Servlet入门

本章内容：共1小节，11个知识点

- 第1节：Servlet入门

本章目标

- 理解Servlet的线程特性；
- 能够编写Servlet，对于不同类型的请求进行响应；
- 能够配置Servlet的初始化参数、启动项、全局参数；
- 掌握请求和响应接口的作用和基础方法；
- 能够处理请求参数；
- 能够获得常用的请求头属性信息；
- 能够配置错误页面；

第1节【Servlet入门】-1

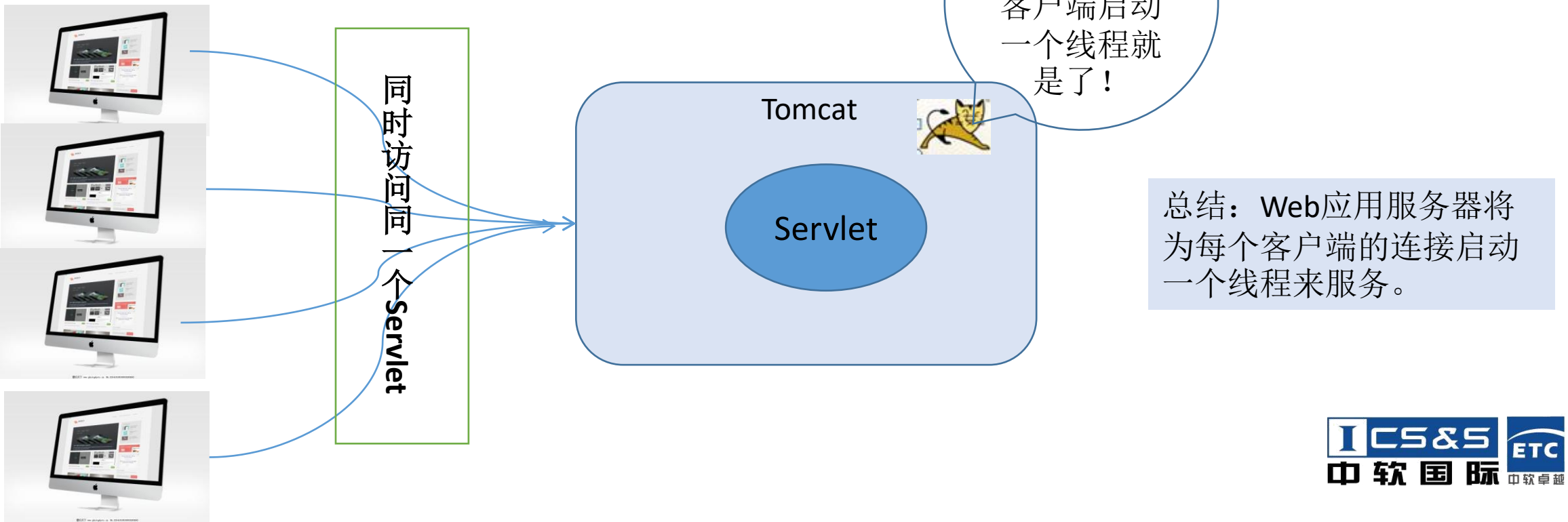
- 知识点1：Servlet线程特性
- 知识点2：请求和响应接口
- 知识点3：利用Servlet对客户端不同方式请求作出动态响应
- 知识点4： Servlet中获取普通请求不同名或同名的参数（name1=value1&name2=value2&...）的方法
- 知识点5： Servlet初始化参数
- 知识点6： 全局参数
- 知识点7： Servlet加载启动选项

第1节【Servlet入门】-2

- 知识点8： Servlet配置中通配符*的用法
- 知识点9： web.xml中首页及错误页面等其他配置信息
- 知识点10：Servlet中获取请求头属性的方法
- 知识点11：重要的请求头属性（请求长度、请求MIME类型、请求referer等）

知识点1【Servlet线程特性】-1

- Servlet是运行在服务器端的组件，能够给客户端返回动态页面；
- 那么问题来了：肯定会有多个客户端同时请求访问一个Servlet，Web服务器怎么处理多个请求呢？

