



# 今克

## Java EE框架 ---MVC

Java EE framework -环境搭建、Servlet、链接数据库、MVC

王磊

研发部

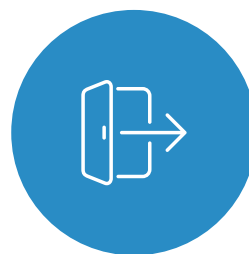
# CONTENTS



开发环境设置



Servlet



SQLHelper工具类



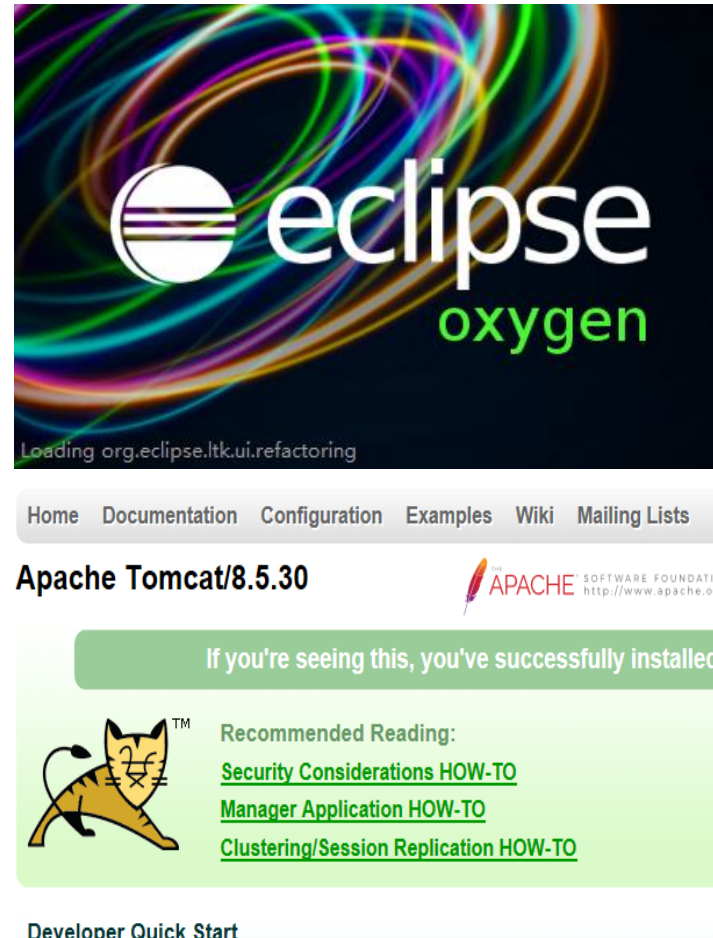
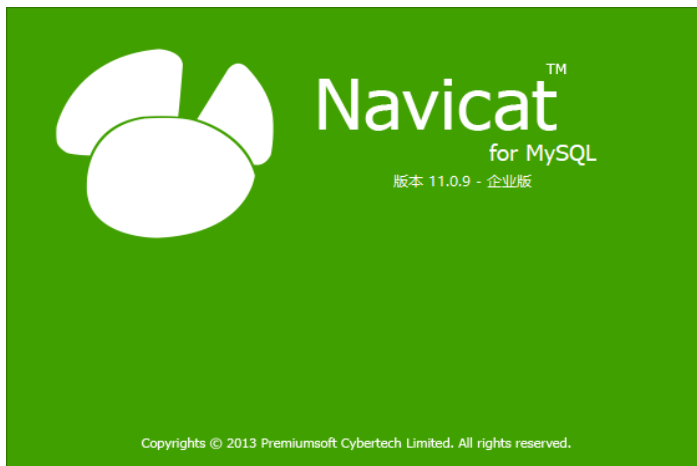
MVC项目

# • 开发环境设置 •

- 1、操作系统：Windows7、Windows10
- 2、应用软件：Jdk1.8.0\_162 Eclipse Oxygen Release Milestone 5 (4.7.0 M5)、MySQL5.5、Navicat for MySQL11.0
- 3、服务器：Tomcat8.5.30

```
C:\Users\Administrator>java -version
java version "1.8.0_162"
Java(TM) SE Runtime Environment (build
Java HotSpot(TM) 64-Bit Server VM (bu
mode)
```

