今日内容：

1. Servlet
2. HTTP协议
3. Request

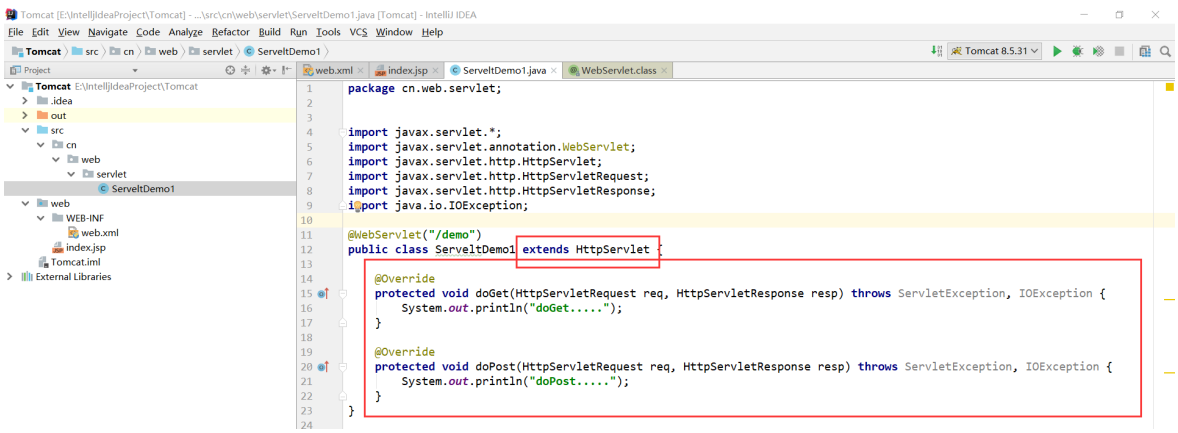
Servlet：

1. 概念
2. 步骤
3. 执行原理
4. 生命周期
5. Servlet3.0 注解配置

Servlet的体系结构

Servlet -- 接口 | GenericServlet -- 抽象类 | HttpServlet -- 抽象类

- GenericServlet：将Servlet接口中其他的方法做了默认空实现，只将service()方法作为抽象
- 将来定义Servlet类时，可以继承GenericServlet，实现service()方法即可
    - HttpServlet：对http协议的一种封装，简化操作，开发用这个
        1. 定义类继承HttpServlet
    2. 复写doGet/doPost方法



Servlet相关配置

1. urlpartten:Servlet访问路径
    1. 一个Servlet可以定义多个访问路径： @WebServlet({"/d4","/dd4","/ddd4"})

    当然，一般情况下用一个就好了

2. 路径定义规则：
3. /xxx：路径匹配：@WebServlet("/demo")
4. /xxx/xxx:多层路径，目录结构@WebServlet("/user/demo")，访问资源的时候写上这两层路径，@WebServlet("/user/*")，表示在访问资源路径的时候/user/xxx，xxx随便填就可以访问到资源
5. *.do：扩展名匹配，@WebServlet("*.do")，注意不要加/，那么写上xxx.do就可以访问资源，当然，do是随便写的，换个别的也成，比如"*.hehe"

HTTP：

概念

> *Hyper Text Transfer Protocol 超文本传输协议*

传输协议

> *定义了，客户端和服务器端通信时，发送数据的格式*

特点：

1. 基于TCP/IP的高级协议
2. 默认端口号:80
3. 基于请求/响应模型的:一次请求对应一次响应
4. 无状态的：每次请求之间相互独立，不能交互数据

历史版本：

- 1.0：每一次请求响应都会建立新的连接
- 1.1：复用连接

请求消息数据格式

请求行

请求方式 请求url 请求协议/版本 GET /login.html HTTP/1.1

- 请求方式：
    - HTTP协议有7种请求方式，常用的有2种
        - GET：
            1. 请求参数在请求行中，在url后。
            2. 请求的url长度有限制的
            3. 不太安全
        - POST：

