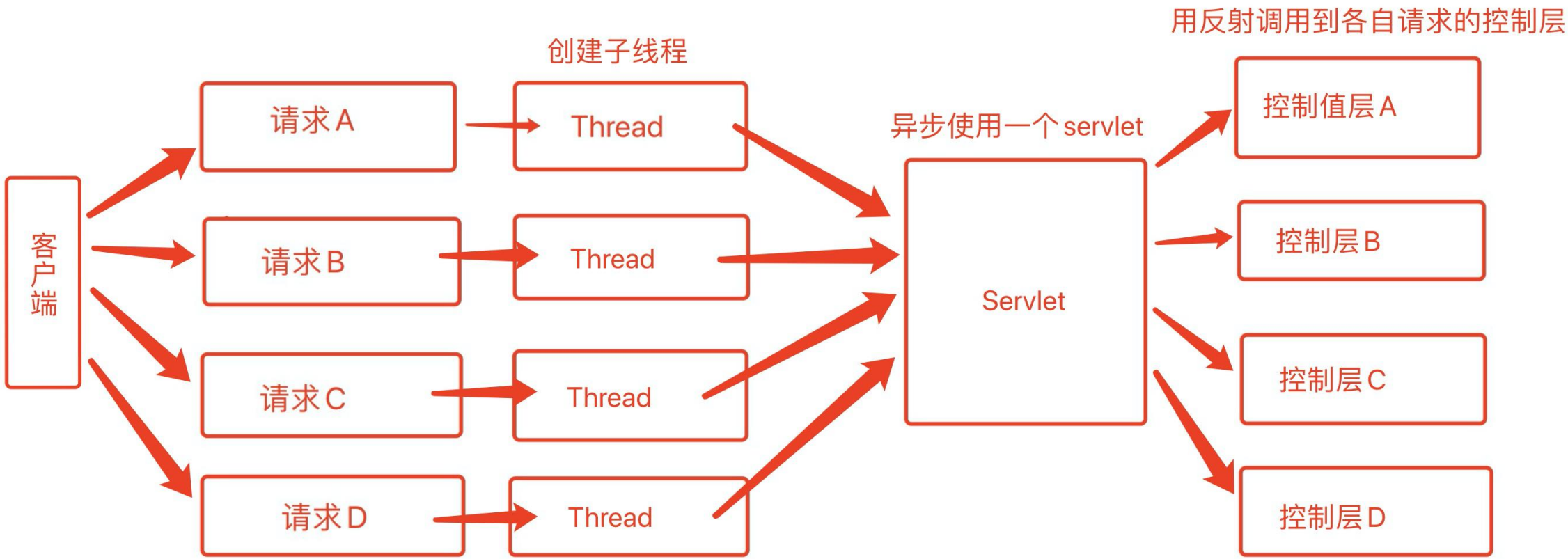
```
public class Test2{
        ...
        String s ;
        public void fun1(){
                System.out.print(s);
    }
}
```