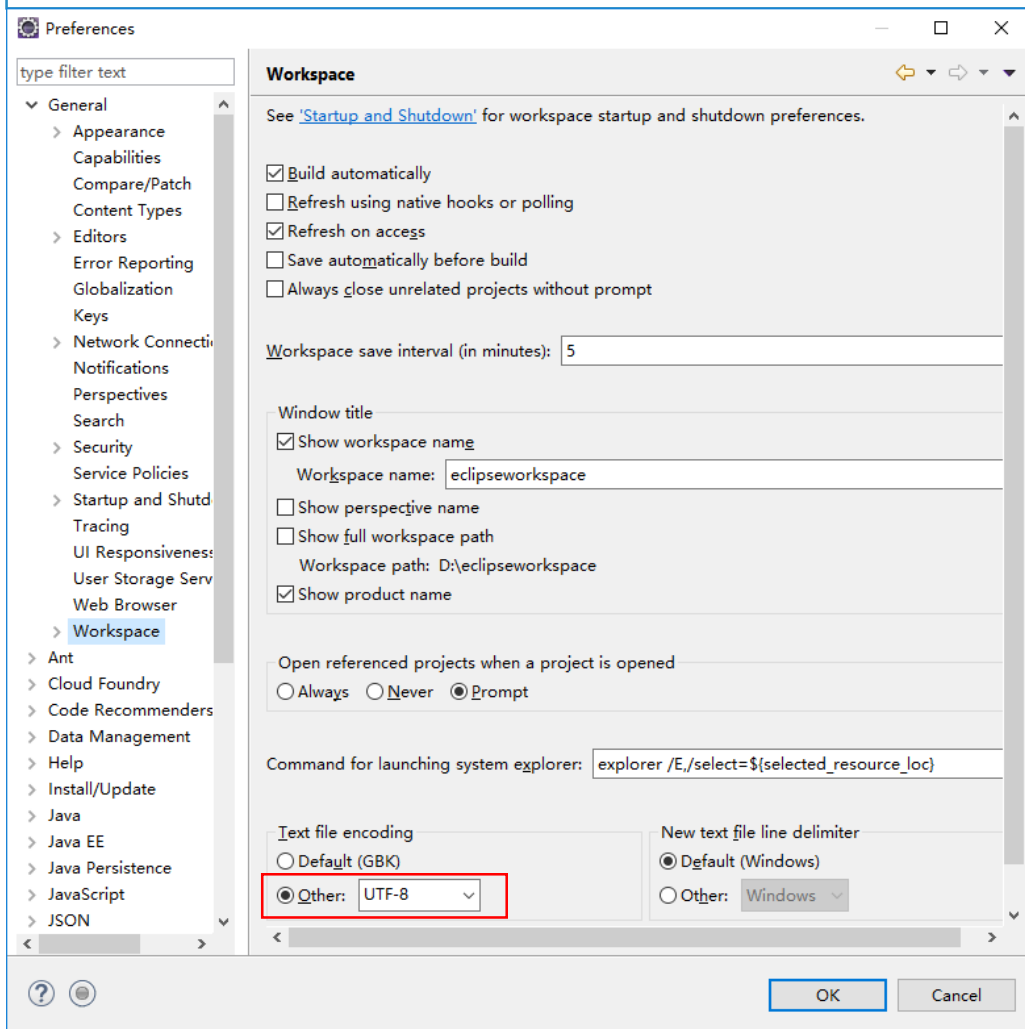
C:\Users\Administrator>



# 开发环境设置

## 设置Eclipse环境

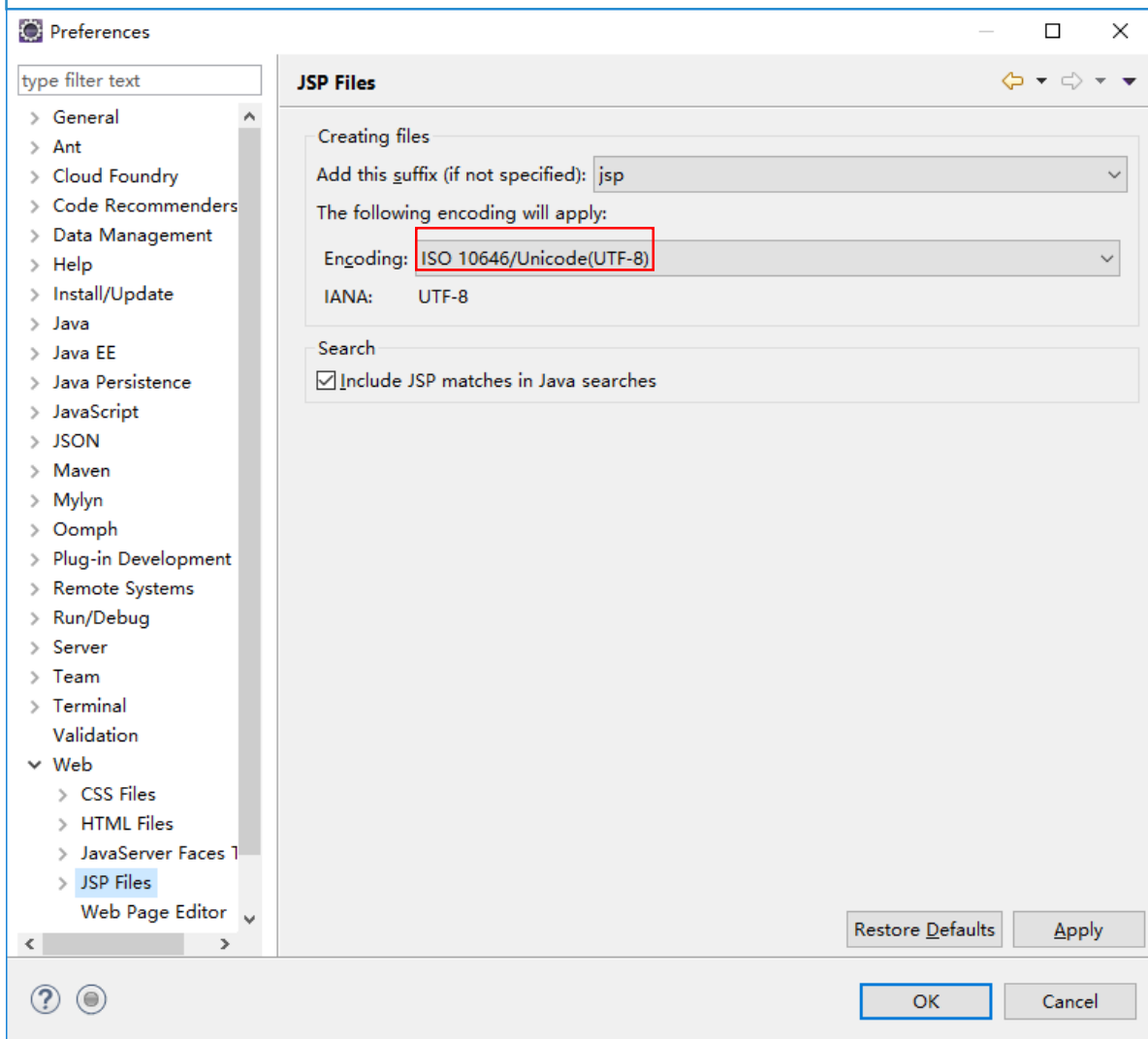
### 1、Eclipse→Windows->Preferences->General->Workspace



