**数学与信息技术学院**

**《Java Web应用开发》**

实 验 报 告

**实验项目名称：** 实验2 Servlet技术

**学 号：**  202211701140

**姓 名：**  刘葭琪

**指 导 教 师：**  宋清阁

**实验目的：**

1. 掌握Servlet配置方式（注解方式和配置文件两种方式）

2. 熟悉Servlet生命周期

3. 熟悉ServletConfig接口

4. 掌握ServletContext接口获取Web应用程序的初始化参数

5. 掌握ServletContext接口实现多个Servlet对象数据共享

6. 掌握ServletContext接口读取web应用下的资源文件

7. 熟悉HttpServletResponse应用

8. 掌握HttpServletResponse解决中文乱码

9. 掌握HttpServletRequest请求转发

10. 掌握HttpServletReques获取请求参数

11. 掌握HttpServletReques解决请求参数的中文乱码

**实验内容：**

1. 根据所学的知识，分别使用注解方式和配置文件两种方式进行Servlet配置。

2. 重写init(), service(), destroy()方法，了解其调用的时间结点。

3. 通过ServletConfig接口输出初始化参数。

4. 通过ServletContext接口获取Web应用程序的初始化参数。

5. 通过ServletContext接口实现多个Servlet对象数据共享。

6. 通过ServletContext接口读取web应用下的资源文件。

7. 创建登录HTML页面，使用Servlet处理输出登录时提交的请求与数据。

8. 使用response.setContentType("text/html; charset=utf-8");解决HttpServletResponse的中文乱码。

9. 实现HttpServletRequest请求转发。

10. 通过表单页面通过HttpServletReques获取请求参数。

11. 使用request.setCharacterEncoding("utf-8");解决HttpServletReques请求参数的中文乱码。

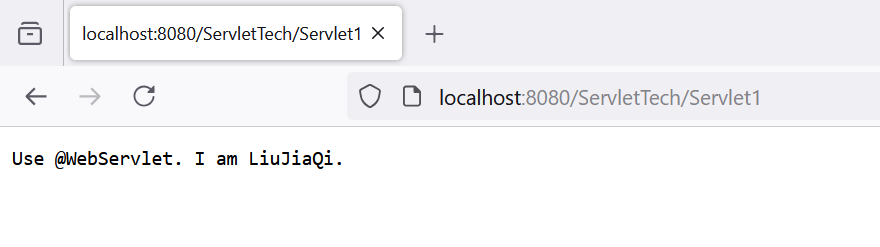
**实验步骤和结果：（在这里贴代码截图和效果截图）**

一、使用注解方式(@WebServlet)配置Servlet

**Servlet1Annotation.java**



**效果**



**解释**

配置Servlet有两种方式，分别是注解方式@WebServlet和配置文件web.xml。Servlet至少需要指定其实体名、类名、路径。当网页收到请求时，会根据映射在该网页URL路径上的Servlet类创建实体并进行响应。

注解中的name是Servlet的实体名，一般情况下与路径名相同。Servlet的名字必须唯一且不空。

在注解方式中，Servlet的类名默认为注解所在的类，类名可以与其他Servlet的类重复。如果没有指定类则会在处理网页请求时返回500状态码并抛出ServletException异常。

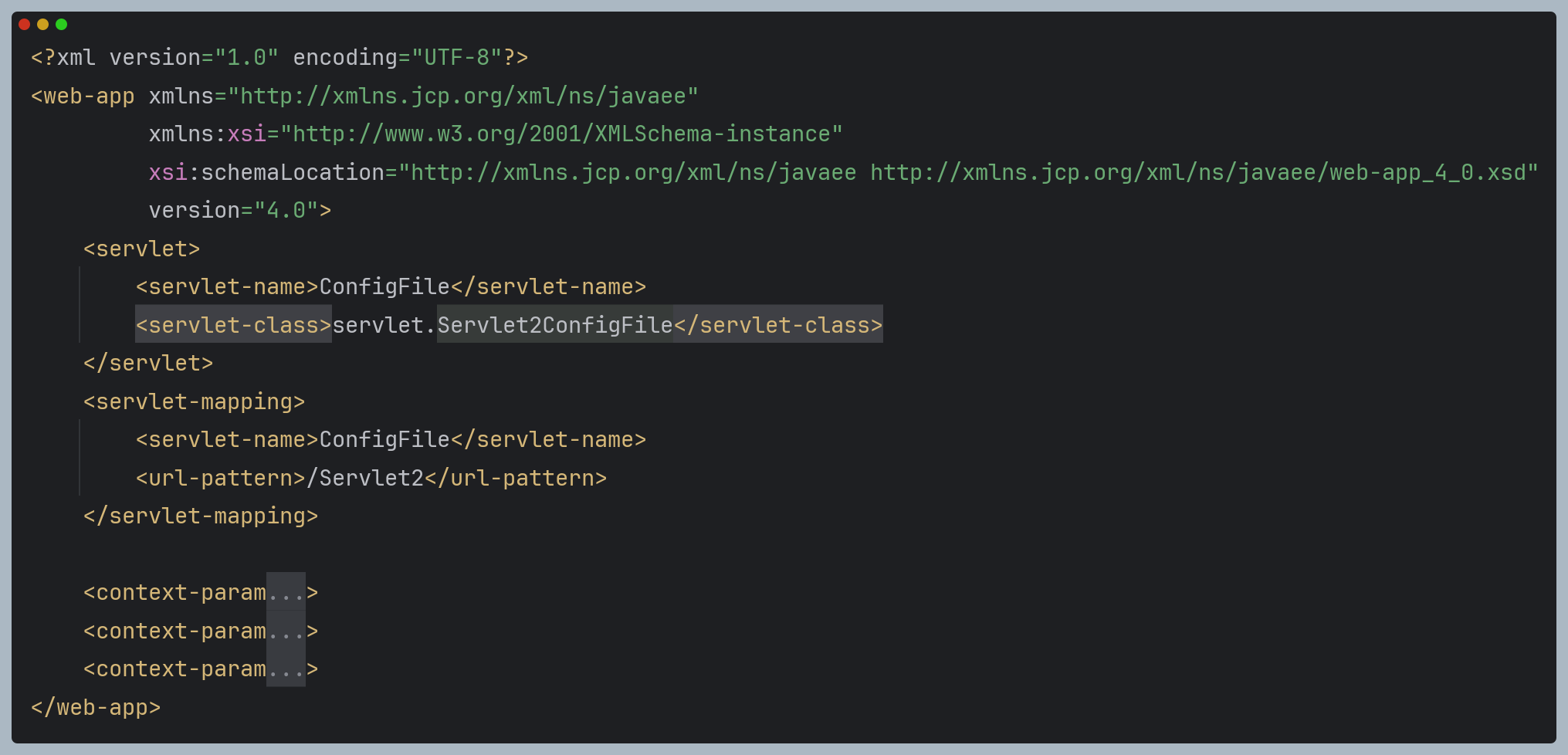
urlPatterns和value二者作用相同，都用于映射URL路径，即说明访问哪个页面会调用该Servlet。路径的根目录为Tomcat的war包所在的路径。不指定路径或两个Servlet指定了相同路径，都会在运行服务器时在控制台中抛出IllegalArgumentException异常。

二、使用配置文件(web.xml)配置Servlet

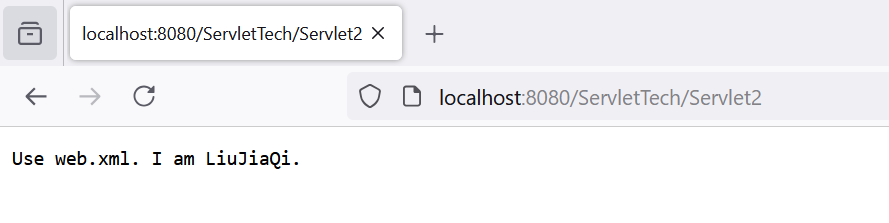
**Servlet2ConfigFile.java**



**web/WEB-INF/web.xml**



**效果**



**解释**

配置文件方式和注解方式类似，<servlet-name>对应name，<url-pattern>对应value或urlPatterns。因为配置文件不是写在类文件里的，所以类名需要使用<servlet-class>进行指定，类的根目录为src。