1. 请求参数在请求体中
2. 请求的url长度没有限制的
3. 相对安全

请求头：客户端浏览器告诉服务器一些信息

请求头名称: 请求头值

- 常见的请求头：
    1. User-Agent：浏览器告诉服务器，我访问你使用的浏览器版本信息
        - 可以在服务器端获取该头的信息，解决浏览器的兼容性问题
    2. Referer：http://localhost/login.html
        - 告诉服务器，我(当前请求)从哪里来?
            - 作用：
                1. 防盗链：
                2. 统计工作：

请求空行

空行，就是用于分割POST请求的请求头，和请求体的。

请求体(正文)：

- 封装POST请求消息的请求参数的
- Get没有请求体，请求在网址以明文显示
- 字符串格式：

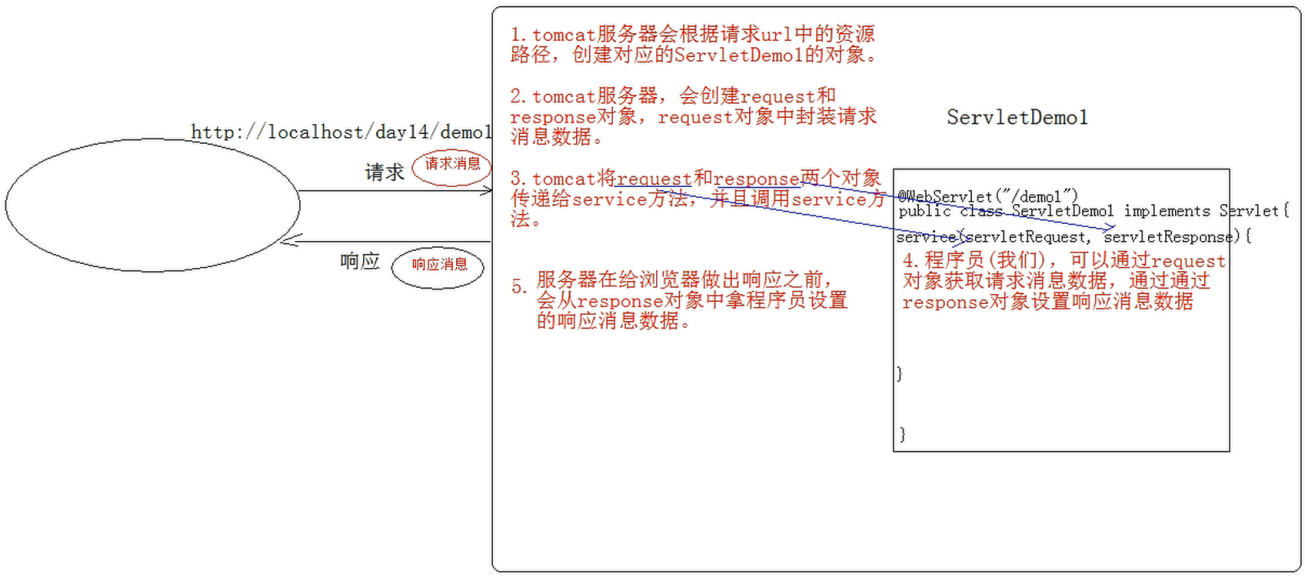
```
POST /login.html    HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-
US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://localhost/login.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1

username=zhangsan
```

- 响应消息数据格式

Request：

request对象和response对象的原理

1. request和response对象是由服务器创建的。我们来使用它们
2. request对象是来获取请求消息，response对象是来设置响应消息



3. request对象继承体系结构： ServletRequest -- 接口 | 继承 HttpServletRequest -- 接口 | 实现 org.apache.catalina.connector.RequestFacade --类(tomcat)

request功能：

获取请求消息数据

获取请求行数据

- GET /day14/demo1?name=zhangsan HTTP/1.1
- 方法：

获取请求方式 ：

- String getMethod()

(重点掌握)获取虚拟目录：

- String getContextPath()

获取Servlet(资源)路径:

- String getServletPath()

获取get方式请求参数：

- String getQueryString()

(重点掌握)获取请求URI：/day14/demo1

- String getRequestURI(): /day14/demo1

- StringBuffer getRequestURL() :http://localhost/day14/demo1

> - *URL:统一资源定位符： http://localhost/day14/demo1*
> - *URI：统一资源标识符: /day14/demo1*
> - *URI要比URL的范围更大*

获取协议及版本：HTTP/1.1

- String getProtocol()