# 开发环境设置

设置Eclipse环境

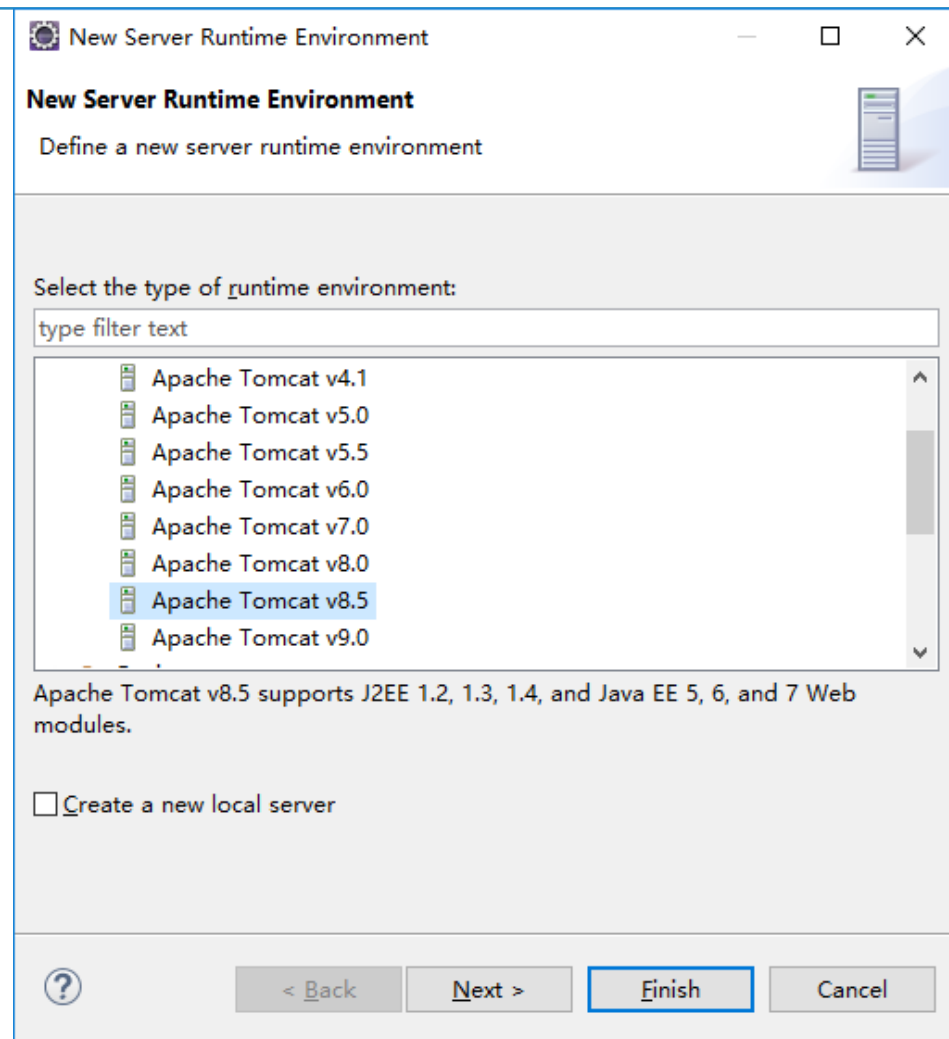
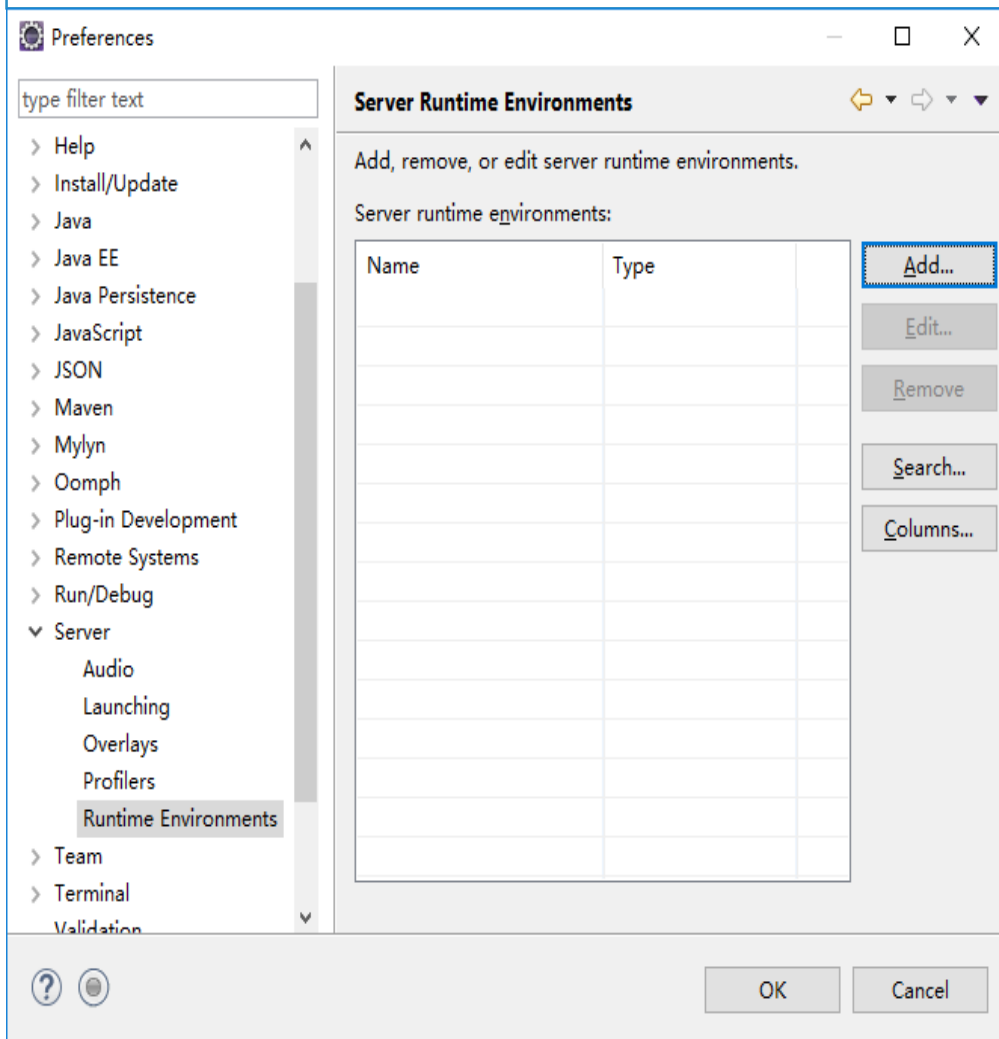
2、Eclipse→Windows->Preferences->Web->Jsp Files



## 开发环境设置

## 设置Eclipse环境

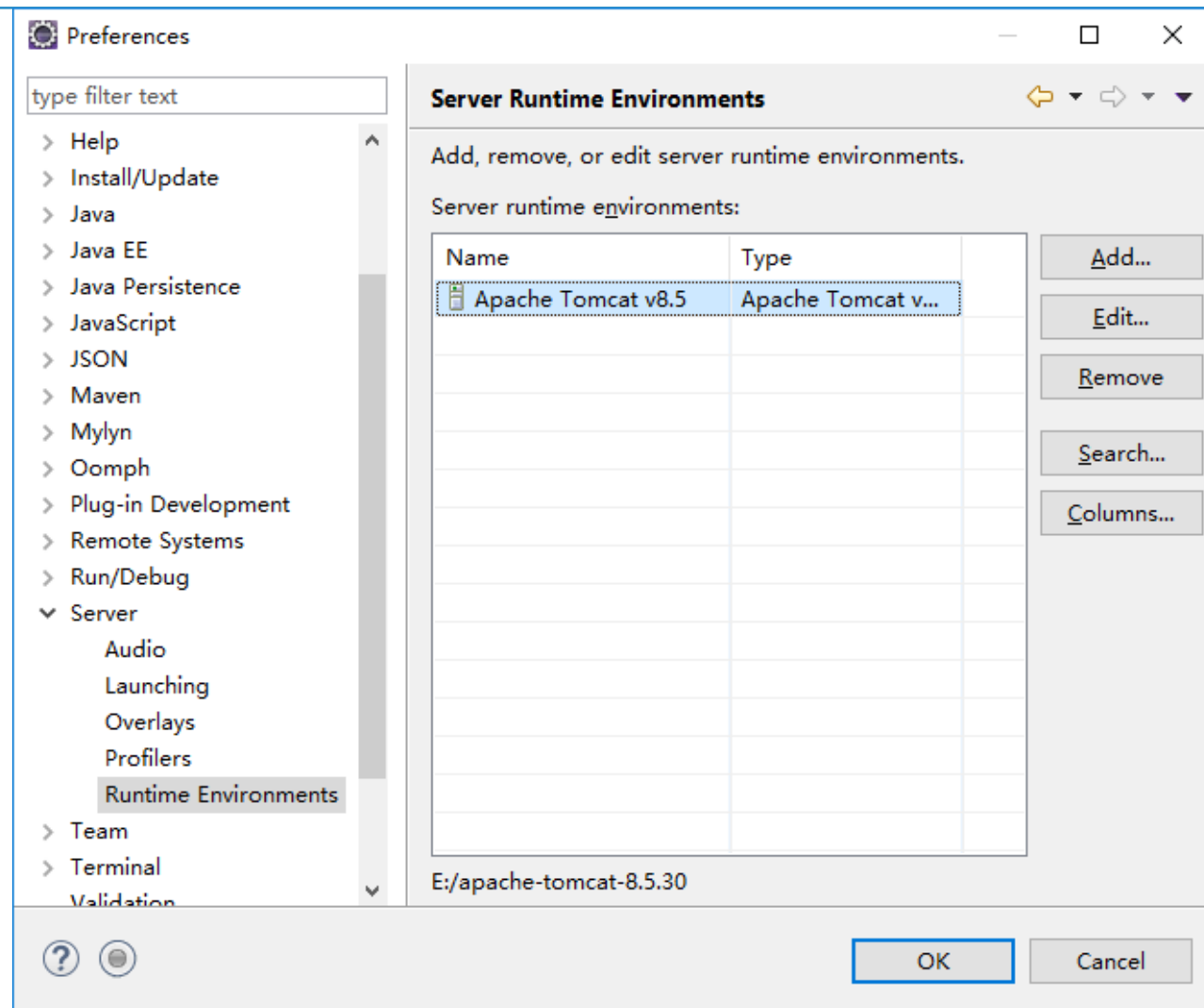
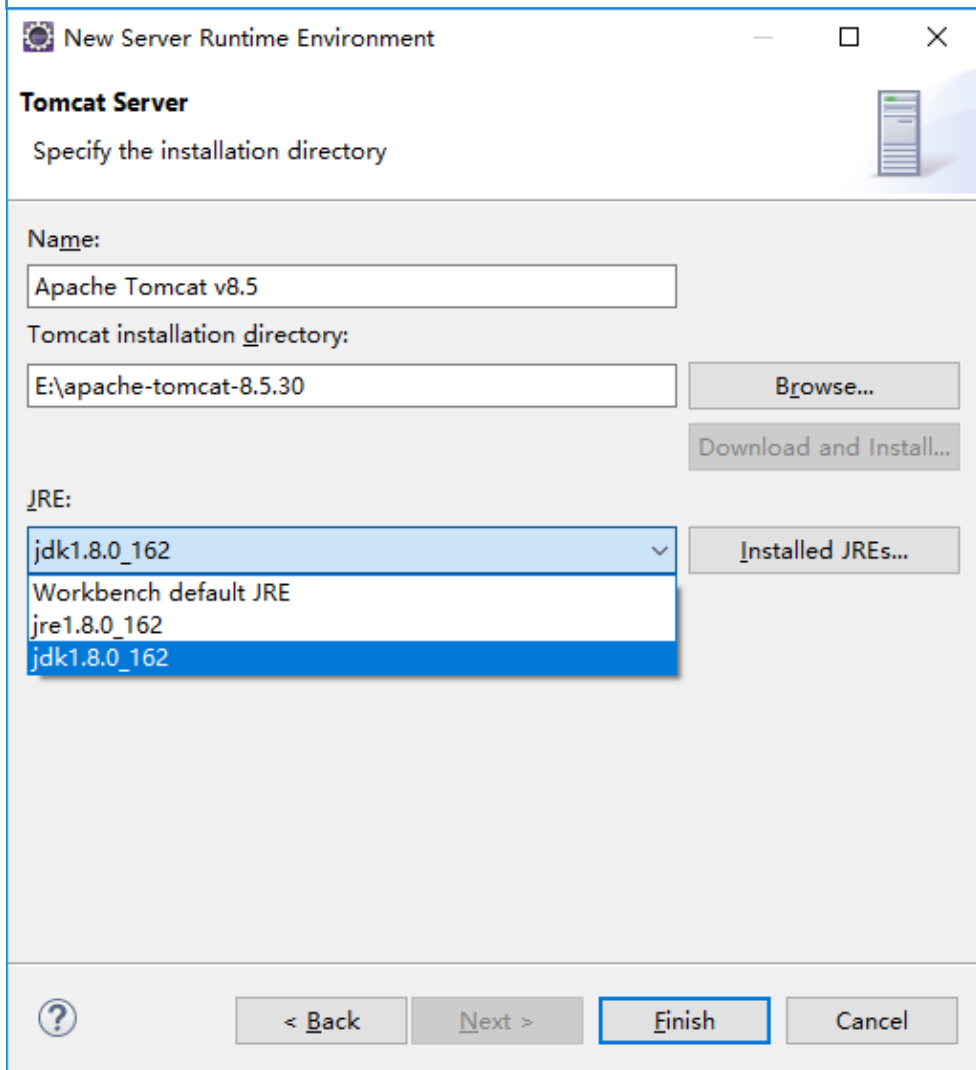
### 3、 Eclipse→Windows->Preferences->Server->Runtime Environments