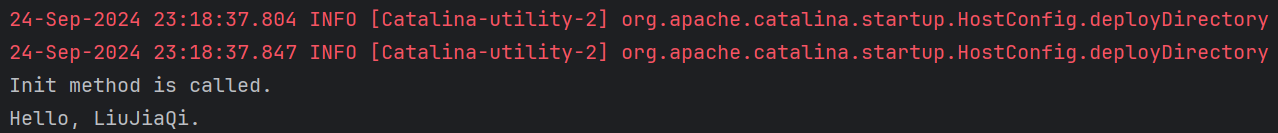
三、Servlet生命周期

**Servlet3LifeCycle.java**



**效果**

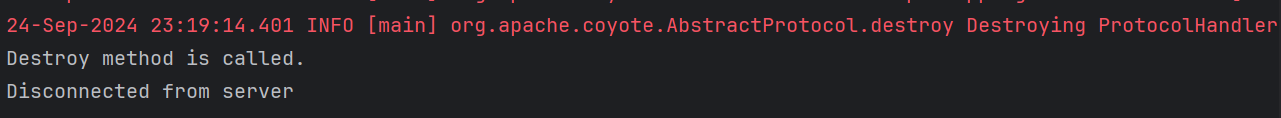
初次访问网页时会调用init()：



每次刷新网页都会调用service()：



关闭服务器时调用destroy()：



**解释**

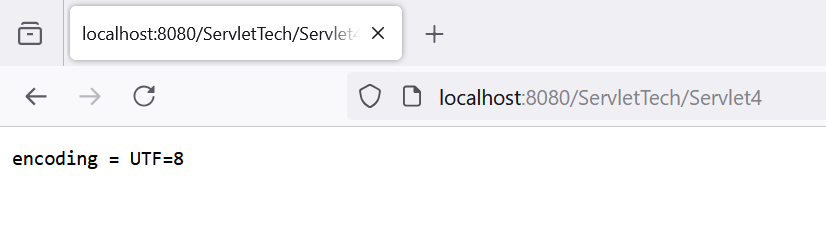
当网页被初次访问时创建一个Sevrlet实体，并调用初始化方法init()。每次访问网页（包括初次访问）都会调用service()方法。在服务器关闭或Web应用被移出Servlet容器时调用销毁方法destroy()。Servlet实体的生命周期即从init()被调用到destroy()被调用。init()和destroy()只会被调用一次，而service()会调用多次。

四、ServletConfig接口

**Servlet4Config.java**



**效果**

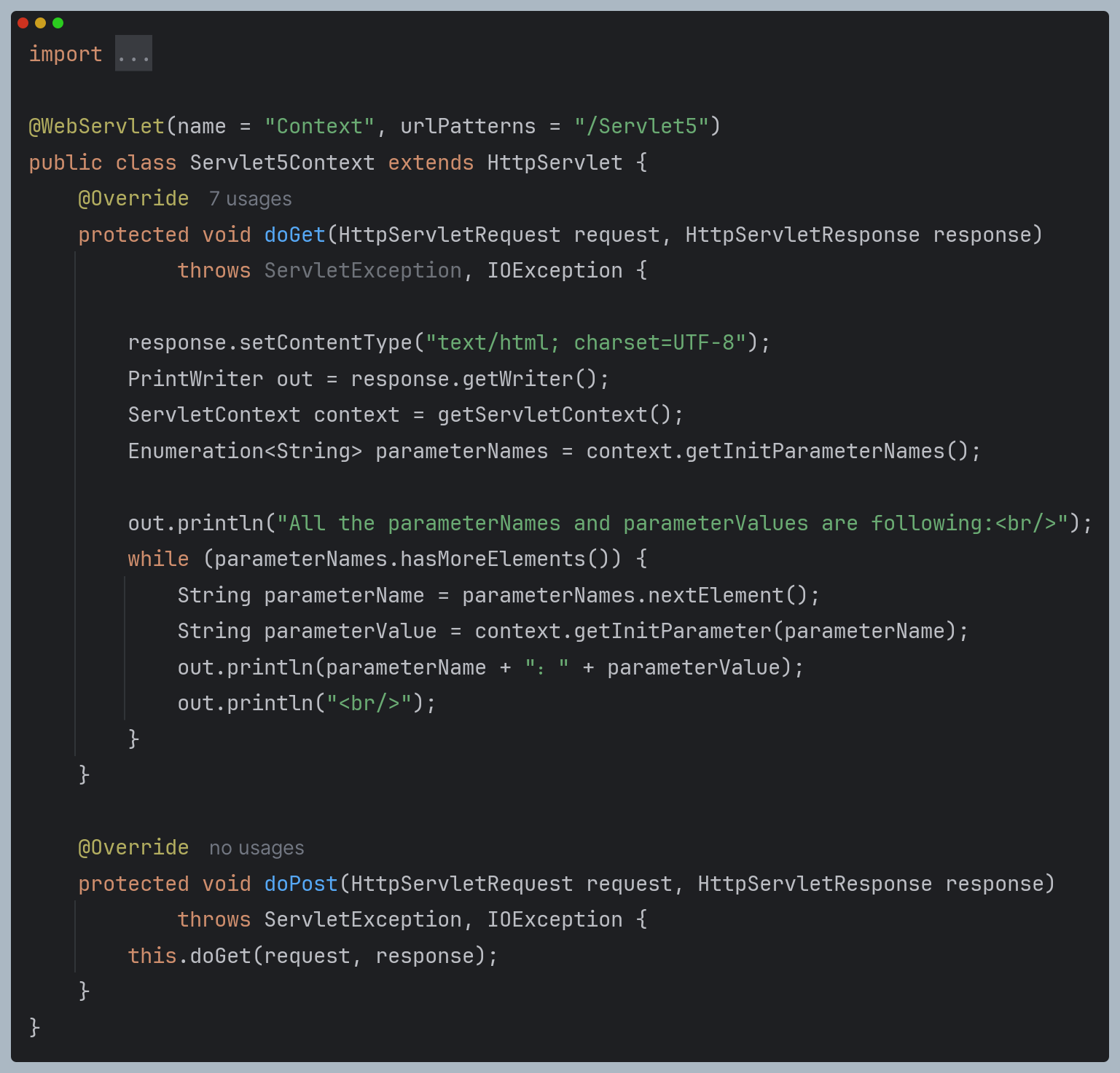


**解释**

ServletConfig类用于封装Servlet的配置信息。同样可以使用注解和配置文件两种方式配置。WebInitParam用于配置初始化参数，当容器创建实例对象时会将这些参数封装到ServletConfig中，并通过init()传递给Servlet。可以使用getServletConfig()以获取参数信息列表。getInitParameter()用于在列表中获取指定名称的初始化参数值。

