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

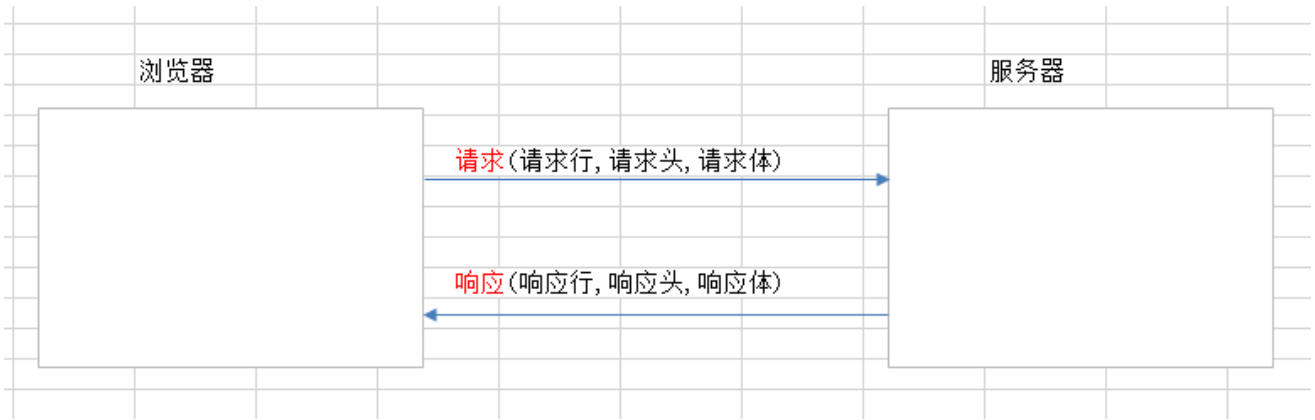
一,案例需求

- 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　　HttpServletRequest

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.



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

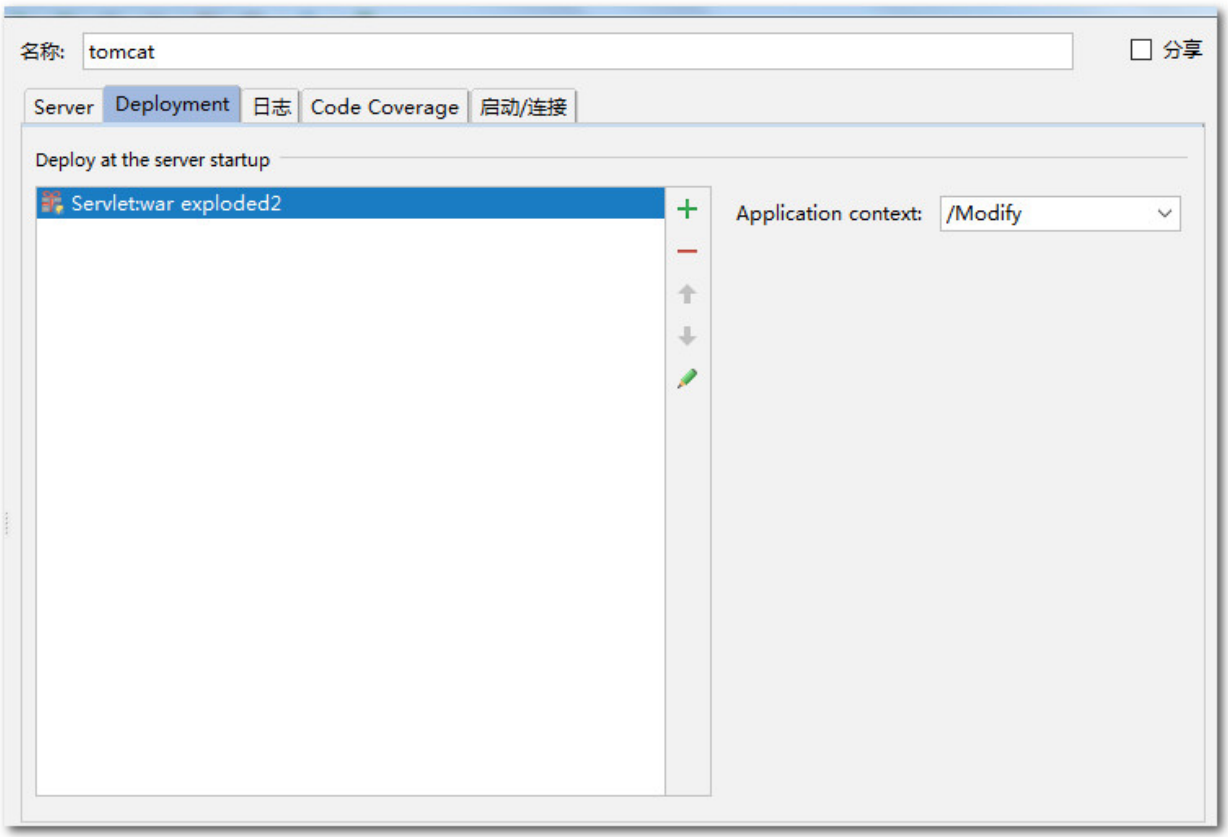
2.1获取客户机信息(操作请求行)