# 开发环境设置

设置Eclipse环境

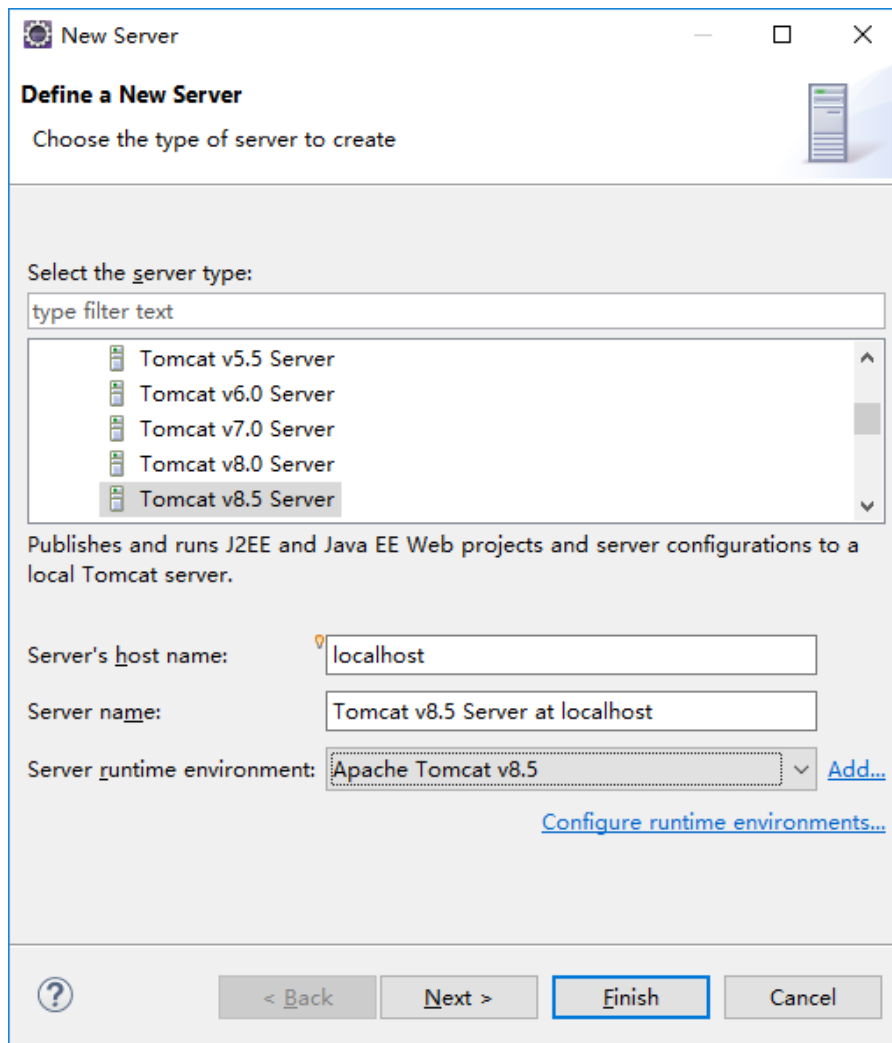
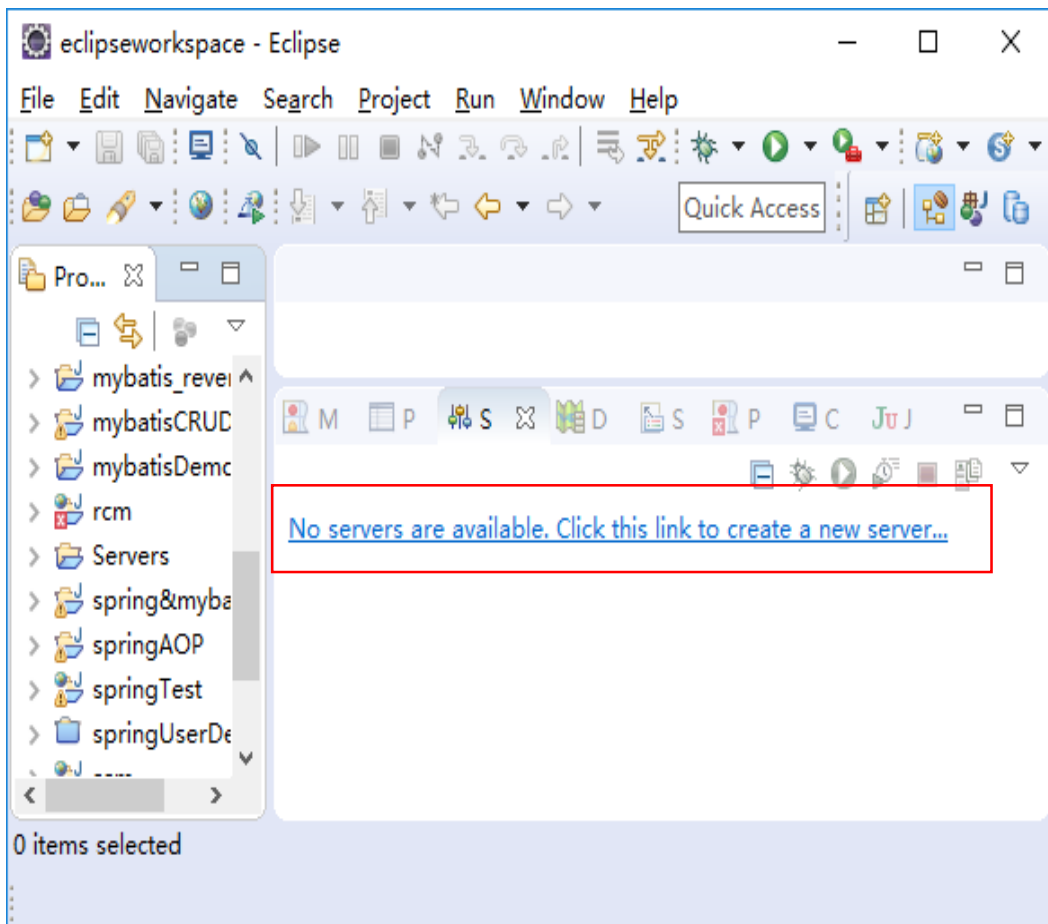
4、Eclipse→Windows->Preferences->Server->Runtime Environments