五、ServletContext接口获取Web应用程序的初始化参数

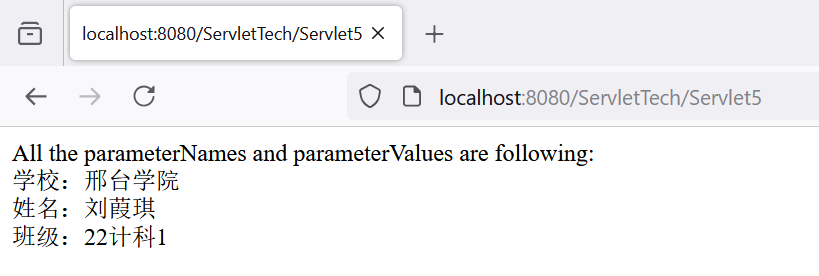
**Servlet5Context.java**



**web/WEB-INF/web.xml**



**效果**



**解释**

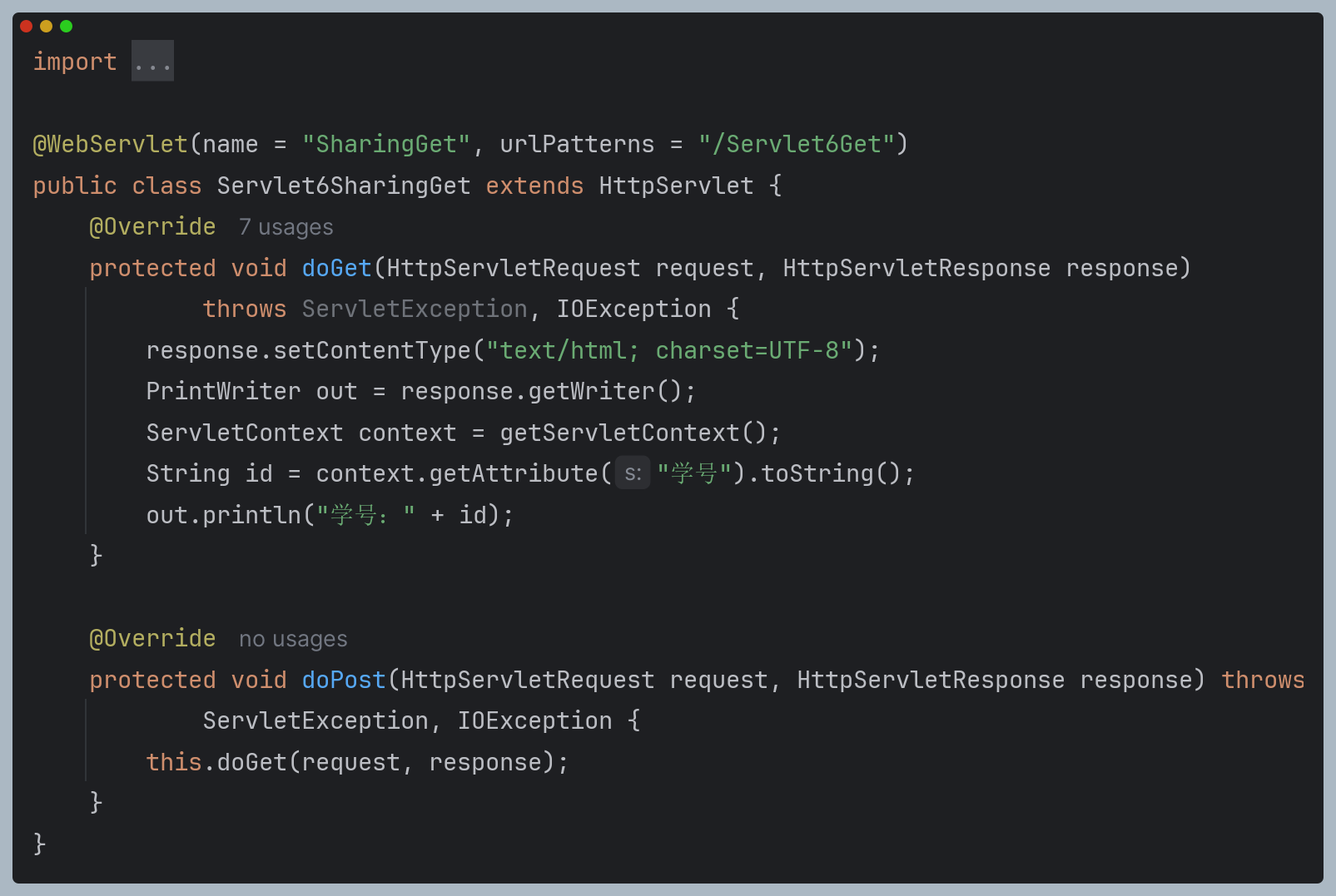
ServletContext会包含整个Web应用程序的参数，因此在不同的Servlet之间可以通过ServletContext实现数据共享，参数在web.xml中配置。每个参数(context-param)都有参数名(param-name)和参数值(param-value)两项组成。通过getServletContext()获取参数列表。通过getInitParameterNames()从参数列表中获取全部的参数名，并枚举每一个参数名，使用getInitParameter()获取与参数对应的参数值并输出。

1. ServletContext接口实现多个Servlet对象数据共享

**Servlet6SharingSet.java**

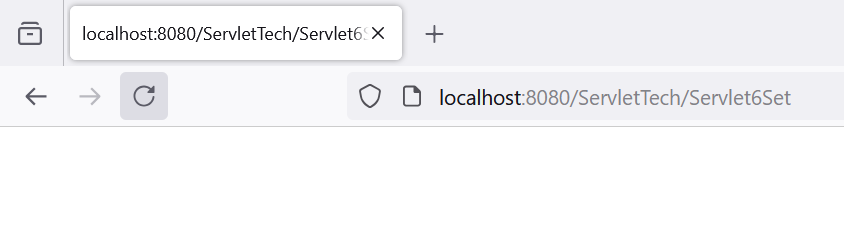


**Servlet6SharingGet.java**

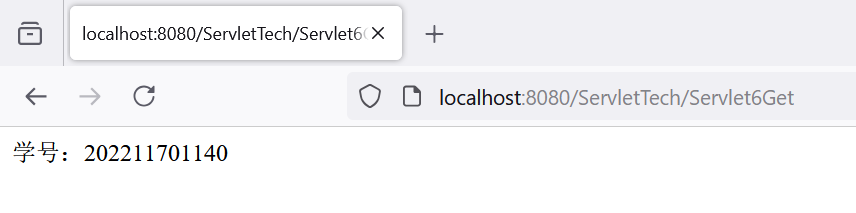


**效果**

先访问Servlet6Set添加参数信息。



再访问Servlet6Get获取上一步添加的参数值。

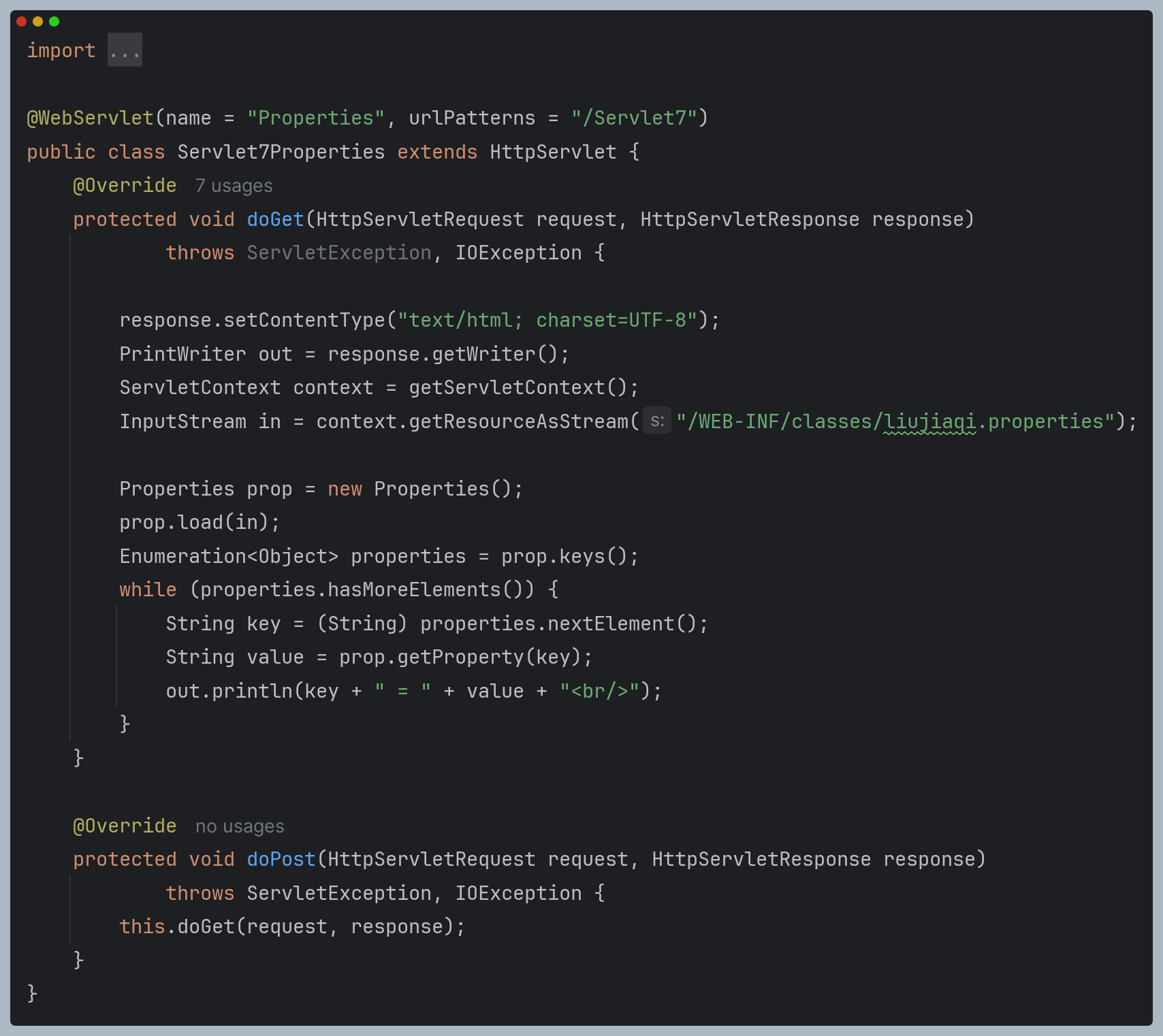


**解释**

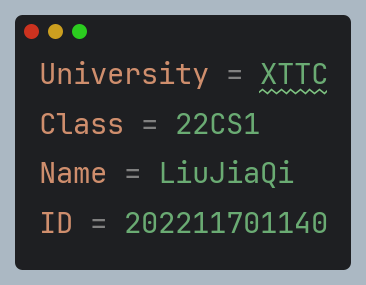
ServletContext会包含整个Web应用程序的参数，因此在不同的Servlet之间可以通过ServletContext实现数据共享。即可以在一个Servlet中添加属性，再使用另一个Servlet读取该属性参数。当然要先访问setAttribute再getAttribute，否则会因为没有设置参数而在调用getAttribute()时返回空值，而空值调用toString()则会抛出NullPointerException异常。