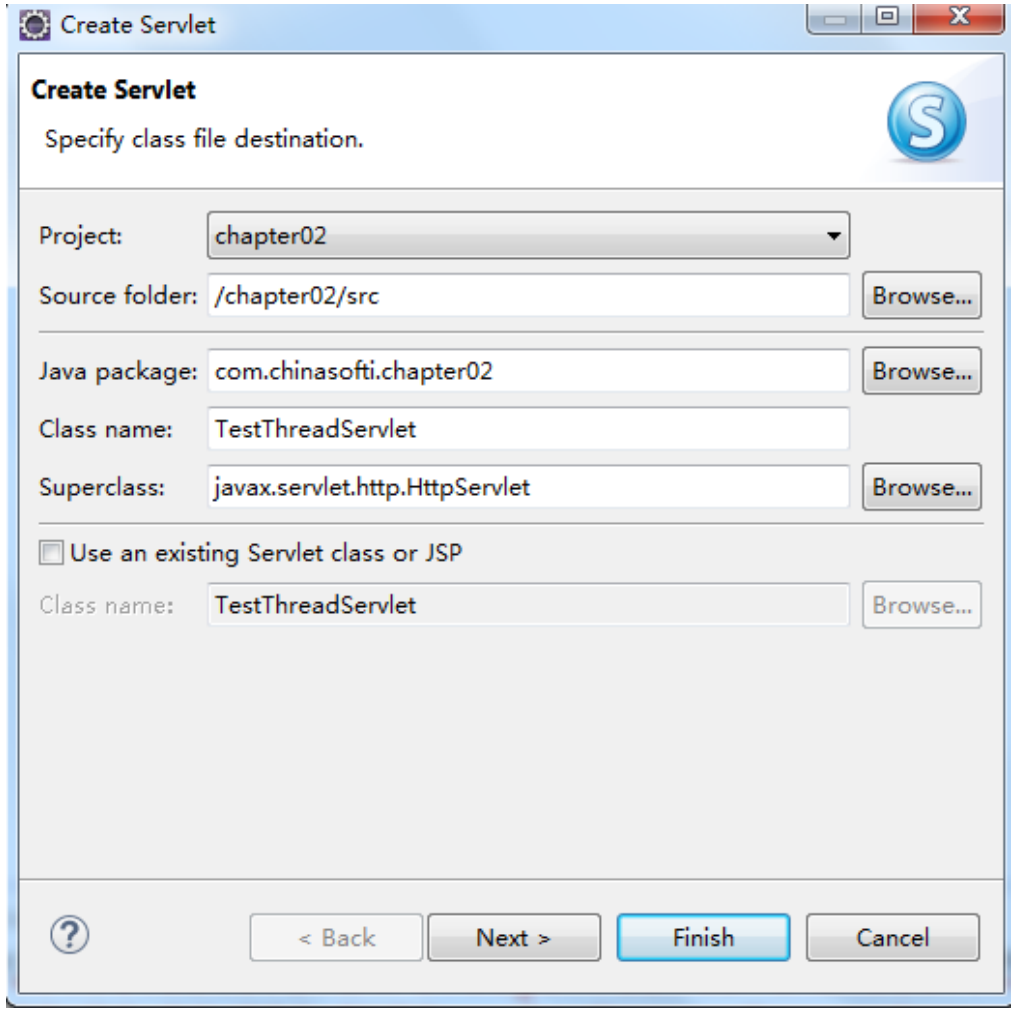


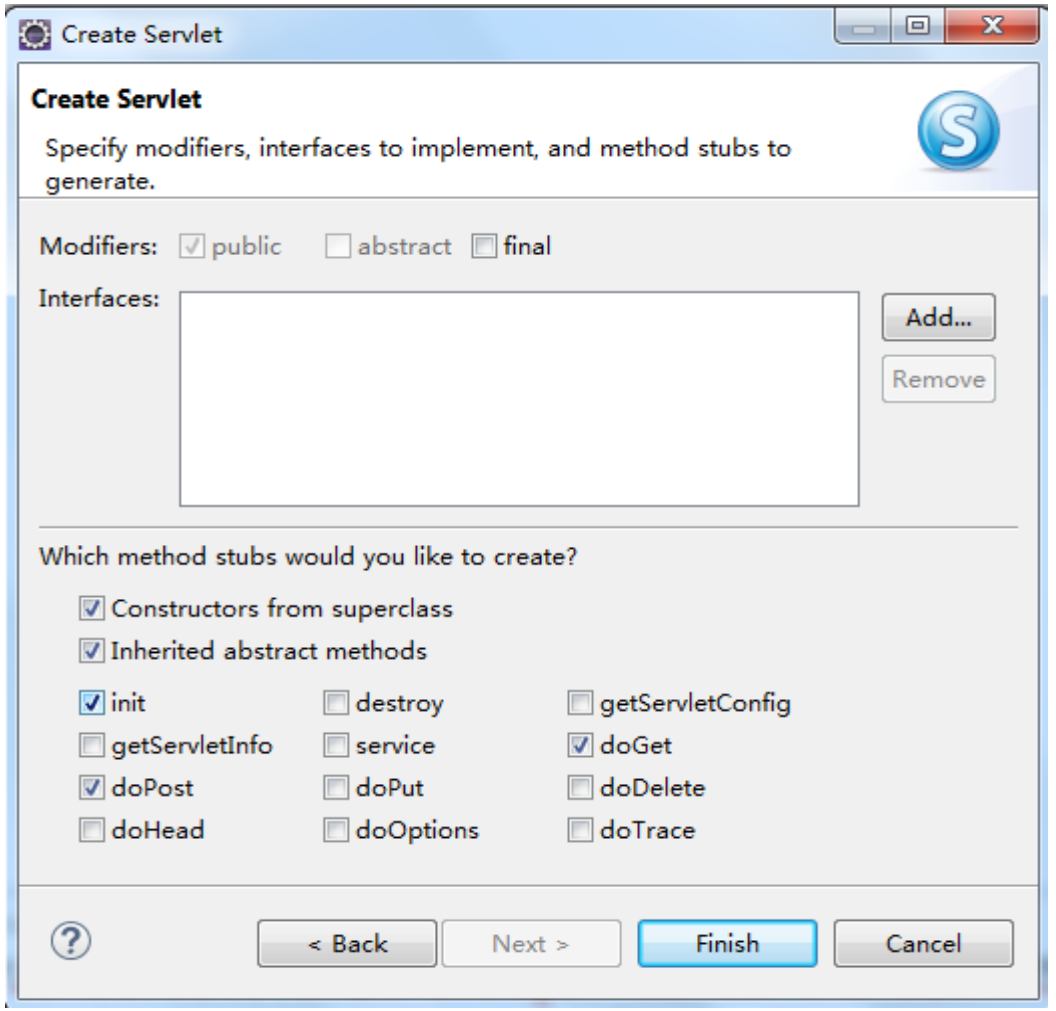
知识点1【Servlet线程特性】-2

课堂案例：
TestThreadServlet.java

- 思考：多个浏览器客户端访问同一个Servlet，服务器会创建多少个Servlet对象吗?
- 让我们编写简单的Servlet来进行验证;
- 分别在构造方法、无参init方法、doGet方法、doPost方法中编写打印输出语句：

Eclipse工具自动创建Servlet





步骤：右键点击工程下的包----->创建----->Servlet

课堂案例：
TestThreadServlet.java

```
@WebServlet("/TestThreadServlet")
public class TestThreadServlet extends HttpServlet {
		private static final long serialVersionUID = 1L;

    /**
     * Default constructor.
     */
    public TestThreadServlet() {
		System.out.println("调用无参构造方法！！！");
    }
```

课堂案例：
TestThreadServlet.java

```
/**
             * @see Servlet#init(ServletConfig)
             */
            public void init() throws ServletException {
        System.out.println("调用init()方法!!!");
    }