不同用户访问的s变量都是同一个变量

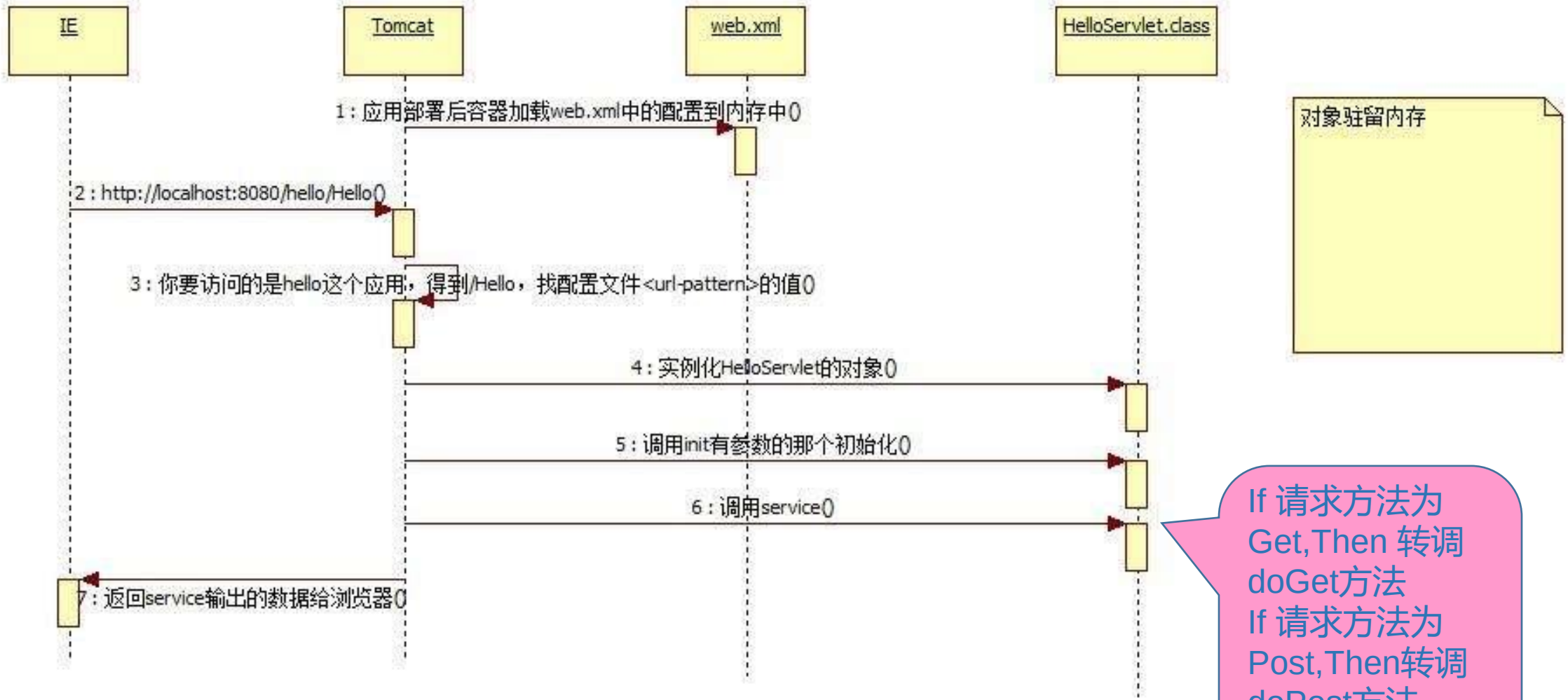
2. web.xml声明多次：

创建多个实例，通过不同的servlet-name区分

Servlet工作原理—— Servlet单例模式



Servlet工作原理—— Servlet执行顺序图



Servlet工作原理—— Servlet参数获取方法

启动一个servlet线程时，自动调用servlet的初始化函数。可以在初始化函数中读取配置文件（web.xml）中一些参数。

1、读取当前网站实际物理路径

```
String path=config.getServletContext().getRealPath("/");
```

Servlet工作原理—— Servlet参数获取方法

2、读取web.xml文件参数

```
<servlet>
  <servlet-name>ServletTest</servlet-name>
  <servlet-class>test. ServletTest</servlet-class>
  <init-param>
    <param-name>user</param-name>
    <param-value>root</param-value>
  </init-param>
</servlet>
```

该servlet中定义了一个参数名为user值为root,在servlet中如何读取呢？可以在servlet类中init(ServletConfig config)函数：

```
public void init(ServletConfig config) throws
ServletException {
        super.init(config);
    String value=config.getInitParameter("user");
    System.out.println(value);
}
```