1. ServletContext接口读取web应用下的资源文件

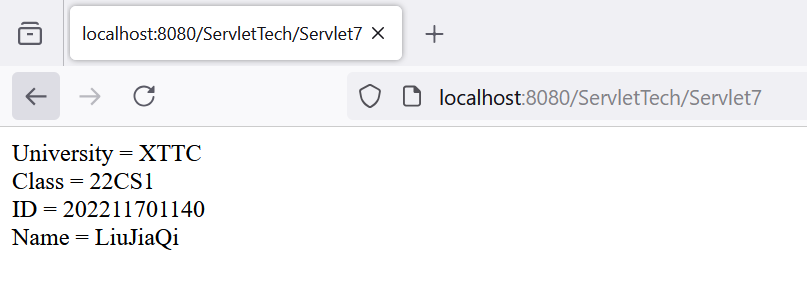
**Servlet7Properties.java**



**web/WEB-INF/classes/liujiaqi.properties**



**效果**



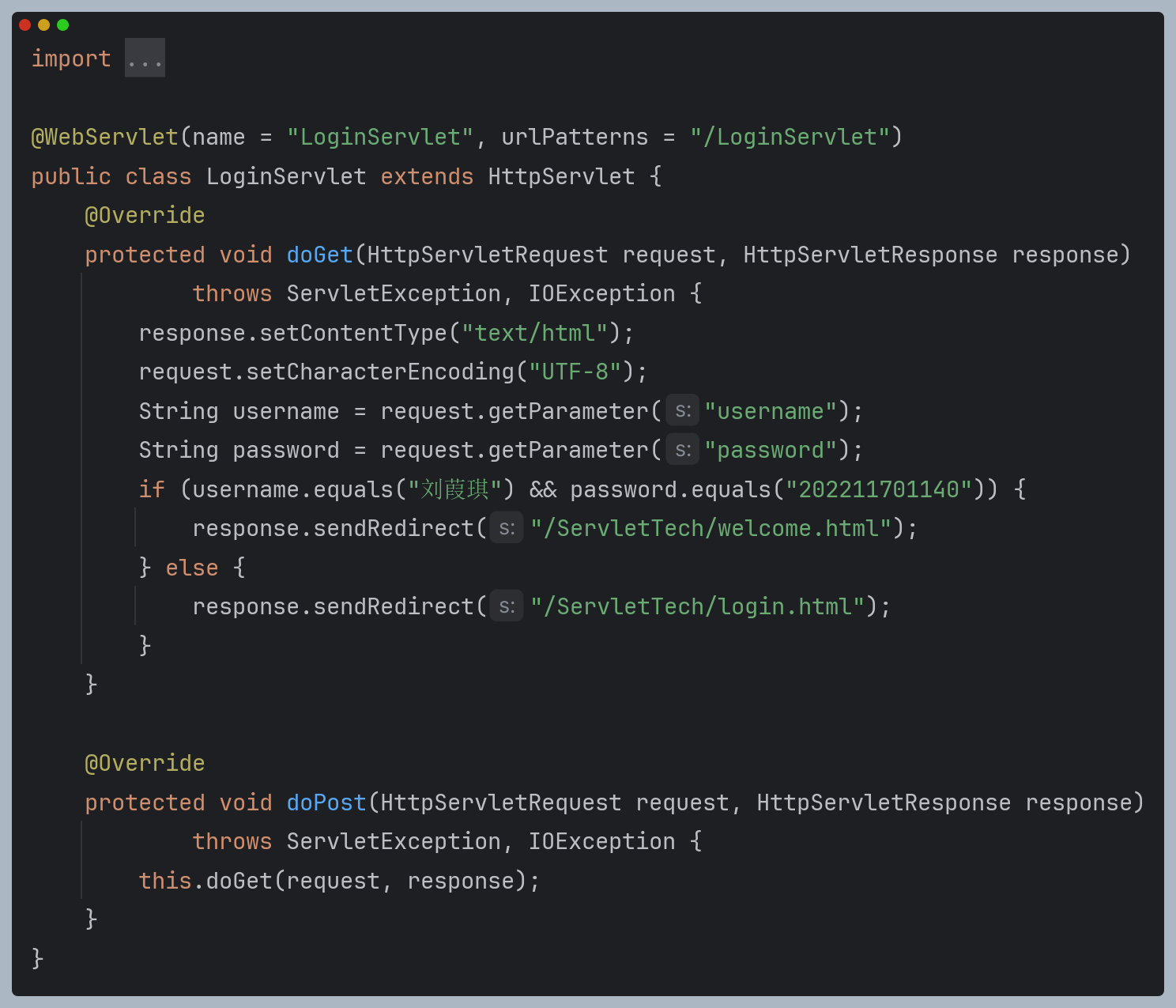
**解释**

properties是web应用下的资源文件，因此可以通过ServletContext进行访问。资源文件中的参数以key = value的形式进行设置。读取时先将资源文件转换为字节流的形式。再从流中读入文件内容。

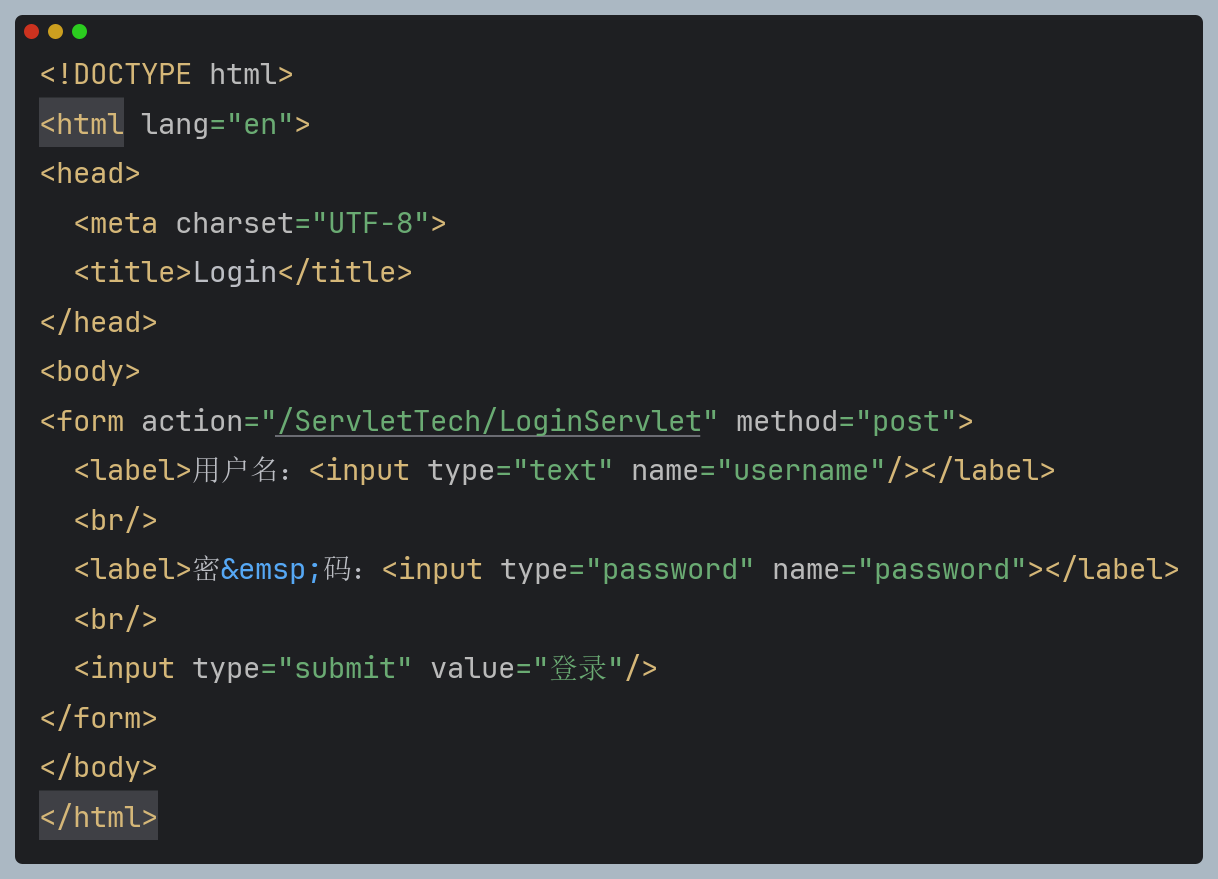
如果在properties文件中使用了中文而输出是乱码，首先检查response是否有setContentType()。其次在Idea-Settings-File Encoding中修改文件的默认编码格式为UTF-8。