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

  将路径更改后



  html

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>response响应行</div>
<a href="http://localhost/Modify/q">操作</a><br/>
<form action="http://localhost/Modify/q"method="post">
    <input type="submit"value="123"/>
```

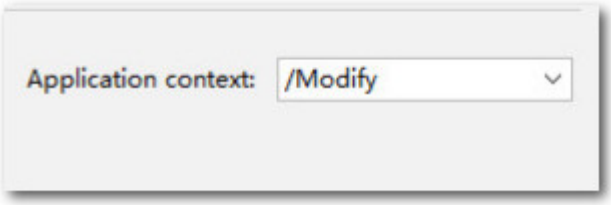
```
</form>
<img src="img/1.jpg"width="100%"/>
</body>
</html>
```

java

```
@WebServlet("/q")
public class ServletQ extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println(request.getMethod());
        System.out.println("address="+request.getRemoteAddr());
        System.out.println("工程名"+request.getContextPath());
        System.out.println(request.getRequestURI());
        System.out.println(request.getRequestURL());
        System.out.println(request.getServerPort());
        System.out.println(request.getQueryString());
    }
}
```

- **getRemoteAddr()** ；获取客户机的IP地址 来自父类**ServletRequest**的方法
- **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 来自父类**ServletRequest**的方法
- 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)

`getHeader(String name)`;

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

```
<!DOCT<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>Title</title>
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>response响应行</div>
<a href="http://localhost/Modify/q">操作</a><br/>
<form action="http://localhost/Modify/q"method="post">
    <input type="submit"value="123"/>
</form>
<img src="img/1.jpg"width="100%"/>
</body>
</html>
```

```
@WebServlet("/q")
public class ServletQ extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println(request.getHeader("User-Agent"));
        System.out.println(request.getHeader("Referer"));
        if(!request.getHeader("Referer").contains("localhost")){
            return;
        }
    }
}
```

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)	来自父类**ServletRequest**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)	来自父类**ServletRequest**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<String,String[]> getParameterMap()	来自父类**ServletRequest**获得所有的请求参数。key为参数名,value为key对应的所有的值。

```
<!DOCT<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div>response响应行</div>
<form action="http://localhost/Modify/q"method="post">
    用户名<input type="text"name="username"/><br/>
    密码<input type="password"name="psw"/><br/>
    爱好<br/>
    <input type="checkbox"name="bas"value="basket">篮球<br/>
    <input type="checkbox"name="bas"value="ali">奥迪<br/>
    <input type="checkbox"name="bas"value="teng">奔驰<br/>
    <input type="submit"value="提交">
</form>
</body>
</html>
```