Servlet工作原理—— Servlet参数获取方法

上面初始化参数其实只能由该ServletTest这个servlet读取，类似于局部变量。如果要定义全局的初始化参数呢，可以在web.xml中这样定义.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app >
  <servlet>
    <servlet-name>ServletTest</servlet-name>
    <servlet-class>test. ServletTest</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ServletTest</servlet-name>
    <url-pattern>/ServletTest</url-pattern>
  </servlet-mapping>
   <context-param>
      <param-name>url</param-name>
      <param-value>
            jdbc:mysql://127.0.0.1:3306/support</param-value>
   </context-param>
</web-app>
```

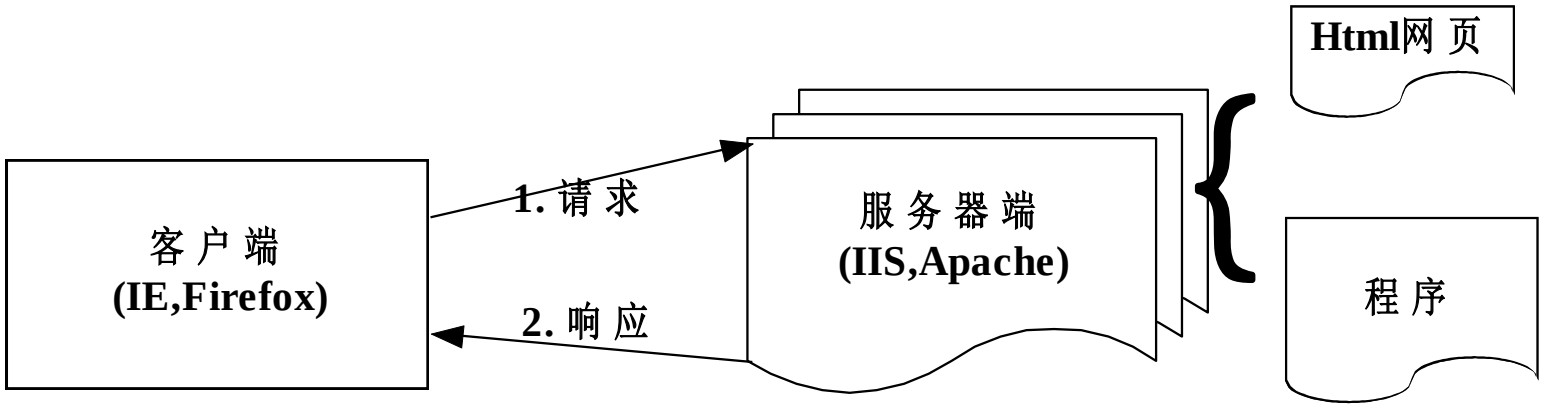
在 Servlet读取代码