1. HttpServletResponse应用

**LoginServlet.java**



**login.html**

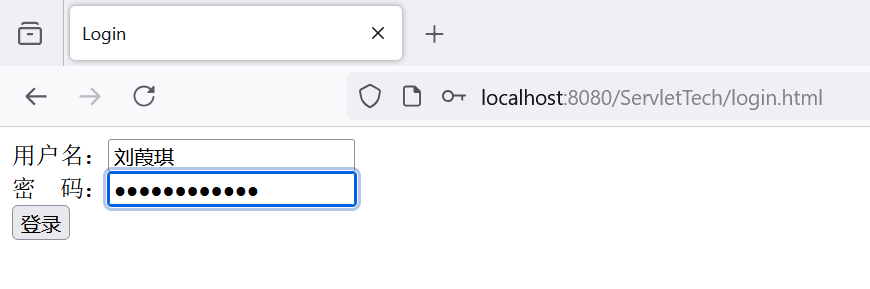


**welcome.html**

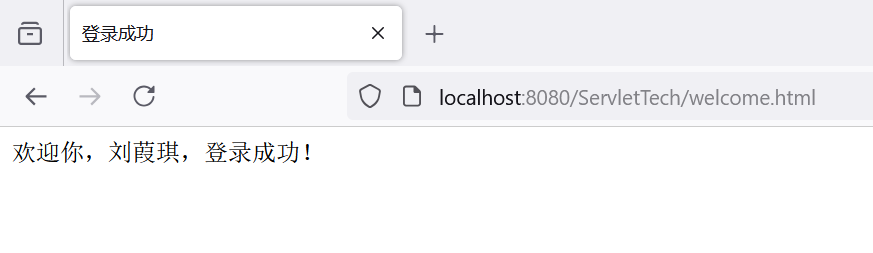


**效果**

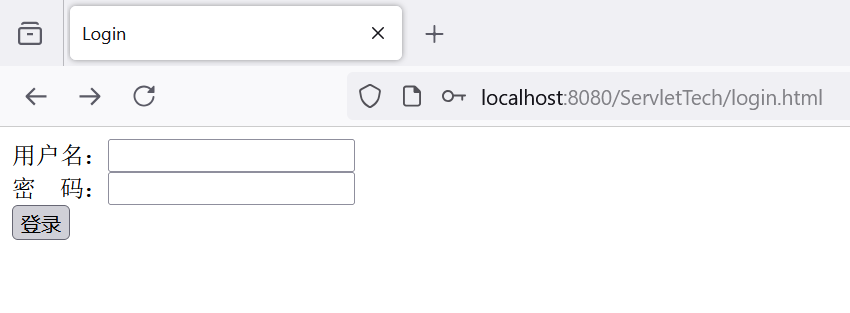
输入用户名和密码：



用户名和密码正确时会跳转到welcome.html：



用户名或密码不正确时会跳转回login.html：



**解释**

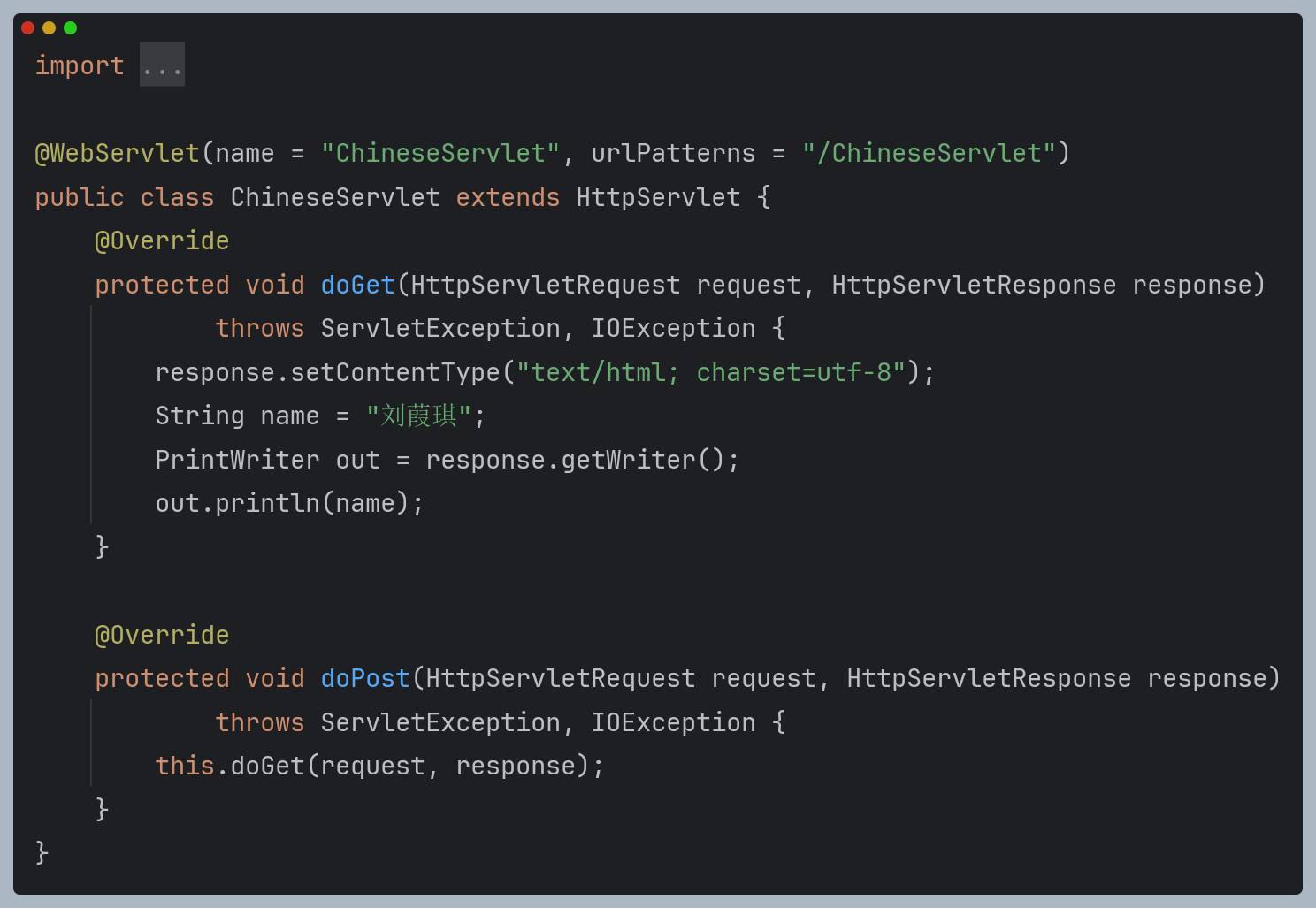
Web应用下，html文件的根目录为主机目录，即localhost:8080。而Servlet的根目录为war包路径（不一定与项目名相同），即localhost:8080/ServletTech。因此在html中访问Sevrlet时需要在路径前加上war包路径。

sendRedirect()用于重定向，括号内参数为重定向地址。重定向会在处理请求时向浏览器发送一个新的url并让浏览器访问该url。即重定向是发生在浏览器中的行为，因此当重定向到其他页面时，浏览器地址栏的url会发生改变。

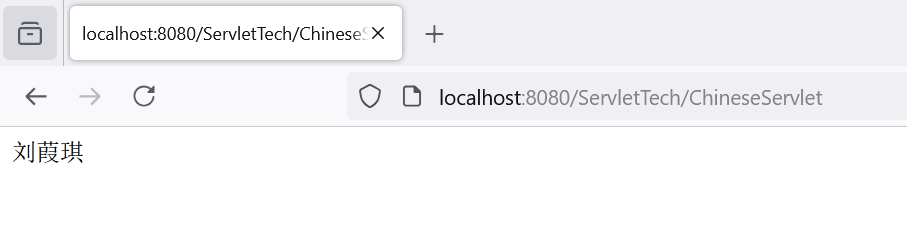
用户名输入为中文时，其作为请求数据发送到处理请求的Servlet中，因此需要指定请求数据的解码方式即setCharacterEncoding()，否则可能会因为编码方式不同而不能成功登录。