                        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                        response.getWriter().append("Served at: ").append(request.getContextPath());
          this.doPost(request, response);
                        System.out.println("调用doGet()方法");
}
```

课堂案例：
TestThreadServlet.java

```
	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
		// TODO Auto-generated method stub
	  System.out.println("调用doPost()方法！！！");
	}
}
```

知识点1【Servlet线程特性】-3

课堂案例：
TestThreadServlet.java

- 测试步骤
  - 第一次访问TestThreadServlet，结果为：

```
调用构造方法TestThreadServlet()
调用无参init()方法
调用doGet方法
```

服务器调用构造方法创建对象，再调用有参init方法，再调用无参init方法。

  - 打开新的浏览器窗口，访问TestThreadServlet，结果为：

```
调用doGet方法
```

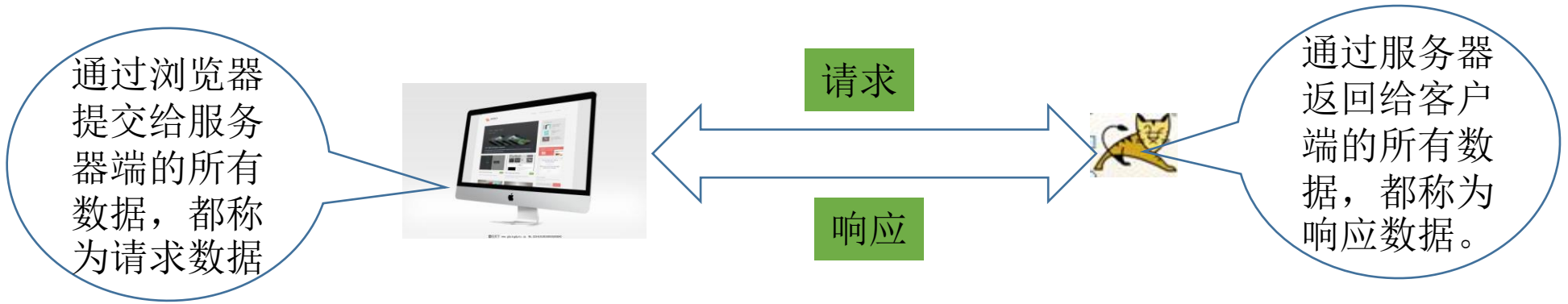
  - 不管再有多少个新的浏览器窗口访问TestThreadServlet，结果都是：

```
调用doGet方法
```