获取客户机的IP地址：

- String getRemoteAddr()

```
package cn.web.request;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

    /*
    * 使用Request对象进行请求数据
    * */
    @WebServlet("/requestDemo1")
    public class Servlet extends HttpServlet {
        protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        }

        protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

            /*获取请求方式*/
            String method = request.getMethod();

            /*获取虚拟目录*/
            String contextPath = request.getContextPath();

            /*获取Servlet资源路径*/
            String servletPath = request.getServletPath();

            /*获取get方式请求参数*/
            String queryString = request.getQueryString();

            /*获取请求URI*/
            String requestURI = request.getRequestURI();
```

```
        /*获取请求URI，获取URL*/
        StringBuffer requestURL = request.getRequestURL();

        /*获取协议和版本*/
        String protocol = request.getProtocol();

        /*获取客户机的IP地址*/
        String remoteAddr = request.getRemoteAddr();

        System.out.println(method);

        System.out.println("-------------");
        System.out.println(contextPath);

        System.out.println("-------------");
        System.out.println(servletPath);

        System.out.println("-------------");
        System.out.println(queryString);

        System.out.println("-------------");
        System.out.println(requestURI);

        System.out.println("-------------");
        System.out.println(requestURL);

        System.out.println("-------------");
        System.out.println(protocol);

        System.out.println("-------------");
        System.out.println(remoteAddr);

    }
}
/*
GET
-------------
/demo
-------------
/requestDemo1
-------------
null
-------------
/demo/requestDemo1
-------------
http://localhost:8080/demo/requestDemo1
-------------
HTTP/1.1
-------------
0:0:0:0:0:0:0:1
```

```
     */
```

获取请求头数据

- 方法:
- (重点掌握)String getHeader(String name):通过请求头的名称获取请求头的值
- Enumeration<String> getHeaderNames():获取所有的请求头名称，当成迭代器使用就行

案例一:

```java
package cn.web.request;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Enumeration;

@WebServlet("/requestDemo1")
public class Servlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        /*获取所有的请求头的名称*/
        Enumeration<String> headerNames = request.getHeaderNames();

        /*遍历*/
        while (headerNames.hasMoreElements()){
            /*获取下一个请求头的名称*/
            String name = headerNames.nextElement();

            /*根据请求头名称获取值*/
            String value = request.getHeader(name);

            System.out.println(name+"-->"+value);
        }

    }
}
```

输出:

```
  host-->localhost:8080
  connection-->keep-alive
  cache-control-->max-age=0
  upgrade-insecure-requests-->1
  user-agent-->Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/74.0.3729.131 Safari/537.36
  accept--
>text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*
;q=0.8,application/signed-exchange;v=b3
  accept-encoding-->gzip, deflate, br
  accept-language-->zh-CN,zh;q=0.9
  cookie-->JSESSIONID=28CF8526A0A667396AA102A65C3ED80B; Idea-eef7716b=dd12e287-
a58b-4a7b-8ed9-91ed4334fe90; JSESSIONID=D7214FD1E1E7C36BBE57AD9D5B22CBF6
```

案例二:

```
  package cn.web.request;

  import javax.servlet.ServletException;
  import javax.servlet.annotation.WebServlet;
  import javax.servlet.http.HttpServlet;
  import javax.servlet.http.HttpServletRequest;
  import javax.servlet.http.HttpServletResponse;
  import java.io.IOException;
  import java.util.Enumeration;

  @WebServlet("/requestDemo1")
  public class Servlet extends HttpServlet {
      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

      }

      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

          /*
          * 根据名字获取到值
          * 名字不区分大小写
          * */
          String header = request.getHeader("user-agent");

          /*假如header里面包含Chrome这个字样*/
          if (header.contains("Chrome")){

              System.out.println("谷歌浏览器...");
          }

      }
  }
```

```
//谷歌浏览器...
```

获取请求体数据:

- 请求体：只有POST请求方式，才有请求体，在请求体中封装了POST请求的请求参数
- 步骤:

  获取流对象
  - BufferedReader getReader()：获取字符输入流，只能操作字符数据
    - ServletInputStream getInputStream()：获取字节输入流，可以操作所有类型数据
- 在文件上传知识点后讲解