1. HttpServletResponse解决中文乱码

**ChineseServlet.java**



**效果**

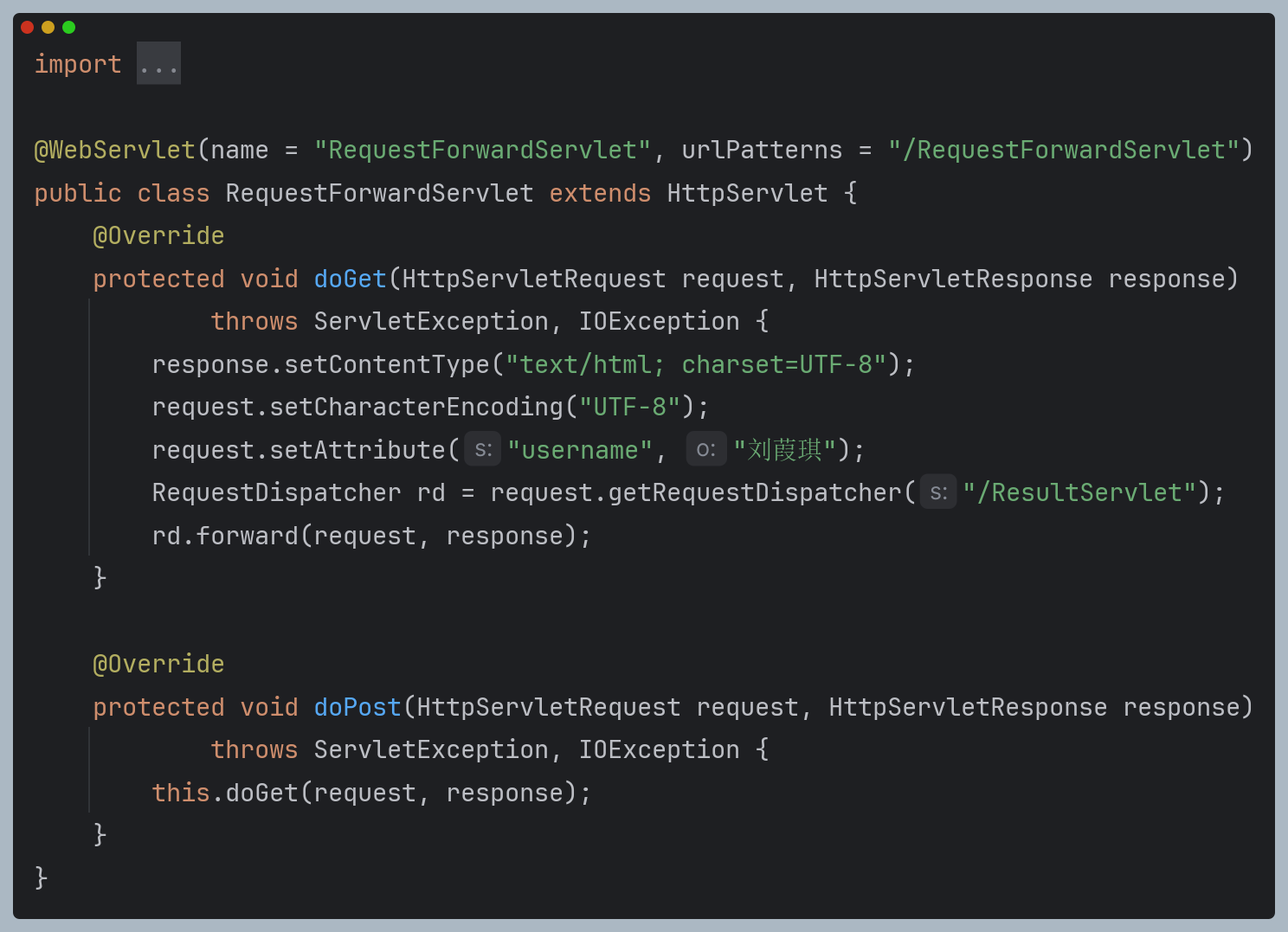


**解释**

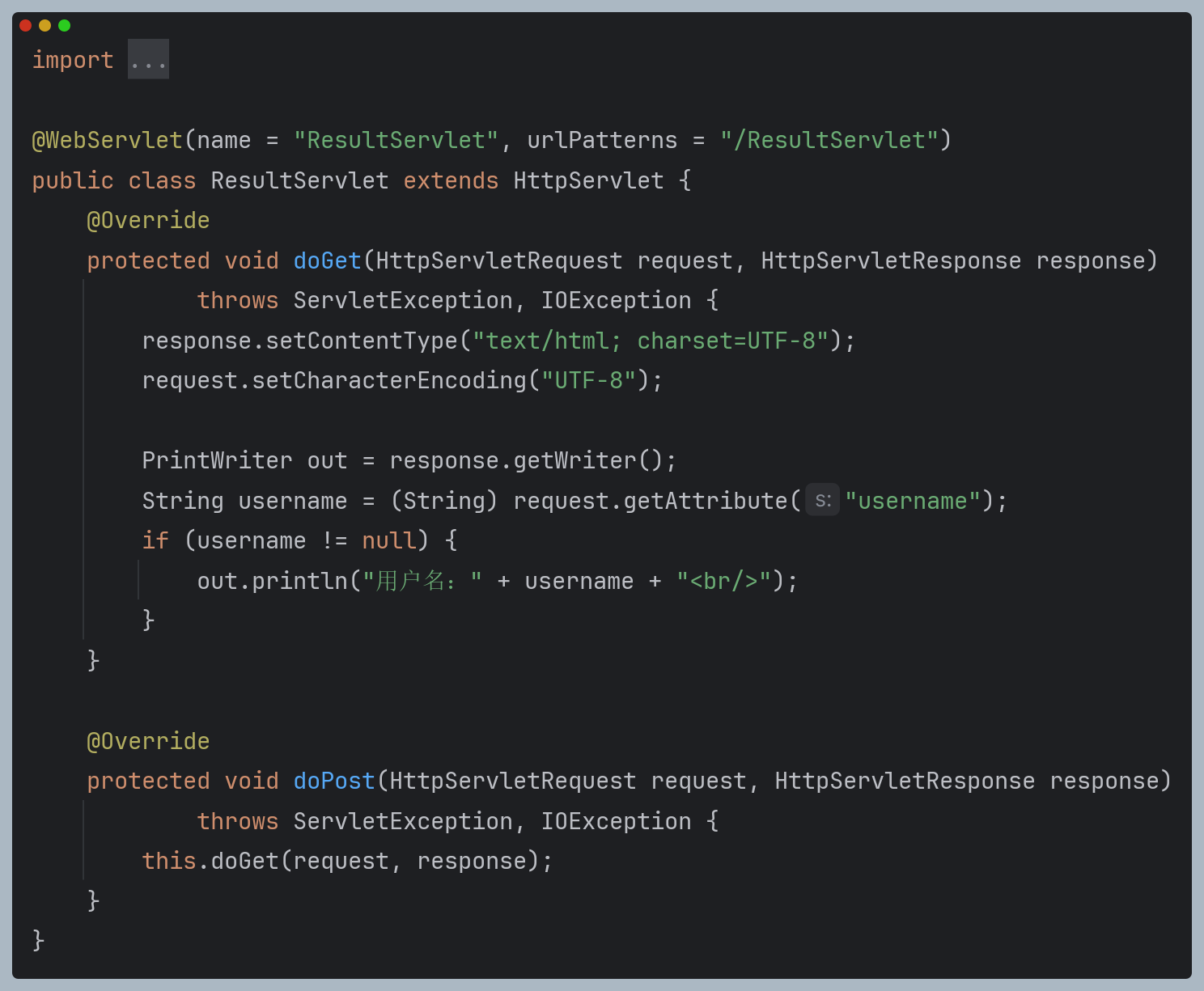
response.getWriter()会返回response的字节流，通过println在response中输出信息。因此如果出现了中文乱码，原因是response的编码格式错误，所以使用response.setContentType("text/html; charset=utf-8");更改编码格式。