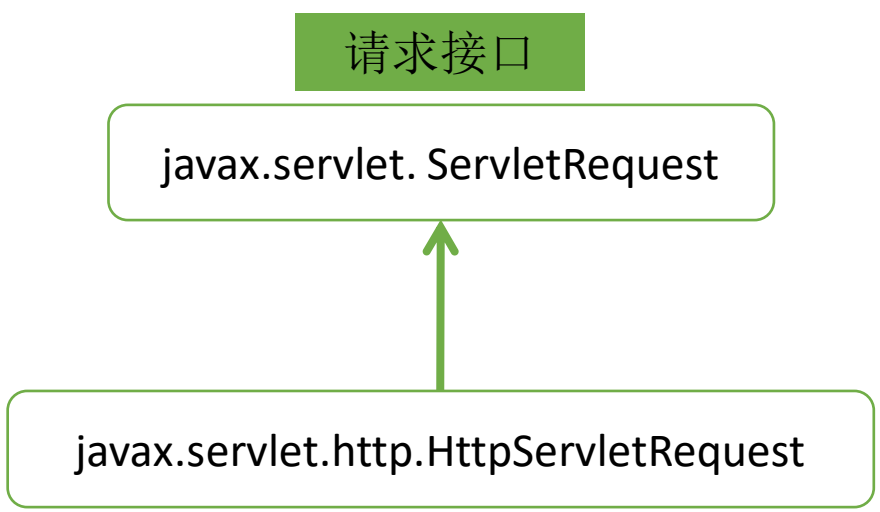
结论：第一次访问Servlet时，服务器将创建一个该Servlet类的对象，并调用doXXX方法生成响应；多个客户端访问同一个Servlet时，不再创建新的对象，而是共用同一个Servlet对象。可以说，Servlet是多线程单实例的。

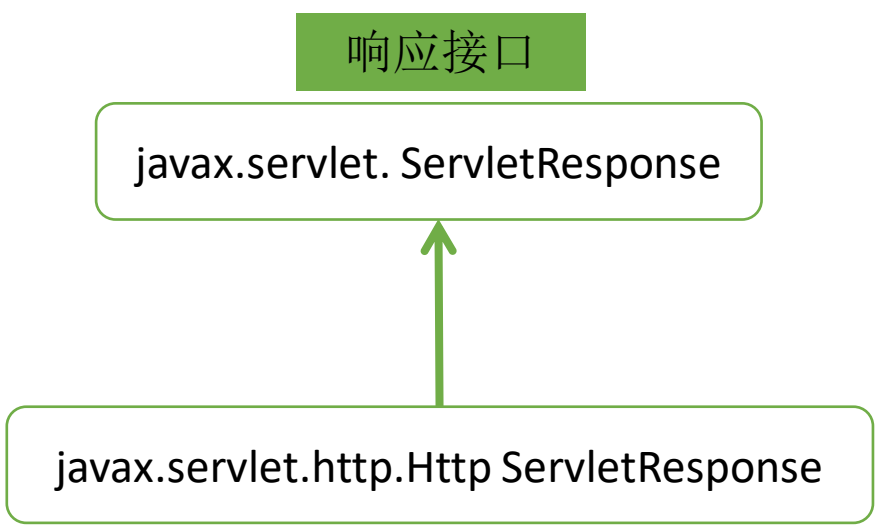
知识点2【请求和响应接口】-1

- Web应用基于HTTP协议，HTTP协议基于请求/响应模型；



- Servlet API中，定义了请求接口和响应接口，用来封装和操作请求和响应数据；





知识点2【请求和响应接口】-2

- Servlet类使用doXXX方法提供服务，这些方法继承于HttpServlet类；
- 查看API，可见doXXX方法都有两个参数，分别是请求和响应：

protected void	**doDelete**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a DELETE request.
protected void	**doGet**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a GET request.
protected void	**doHead**(HttpServletRequest req, HttpServletResponse resp) Receives an HTTP HEAD request from the protected service method and handles the request.
protected void	**doOptions**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a OPTIONS request.
protected void	**doPost**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a POST request.
protected void	**doPut**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a PUT request.
protected void	**doTrace**(HttpServletRequest req, HttpServletResponse resp) Called by the server (via the service method) to allow a servlet to handle a TRACE request.

- 也就是说：服务器会创建请求对象和响应对象传递给doXXX方法，doXXX方法中可以直接使用请求和响应对象；

知识点2【请求和响应接口】-3

课堂案例：
TestReqResServlet.java

- 编写简单Servlet，验证服务器会创建请求和响应对象：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
    System.out.println("请求对象： "+request);
    System.out.println("响应对象： "+response);
}
```

- 访问该Servlet，打印输出结果：

```
请求对象：org.apache.catalina.connector.RequestFacade@318877
响应对象：org.apache.catalina.connector.ResponseFacade@40fee2
```

- 通过以上结果可见：
  - 服务器实现了请求和响应接口，实现类分别是RequestFacade以及ResponseFacade；
  - 每次访问Servlet，服务器都会创建请求对象和响应对象传递给doXXX方法；

知识点2【请求和响应接口】-4

课堂案例：
TestReqResServlet.java

- 既然服务器确实创建了请求和响应对象，那么就可以在doXXX方法使用该对象：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
    System.out.println("请求对象： "+request);
    System.out.println("响应对象： "+response);
    System.out.println("你的IP地址是： "+request.getRemoteAddr());
}
```

该方法返回客户端的IP地址。

- 访问该Servlet，打印输出结果：

```
你的IP地址是：127.0.0.1
```

- 通过以上结果可见：
  - doXXX方法中可以使用方法参数request，response去调用请求和响应接口中的方法；
  - 后续将逐渐熟悉请求和响应接口中的常用方法；

