day29-request

学习目标

1. 能够使用Request对象获取HTTP协议请求内容
2. 能够处理HTTP请求参数的乱码问题
3. 能够使用Request域对象
4. 能够使用Request对象做请求转发
5. 能够完成登录案例

案例一:完成网站的登录案例

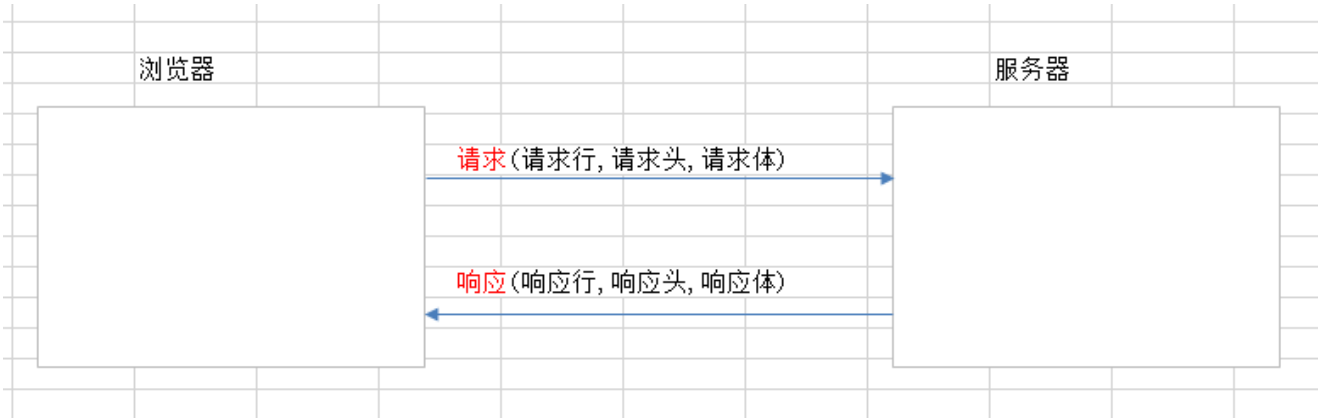
一,案例需求

- 点击登录按钮, 进行登录.
- 登录成功,显示login Success
- 登录失败,显示login failed

二,技术分析

1,request对象的基本概念

在Servlet API中，定义了一个HttpServletRequest接口，它继承自ServletRequest接口，专门用来封装HTTP请求消息。由于HTTP请求消息分为请求行、请求头和请求体三部分，因此，在HttpServletRequest接口中定义了获取请求行、请求头和请求消息体的相关方法.



法名	描述

Web服务器收到客户端的http请求，会针对每一次请求，分别创建一个用于代表请求的request对象、和代表响应的response对象。

2.request操作请求三部分

2.1获取客户机信息(操作请求行)

请求方式 请求路径(URI) 协议版本

POST /day17Request/WEB01/register.htm?username=zs&password=123456 HTTP/1.1

- **getMethod()**;获取请求方式
- **getRemoteAddr()** ：获取客户机的IP地址
- **getContextPath()**;获得当前应用工程名;
- getRequestURI();获得请求地址，不带主机名
- getRequestURL()；获得请求地址，带主机名
- getServerPort()；获得服务端的端口
- getQueryString()；获的请求参数(get请求的,URL的?后面的. eg:username=zs&password=12345)

2.2.获得请求头信息(操作请求头)

long	**getDateHeader**(String name) Returns the value of the specified request header as a long value that represents a Date object.
String	**getHeader**(String name) Returns the value of the specified request header as a String.
Enumeration	**getHeaderNames**() Returns an enumeration of all the header names this request contains.
Enumeration	**getHeaders**(String name) Returns all the values of the specified request header as an Enumeration of String objects.
int	**getIntHeader**(String name) Returns the value of the specified request header as an int.

getHeader(String name);

- User-Agent: 浏览器信息
- Referer:来自哪个网站(防盗链)

2.3接受请求参数(操作请求体)