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

    <form action="/demo/requestDemo1" method="post">
        <input type="text" placeholder="输入文本" name="text">
        <input type="submit" value="提交">
    </form>

</body>
</html>
```

```
    ```java
            package cn.web.request;

            import javax.servlet.ServletException;
            import javax.servlet.annotation.WebServlet;
            import javax.servlet.http.HttpServlet;
            import javax.servlet.http.HttpServletRequest;
            import javax.servlet.http.HttpServletResponse;
            import java.io.BufferedReader;
            import java.io.IOException;
            import java.util.Enumeration;

            @WebServlet("/requestDemo1")
            public class Servlet extends HttpServlet {
                protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

            //          获取请求消息体
```
```

```
                    /*获取字符流*/
                    BufferedReader reader = request.getReader();

                    /*读取数据*/
                    String line = null;
                    while ((line=reader.readLine())!=null){
                        System.out.println(line);
                    }

                }
                protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                }
            }
    //text=adasd
```

> 再从流对象中拿数据

5. 其他功能：

获取请求参数通用方式

不论get还是post请求方式都可以使用下列方法来获取请求参数

1. String getParameter(String name):根据参数名称获取参数值

   > *比如：username=zs&password=123，然后通过username获取zs，通过password获取123*

2. String[] getParameterValues(String name):根据参数名称获取参数值的数组
   比如有hobby=xx&hobby=game，通过hobby获取xx和game，可以用于复选框
3. Enumeration getParameterNames():获取所有请求的参数名称
4. Map<String,String[]> getParameterMap():获取所有参数的map集合

- 中文乱码问题：
  - get方式：tomcat 8 已经将get方式乱码问题解决了
  - post方式：会乱码
    - 解决：在获取参数前，设置request的编码request.setCharacterEncoding("utf-8");

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
```

```
</head>
<body>

    <form action="/demo/requestDemo1" method="post">
        <input type="text" placeholder="请输入用户名" name="username">
        <input type="text" placeholder="请输入密码" name="password">

        <input type="checkbox" name="hobby" value="Game">游戏
        <input type="checkbox" name="hobby" value="Study">学习

        <input type="submit" value="提交">
    </form>

</body>
</html>
```

```
package cn.web.request;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.Enumeration;
import java.util.Map;
import java.util.Set;

@WebServlet("/requestDemo1")
public class Servlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        /*设置字符集，解决中文乱码问题*/
        request.setCharacterEncoding("utf-8");

        /*获取所有请求的参数名*/
        Enumeration<String> parameterNames = request.getParameterNames();

        /*通过参数名来获取参数*/
        while (parameterNames.hasMoreElements()){
            String parameter = parameterNames.nextElement();

            /*通过参数名来获取参数数组*/
            String[] parameterValues = request.getParameterValues(parameter);

            for (String parameterValue : parameterValues) {
                System.out.println(parameter+"-->"+parameterValue);
            }
```

```
        }

        System.out.println("----------------------------");

//          使用Map集合，获取所有参数的map集合
        /*形成了map集合*/
        Map<String, String[]> parameterMap = request.getParameterMap();

        /*获取所有map集合的键*/
        Set<String> keySet = parameterMap.keySet();

        /*获取单个的键*/
        for (String name : keySet) {
            /*通过键获取值*/
            String[] values = parameterMap.get(name);
            for (String value : values) {
                System.out.println(name+"-->"+value);
            }
            System.out.println("-----");
        }

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        this.doPost(request,response);
    }
}
/*
username-->张三
password-->123123
hobby-->Game
hobby-->Study
----------------------------
username-->张三
-----
password-->123123
-----
hobby-->Game
hobby-->Study
-----
*/
```

请求转发

一种在服务器内部的资源跳转方式

1. 步骤：
    1. 通过request对象获取请求转发器对象：RequestDispatcher getRequestDispatcher(String path)

2. 使用RequestDispatcher对象来进行转发：forward(ServletRequest request, ServletResponse response)

2. 特点：
    1. 浏览器地址栏路径不发生变化
    2. 只能转发到当前服务器内部资源中，不能访问其他服务器的资源
    3. 转发是一次请求，虽然两个资源同时被访问了，但是只是同一次请求