```
public void init(ServletConfig config) throws ServletException {
   super.init(config);
   String url=config.getServletContext(). getInitParameter("url");
    System.out.println(url);
}
```

补充知识

HTTP请求简介

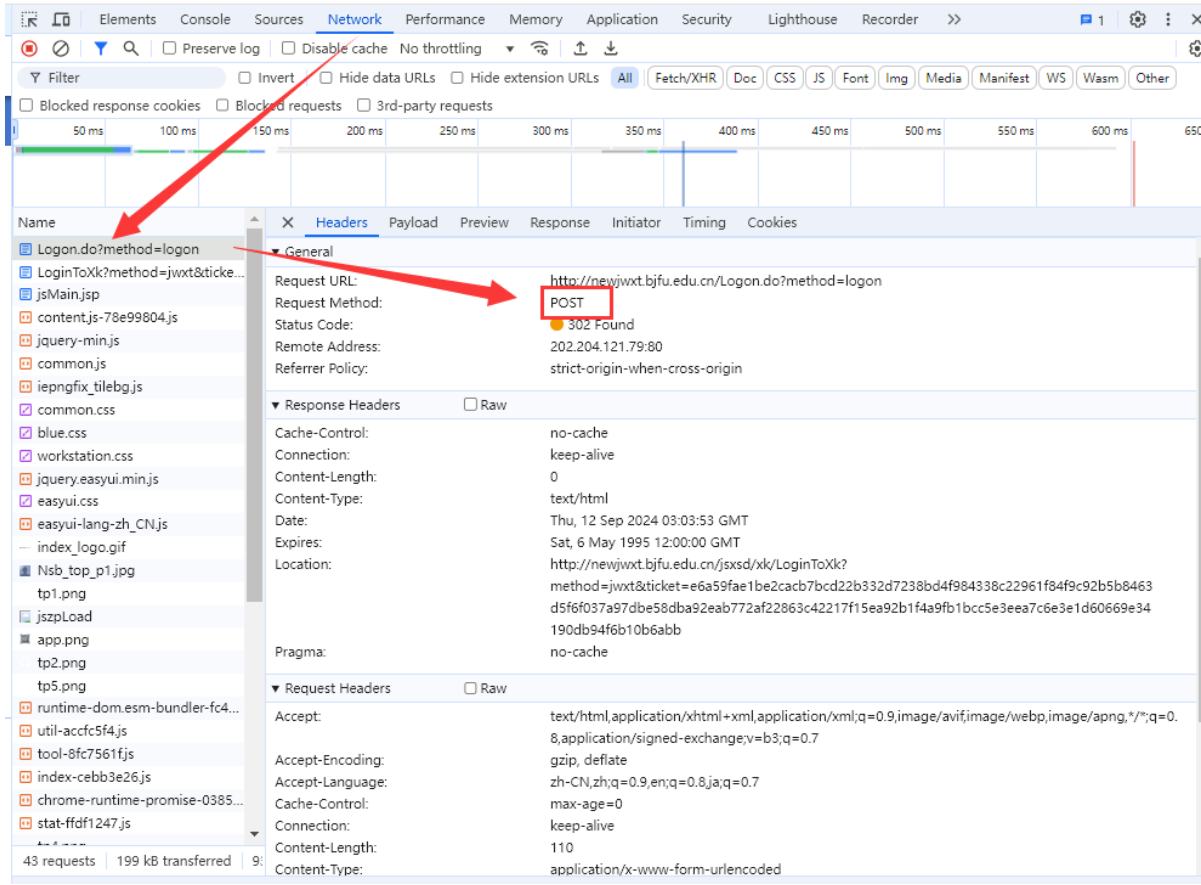
- 客户端同Web服务器之间的协议为http
  - HTTP为超文本传输协议
  - HTPP协议为“请求和响应”协议
- 客户端请求的方法
  - GET、POST、HEAD
  - DELETE、TRACE、PUT
- 提交表单的方法： GET、POST



HTTP请求简介

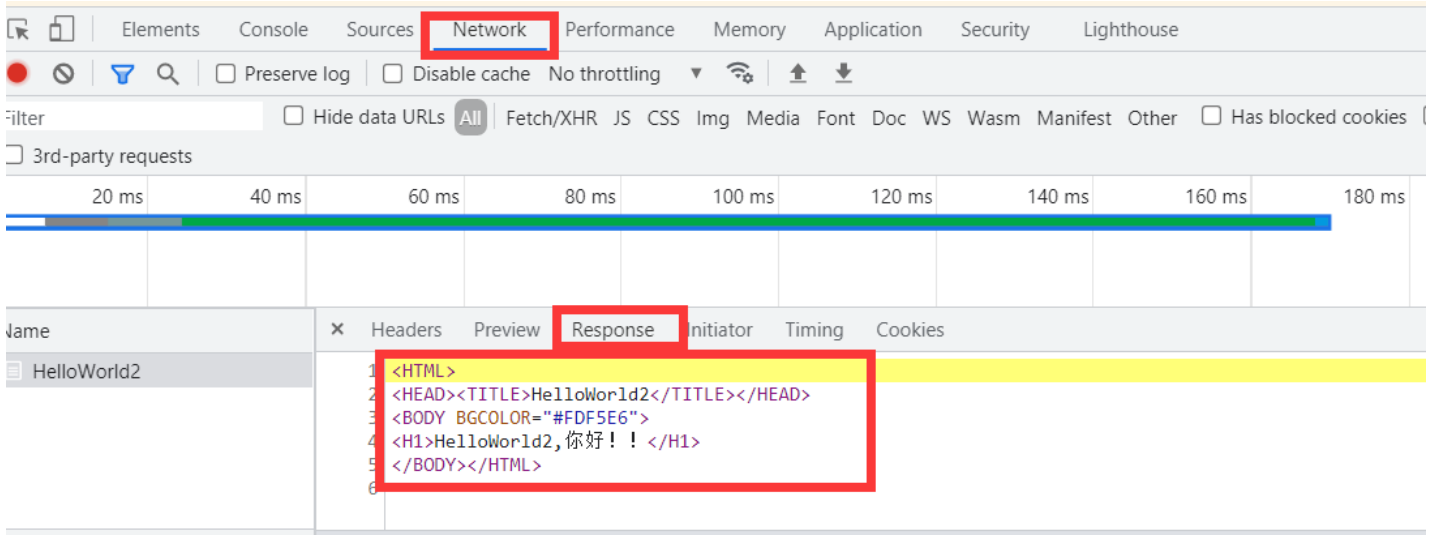
- 客户端请求包括：请求行、头、数据体

GET 实例	POST 实例
GET /servlet/default.jsp? LastName=Franks&FirstName=Michael HTTP/1.1 Accept: text/plain; text/html Accept-Language: en-gb Connection: Keep-Alive Host: localhost Referer: http://localhost/ch8/SendDetails.htm User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98) Content-Length: 33 Content-Type: application/x-www-form-urlencoded Accept-Encoding: gzip, deflate	POST /servlet/default.jsp HTTP/1.1 Accept: text/plain; text/html Accept-Language: en-gb Connection: Keep-Alive Host: localhost Referer: http://localhost/ch8/SendDetails.htm User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98) Content-Length: 33 Content-Type: application/x-www-form-urlencoded Accept-Encoding: gzip, deflate LastName=Franks&FirstName=Michael



HTTP响应简介

- 服务器接收到请求后，返回HTTP响应
- 每个响应：状态行、头、数据体
- 常见状态
  - 404 所请求的文件不存在
  - 500 服务器程序出错
  - 200 OK 成功
- 数据体用于浏览器显示