2.3.1相关的API

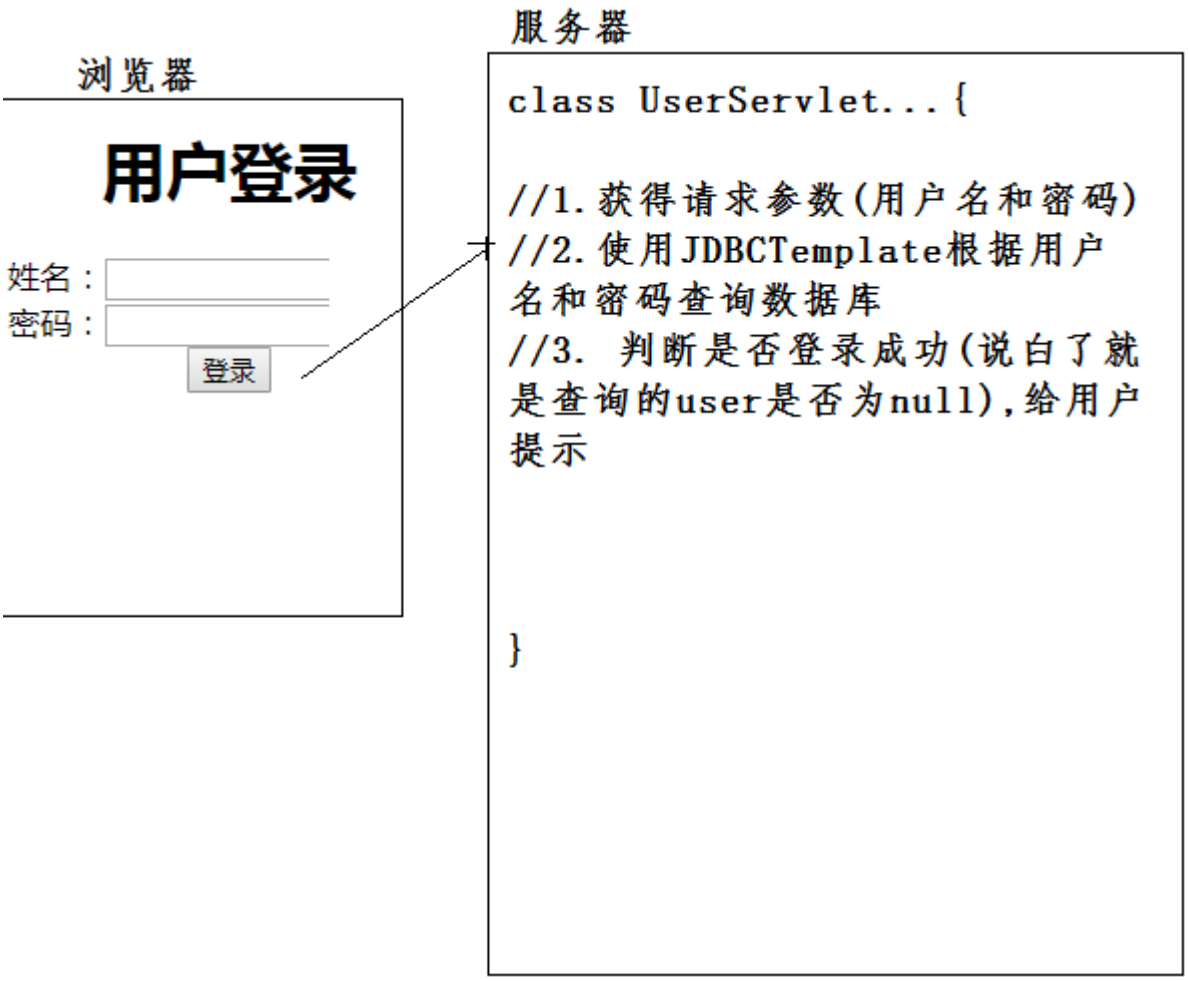
法名	描述
String getParameter(String name)	获得指定参数名对应的值。如果没有则返回null，如果有多个获得第一个。 例如：username=jack
String[] getParameterValues(String name)	获得指定参数名对应的所有的值。此方法专业为复选框提供的。 例如：hobby=抽烟&hobby=喝酒
Map getParameterMap()	获得所有的请求参数。key为参数名,value为key对应的所有的值。