```
@WebServlet("/q")
public class ServletQ extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println(request.getParameter("username"));
        System.out.println(request.getParameter("psw"));
        String hobis=Arrays.toString(request.getParameterValues("bas"));
        System.out.println(hobis.substring(1,hobis.length()-1));
        Map<String, String[]> map = request.getParameterMap();
        Set<String> set = map.keySet();
        for (String s : set) {
            String[] values = map.get(s);
            System.out.println(s+":"+Arrays.toString(values));
        }
    }
}
```

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

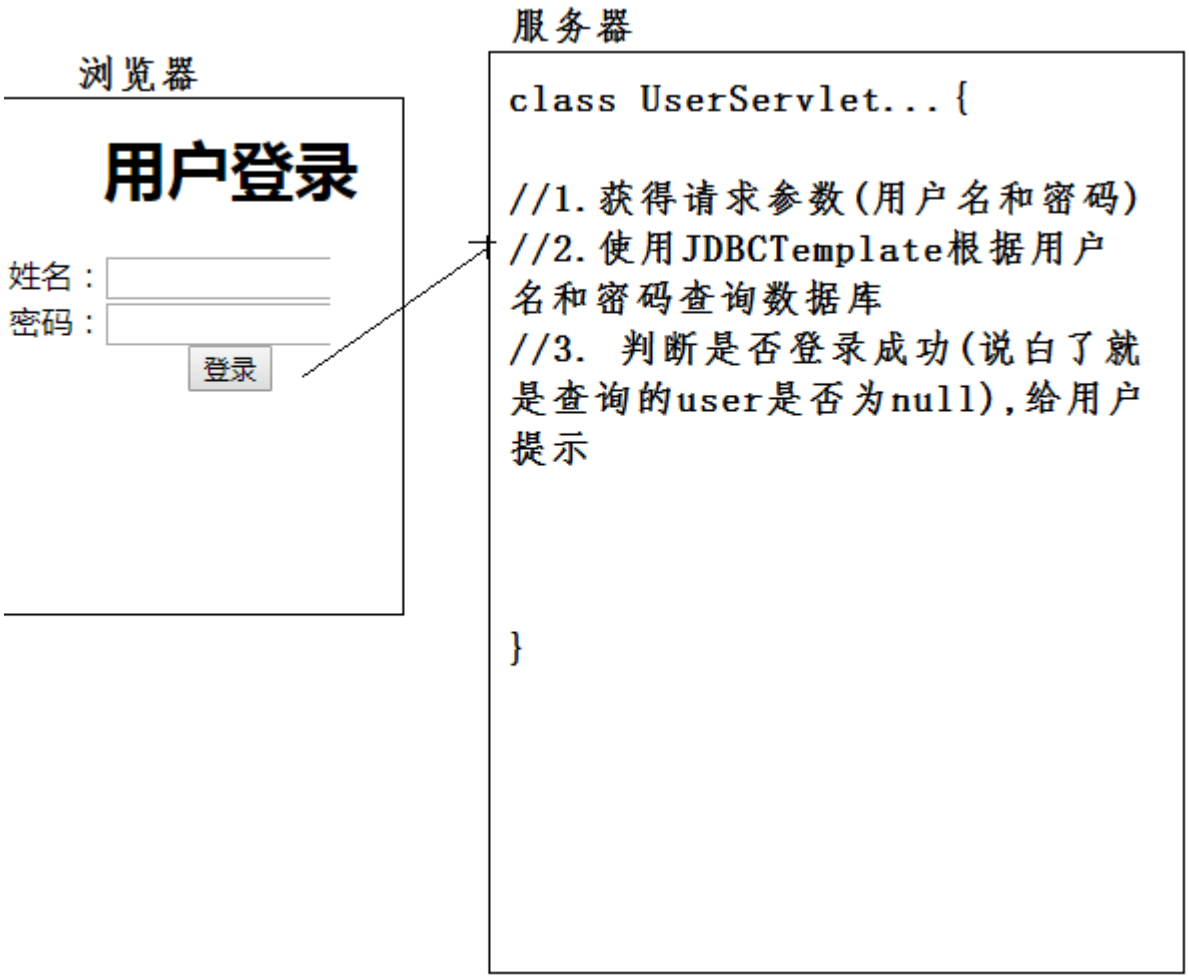
3. 创建Servlet获取请求参数
4. 调用BeanUtils.populate方法封装数据

```
public class User {
    String username;
    String psw;
    String[]bas;
    构造set()/get()
    toString()
}
```

```
@WebServlet("/q")
public class ServletQ extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)  {
        Map<String, String[]> map = request.getParameterMap();
        User user=new User();
        try {
            BeanUtils.populate(user,map);
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println(user);
    }
}
```

三,思路分析



四,代码实现

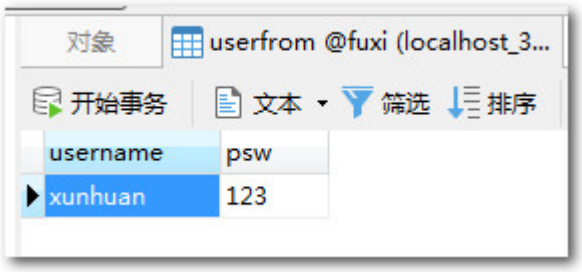
- 页面的准备

