# 今日内容

# Xml概述（了解）

* 什么是XML:

E**X**tensible **M**arkup **L**anguage 可扩展的标记语言.

\* 标记语言：通过一组标签表示语言.<></> </>

\* 可扩展标记语言:HTML中的标签都是预定义好.XML允许用户自定义标签.

* XML的作用：

XML出现替换HTML. -XHTML.

XML用来存储数据(传递数据),作为软件的配置文件.

\* XML可以在任何平台下进行解析的.传递数据.(异步系统之间的传输数据).

\* XML可以作为软件的配置文件.

【XML基本语法】

\* 文档声明:

\* 在XML的文档的第一行第一列编写文档声明.

\* 语法：<?xml ?>

\* 属性:

\* version:版本号.必须的.

\* encoding:字符集.UTF-8,GBK,GB2312,ISO-8859-1

\* standalone:依赖外部的文件.yes-代表这个文档. no-代表这个文档依赖外部的文档.

\* 元素:标签.(可以自定义标签)

\* 语法:

\* 标签区分大小写的.<a> <A>

\* XML语法比较严格,有开始标签,必须有结束标签.<book/>

\* XML的元素名称.需要以字母开始中间不能使用空格 不能使用:

\* 元素的名称不能是以XML,xml,Xml开头.

\* XML文档必须有一个跟元素.

\* 属性：

\* 语法:

\* 不能使用空格，冒号特殊字符.

\* 属性需要以字母开头.

\* 属性必须使用引号.

\* 注释：

\* <!-- 注释 -->

\* 特殊字符和CDATA区:

\* < &lt;

\* > &gt;

\* & &amp;

\* “ &quot;

\* ‘ &apos;

\* CDATA:<![CDATA[ 内容 ]]>

【XML解析】

\* XML的解析方式

\* DOM解析,SAX解析.

\* 针对这两种方式不同的公司提供了不同的API.

\* SUN:JAXP.

\* JDom:Jdom

\* Dom4J:dom4j

\* pull解析-主要应用在android.

\* jsoup解析-主要应用在网络爬虫,应用解析HTML的.

# Xml解析入门

Xml文件

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <books>  <book>  <auther id=*"xzy"*>许志远</auther>  <name>滞后在蓝翔技校文艺学院的那些日子</name>  <price>120</price>  </book>  <book>  <auther>hang</auther>  <name>厚脸皮是怎样炼成的</name>  <price>12</price>  </book>  </books> |

Xml解析快速入门

|  |
| --- |
| public class Demo1 {  public static void main(String[] args) throws DocumentException {  //获取文档的解析器  SAXReader reader = new SAXReader();  //加载文件 demo1.xml 返回document 代表整个文档  Document document = reader.read("src/demo1.xml");  //获取文件的根节点 （根元素）  Element rootElement = document.getRootElement();//<books></books>  Object object = rootElement.elements("book").get(0);//<book></book>  Element book1 = (Element)object ;  Element element = book1.element("auther");//<auther></auther>  //获取文本内容  String auther = element.getText();  System.*out*.println(auther);  }  } |

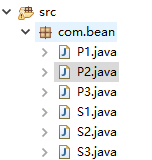
# 使用xml解析模拟spring管理对象

之前的对象 new 现在不用

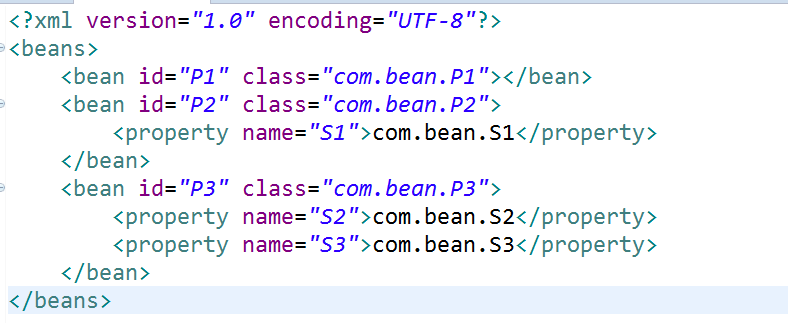
将你所使用的对象配置在配置文件中，通过读取配置文件将所有的对象创建，然后将对象放在容器中（map）。对象和对象之间存在着依赖关系