2.3.2使用BeanUtils封装

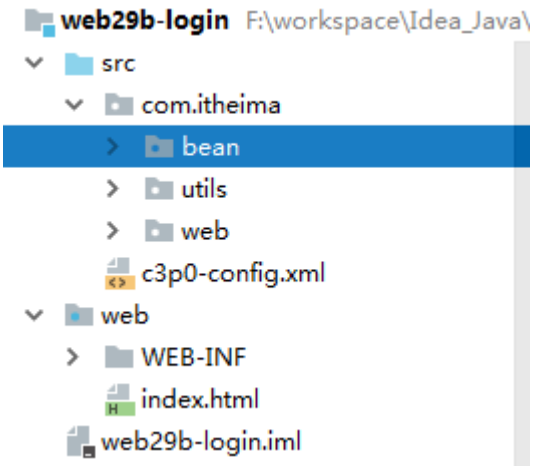
现在我们已经可以使用request对象来获取请求参数，但是，如果参数过多，我们就需要将数据封装到对象。以前封装数据的时候，实体类有多少个字段，我们就需要手动编码调用多少次setXXX方法，因此，我们需要BeanUtils来解决这个问题。

1. 设置一个登录页面准备提交表单数据（username、password）
2. 导入BeanUtils相关jar包
3. 创建Servlet获取请求参数
4. 调用BeanUtils.populate方法封装数据

三,思路分析



四,代码实现



- 页面的准备

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body >
    <center>
        <h1>用户登录</h1>
        <form action="" method="post">
            姓名: <input type="text" name="username" /><br/>
            密码: <input type="password" name="password"/><br/>
            <input type="submit" value="登录"/>
        </form>
    </center>

</body>
</html>
```

- 数据库的创建

```
create database web28;
use web28;
create table t_user(
    id int primary key auto_increment,
    username varchar(20),
    password varchar(20)
);
```

- JavaBean

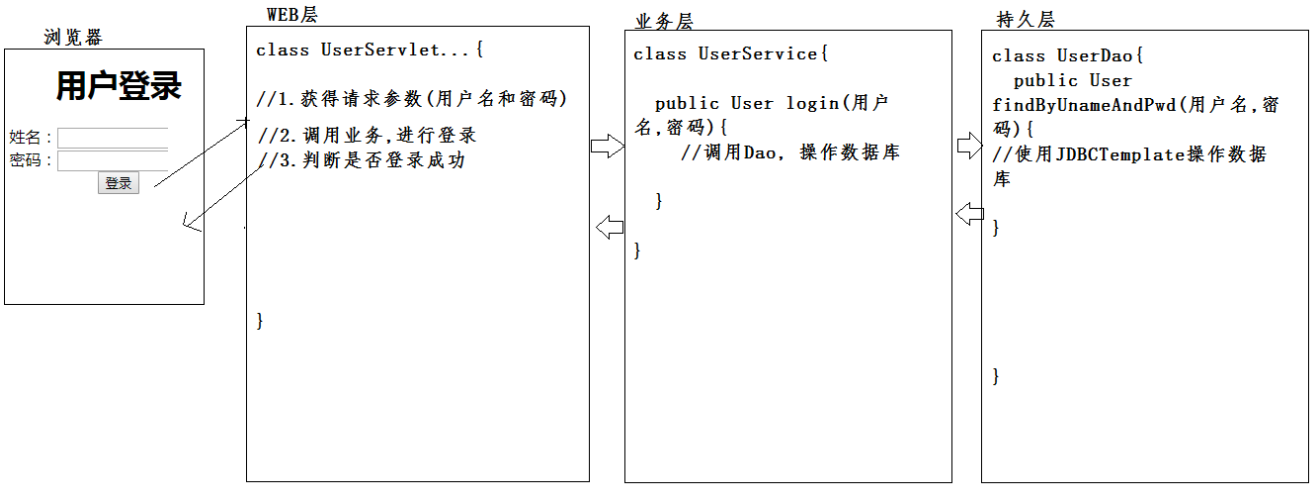
```
public class User {
    private int id;
    private String username;
    private String password;
    //set/get

}
```

- UserServlet

```
//Ctrl+Alt+T
@WebServlet("/userServlet")
public class UserServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        try {
            //1. 获得请求参数(用户名和密码)
            String username = request.getParameter("username");
            String password = request.getParameter("password");