```
<!DOCT<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>用户登录</title>
    <style>
        .log{
            margin-top: 100px;
        }
    </style>
</head>
<body align="center">
<div class="log">
登录
<form action="http://localhost/Modify/Sel"method="post"><br/>
    用户名<input type="text"name="username"/><br/><br/>
    密码<input type="password"name="psw"/><br/><br/>
    <input type="submit"value="提交"><br/><br/>

</form>
```

```
</div>
</body>
</html>
```

- 数据库的创建



- JavaBean

```
public class User {
    private String username;
    private String psw;
    构造set/get
}
```

- UserServlet

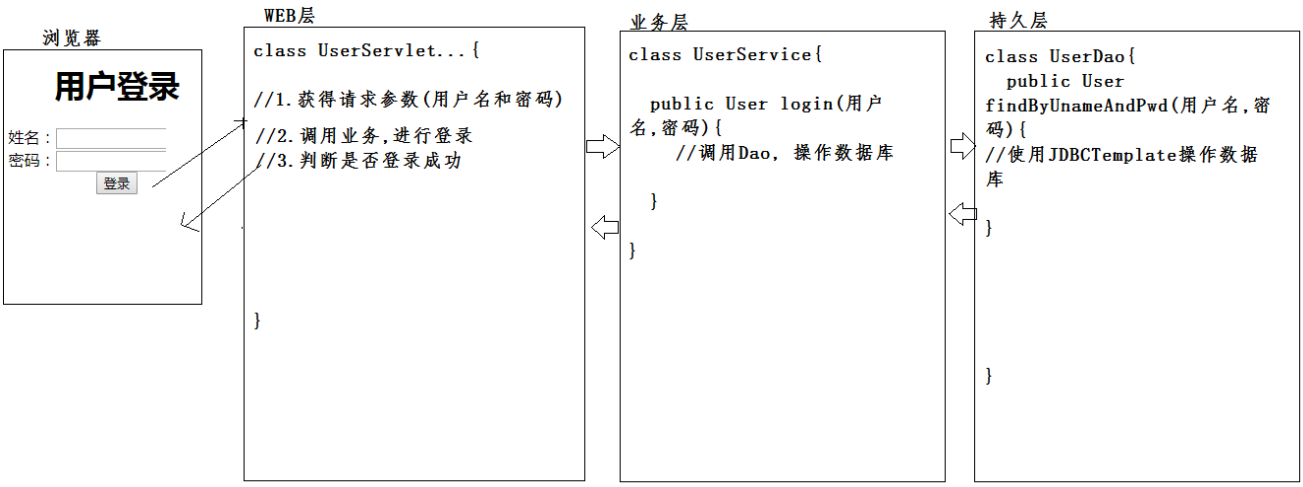
```
@WebServlet("/Sel")
public class ServletSelect extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        try {
            String username = request.getParameter("username");
            String psw = request.getParameter("psw");
            JdbcTemplate jt=new JdbcTemplate(C3P0Utiles.getDataSource());
            User newUser=jt.queryForObject("select * from userfrom where username=? and psw=
?",new BeanPropertyRowMapper<>(User.class),username,psw);
            if(newUser!=null){
                response.getWriter().print("Login");

            }else{
                response.getWriter().print("Default");
            }
        } catch (Exception e) {
            response.getWriter().print("Default");
            System.out.println("账号密码不一致");
        }
    }
}
```

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("/Sel")
public class ServletSelect extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            String username = request.getParameter("username");
            String psw = request.getParameter("psw");
        SelectServlet ss=new SelectServlet();
        User newUser = ss.getUser(username, psw);
        if(newUser!=null){
                response.getWriter().print("Login");
            }else{
                response.getWriter().print("Default");
            }
    }
}
```

- 业务层(UserService)

```
public class SelectServlet {
    public static User getUser(String username,String psw) {
        Dao dao=new Dao();
        return dao.login(username,psw);
    }
}
```