当你使用的时候直接去获取

1 创建需要使用的实体类



2 Bean之间的依赖关系



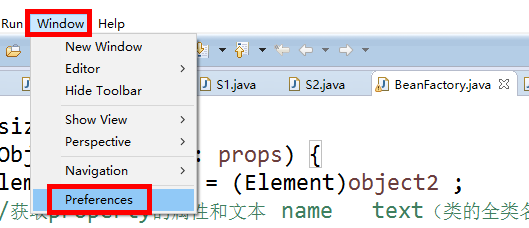
解析xml文件 按照顺序将所有的bean对象创建

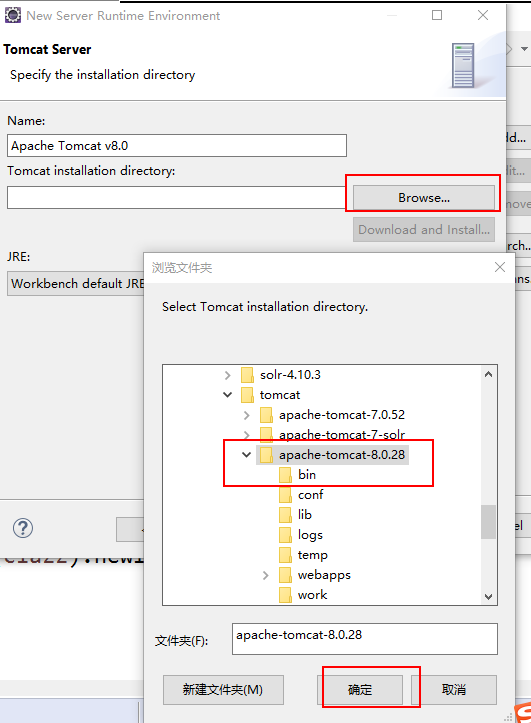
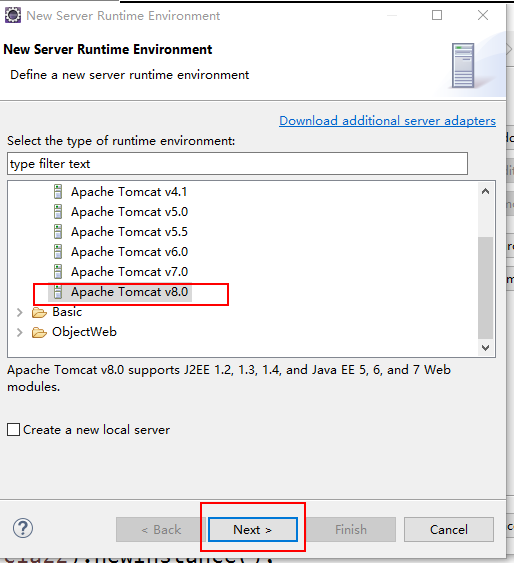
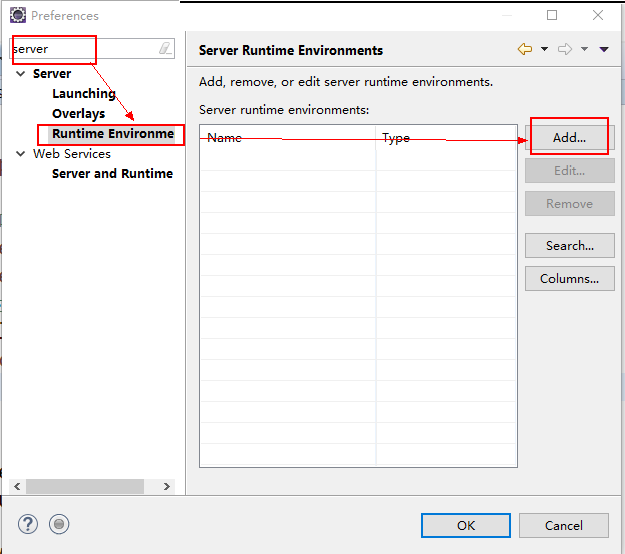
|  |
| --- |
| public class BeanFactory {  public static Object getBean(String beanName) throws Exception{  //1创建一个容器来存放对象  Map<String , Object> map = new HashMap<>();  //2读取配置文件 以此创建对象  //2。1创建解析器  SAXReader reader = new SAXReader();  //2.2加载配置文件  Document document = reader.read("src/application.xml");  //2.3获取根节点  Element rootElement = document.getRootElement();  //2.4获取元素 bean元素  List beans = rootElement.elements("bean");  //2.5遍历bean  for (Object object : beans) {  Element bean = (Element)object ;  //获取bean下面的子元素 property  List props = bean.elements("property");  //判断  if(props.size()>0){  for (Object object2 : props) {  Element property = (Element)object2 ;  //获取property的属性和文本 name text（类的全类名）  String name = property.attribute("name").getValue();//key  String text = property.getText();  //创建property下面的对象  Object object3 = Class.*forName*(text).newInstance();//对象作为value  map.put(name, object3) ;  }  }  //拿属性 id class  String key = bean.attribute("id").getValue();//key  String clazz = bean.attribute("class").getValue();  //创建对象  Object object4 = Class.*forName*(clazz).newInstance();  //将对象放在map中  map.put(key, object4) ;  }  return map.get(beanName);  }  } |

