# 实例servletDemoLifeCycle

知晓Web项目的创建过程和方法

理解Servlet的运行生命周期

学会创建Servlet

学会配置Servlet

学会运行服务，部署项目

学会读UML时序图

学会画UML时序图

## 1创建项目

打开eclipse开发工具

File>New>Other

|  |
| --- |
|  |

创建一个动态Web项目

在Wizards处，输入web，筛选出与Web相关的所有对象

|  |
| --- |
|  |

选中Dynamic Web Project，即将要创建的项目类型，然后Next下一步

假设创建的项目名称是servletDemoLifeCycle，在相应的位置输入这个名称，然后查看当前配置环境的一些参数。

比如项目的存放位置，目标运行环境要求，容器版本号等。

|  |
| --- |
|  |

其中，本地存放位置，可以自行在eclipse配置参数中去配置，点击Finish完成创建。

## 2观察项目

然后得到如下的目录结构的项目

|  |
| --- |
|  |

在项目名上右键，点击Go Into

|  |
| --- |
|  |

然后，就可以集中精力对付这一个刚刚创建的项目了。

|  |
| --- |
|  |

可以看到该项目的JRE版本号，如果运行或部署，相应的服务器版本号，这里看到的分别是

JREj 1.8

Tomcat 9.0

## 3创建文件

### 3.1创建html文件

首先，创建一个html文件，主要用于给出请求。

在WebContent目录上，右键，New>HTML File

|  |
| --- |
|  |

然后按提示，在相应的位置输入login.html，即将要创建的HTML文件名。

|  |
| --- |
|  |

点击Finish之后，就可以看到login.html文件出现在了项目目录下。右边的编辑器中可以看到相应的HTML文件的源代码

|  |
| --- |
|  |

下面开始编写相应的内容。

在<title></title><body></body>标签中，输入以下相应的内容。

|  |
| --- |
| **<title>Login</title>**  **<body>**  **<form action=*"/login"*>**  **用户名：<input type=*"text"* name=*"username"*> <br>**  **密　码：<input type=*"password"* name=*"password"*> <br>**  **<input type=*"submit"* value=*"登录"*>**  **</form>**  **</body>** |

输入完成以后，如下图所示：

|  |
| --- |
|  |

输入完成以后，一定要记得点击保存。

### 3.2创建xml配置文件

再创建一个xml文件，主要用于配置Serlvet。

在WEB-INF目录上点右键，然后New>Other

|  |
| --- |
|  |

在弹出的窗体中，输入xml，查找相应的xml文件的模板

|  |
| --- |
|  |

选中XML File之后，点击Next，得到如下提示界面

|  |
| --- |
|  |

最上面，是文件所在的目录位置，或是理解为项目内的位置。这是一个xml配置文件。

按约定，web.xml配置文件就在WEB-INF目录下，而且文件名必须是web.xml。输入这个文件名，然后Finish即可。

|  |
| --- |
|  |

看到web.xml文件存在于项目中后，在右边的空白处，即可开始编辑相应的配置内容。

在空白的位置输入以下内容。

|  |
| --- |
| **<web-app xmlns=*"http://java.sun.com/xml/ns/javaee"***  **xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"***  **xsi:schemaLocation=*"***  ***http://java.sun.com/xml/ns/javaee***  ***http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"***  **version=*"2.5"*>**    **<servlet>**  **<servlet-name>MyServlet</servlet-name>**  **<servlet-class>loginServlet</servlet-class>**  **</servlet>**  **<servlet-mapping>**  **<servlet-name>MyServlet</servlet-name>**  **<url-pattern>/login</url-pattern>**  **</servlet-mapping>**  **</web-app>** |

然后得到如下的结果，点击保存。

|  |
| --- |
|  |

### 3.3Servlet配置文件读解

在web.xml文件中，有如下一些节点。

<servlet>节点是对本项目中所有Servlet的配置。

每一个<servlet>节点，都会有至少一个<servlet-mapping>节点配置与之对应。

<servlet\_mapping>节点，是对应的URL映射地址的配置。

对应的方式是依赖于相同的Servlet节点名字，即<serlvet-name>。这里的具体名字就是“MyServlet”。

在<servlet>节点内，除了指定节点名字，还要指定这个Servlet将与哪一个具体的Servlet对应起来。

即要指定<servlet-class>节点里的具体的Servlet的名字，这是一个类的名字。

这里的具体的Servlet名字即为loginServlet，而这个Servlet还没有创建，但这里可以先这样完成配置。等一下就会创建它。

在<servlet-mapping>节点内，指定名字，还要指定路径。

即要指定<url-pattern>节点里的具体的映射路径。这里”/”符号，代表了项目本身的路径。

从“/”路径开始，如果没有额外加入子目录，就可以直接指定相应的访问路径了。

这种指定路径的方式，可以认为是绝对路径方式。

如果这里不指定”/”，则可以为是相对路径方式。即与当前给出某个链接URL的资源文件所在的位置保持一致。【这里，对于初学者，可能不容易理解，请在以后的示例中，多对比，多思考，多试验】。