# 开发环境设置

设置Eclipse环境

5、启动Eclipse，关联Tomcat服务器





# • Web 技术 •

- 1、Web技术起源于八十年代，1991年CERN(European Organization for Nuclear Research)正式发布了Web技术标准。目前与web相关的各种标准都是由著名的W3C组织(World wide web consortium)管理和维护的。
- 2、Web开发技术大体上也可以被分为客户端和服务端两个大类。
- 3、Web客户端的主要任务是展现信息内容：常用的客户端信息显示语言是html，它是目前应用最广泛且被大多数浏览器支持的语言。（CSS, Javascript, vbscript）
- 4、Web服务器端响应客户的请求：服务器响应浏览器发来的http请求，动态响应客户端请求后，返回结果。从最早的CGI技术到目前的PHP，ASP，JSP/Servlet等。

# • 什么是 JavaEE •

- 1、**Java EE**：Java 平台企业版（Java Platform Enterprise Edition），之前称为Java 2 Platform, Enterprise Edition (**J2EE**)，2018年3月更名为 **Jakarta EE**(英 [dʒə'kɑ:tə]，这个名称应该还没有得到群众认可)。
- 2、**Java EE**是 Sun 公司为企业级应用推出的标准平台，用来开发B/S架构软件。
- 3、Java EE 可以说是一个**框架**，也可以说是一种**规范**。
- 4、所有的 **Java EE API**, 都是按照领域专家们所确定的标准发布的每个 Java EE API 都经过了 Java Community Process 的严谨审核。
- 5、**Application Server**应用服务器,是 Java EE 规范的完整实现。可以将 Java EE 程序部署到任意一种 Application Server 上。如 Apache **Tomcat**, IBM WebSphere, Oracle WebLogic, JBoss Wildfly, Payara Server 等

# • 什么是Servlet •

Servlet是sun公司提供的一门用于开发动态web资源的技术。

Sun公司在其API中提供了一个servlet接口，用户若想开发一个动态web资源(即开发一个Java程序向浏览器输出数据)，需要完成以下2个步骤：

- 1、编写一个Java类，实现servlet接口。
- 2、把开发好的Java类部署到web服务器中。

按照一种约定俗成的称呼习惯，通常也把实现了servlet接口的java程序，称之为Servlet

## • Servlet的运行过程 •

Servlet程序是由WEB服务器调用，web服务器收到客户端的Servlet访问请求后：

- 1、Web服务器首先检查是否已经装载并创建了该Servlet的实例对象。如果是，则直接执行第④步，否则，执行第②步。
- 2、装载并创建该Servlet的一个实例对象。
- 3、调用Servlet实例对象的init()方法。
- 4、创建一个用于封装HTTP请求消息的HttpServletRequest对象和一个代表HTTP响应消息的HttpServletResponse对象，然后调用Servlet的service()方法并将请求和响应对象作为参数传递进去。
- 5、WEB应用程序被停止或重新启动之前，Servlet引擎将卸载Servlet，并在卸载之前调用Servlet的destroy()方法。

# • Java链接数据库 •

## 1、创建MySQL数据库

drop database if exists userdb;

create database userdb DEFAULT CHARSET utf8;

use userdb;

create table user

(

id int auto\_increment primary key,

username varchar(255),

password varchar(255),

email varchar(255),

identity varchar(255)

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

insert into user(`id`,`username`,`password`,`email`,`identity`) values

(0,'admin','123456','adminl@126.com','admin'),

(0,'test1','123456','test1@qq.com','user'),

(0,'test2','123456','test2@qq.com','user'),

(0,'test3','123456','test3@qq.com','user'),

(0,'test4','123456','test4@qq.com','user');

id	username	password	email	identity
1	admin	123456	adminl@126.com	admin
2	test1	123456	test1@qq.com	user
3	test2	123456	test2@qq.com	user
4	test3	123456	test3@qq.com	user
5	test4	123456	test4@qq.com	user