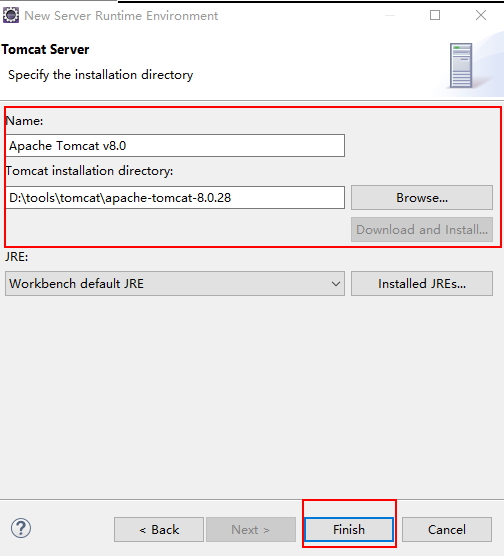
测试类测试

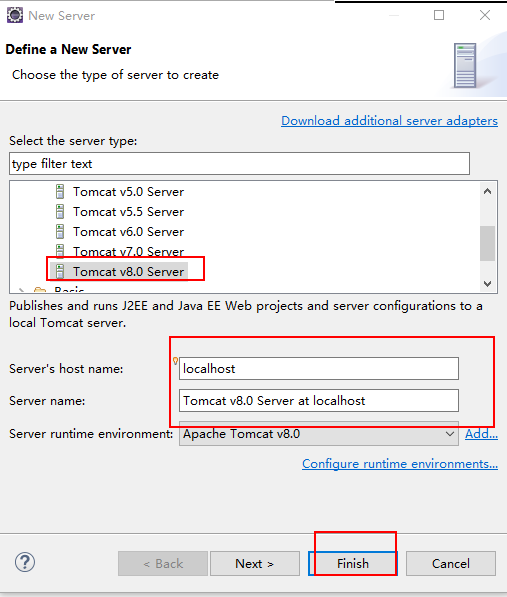
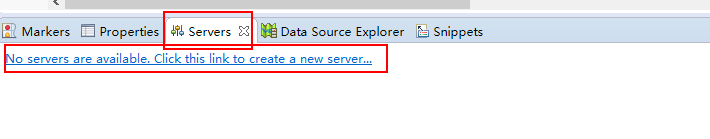
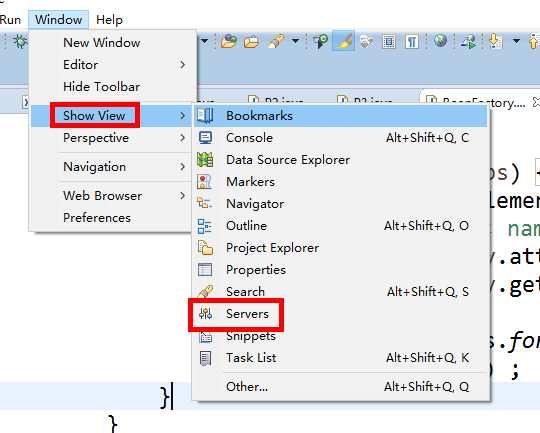
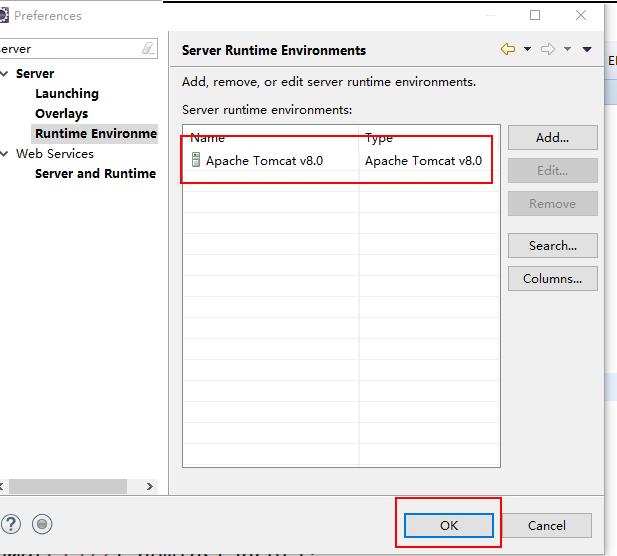
|  |
| --- |