知识点3【利用Servlet对客户端不同方式请求作出动态响应】-1

- 客户端访问服务器端Servlet的方式有三种，分别是：
  - 直接从地址栏输入URL访问；
  - 在网页中点击超级链接访问；
  - 在网页中通过表单提交访问；
- 编写Servlet，返回客户端IP地址，以测试三种请求方式；

知识点3【利用Servlet对客户端不同方式请求作出动态响应】-2

课堂案例：
TestThreeMethodsServlet.java

- 编写Servlet，返回客户端IP地址，以测试三种请求方式；
- doGet和doPost方法实现相同功能，向客户端返回IP地址；

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                        response.setContentType("text/html;charset=utf-8");
                        PrintWriter out=response.getWriter();
                        String ip=request.getRemoteAddr();
                        out.println("您好，目前调用的是doGet方法，您的IP地址是："+ip);
                        out.close();
            }
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
                        response.setContentType("text/html;charset=utf-8");
                        PrintWriter out=response.getWriter();
                        String ip=request.getRemoteAddr();
                        out.println("您好，目前调用的是doPost方法，您的IP地址是："+ip);
                        out.close();
            }
```

知识点3【利用Servlet对客户端不同方式请求作出动态响应】-3

- 编写index.html，定义超级链接、表单；访问Servlet；

课堂案例：
index.html

- 一个表单使用默认的get方式，一个使用post方式；TestThreeMethodsServlet是Servlet的url-pattern；

```
<a href="TestThreeMethodsServlet">访问TestThreeMethodsServlet</a><br><br>

<form action="TestThreeMethodsServlet" >
        用户名：<input name="username" type="text"><br><br>
        密　码：<input name="pwd" type="password"><br>
        <input type="submit" value="提交"/><br><br>
</form>

<form method="post" action="TestThreeMethodsServlet">
        用户名：<input name="username" type="text"><br><br>
        密　码：<input name="pwd" type="password"><br>
        <input type="submit" value="提交"/><br>
</form>
```

知识点3【利用Servlet对客户端不同方式请求作出动态响应】-4

- 方式一：直接在浏览器输入URL

  您好，目前调用的是doGet方法，您的IP地址是：127.0.0.1

- 方式二：点击index.html上的超级链接访问

  您好，目前调用的是doGet方法，您的IP地址是：127.0.0.1

- 方式三：点击表单的按钮
  - 没有指定method的表单，默认是get方式

    您好，目前调用的是doGet方法，您的IP地址是：127.0.0.1
  - 指定method为post的表单

    您好，目前调用的是doPost方法，您的IP地址是：127.0.0.1

总结：

1、直接使用URL访问，是GET方式，调用doGet方法；

2、超级链接访问，是GET方式，调用doGet方法；

3、表单提交访问，取决form的method属性的值，默认是get，指定为post时，调用doPost方法；

知识点4【Servlet中获取请求参数的方法】-1

- 当客户端请求服务器端的Servlet时，可以向Servlet传递请求参数，有以下几种方式：
  - 可以在URL后使用name=value&name=value的形式传递，例如：

    <a href="TestPramServlet?page=1&author=wangxh">传递两个请求参数，名字分别为page和author，值分别为1和wangxh；
  - 可以在使用表单提交，表单中的元素值将作为请求参数传递，元素的name是参数名字，value的值是参数的值，例如：testParam.html