```
package cn.web.request;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.Enumeration;
import java.util.Map;
import java.util.Set;

@WebServlet("/requestDemo1")
public class Servlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        request.setCharacterEncoding("utf-8");

        System.out.println("demo1....");

        /*进行资源请求转发*/
        request.getRequestDispatcher("/requestDemo2").forward(request,response);

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        this.doPost(request,response);
    }
}
/*
demo1....
demo2....
*/
```

```
package cn.web.request;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/requestDemo2")
public class Servlet2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        request.setCharacterEncoding("utf-8");

        /*请求转发过来的时候应该会输出这一句话*/
        System.out.println("demo2....");


    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        this.doPost(request,response);
    }
}
```

共享数据：

- 域对象：一个有作用范围的对象，可以在范围内共享数据
- request域：代表一次请求的范围，一般用于请求转发的多个资源中共享数据
- 方法：

1. void setAttribute(String name,Object obj):存储数据
2. Object getAttitude(String name):通过键获取值
3. void removeAttribute(String name):通过键移除键值对
4. 获取ServletContext：

- ServletContext getServletContext()

案例：用户登录

- 用户登录案例需求： 1.编写login.html登录页面 username & password 两个输入框 2.使用Druid数据库连接池技术,操作mysql，day14数据库中user表 3.使用JdbcTemplate技术封装JDBC 4.登录成功跳转到SuccessServlet展示：登录成功！用户名,欢迎您 5.登录失败跳转到FailServlet展示：登录失败，用户名或密码错误

- 分析
- 开发步骤
    1. 创建项目，导入html页面，配置文件，jar包
    2. 创建数据库环境

```sql
CREATE DATABASE day14;
USE day14;
CREATE TABLE USER(
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(32) UNIQUE NOT NULL,
    PASSWORD VARCHAR(32) NOT NULL
);
```

3. 创建包cn.itcast.domain,创建类User

```java
package cn.itcast.domain;


/*对应着mysql数据库中day14中的user表*/
public class User {

    private int id;
    private String username;
    private String password;

    public User() {
    }

    public User(int id, String username, String password) {
        this.id = id;
        this.username = username;
        this.password = password;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
```

```
        this.password = password;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                ", password='" + password + '\'' +
                '}';
    }
}
```

4. 创建包cn.itcast.util,编写工具类JDBCUtils

```
package cn.itcast.util;


import com.alibaba.druid.pool.DruidDataSourceFactory;

import javax.sql.DataSource;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

/*
* JDBC的工具类  使用Druid连接池
* */
public class JDBCUtils {


    private static DataSource dataSource;

    /*
    * 静态代码块获取配置文件
    * */
    static {
        try {
            /*创建properties对象*/
            Properties properties = new Properties();
            /*获取文件路径*/
            String path =
JDBCUtils.class.getClassLoader().getResource("druid.properties").getPath();
            /*加载配置文件*/
            properties.load(new FileInputStream(path));
```

```
                    /*获取DataSource*/
                    dataSource =
DruidDataSourceFactory.createDataSource(properties);
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }


            /*
            * 获取连接池对象
            * */
            public static DataSource getDataSource(){
                return dataSource;
            }


            /*
            * 获取Connection对象
            * */
            public static Connection connection(){

                Connection connection=null;

                try {
                    connection = dataSource.getConnection();
                } catch (SQLException e) {
                    e.printStackTrace();
                }

                return connection;
            }


            /*关闭连接池的方法*/
            public static void close(Statement statement,Connection connection){
                if (statement!=null){
                    try {
                        statement.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }

                if (connection!=null){
                    try {
                        connection.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
```

```
                    }
                }
            }

            /*关闭连接池的方法重载*/
            public static void close(ResultSet resultSet,Statement
statement,Connection connection){
                if (resultSet!=null){
                    try {
                        resultSet.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }

                if (statement!=null){
                    try {
                        statement.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }

                if (connection!=null){
                    try {
                        connection.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
```

5. 创建包cn.itcast.dao,创建类UserDao,提供login方法

```
package cn.itcast.dao;


import cn.itcast.domain.User;
import cn.itcast.util.JDBCUtils;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;