| public static void main(String[] args) throws Exception {  S1 bean = (S1)BeanFactory.*getBean*("S1");  bean.run();  } |

# Tomcat的基本使用

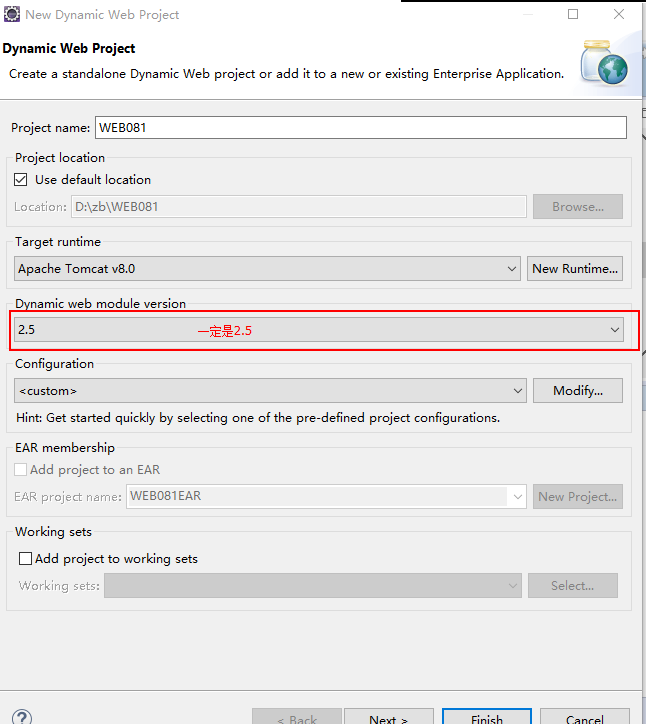
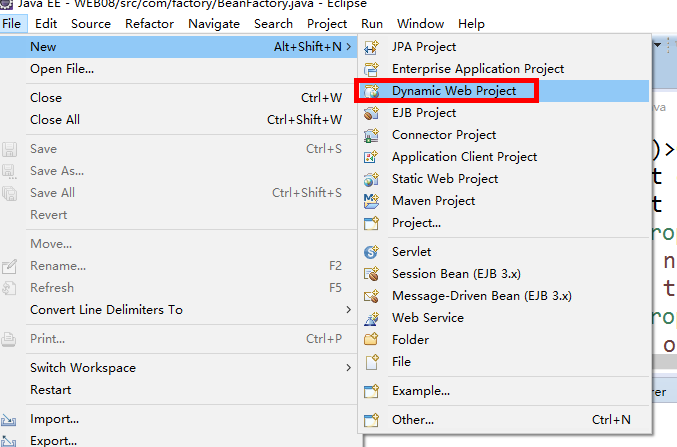
1. 