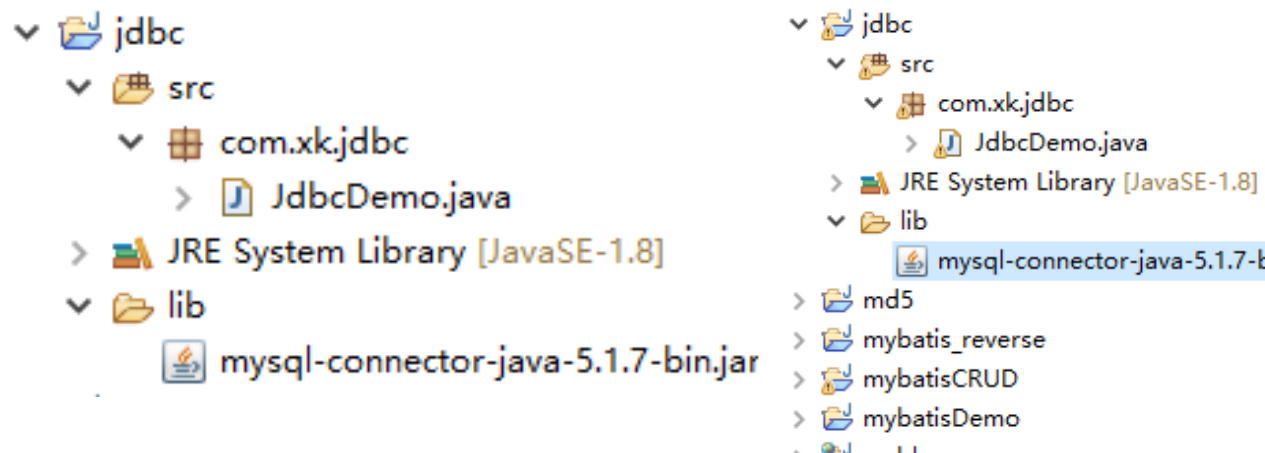
# • Java链接数据库 •

## 2、创建Java项目jdbc

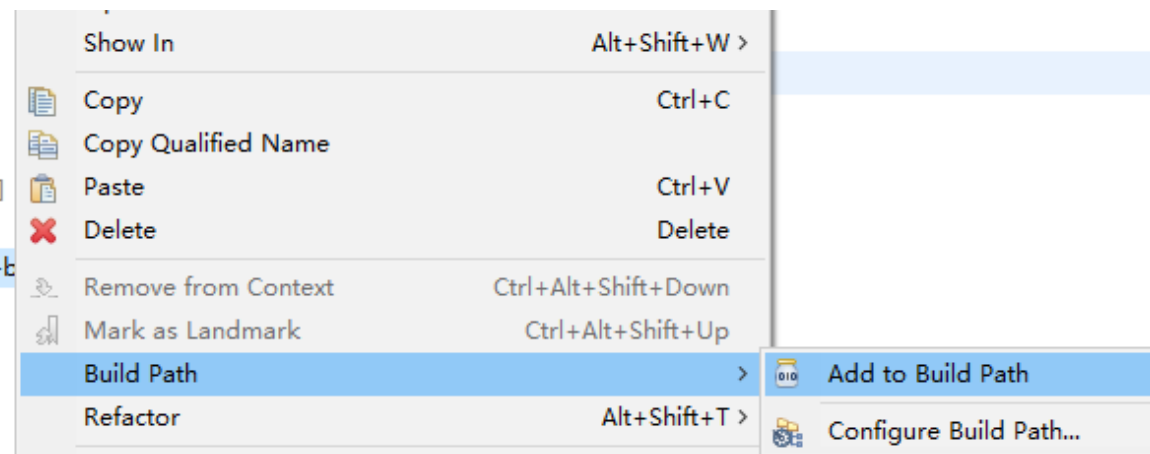
### 2.1 创建文件夹lib，置入mysql驱动jar包

### 2.2 创建com.xk.jdbc包，在包下面创建JdbcDemo类

### 2.3 build path:构建路径



( 1 )



( 2 )

# Java链接数据库

## 3、编写链接数据库代码

```
package com.xk.jdbc;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class JdbcDemo {
    public static void main(String[] args) throws SQLException {
        Connection ct=null;
        PreparedStatement ps=null;
        ResultSet rs=null;
        try {
            //1 加载驱动
            Class.forName("com.mysql.jdbc.Driver");
            //2 创建链接
            ct = DriverManager.getConnection("jdbc:mysql://localhost:3306/"
                + "userdb?characterEncoding=utf-8","root", "root");
            //创建sql对象
            ps=ct.prepareStatement("select * from user");
            rs=ps.executeQuery();
        }
    }
}
```

# • Java链接数据库 •

## 3、编写链接数据库代码

```
while(rs.next()) {
    System.out.println(rs.getString("id")+" "
                        +rs.getString("username")
                        +rs.getString("password")+" "+
                        rs.getString("email")+" "
                        +rs.getString("identity"));
}
//ps.executeUpdate();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}finally{
    //关闭资源
    if(rs!=null){
        rs.close();
    }
    if (ps!=null){
        try {
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

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



## • Java链接数据库 •

### 4、运行程序，观察结果

```
1 admin123456 admin1@126.com admin
2 test1123456 test1@qq.com user
3 test2123456 test2@qq.com user
4 test3123456 test3@qq.com user
5 test4123456 test4@qq.com user
```

# • Java链接数据库 •

## 5、插入一条数据，观察结果

### 5.1需要改写源代码，重新编译和运行

```
ps = ct.prepareStatement("insert into user values(?,?,?,?)");
```

```
ps.setString(1, "12");
```

```
ps.setString(2, "mary");
```

```
ps.setString(3, "123456");
```

```
ps.setString(4, "mary@sohu.com");
```

```
ps.setString(5, "admin");
```

```
//执行sql语句
```

```
ps.executeUpdate();
```

```
ps = ct.prepareStatement("select * from user");
```

```
rs = ps.executeQuery();
```

```
while(rs.next()) {
```

```
    System.out.println(rs.getString("id")+" "
```

```
        +rs.getString("username")
```

```
        +rs.getString("password")+" "+
```

```
        rs.getString("email")+" "
```

```
        +rs.getString("identity"));
```

```
}
```

```
} catch (ClassNotFoundException e) {
```

```
    e.printStackTrace();
```

```
1 admin123456 admin1@126.com admin
```

```
2 test1123456 test1@qq.com user
```

```
3 test2123456 test2@qq.com user
```

```
4 test3123456 test3@qq.com user
```

```
5 test4123456 test4@qq.com user
```

```
12 mary123456 mary@sohu.com admin
```

# 访问数据库工具类代码封装

需求：写出通用链接数据库工具类

```
package com.wl.util;
import java.io.IOException;
public class SQLHelper {
    private static Connection ct=null;
    private static PreparedStatement ps=null;
    private static ResultSet rs=null;
    private static String driver="";
    private static String url="";
    private static String user="";
    private static String password="";
    private static Properties pp=null;
    private static InputStream fis=null;
    static{
        try {
            pp=new Properties();
            fis=SQLHelper.class.getClassLoader().getResourceAsStream("dbinfo.properties");
            pp.load(fis);
            driver=pp.getProperty("driver");
            url=pp.getProperty("url");
            user=pp.getProperty("user");
            password=pp.getProperty("password");
            Class.forName(driver);
        } catch (Exception e) {
            e.printStackTrace();
        } finally{
            if(fis!=null){
                try {
                    fis.close();
                } catch (IOException e){
                    e.printStackTrace();
                }
            }
        }
    }
}
```

## • 访问数据库工具类代码封装 •

需求：写出通用链接数据库工具类

```
public static Connection getConnection() {  
    try {  
        ct=DriverManager.getConnection(url, user, password);  
    } catch (SQLException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
    return ct;  
}
```

# 访问数据库工具类代码封装

需求：写出通用链接数据库工具类

```
public static void close(ResultSet rs, Statement ps, Connection ct) {  
    if(rs!=null){  
        try {  
            rs.close();  
        } catch (SQLException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
    }  
    if(ps!=null){  
        try {  
            ps.close();  
        } catch (SQLException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
    }  
    if(ct!=null){  
        try {  
            ct.close();  
        } catch (SQLException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
    }  
}
```

```
public static Connection getCt(){  
    return ct;  
}  
public static PreparedStatement getPs(){  
    return ps;  
}  
public static ResultSet getRs(){  
    return rs;  
}
```

## • 访问数据库工具类代码封装 •

需求：写出通用链接数据库工具类

```
public static ResultSet executeQuery(String sql,String[]parameters){
    try {
        ct=getConnection();
        ps=ct.prepareStatement(sql);
        if(parameters!=null){
            for(int i=0;i<parameters.length;i++){
                ps.setString(i+1, parameters[i]);
            }
        }
        rs=ps.executeQuery();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return rs;
}
```

## • 访问数据库工具类代码封装 •

需求：写出通用链接数据库工具类