            //2. 使用JDBCTemplate根据用户名和密码查询数据库
            //a. 创建JDBCTemplate对象
            JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Utils.getDataSource());
            //b. 编写sql语句, 执行
            String sql = "SELECT * FROM t_user WHERE username = ? AND password = ?";
            User user = jdbcTemplate.queryForObject(sql, new BeanPropertyRowMapper<>(User.class),
username, password);

            //3. 判断是否为null
            if(user != null){
                response.getWriter().print("Login Success");
            }else{
                response.getWriter().print("Login Failed");
            }
        } catch (Exception e) {

            e.printStackTrace();
            response.getWriter().print("Login Failed");
        }


    }

    protected void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        doGet(request,response);
    }



}
```

五,使用三层架构来改写登录案例

1.改造思路



2.代码实现

- WEB层; UserServlet

```
@WebServlet("/userServlet")
public class UserServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        try {
            //1. 获得请求参数(用户名和密码)
            String username = request.getParameter("username");
            String password = request.getParameter("password");

            //2.调用业务, 进行登录逻辑
            UserService userService = new UserService();
            User user =  userService.login(username,password);
            //3. 判断是否为null
            if(user != null){
                response.getWriter().print("Login Success");
            }else{
                response.getWriter().print("Login Failed");
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().print("Login Failed");
        }


    }

    protected void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        doGet(request,response);
    }


}
```

- 业务层(UserService)

```
public class UserService {

    public User login(String username, String password) throws Exception {
        //编写业务逻辑的代码
        //调用Dao
        UserDao userDao = new UserDao();
        User user =  userDao.findByUnameAndPwd(username,password);
        return  user;
    }
}
```

- 持久层(UserDao)

```
public class UserDao {

    public User findByUnameAndPwd(String username, String password) throws Exception {

        JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Utils.getDataSource());
        String sql = "select * from t_user where username = ? and password = ?";
        User user = jdbcTemplate.queryForObject(sql, new BeanPropertyRowMapper<>(User.class),
username, password);
        return  user;
    }
}
```

六,request总结

1.转发