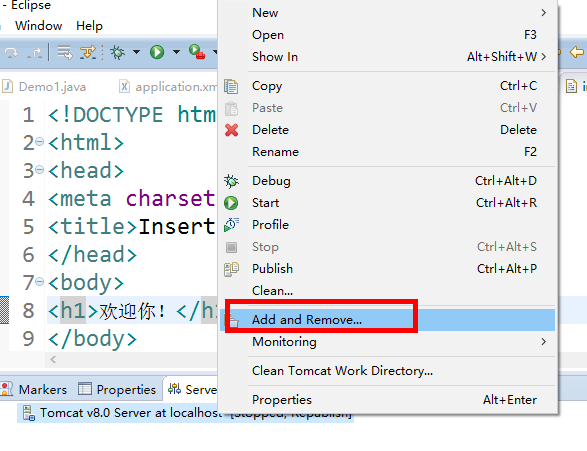


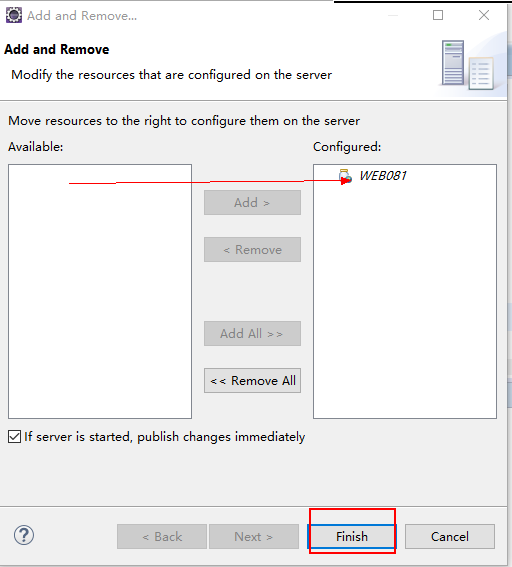


创建项目

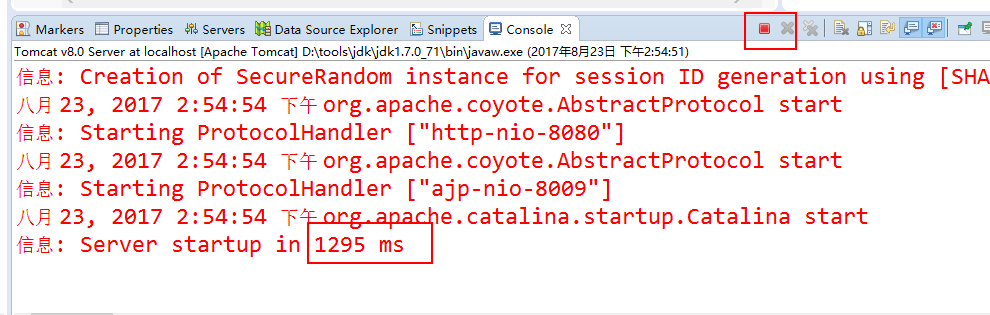


发布项目

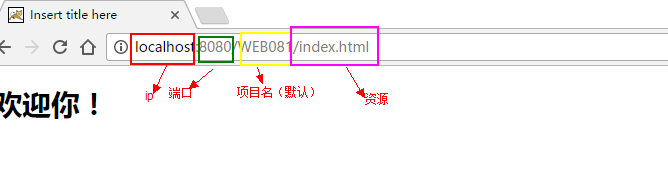




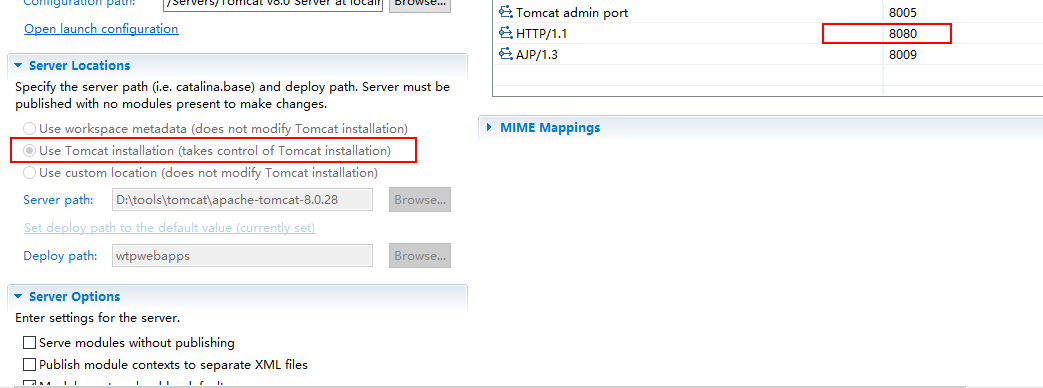
启动成功



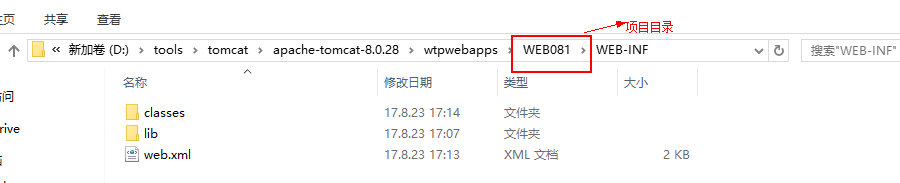
访问



修改发布的位置



所有的SRC下面的资源都在 WEB-INf/classes/下



# 网络资源的访问

1. http://localhost:8080/WEB081/index.html

# HTTP传输协议(知道)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 什么是Http协议 HTTP，超文本传输协议（HyperText Transfer Protocol)是互联网上应用最为广泛的 一种网络协议。所有的WWW文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法  超文本传输协议  请求 相应 Http协议的组成 Http协议由Http请求和Http响应组成，当在浏览器中输入网址访问某个网站时， 你的浏览器会将你的请求封装成一个Http请求发送给服务器站点，服务器接收到请 求后会组织响应数据封装成一个Http响应返回给浏览器。即没有请求就没有响应。   Http请求编辑一个form.html的表单页面  抓包： **页面按下F12检测到的内容** 请求行 请求方式：POST、GET  请求的资源：/DemoEE/form.html  协议版本：HTTP/1.1  HTTP/1.0，发送请求，创建一次连接，获得一个web资源，连接断开。  HTTP/1.1，发送请求，创建一次连接，获得多个web资源，保持连接。 请求头 请求头是客户端发送给服务器端的一些信息，使用键值对表示key：value   |  |  | | --- | --- | | 常见请求头 | 描述 （红色掌握，其他了解） | | Referer | 浏览器通知服务器，当前请求来自何处。如果是直接访问，则不会有这个头。常用于：防盗链 | | If-Modified-Since | 浏览器通知服务器，本地缓存的最后变更时间。与另一个响应头组合控制浏览器页面的缓存。 | | **Cookie** | 与会话有关技术，用于存放浏览器缓存的cookie信息。 | | **User-Agent** | 浏览器通知服务器，客户端浏览器与操作系统相关信息 | | Connection | 保持连接状态。Keep-Alive 连接中，close 已关闭 | | Host | 请求的服务器主机名 | | Content-Length | 请求体的长度 | | Content-Type | 如果是POST请求，会有这个头，默认值为application/x-www-form-urlencoded，表示请求体内容使用url编码 | | Accept： | 浏览器可支持的MIME类型。文件类型的一种描述方式。  MIME格式：大类型/小类型[;参数]  例如：  text/html ，html文件  text/css，css文件  text/javascript，js文件  image/\*，所有图片文件 | | Accept-Encoding | 浏览器通知服务器，浏览器支持的数据压缩格式。如：GZIP压缩 | | Accept-Language | 浏览器通知服务器，浏览器支持的语言。各国语言（国际化i18n） |  请求体 当请求方式是post的时，请求体会有请求的参数，格式如下：  username=zhangsan&password=123  如果请求方式为get，那么请求参数不会出现在请求体中，会拼接在url地址后面  <http://localhost:8080...?username=zhangsan&password=123> Http响应  响应行 Http协议  状态码：  常用的状态码如下：  200 ：请求成功。  302 ：请求重定向。  304 ：请求资源没有改变，访问本地缓存。  404 ：请求资源不存在。通常是用户路径编写错误，也可能是服务器资源已删除。  500 ：服务器内部错误。通常程序抛异常。  状态信息：状态信息是根据状态码变化而变化的 响应头 响应也都是键值对形式，服务器端将信息以键值对的形式返回给客户端   |  |  | | --- | --- | | 常见请求头 | 描述 | | **Location** | 指定响应的路径，需要与状态码302配合使用，完成跳转。 | | Content-Type | 响应正文的类型（MIME类型）  取值：text/html;charset=UTF-8 | | **Content-Disposition** | 通过浏览器以下载方式解析正文  取值：attachment;filename=xx.zip | | **Set-Cookie** | 与会话相关技术。