1. HttpServletRequest请求转发

**RequestForwardServlet.java**

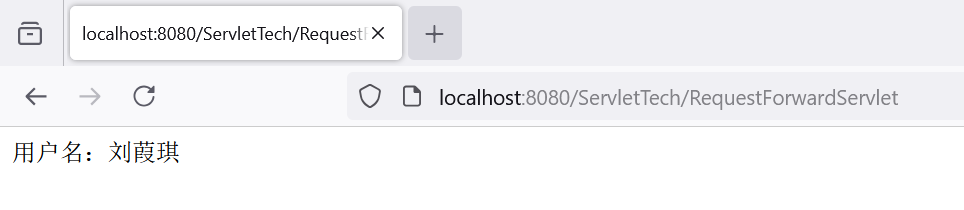


**ResultServlet.java**



**效果**

访问/RequestForwardServlet时显示出的内容来源是

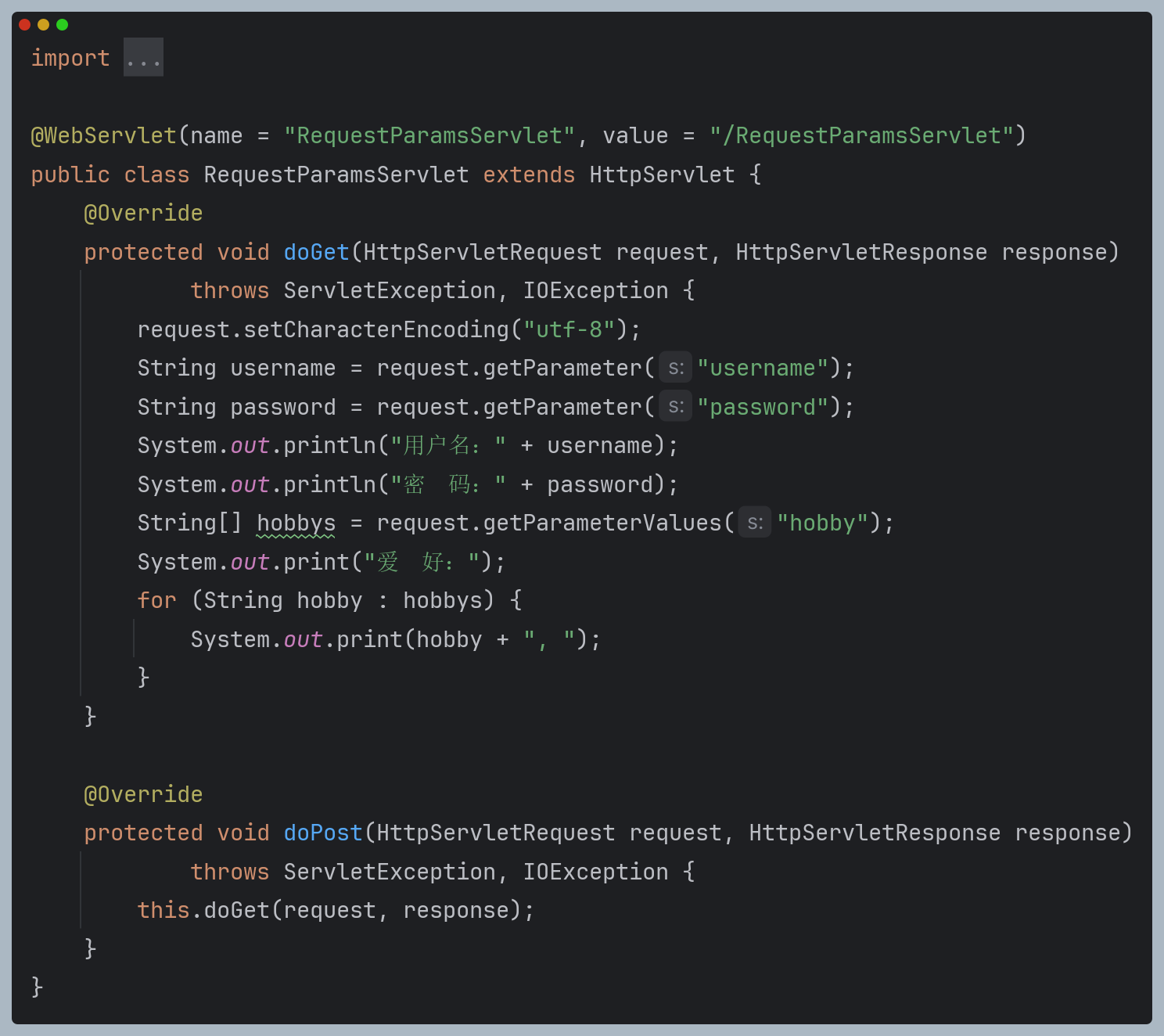


**解释**

请求转发与重定向类似，但是请求转发发生在服务器内部，与浏览器无关，因此地址栏也不会发生变动。使用getRequestDispatcher建立一个指向指定路径的实例，通过forward函数在服务器内部向该路径发送请求。最后响应会在原界面呈现出来。