```
public static void executeUpdate(String sql,String[]parameters){
    try {
        ct=getConnection();
        ps=ct.prepareStatement(sql);
        if(parameters!=null){
            for(int i=0;i<parameters.length;i++){
                ps.setString(i+1, parameters[i]);
            }
            ps.executeUpdate();
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException();
    }finally{
        close(rs,ps,ct);
    }
}
```

## • 访问数据库工具类代码封装 •

总结：

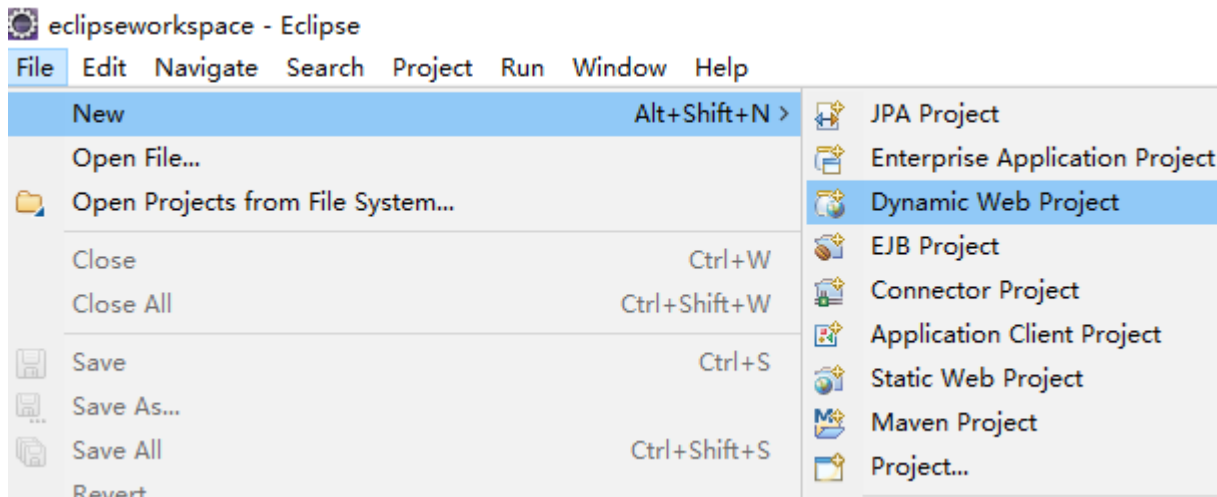
- 1、编写配置文件（解耦）
- 2、利用java类的反射，在静态代码块中读取配置文件，动态加载
- 3、编写getConnection、executeUpdate、executeQuery、close方法
- 4、封装成SQLHelve类，供项目调用



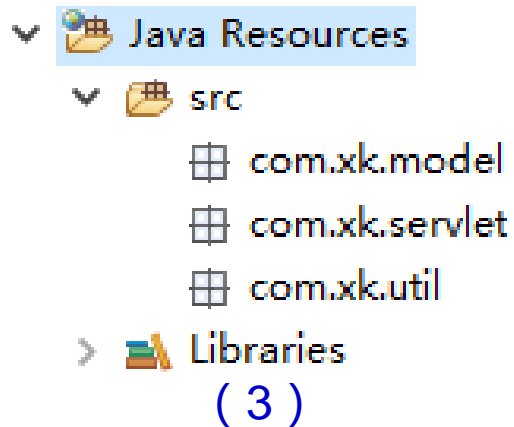
# MVC架构demo

1、创建项目xkuser

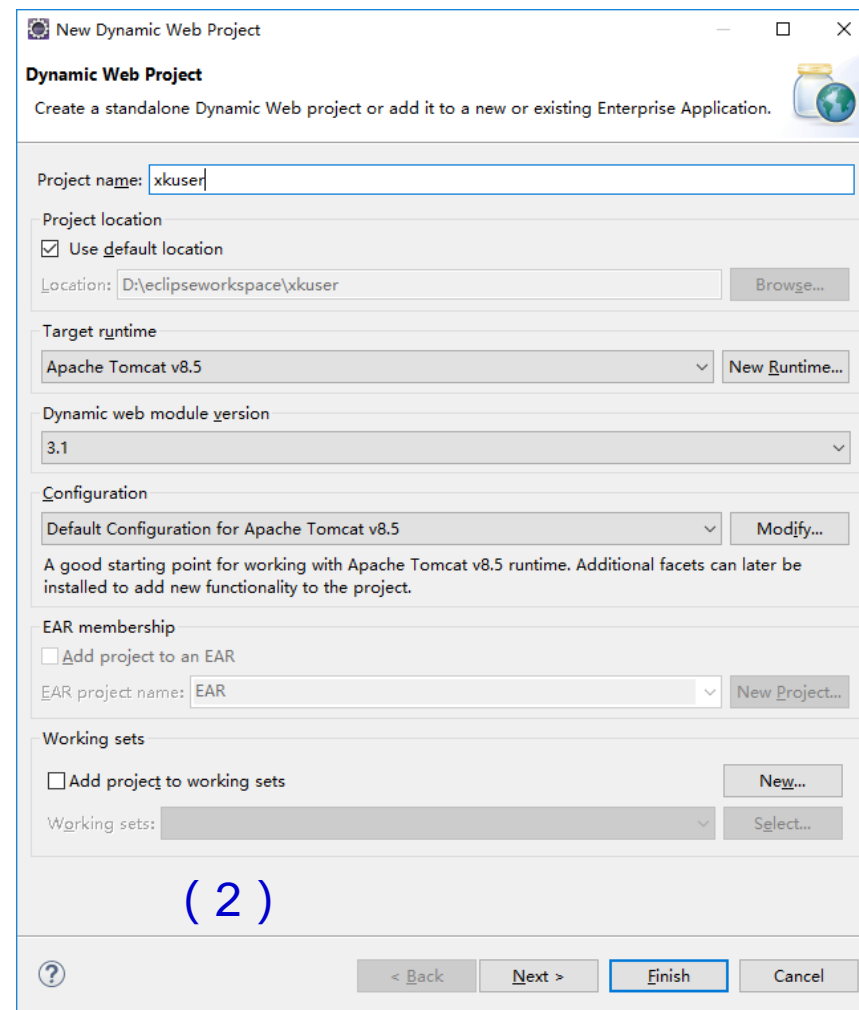
2、项目下创建com.xk.model com.xk.servlet com.xk.service com.xk.util 包



( 1 )



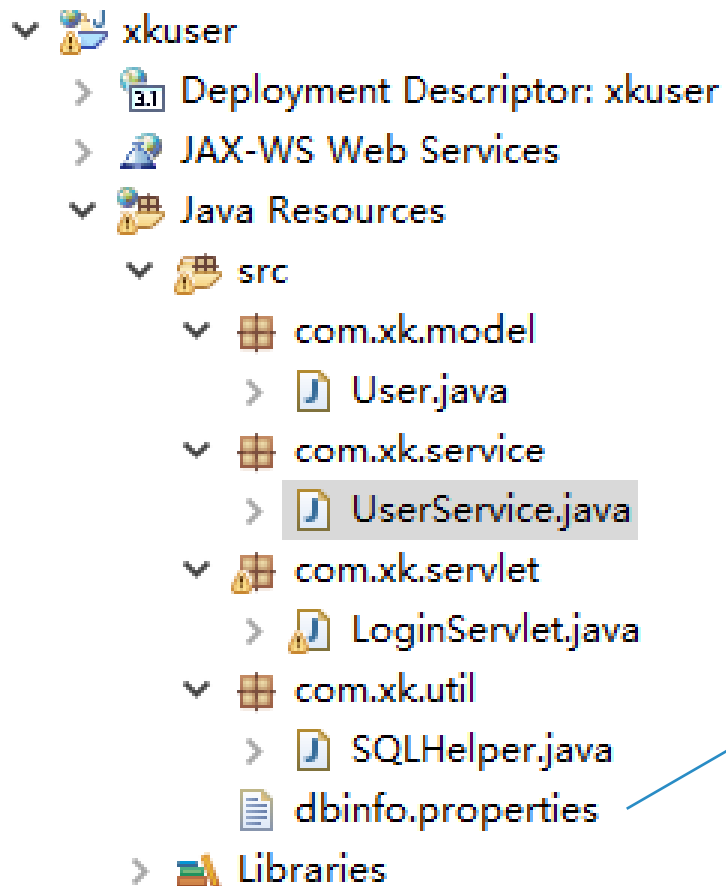
( 3 )



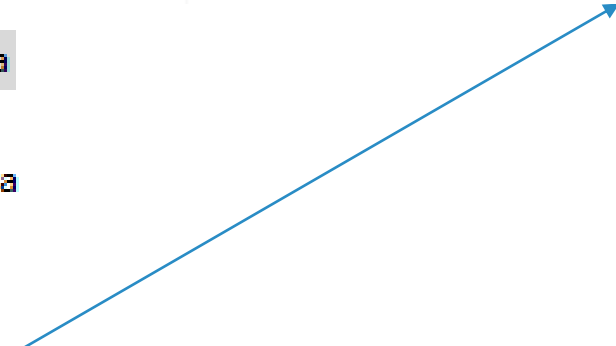
( 2 )

# • MVC架构demo •

- 3、在WEB-INF->lib文件夹下引入mysql 驱动jar包， Build Path
- 4、在src文件夹下，引入dbinfo.properties配置文件
- 5、将SQLHelper.java引入com.xk.util包中