服务器向浏览器写入cookie | | Content-Encoding | 服务器使用的压缩格式  取值：gzip | | Content-length | 响应正文的长度 | | Refresh | 定时刷新，格式：秒数;url=路径。url可省略，默认值为当前页。  取值：3;url=www.itcast.cn //三秒刷新页面到www.itcast.cn | | Server | 指的是服务器名称，默认值：Apache-Coyote/1.1。可以通过conf/server.xml配置进行修改。<Connector port="8080" ... server="itcast"/> | | Last-Modified | 服务器通知浏览器，文件的最后修改时间。与If-Modified-Since一起使用。 |  响应体 响应体是服务器回写给客户端的页面正文，浏览器将正文加载到内存，然后解析渲染 显示页面内容 |

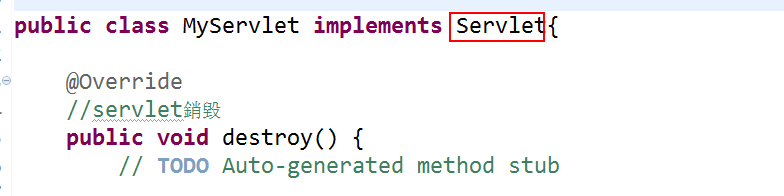
# Servlet 快速入门

什么是servlet

A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.

怎么 创建一个servlet

To implement this interface, you can write a generic servlet that extends javax.servlet.GenericServlet or an HTTP servlet that extends javax.servlet.http.HttpServlet.



怎么访问servlet

在web.xml中配置

|  |
| --- |
| <servlet>  <servlet-name>MyServlet</servlet-name>  <servlet-class>com.web.MyServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>MyServlet</servlet-name>  <url-pattern>/MyServlet</url-pattern>  </servlet-mapping> |



Servlet的生命周期

实例化 java--->.class ----->初始化 ---->对外服务....------>销毁

Init() ------> service()..多次服务 --------->destory()