```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Mon, 3 Jan 1998 13:13:33 GMT
Content-Type: text/html
Last-Modified: Mon, 11 Jan 1998 13:23:42 GMT
Content-Length: 112

<html>
<head>
<title>HTTP Response Example</title></head><body>
Welcome to Brainy Software
</body>
</html>
```

表单中的Get方法和Post方法

- 两个方法都是请求另一个URL，并提交数据
  - 使用Get请求时，需声明
    - ✓<FORM ACTION= " *http://localhost:8080/ServletTest/HelloWorld1*" >，默认为Get方法
  - 使用Post请求时，需声明
    - ✓<FORM ACTION="*http://localhost:8080/ServletTest/HelloWorld1*" METHOD="*POST*">
  - 具体可参见示例工程ServletTest中的simpleform.html
- Get发送的数据和请求的地址一起显示在地址栏
  - http://localhost:8080/ServletTest/GetNameandSchool?name=wxy&school=bjfu
  - 具体可参见示例工程ServletTest中的formgettest.html
- Post请求的数据会隐藏在请求报文的body部分
  - 在地址栏
  - http://localhost:8080/ServletTest/GetHelloForm
  - 具体可参见示例工程ServletTest中的HelloForm.html

http中的Get方法和Post方法

- 数据传输的保密性上
  - Get方法弱
  - Post方法强
- 数据传输的数量上
  - Get方法弱，url之后一般不能多于256个字符
  - Post方法强

课堂复习1

- 掌握Web项目建立
  - 创建Web项目→建立HTML请求页面→建立Servlet应答文件→配置web.xml→调试运行→项目打包→项目发布到服务器上
- 掌握HTML请求页面的撰写
  - <form action="???" method="???"
  - <input type="???" name="???" value="???">…</input>
  - <select>…</select>
  - <textarea>…</textarea>

课后练习

- 在自己机器上练习
  - **安装并配置JRE,Tomcat,Java EE，MyEclipse**（包括中文和**Tomcat**插件）
  - 把提供的**ServletTest**文件导入**Eclipse**,在**Eclipse**下能运行并发布到**Tomcat**的**webapp**目录下

感谢聆听

Thanks For Your Listening!