```
<form action="TestPramServlet" method="post">
        用户名：<input name="userName" type="text"><br><br>
        密　码：<input name="pwd" type="password"><br>
        技术方向：
        <input type="checkbox" name="techs" value="Java">1、Java
        <input type="checkbox" name="techs" value="大数据">2、大数据
        <input type="checkbox" name="techs" value="人工智能">3、人工智能
        <input type="checkbox" name="techs" value="HTML5">4、HTML5<br>
        <input type="hidden" name="action" value="test"/>
        <input type="submit" value="提交"/><br><br></form>
```

总结：

1、请求参数username和pwd的值取决于用户的输入；

2、请求参数techs可能包含多个值，取决于用户选择情况；

3、请求参数action是一个隐藏参数，也会传递到服务器端，不过不在浏览器显示；

知识点4【Servlet中获取请求参数的方法】-2

- 当客户端请求服务器端的Servlet时，请求参数都会被发送到服务器，服务器会将请求参数封装到请求对象中；
- 请求接口中提供了四个与请求参数相关的方法，前两个方法常用：

方法声明	方法描述
java.lang.String getParameter(java.lang.String name)	返回某个指定名字的请求参数的值，值为String类型；
java.lang.String[] getParameterValues(java.lang.String name)	返回指定名字的请求参数的值，值为String[]类型，一般用于一个名字对应多个值情况；
java.util.Map<java.lang.String,java.lang.String[]> getParameterMap()	将所有请求参数的name和value作为键值对返回，存储在Map对象中；
java.util.Enumeration<java.lang.String> getParameterNames()	返回所有的请求参数的名字，存在集合对象中；

知识点4【Servlet中获取请求参数的方法】-3

课堂案例：
TestParamServlet.java
testParam.html

- 在Servlet中可以获得请求参数：

testParam.html

TestPramServlet.java

用户名：Brain
密码：111111
Java
大数据
隐藏参数action的值是：test

课堂案例：
TestParamServlet.java

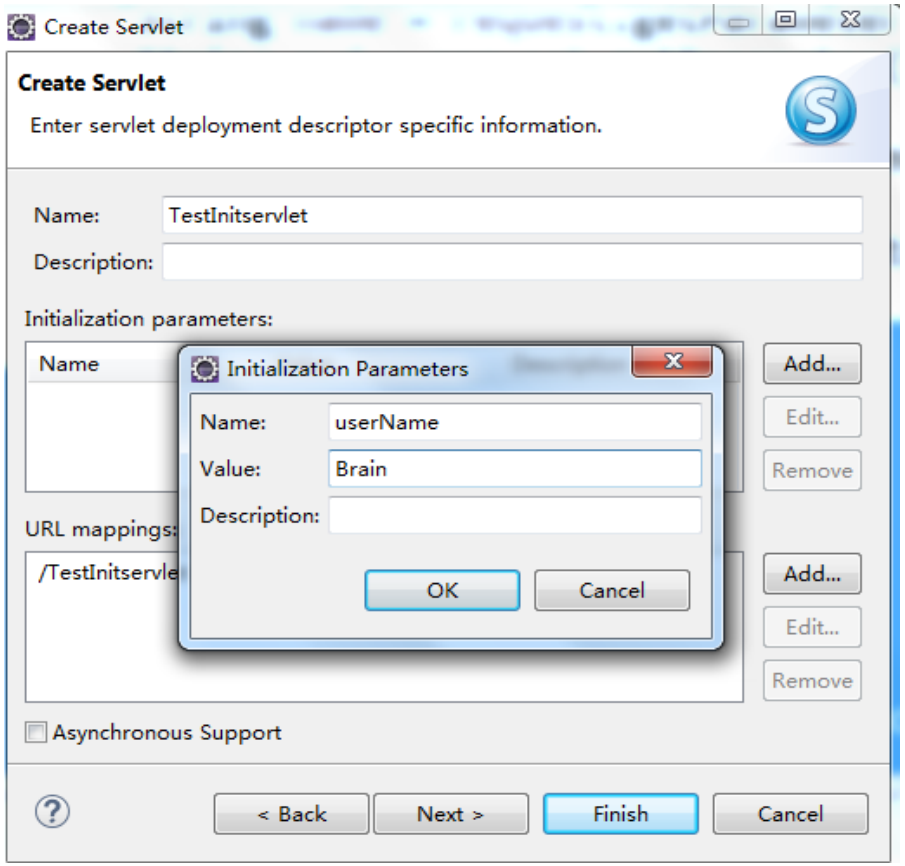
```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
                    // 设置浏览器向服务器发送字节的编码
                    request.setCharacterEncoding("utf-8");
                    // 响应 设置服务器返回给浏览器时的字符编码
            response.setContentType("text/html;charset=utf-8");
                    PrintWriter out = response.getWriter();              //获得请求参数
                    String name = request.getParameter("userName");
                    String pwd = request.getParameter("password");
                    String [] techs = request.getParameterValues("techs");
                    String action = request.getParameter("action");
```

课堂案例：
TestParamServlet.java

```
out.println("用户名： "+name+"<br>");
                    out.println("密码： "+pwd+"<br>");
                    out.println("技术方向有： "+"<br>");
                    for(String t:techs) {
                                out.println(t+"<br>");
                    }
                    out.println("隐藏参数action的值是： "+action+"<br>");
                    out.close();
          }
```

知识点5【Servlet初始化参数】-1

- 如果Servlet需要使用一些可以配置的参数，可以在创建Servlet时进行设置称为初始化参数；
- 一个Servlet可以配置多个初始化参数，所有的初始化参数只能在当前Servlet类中使用；




```
@WebServlet(
        urlPatterns = { "/TestInitServlet" },
        initParams = {
                @WebInitParam(name = "userName", value = "Brain"),
                @WebInitParam(name = "password", value = "111111")
        })
public class TestInitServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

知识点5【Servlet初始化参数】-2

课堂案例：
TestInitServlet.java

- ServletConfig接口中定义了两个与Servlet初始化参数有关的方法；
- 由于HttpServlet类已经间接实现了该接口，因此默认可以直接使用这两个方法；

方法声明	方法描述
java.lang.String getInitParameter(java.lang.String name)	返回某个指定名字的初始化参数的值，值为String类型；
java.util.Enumeration<java.lang.String> getInitParameterNames()	返回所有初始化参数的名字；