### 3.4创建serlvet文件

再创建一个java文件，即创建一个Servlet，它本质上就是一个java类，用于实现Servlet。

这个Servet将对请求作出响应。

在src目录下，右键，New>Servlet

|  |
| --- |
|  |

在弹出的窗体中，完成以下设置和指定。

|  |
| --- |
|  |

红色打勾的，就是要思考和指定的内容。

对应的包名需要指定。现不仿指定为servlet。一般建议是以小写开头。这个是一个习惯约定。

【注意，在这里，建立了一个包！！！！】

而对于类名的指定，则要与刚才web.xml中指定的类名一至，否则，就要再去修改配置。

也就是说，这里指业Class name为LoginServet，注意，这个名称是大小写敏感的。要严格指定。

命名习惯是这样

包名一般以小写开头

类名一般以大写开头

【要特别注意截图中的小问题哟。】

然后点击Finish完成。开发环境会根据相应的Servlet的生成模板生成一个.java源文件。

|  |
| --- |
|  |

### 3.5观察Servlet类的源文件

在这个源文件中，可以看到模板生成出来的类，重点要用户来处理的部分就是doGet部分。

而这里有一点需要提出来的就是，这里，由于版本的原因，它自动生成的注解也在文件中。现在可以先将注解行删除。

暂时不用的也删除掉。整理后的文件如下。

|  |
| --- |
|  |

将在doGet处来编写代码，完成对请求的处理和响应。

请将如下代码编写到doGet函数体中。

|  |
| --- |
| **//从请求中获取相应的参数**  **String username = request.getParameter("username");**  **String password = request.getParameter("password");**  **//在控制台输出相应的参数**  **System.*out*.println("username = " + username );**  **System.*out*.println("password = " + password );**  **//给出响应**  **response.setContentType("text/html");**  **response.getWriter().println("Login Success!");** |

编写完成后，如下图所示。

|  |
| --- |
|  |

到这里，保存文件即可。

现在，可以开始运行和测试这个项目了。

## 4运行和测试

在配置好服务器以后，eclipse环境下就可以直接使用服务器，即tomcat服务器。

点击菜单Window>Show View>Servers

|  |
| --- |
|  |

然后就可以看到透视如下

|  |
| --- |
|  |

在Servet页，点击”No servers are availabe,Click this link to create a new server…”