- 持久层(UserDao)

```
public class Dao {
    public User login(String username,String psw){
        User newUser= null;
        try {
            JdbcTemplate jt=new JdbcTemplate(C3P0Utiles.getDataSource());
            newUser = jt.queryForObject("select * from userfrom where username=? and psw= ?",new
BeanPropertyRowMapper<>(User.class),username,psw);
        } catch (DataAccessException e) {
            System.out.println("账号密码不一致");
        }
        return newUser;
    }
}
```

六,request总结

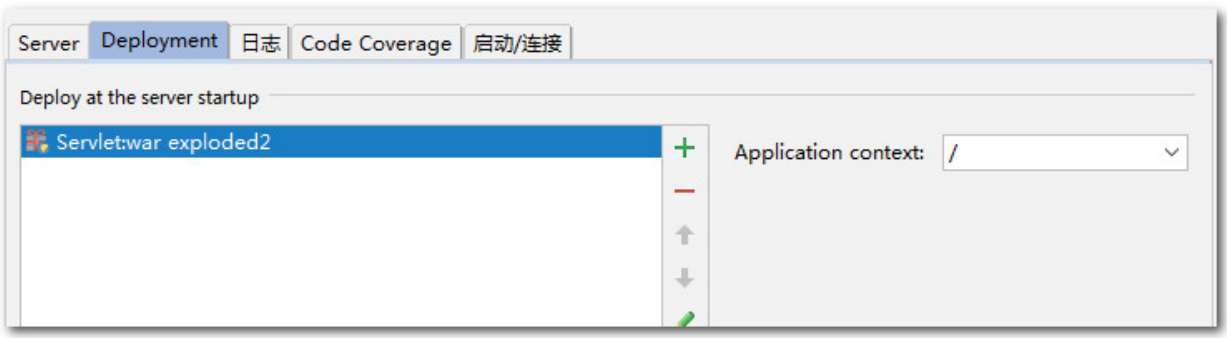
1.转发

转发对象**getRequestDispatcher** 来自于父类 **RequestDispatcher**类有个方法forward()

```
request.getRequestDispatcher(url).forward(request, response);  //转发  url为要跳转的内部路径
```

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



```
<!DOCT<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>转发</title>
</head>
<body align="center">
<a href="http://localhost/ts">转发规则</a>
</body>
</html>
```

```
@WebServlet("/ts")
public class TServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.getWriter().print("I have enough money");
        RequestDispatcher rd = request.getRequestDispatcher("/bank");
        rd.forward(request,response);
    }
}
```

```
@WebServlet("/bank")
public class Bank extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request, response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.getWriter().print("Wellcome to bank");
    }
}
```

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)；

```
@WebServlet("/ts")
public class TServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        request.setAttribute("name","xunhuan");
        RequestDispatcher rd = request.getRequestDispatcher("/bank");
        rd.forward(request,response);
    }
}
```

```
@WebServlet("/bank")
public class Bank extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request, response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.getWriter().print(request.getAttribute("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。

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");
```

```
@WebServlet("/ts")
public class TAServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        String username = request.getParameter("username");
        response.setContentType("text/html;charset=utf-8");
        response.getWriter().print(username);
    }
}
```

```
<!DOCT<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>中文表单</title>
<style>
    .al{
        margin-top: 100px;
    }
</style>
</head>
<body align="center">
<div class="al">
    <form action="http://localhost/ts"method="post">
        用户名<input type="text"name="username"/><br/>
        密码<input type="password"name="psw"/><br/>
        <input type="submit"value="提交">
    </form>
</div>
</body>
</html>
```

七, 生成验证码

- 导入jar ValidateCode.jar
- CodeServlet

```
@WebServlet("/checkS")
public class checkS extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        ValidateCode vc = new ValidateCode(100, 40, 4, 10);
        vc.write(response.getOutputStream());
    }
}
```

- 页面

```
<!DOCT<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>中文表单</title>
<style>
    .al{
        margin-top: 100px;
    }
</style>
</head>
<body align="center">
<div class="al">
    <form action="#"method="post">
        用户名<input type="text"name="username"/><br/>
        密码<input type="password"name="psw"/><br/>
        <img src="/checkS" onclick="changimg(this)"/><br/>
        <input type="text"name="checktext"><br/>
        <input type="submit"value="提交">
    </form>
</div>
</body>
<script>
    function changimg(o){
        o.src="/checkS?a="+new Date().getMilliseconds();
    }
</script>
</html>
```