/*
* 操作数据库的User表的类
* */
public class UserDao {

//     声明JdbcTemplate对象共用
```

```
    private JdbcTemplate jdbcTemplate = new
JdbcTemplate(JDBCUtils.getDataSource());


    /**
     * 登录方法
     * @param loginUser 只有用户名和密码
     * @return  user    包含用户全部数据，如果没有查询到数据返回null
     */
    public User login(User loginUser){

        try {
            /*声明sql语句*/
            String sql = "select * from user where username=? and password=?";

            /*调用query方法并且给?赋值*/
            User user = jdbcTemplate.queryForObject(sql,new
BeanPropertyRowMapper<>(User.class),
                    loginUser.getUsername(),loginUser.getPassword());

            return user;
        } catch (DataAccessException e) {
            e.printStackTrace();
            return null;//如果没有查询到数据返回null
        }
    }
}
```

5_2：在这里新建一个测试类，测试类里面测试一下UserDao是否成功执行

```
package cn.itcast.test;

import cn.itcast.dao.UserDao;
import cn.itcast.domain.User;
import org.junit.Test;

public class UserDaotest {



    @Test
    public void testLogin(){
        User loginUser = new User();
        loginUser.setUsername("superbaby");
        loginUser.setPassword("123");


        UserDao dao = new UserDao();

        User login = dao.login(loginUser);
```

```
        System.out.println(login);
    }

}
```

6. 编写html界面，login.html中form表单的action路径的写法：虚拟目录+Servlet的资源路径

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <!--注意form表单的action应该写 虚拟路径+资源路径-->
    <form action="/TomcatTest/LoginServlet" method="post">
        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>
</html>
```

7. 编写LoginServlet，完成登录的具体逻辑

```
package cn.itcast.web.servlet;

import cn.itcast.dao.UserDao;
import cn.itcast.domain.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

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

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        /*设置编码*/
```

```
        request.setCharacterEncoding("utf-8");
        /*获取请求参数*/
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        /* 封装user对象*/
        User loginUser = new User();
        loginUser.setUsername(username);
        loginUser.setPassword(password);
        /*调用UserDao的login方法*/
        UserDao userDao = new UserDao();
        User login = userDao.login(loginUser);
        /*判断login*/
        if (login==null){
            /*登录失败*/

 request.getRequestDispatcher("/failServlet").forward(request,response);
        }else {
            /*登录成功*/
                /*存储数据*/
                request.setAttribute("user",login);

 request.getRequestDispatcher("/successServlet").forward(request,response);
        }
    }
}
```