|  |
| --- |
|  |

添加服务器，并将本项目添加到容器中。Finish之后，就可以看到服务器和服务器里的项目了。点击展开。如下。

|  |
| --- |
|  |

然后选中服务器，点击右上的启动按钮，就可以启动服务器了。

|  |
| --- |
|  |

当服务器启动完成以后，在控制台端，可以看到如下信息

|  |
| --- |
|  |

即，服务器的服务端口号和服务器启动所用的时间。

这里，用的是tomcat配置的默认的端口号，即8080，而这一次服务器启动用了9651ms。

再观察服务器信息。

|  |
| --- |
|  |

服务器已经完成启动

服务器已经完成了同步。

servletDemoLifeCycle也已经完成了同步。

下面就可以用浏览器来进行测试了。

由浏览器先访问login.html页，然后点击登录后向loginSerlvet发出请求，观察运行的情况和结果。

|  |
| --- |
|  |

在用户名和密码处输入：peter和123456

|  |
| --- |
|  |

点击登录，会得到一个访问出错的结果，如下。

|  |
| --- |
|  |

观察这个URL就可以得知，将到请求的Servlet是在servletDemoLifeCycle项目下的login

即/servletDemoLifeCycle/loign

而这里的地址栏里显示的是服务器根目录下的/login

URL地址是不对的，但参数是给出来了

### 4.1确定URL地址

调整serlvet的映射，与实际的项目中的servlet对应上。

调整login.html文件如下

|  |
| --- |
|  |

删除<form action=”/login”>节点中的“/”根标记，即使用相对于该项目下的login路径来做映射地址。

修改以后，点击保存。

### 4.2确定Servlet类和映射正确

|  |
| --- |
|  |

再一次得到出错，不过这一次不是404，而是500，是服务器内部错误。

提示是：找不到对应的类，即服务器没有找到相应的Servlet来处理该请求。

哪里错了呢？

查看web.xml文件，发现指定的Servlet类在项目下，但容器不会在项目的包下去进一步查找相应的类。

因此，这里需要指定类所在的包名。

|  |
| --- |
|  |

保存，重新启动tomcat服务器，再访问html页，提交请求。

### 4.3在调试过程中确认服务器重启

确保在调整了类和配置文件以后，要重启服务器。

然后再观察运行情况。

## 5观察结果

关闭浏览器再打开，输入对应的访问服务器的URL地址：

<http://localhost:8080/servletDemoLifeCycle/login.html>

给入相应的测试数据peter,123456

在浏览器端，得到如下的结果

|  |
| --- |
|  |

访问成功。

观察地址栏可以确认

请求地址正确

请求传递的参数正确

再观察控制台结果

|  |
| --- |
|  |

可以看到，从浏览器端，请求到达Servlet后，Servlet可以获取参数，进行处理，并给出响应。

## 6部署再运行

将项目导出为一个war包。需要记住的就是项目名和里面的访问页面。

即servletDemoLifeCycle和login.html

导出项目，可以如下进行操作

在eclipse下，回到项目浏览页，建议先停止tomcat的服务。然后在该项目上右键

Export>WAR file

|  |
| --- |
|  |

弹出如下的窗体

|  |
| --- |
|  |

浏览本地机器，在磁盘上找个位置用于导出，生成相应的文件。

|  |
| --- |
|  |

这样，在D盘的exportWars目录下，就可以看到该war文件了。

|  |
| --- |
|  |

复制该文件到tomcat服务器的相应的发布目录下。

|  |
| --- |
|  |

然后启动tomcat服务器

然后就可以在浏览器端进行访问了测试了。

在浏览器中输入：

<http://localhost:8080/servletDemoLifeCycle/login.html>

运行的结果和在IDE环境下的一样。

## 7小结

核心：

理解浏览器与服务器的交互

理解请求

理解响应

Servlet的生命周期是从创建到销毁的一个全过程，但在应用中更需要关注的是它处理请求，做出响应的过程。

### 7.1本项目运行的时序

对应的时序图分析如下

|  |
| --- |
|  |

### 7.2servlet生命周期小结

Servlet是一个运行了面向请求/ 响应服务器中的网络模块。

请求是客户的一个调用，可能是远程的。

请求包含了客户要发送给服务器的数据。

响应是服务器返回给客户的回答该请求的数据。

Servlet是一个JAVA对象，以请求为输入，分析其数据，执行一些逻辑运算，并给客户发回一个响应。

Servlet一旦开始了第一次的实例化运行，就一直驻留在内存中。

当服务器关闭时，Servlet会自动销毁。

Servlet生命周期:Servlet从创建到销毁的过程.

何时创建:用户第一次访问Servlet创建Servlet的实例（默认情况下）

可以通过在<servlet>标签中设置<load-on-startup>2</load-on-startup>让servlet随服务器启动而加载。设置一个正整数即可(不要是1)。

何时销毁:当项目从服务器中移除的时候，或者关闭服务器的时候.

用户第一次访问Servlet的时候,服务器会创建一个Servlet的实例,那么Servlet中init方法就会执行.任何一次请求服务器都会创建一个新的线程访问Servlet中的service的方法.在service方法内部根据请求的方式的不同调用doXXX的方法.(get请求调用doGet,post请求调用doPost).当Servlet中服务器中移除掉,或者关闭服务器,Servlet的实例就会被销毁,那么destroy方法就会执行

### 7.3Servlet 执行以下主要任务

• 读取客户端（浏览器）发送的显式的数据。这包括网页上的 HTML 表单，或者也可以是来自 applet 或自定义的 HTTP 客户端程序的表单。

• 读取客户端（浏览器）发送的隐式的 HTTP 请求数据。这包括 cookies、媒体类型和浏览器能理解的压缩格式等等。

• 处理数据并生成结果。这个过程可能需要访问数据库，执行 RMI 或 CORBA 调用，调用 Web 服务，或者直接计算得出对应的响应。

• 发送显式的数据（即文档）到客户端（浏览器）。该文档的格式可以是多种多样的，包括文本文件（HTML 或 XML）、二进制文件（GIF 图像）、Excel 等。

• 发送隐式的 HTTP 响应到客户端（浏览器）。这包括告诉浏览器或其他客户端被返回的文档类型（例如 HTML），设置 cookies 和缓存参数，以及其他类似的任务。

### 7.4 url-pattern配置方式

共有三种

1. 完全路径匹配 ：以 / 开始 例如: /ServletDemo4 , /aaa/ServletDemo5 , /aaa/bbb/ServletDemo6

2. 目录匹配 ：以 / 开始 需要以 \* 结束. 例如:/\* ,/aaa/\* ,/aaa/bbb/\*

3. 扩展名匹配 ：不能以 / 开始 以 \* 开始的. 例如: \*.do , \*.action

错误的写法 ：/\*.do

完全路径匹配 > 目录匹配 > 扩展名匹配

### 7.5通常Servlet的执行流程

|  |
| --- |
|  |

7.6注意事项

* 项目名，包名，类名
* 映射路径的绝对性和相对性
* 配置文件中的类的指定和路径的指定要一致对应
* 对于html的改变，需要刷新浏览器即可
* 对于java和xml的改变，需要重启服务器，重新做同步
* 对于发布新部署，一定要在全部测试通过之后再进行