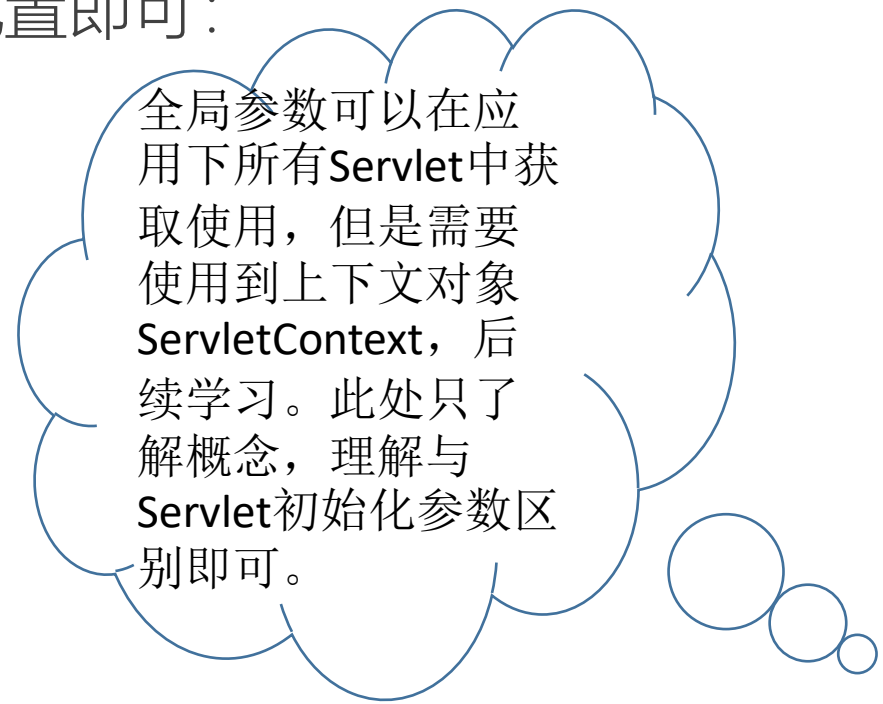
- 在当前的Servlet中可以获得初始化参数进行使用，其他Servlet中无法使用；

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                request.setCharacterEncoding("UTF-8");
                response.setContentType("text/html;charset=utf-8");
                PrintWriter out = response.getWriter();

                String name = this.getInitParameter("userName");
                String pwd = this.getInitParameter("password");

                out.println("用户名：  "+name);
                out.println("密码：  "+pwd);   }
```

知识点6【全局参数】

- Servlet的初始化参数只能在当前Servlet中使用；
- 如果需要在应用下所有Servlet中都能够使用某个参数，可以定义全局参数；
- 在web.xml的根节点下使用<context-param>配置即可：

```
<context-param>
  <param-name>propsFile</param-name>
  <param-value>config.properties</param-value>
 </context-param>
```



知识点7【Servlet加载启动选项】-1

课堂案例：
TestLoadServlet.java

- 默认情况下，只有当第一次访问Servlet时，服务器才会初始化Servlet实例；
- 如果需要更早实例化Servlet，可以在web.xml中进行配置，使得在启动容器的时候就能初始化Servlet实例；
- 在TestLoadServlet类中的构造方法，写打印语句，观察何时调用构造方法：

```
public TestLoadServlet() {
    super();
    System.out.println("调用了TestLoadServlet()，创建实例了~~~");
  }
```

知识点7【Servlet加载启动选项】-2

- 在web.xml中进行配置，使得在启动容器的时候就能对Servlet实例化；

```
<servlet>
  <description></description>
  <display-name>TestLoadServlet</display-name>
  <servlet-name>TestLoadServlet</servlet-name>
  <servlet-class>com.chinasofti.chapter02.section01.TestLoadServlet</servlet-class>
  <load-on-startup>2</load-on-startup>
 </servlet>
```

数字2不是表示个数，而是顺序。Servlet永远只被实例化1个对象。

- 启动Tomcat，不访问该Servlet，可以在启动日志中看到：

```
信息: 正在启动 Servlet 引擎: [Apache Tomcat/9.0.29]
调用了TestLoadServlet()，创建实例了~~~
十二月 27, 2019 3:28:35 下午 org.apache.coyote.AbstractProtocol start
```

说明启动的同时已经创建了一个Servlet的实例，而不是第一次访问才创建。

知识点8【Servlet配置中通配符*的用法】

- 要访问Servlet，必须为Servlet配置<url-pattern>，可以使用通配符*进行配置，从而通配多种访问模式；
- *使用有两种方式
  - *.扩展名： 比如 *.do、*.action
  - 以 / 开头，同时以 /* 结尾，比如 /* 、/admin/*

```
<servlet-mapping>
  <servlet-name>TestPatternServlet</servlet-name>
  <url-pattern>*.action</url-pattern>
  <url-pattern>/admin/*</url-pattern>
 </servlet-mapping>
```

这些URL都可以访问到TestPatternServlet

http://localhost:8080/chapter02/admin/hello
http://localhost:8080/chapter02/admin/file/upload
http://localhost:8080/chapter02/hello.action
http://localhost:8080/chapter02/admin/hello.action

知识点9【web.xml中首页及错误页面等其他配置信息】-1

- web.xml中还可以配置很多其他信息
- 配置默认首页：当不指定具体访问路径时，默认访问默认首页
- 将按照顺序访问，都不存在将报404错误

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

知识点9【web.xml中首页及错误页面等其他配置信息】-2

- web.xml中还可以配置很多其他信息；
- 配置错误页面：当应用中出现响应错误或者异常时，可以跳转到错误页面；