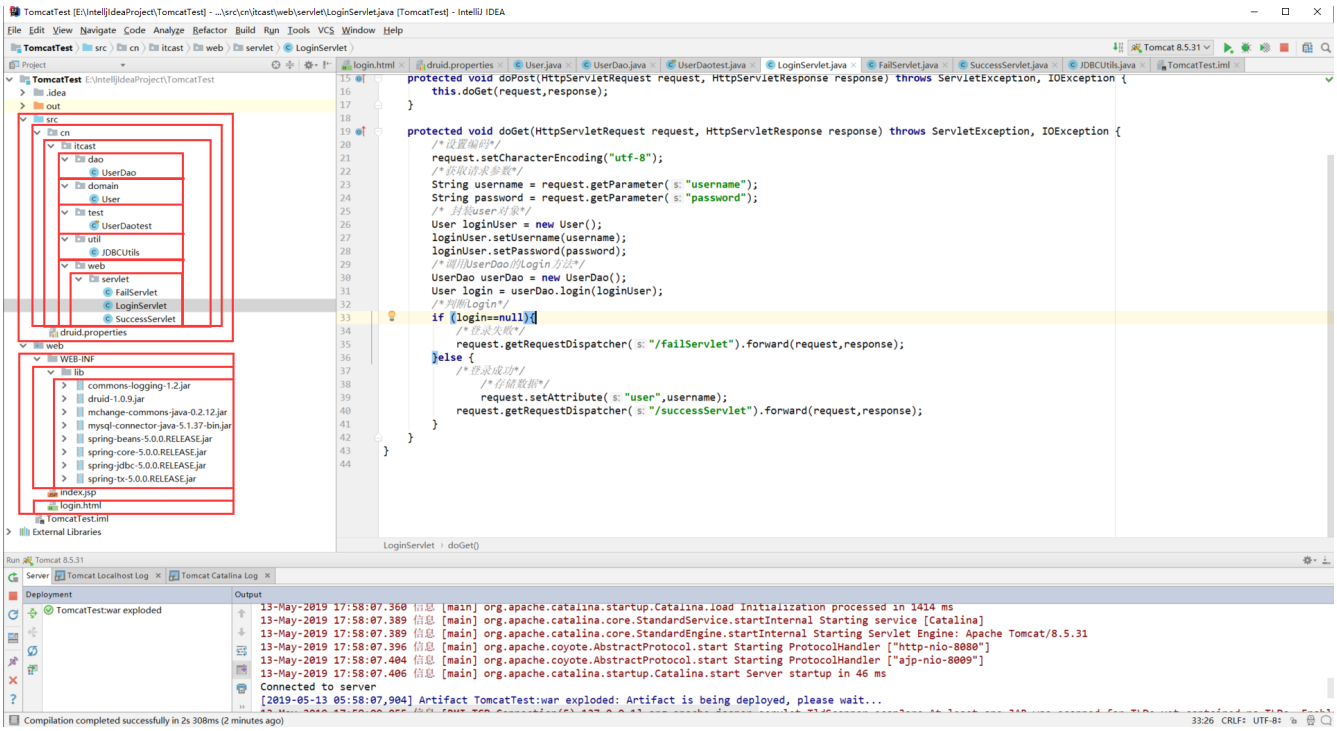
8. 编写FailServlet和SuccessServlet类

```
package cn.itcast.web.servlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/failServlet")
public class FailServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        /*给页面写一句话，提前剧透*/
            /*设置编码*/
        response.setContentType("text/html;charset=utf-8");
        response.getWriter().write("登录失败，用户名或密码错误");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
```

```
        this.doPost(request,response);
    }
}
```

```
package cn.itcast.web.servlet;

import cn.itcast.domain.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/successServlet")
public class SuccessServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        /*获取存储的数据*/
        User user = (User) request.getAttribute("user");
        if (user!=null) {
            /*设置编码*/
            response.setContentType("text/html;charset=utf-8");
            /*给页面写一句话，提前剧透*/
            response.getWriter().write("登录成功，"+user.getUsername()+"，欢迎您");
        }
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        this.doPost(request,response);
    }
}
```

目录分级：



BeanUtils工具类

> *简化数据封装，导入jar包：commons-beanutils-1.8.0.jar*

- 用于封装JavaBean的
- 包：org.apache.commons.beanutils.BeanUtils

JavaBean：标准的Java类

要求：

1. 类必须被public修饰
2. 必须提供空参的构造器
3. 成员变量必须使用private修饰
4. 提供公共setter和getter方法

功能：封装数据

概念：

- 成员变量：
- 属性：setter和getter方法截取后的产物 例如：getUsername() --> Username--> username
- 操作的是属性而不是成员变量

方法：

1. setProperty()
2. getProperty()

3. populate(Object obj,Map map)：将map集合的键值对信息封装到对应的JavaBean对象中

Object其实就是mysql表所对应的那个类创建的对象

```
package cn.itcast.web.servlet;

import cn.itcast.dao.UserDao;
import cn.itcast.domain.User;
import org.apache.commons.beanutils.BeanUtils;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.util.Map;

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

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
//        获取请求参数，使用BeanUtils工具
        /*使用 .getParameterMap()方法获取所有的请求参数并且封装为Map对象*/
        Map<String, String[]> parameterMap = request.getParameterMap();
        /*创建User对象*/
        User loginUser = new User();
        /*使用BeanUtils封装*/
        try {
            /*import org.apache.commons.beanutils.BeanUtils;别导错包*/
            BeanUtils.populate(loginUser,parameterMap);
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }

        UserDao userDao = new UserDao();
        User login = userDao.login(loginUser);

        if (login==null){

 request.getRequestDispatcher("/failServlet").forward(request,response);
        }else {
            request.setAttribute("user",login);
```

```
request.getRequestDispatcher("/successServlet").forward(request,response);
            }
        }
    }
```