```
1 url=jdbc:mysql://localhost:3306/userdb?characterEncoding=utf-8
2 driver=com.mysql.jdbc.Driver
3 user=root
4 password= root
```



# • MVC架构demo •

## 6、在com.xk.model包中编写JavaBean

```
package com.xk.model;
public class User {
    private String id;
    private String username;
    private String password;
    private String email;
    private String identity;
    public String getId() {
        return id;
    }
    public void setId(String 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;
}
public String getIdentity() {
    return identity;
}
public void setIdentity(String identity) {
    this.identity = identity;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
}
```

# • MVC架构demo •

## 7、在com.xk.servlet包中编写LoginServlet，修剪原始Servlet

LoginServlet.java

```
1 package com.xk.servlet;
2 import java.io.IOException;
3 import javax.servlet.ServletException;
4 import javax.servlet.annotation.WebServlet;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 @WebServlet("/LoginServlet")
9 public class LoginServlet extends HttpServlet {
10     private static final long serialVersionUID = 1L;
11     protected void doGet(HttpServletRequest request, HttpServletResponse response)
12         throws ServletException, IOException {
13         doPost(request, response);
14     }
15
16     protected void doPost(HttpServletRequest request, HttpServletResponse response)
17         throws ServletException, IOException {
18
19     }
20 }
```

注解

# • MVC架构demo •

## 9、编写login.jsp登录页面

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <center>
        <form action="LoginServlet" method="post">
            用户姓名: <input type="text" name="username"> <br/>
            用户密码: <input type="password" name="password"> <br/>
            <input type="submit" value="登录"/>
        </form>
    </center>
</body>
</html>
```

# • MVC架构demo •

## 10、(1)在com.xk.service包下面编写UserService类

```
package com.xk.service;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import com.xk.model.User;
import com.xk.util.SQLHelper;
public class UserService {
    //checkUser
    public boolean checkUser(User user) {
        boolean flag = false;
        String sql = "select * from user where username = ? and password = ?";
        String []parameters = {user.getUsername(),user.getPassword()};
        ResultSet rs = SQLHelper.executeQuery(sql, parameters);
        try {
            if(rs!=null) {
                flag = true;
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            SQLHelper.close(rs, SQLHelper.getP(),SQLHelper.getC());
        }
        return flag;
    }
}
```

# • MVC架构demo •

10、(2)在com.xk.service包下面编写UserService类

```
//getAllUsers
public ArrayList<User> getAllUsers(){
    ArrayList<User> allUsers = new ArrayList<User>();
    String sql = "select * from user";
    String []parameters = null;
    ResultSet rs = SQLHelper.executeQuery(sql, parameters);
    try {
        while(rs.next()){
            User user=new User();
            user.setId(rs.getString(1));
            user.setUsername(rs.getString(2));
            user.setPassword(rs.getString(3));
            user.setEmail(rs.getString(4));
            user.setIdentity(rs.getString(5));
            allUsers.add(user);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        SQLHelper.close(rs, SQLHelper.getP(),SQLHelper.getCt());
    }
    return allUsers;
}
```

# • MVC架构demo •

## 11、完善LoginServlet

```
package com.xk.servlet;
import java.io.IOException;
import java.util.ArrayList;
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 com.xk.model.User;
import com.xk.service.UserService;
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doPost(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //从login.jsp页面获取登录用户名和密码
        String username = request.getParameter("username");
        String password = request.getParameter("password");
```



# • MVC架构demo •

## 11、完善LoginServlet

```
//创建userService类，业务逻辑类
UserService userService = new UserService();
//封装user，登录对象
User user = new User();
user.setUsername(username);
user.setPassword(password);
//验证登录是否合法
if(userService.checkUser(user)) {
    //合法，则从数据库中取出全体数据，植入request的attribute域中，页面跳转到main.jsp
    ArrayList<User> allUsers = userService.getAllUsers();
    request.setAttribute("allUsers", allUsers);
    request.getRequestDispatcher("main.jsp").forward(request, response);
} else {
    //非法，则重新定位到登录页面
    request.getRequestDispatcher("login.jsp").forward(request, response);
}
}
```

# • MVC架构demo •

## 12、编写main.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page language="java" import="java.util.*,com.xk.model.*"%>
<body>
    <% ArrayList<User> allUsers = (ArrayList<User>)request.getAttribute("allUsers"); %>
    <center>
        <table border="1px" cellspacing="0" cellpadding="5">
            <%
                for(int i = 0;i < allUsers.size();i++){
                    User user = allUsers.get(i);
            %>
                <tr>
                    <td><%=user.getId() %></td>
                    <td><%=user.getUsername() %></td>
                    <td><%=user.getPassword() %></td>
                    <td><%=user.getEmail() %></td>
                    <td><%=user.getIdentity() %></td>
                </tr>
            <% } %>
        </table>
    </center>
</body>
```

# • MVC架构demo •

12、运行，查看结果

<http://localhost:8080/xkuser/login.jsp>

用户姓名:

用户密码:

1	admin	123456	admin1@126.com	admin
2	test1	123456	test1@qq.com	user
3	test2	123456	test2@qq.com	user
4	test3	123456	test3@qq.com	user
5	test4	123456	test4@qq.com	user
12	mary	123456	mary@sohu.com	admin

## • 作业 •

- 1、在windows实体机或者虚拟机中配置开发环境。
- 2、理解MVC架构体系。
- 3、**编写用户登录功能**，并在欢迎页面显示**欢迎xxx登录**字样，显示全体用户信息。  
用IE8或firefox浏览器调试程序，记录并调试过程中出现的问题。
- 4、项目陈述。