1. HttpServletReques获取请求参数

**RequestParamsServlet.java**

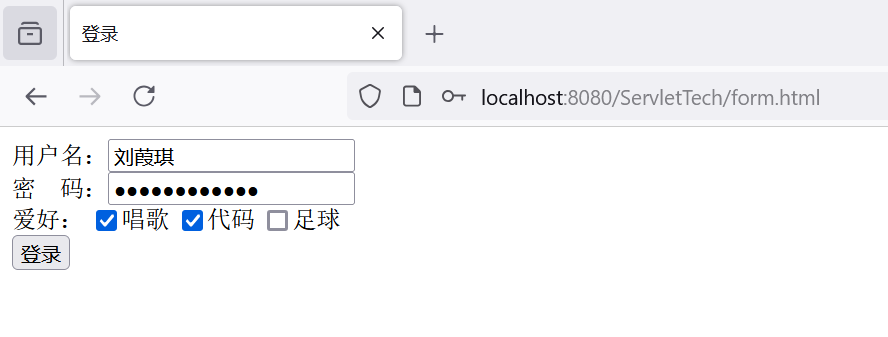


**form.html**

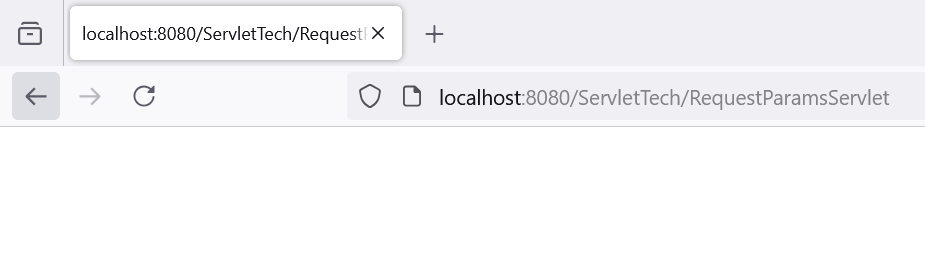


**效果**

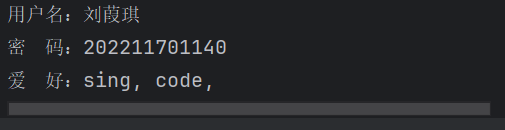
输入数据：



点击登录会会跳转到对应的Servlet路径：



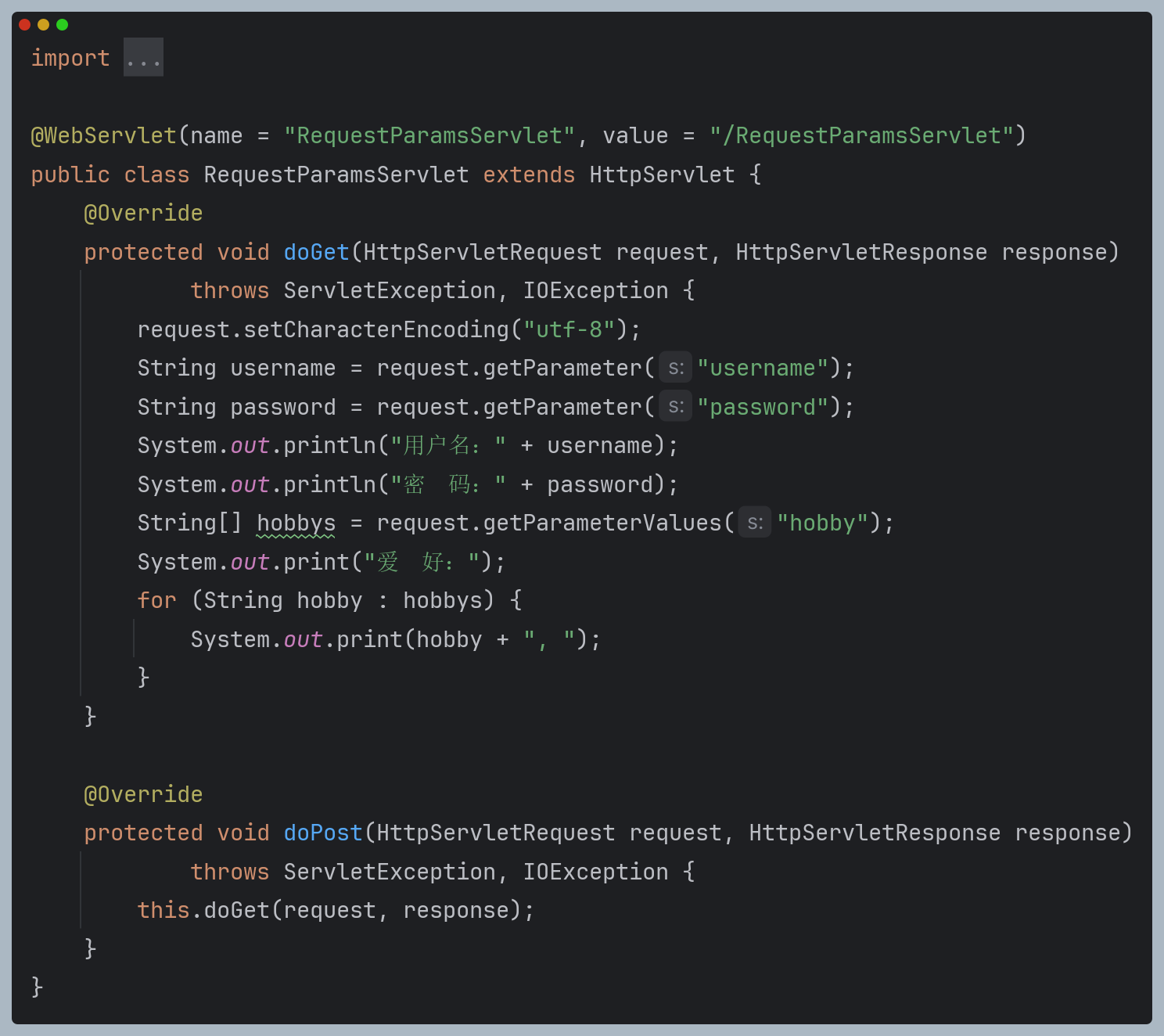
System.out在控制台输出：



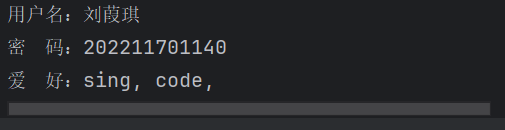
**解释**

表单<form>中，action属性表示提交时处理请求的Servlet路径。name属性为参数名，value为参数值。多选框的参数名相同，而value不同。提交后可通过getParamter()获取具有单个值的参数名对应的参数值，而有多个值时（比如出现多选框时，则需要使用getParameterValues()获取，返回值类型为字符串数组String[]。

十二、HttpServletReques解决请求参数的中文乱码



**效果**



**解释**

从html中提交的数据编码格式为utf-8，因此处理请求并输出时，请求的解码方式需要保持一致，因此添加request.setCharacterEncoding("utf-8");。

**实验总结：（写出自己的心得或收获）**

1. 配置Tomcat服务器的过程。
2. 官网下载Tomcat安装包，到bin目录中通过cmd启动startup.bat以开启服务，访问localhost:8080检查能否正常运行。
3. 新建Java项目（不是Maven项目）。
4. 选中项目的根目录，右键选择Add Framework Support...，选择Web Application(4.0)，OK。
5. 右上角Current File处下拉菜单（或去菜单栏中选择Run），选择Edit Configurations...。点击Add new...（或左上角的加号），选择Tomcat Server - Local。
6. 初次添加时需要在Application Server右边的Configure...处指定Tomcat路径。随后选择Deployment - Add - Artifact新建war包。路径在下方Application context处更改。OK。
7. 打开File - Project Structure，选择Modules - Dependencies - Add，选择“2 Library”，选中Tomcat 9.0.94，Add Selected。勾选Tomcat前的选框，OK。
8. 右键添加Sevrlet模板。打开File | Settings | Editor | File and Code Templates，在Other中选择Web - Java code templates - Servlet Class.java或Servlet Annotation Class.java。将内容复制并在Files中新建模板，Name改为Servlet并将内容粘贴进去。其中JAVAEE\_TYPE输入除了"jakarta"之外的任何字符串都可以，也可以为空。Class\_Name为类名。如果使用为Servlet Annotation Class.java，则会自动生成注解方式用于配置Servlet，因此需要指定实体名即Entity\_Name，路径默认为 "/实体名"，一般实体名与类名相同。
9. 配置Tomcat服务器过程中出现的问题。
10. bin目录下startup.bat闪退。去官网重新下载安装包，老师给的安装包似乎与某些系统不兼容（无论是x86还是x64的）。
11. 右键没有Add Framework Support，idea2023及以上版本没有，在选中根目录的情况下按两下Shift搜索Add Framework Support。
12. Tomcat已配置，且访问也没问题的情况下却始终找不到Tomcat Server - Local选项；或添加Library后Tomcat路径路径正确但是jar包的路径错误，最简单的解决方法是把idea卸载重装。
13. Add-Artifact没有Artifact选项，可能是因为没有添加Web Application框架。
14. 通过模板新建Servlet类后，所有与Servlet相关类标红并报错找不到，可能是因为没有导入Tomcat的jar包。如果String类标红并报错找不到是jdk错误（比如在机房电脑写的文件直接在自己电脑上打开，而二者jdk版本不同），在Project Structure-SDK中删除旧jdk并重新添加jdk。
15. 实验过程中出现的问题。
16. Servlet访问404，原因忘记配置@WebServlet或web.xml。从login跳转到Servlet返回404，原因同上或form action的路径错误，html中根目录基于主机目录而Servlet基于Tomcat路径，Tomcat路径不一定等于项目名称，而是Edit Configuration...中的URL。
17. 启动Tomcat时，在控制台中报错Error during artifact deployment.原因Servlet名字不合法，比如在使用Servlet Annotation Class.java建立Servlet类，但是没有指定Entity\_Name，也没有补充注解内容，此时name = ""即为不合法。
18. properties文件中出现中文，且已指定响应的编码格式为utf-8依旧乱码。Settings - File Encodings，将下方Default encoding for properties files改为UTF-8，并将properties文件删除重建。
19. html中内容改变，但是即使重启了Tomcat，通过浏览器访问时查询源代码内容还是更改前的内容。比如直接从外部将文件复制到项目中会出现这种情况。解决方法将旧文件删除并新建一个同名文件。
20. 一系列的中文乱码问题。
21. 响应中输出的中文乱码。添加response.setContentType("text/html; charset=utf-8");
22. 接收中文请求并在响应中输出该请求时乱码。request.setCharacterEncoding("utf-8");
23. 读取含有中文的properties文件并在响应中输出时乱码。Settings - File Encodings，将下方Default encoding for properties files改为UTF-8，并将properties文件删除重建。
24. Tomcat控制台红字部分乱码。打开Tomcat安装目录下conf/logging.properties，将第51行java.util.logging.ConsoleHandler.encoding设置为GBK。
25. Tomcat控制台白字部分使用System.out输出乱码，但是在网页端的所有内容和在控制台输出的请求信息没有乱码。经过测试可以选择将str转成GBK格式，即String after = new String(str.getBytes("utf-8"), "GBK");在网页输出时输出utf-8格式的str，在控制台输出时输出GBK格式的after。正解的解决方法是，首先确保红字部分没有乱码。之后打开File Encodings，将Global Encoding和Project Encoding设置为UTF-8。在Edit Configurations - VM options中添加-Dfile.encoding=utf-8，等于号后的编码格式需要与logging.properties中的控制台编码格式相同，其他编码格式需要统一为UTF-8。
26. idea中的可设置的编码格式有全局编码Global Encoding，项目编码Project Encoding，单独适用于properties文件的编码，这三个尽量都设置成utf-8。web应用中需要设置请求的解码格式request.setCharacterEncoding和响应的编码格式response.setContentType均设置为utf-8。
27. 在idea中进行web应用开发会涉及到3个控制台，且3个控制台的编码格式相互独立。idea本身的控制台基于idea虚拟机，即不添加任何框架时调用System.out函数。tomcat控制台显示时，启动时的红字基于本机控制台，运行时的白字基于tomcat虚拟机。idea控制台的格式在Help - Edit Custom VM options中修改。红字格式在logging.properties中修改，白字在Edit Configuration - VM options中修改。