```
request.getRequestDispatcher(url).forward(request, response);  //转发
```

转发和重定向区别:

- 转发是一次请求，重定向是二次请求
- 转发地址栏路径不变，重定向地址栏路径改变了
- 转发写跳转路径的时候，不需要加工程名；重定向需要加工程名
- request域对象存取的值在转发(一次请求)中是有效的,在重定向(两次请求)无效的

2.作为域对象存取值

ServletContext: 范围 整个应用

request范围: 一次请求有效

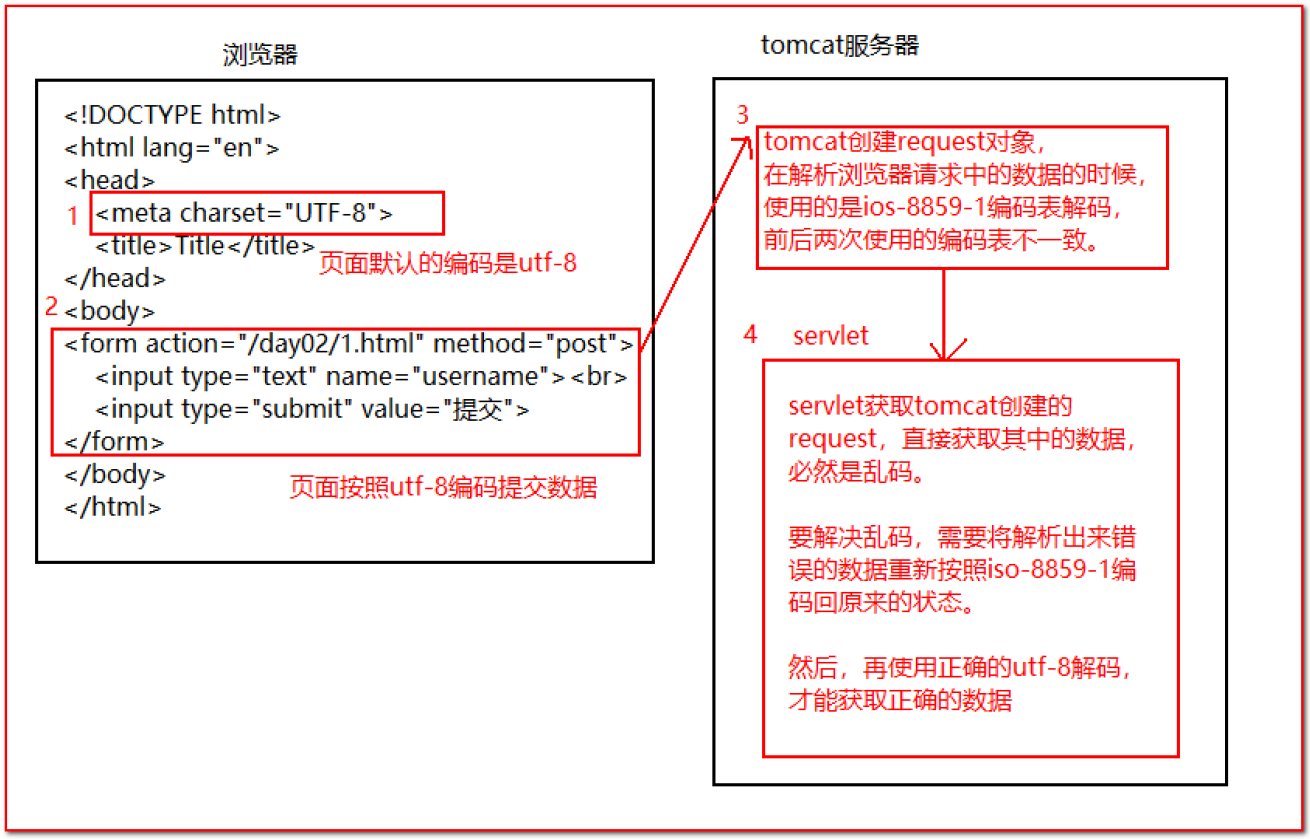
域对象是一个容器，这种容器主要用于Servlet与Servlet/JSP之间的数据传输使用的。

- Object getAttribute(String name) ;
- void setAttribute(String name,Object object) ;
- void removeAttribute(String name) ;

3.请求乱码解决

3.1 请求参数乱码的由来

我们在输入一些中文数据提交给服务器的时候，服务器解析显示出来的一堆无意义的字符，就是乱码。那么这个乱码是如何出现的呢？如下图所示：



3.2 乱码解决

```
void setCharacterEncoding(String env); //设置请求体的编码
```

3.3乱码总结

3.3.1 为什么出现乱码?

编码和解码不一致(iso8859-1不支持中文的)

3.3.2乱码解决

- 响应乱码

```
response.setContentType("text/html;charset=utf-8");
//1. 设置服务器编码为utf-8
//2. 告诉浏览器以utf-8解码
```

- 请求参数乱码

```
get方式不需要处理的(tomcat8之后已经处理了)
post方式,请求参数在请求体里面
request.setCharacterEncoding("utf-8");
```

七, 生成验证码

- 导入jar ValidateCode.jar
- CodeServlet

```
@WebServlet("/codeServlet")
public class CodeServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        //1. 创建验证码对象
        ValidateCode validateCode = new ValidateCode(100, 40, 4, 10);
        //2. 依赖response响应出去
        validateCode.write(response.getOutputStream());

    }

    protected void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
IOException {
        doGet(request,response);
    }


}
```

- 页面

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body >
    <center>
        <h1>用户登录</h1>
        <form action="http://localhost:8080/userServlet" method="post">
            姓名：<input type="text" name="username" /><br/>
            密码：<input type="password" name="password"/><br/>
            验证码：<input type="text" name="code"/><br/><img onclick="changeImg(this)"
src="http://localhost:8080/web29d-code/codeServlet"/>
            <br/>
            <input type="submit" value="登录"/>
        </form>
    </center>

</body>

<script>
    function changeImg(obj) {
        // 改变src的值(如果是同一个路径,读缓存去了) 忽悠浏览器把路径写成不一样的
        obj.src = "http://localhost:8080/web29d-code/codeServlet?a="+new
Date().getMilliseconds();

    }

</script>
</html>
```