```
<error-page>
 <error-code>404</error-code>
 <location>/404.html</location>
 </error-page>

 <error-page>
 <exception-type>java.lang.NullPointerException</exception-type>
 <location>/exception.html</location>
 </error-page>
```

出现404错误时，自动跳转到404.html

发生空指针异常并没有被处理时，跳转到excetption.html

知识点10【Servlet中获取请求头属性的方法】

- 客户端请求服务端的Servlet时，会传递给服务器一系列的HTTP请求头属性，请求接口中定义了系列方法获取请求属性；

方法声明	方法描述
java.lang.String getHeader(java.lang.String name)	返回某个请求头属性的值，值为String类型；
java.util.Enumeration<java.lang.String> getHeaders(java.lang.String name)	返回指定名字的请求头属性的值，值为集合类型，一般用于一个名字对应多个值情况；
int getIntHeader(java.lang.String name)	返回值类型是int类型的请求头属性值；
long getDateHeader(java.lang.String name)	返回日期类型的请求头属性值，返回long型值；
java.util.Enumeration<java.lang.String> getHeaderNames()	返回所有请求头属性的名字；

知识点11【重要的请求头属性】-1

课堂案例：
TestHeadServlet.java
testhead.html

```
<a href="TestHeadServlet?param1=test">访问TestHeadServlet</a>

<form method="get" action="TestHeadServlet">
        用户名：<input name="username" type="text"><br><br>
        密　码：<input name="pwd" type="password"><br>
        <input type="submit" value="提交"/><br>
</form>

<form method="post" action="TestHeadServlet">
        用户名：<input name="username" type="text"><br><br>
        密　码：<input name="pwd" type="password"><br>
        <input type="submit" value="提交"/><br>
</form>
```

知识点11【重要的请求头属性】-1

课堂案例：
TestHeadServlet.java
testhead.html

- 实际应用中，可以使用请求接口中的方法，获取一些常用的请求头属性；
- 例如：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
//                    获得请求长度
                      int length=request.getIntHeader("Content-Length");
                      System.out.println("请求长度："+length);
//                    浏览器能接受的MIME（它全名叫多用途互联网邮件扩展（Multipurpose Internet Mail
//Extensions），最初是为了将纯文本格式的电子邮件扩展到可以支持多种信息格式而定制的。后来被应
//用到多种协议里，包括我们常用的HTTP协议。MIME的常见形式是一个主类型加一个子类型，用斜线分
//隔。比如text/html、application/javascript、image/png等。在访问网页时，MIME type帮助浏览器识别一
//个HTTP请求返回的是什么内容的数据，应该如何打开、如何显示。）类型
                      String mime=request.getHeader("Accept");//如果不存在指定的 header 名，则返回 -1。
                      System.out.println("MIME类型："+mime);
//                    来路路径信息
                      String referer=request.getHeader("Referer");
                      System.out.println("来路路径："+referer);
            }
```

知识点11【重要的请求头属性】-2

课堂案例：
TestHeadServlet.java
testhead.html

- 使用testhead.html中get方式提交表单访问：

```
请求长度：-1
MIME类型：text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
来路路径：http://localhost:8080/chapter02/testhead.html
```

- 使用testhead.html中post方式提交表单访问：

```
请求长度：25
MIME类型：text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
来路路径：http://localhost:8080/chapter02/testhead.html
```

本节总结提问【Servlet入门】

- Servlet的线程性质是什么?
- web.xml中可以配置哪些信息?
- Servlet初始化参数和全局参数有什么区别?
- 如何获得请求参数?
- 如何获得请求属性头?

本节总结【Servlet入门】

- Servlet是多线程，单实例的，不管访问多少次，只有一个Servlet实例；如果在web.xml中配置了启动加载项，则服务器启动时初始化，否则第一次访问时初始化；
- Servlet的初始化参数只能在当前Servlet中使用，全局参数可以在整个应用中使用；
- web.xml中可以配置Servlet相关信息，还可以配置错误页面、默认首页等；
- 对Servlet的url-patttern可以使用通配符*进行配置；
- 请求接口中定义了getParameter等方法获得请求参数；
- 请求接口中定义了getHeader等方法获得请求头属性；

本章作业

- 作业1:
- 题目：编写html页面，提供注册表单，注册表单中至少要用到文本框、多选框、单选框、下拉列表等表单元素，填写信息后提交到Servlet，Servlet读取注册信息进行显示。
- 难度：易
- 作业2:
- 题目：编写Servlet，验证Servlet的初始化过程，包括配置启动项情况。
- 难度：易

本章作业

- 作业3:
- 题目：编写Servlet，验证初始化参数的配置和使用。并同时验证使用通配符配置Servlet的方法。
- 难度：易