SpringMVC笔记

目录

2.配置pom.xml引入项目依赖jar包

实体类：com.yuntu.curd.bean

控制器：com.yuntu.curd.controller

数据访问层：com.yuntu.curd.dao

业务逻辑层：com.yuntu.curd.service

测试：com.yuntu.curd.test

src/main/resources

web.xml

index.jsp

mbg.xml

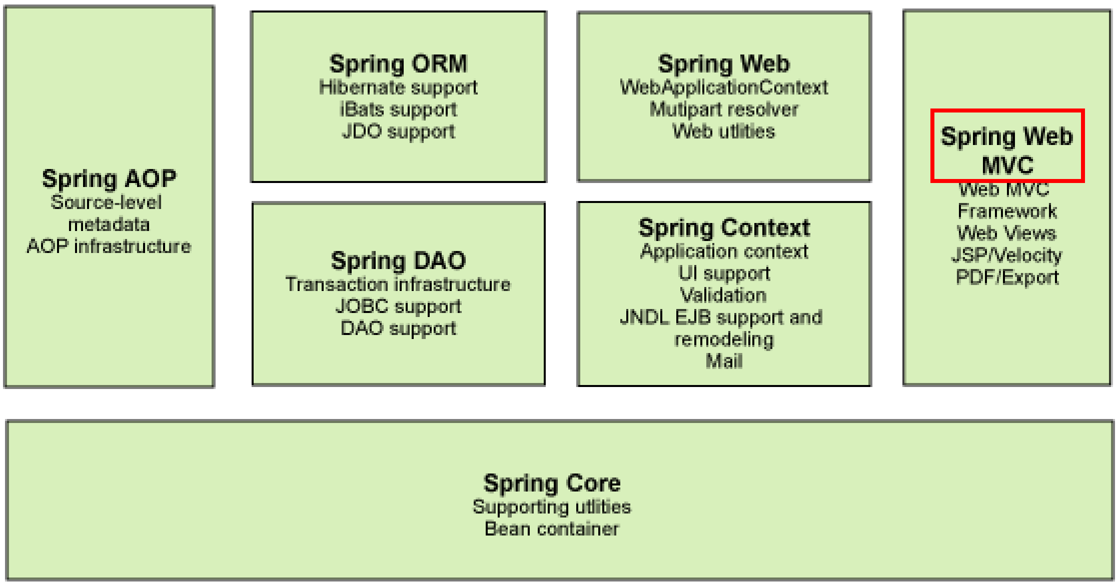
第一章、HelloSpringMVC

什么是SpringMVC?

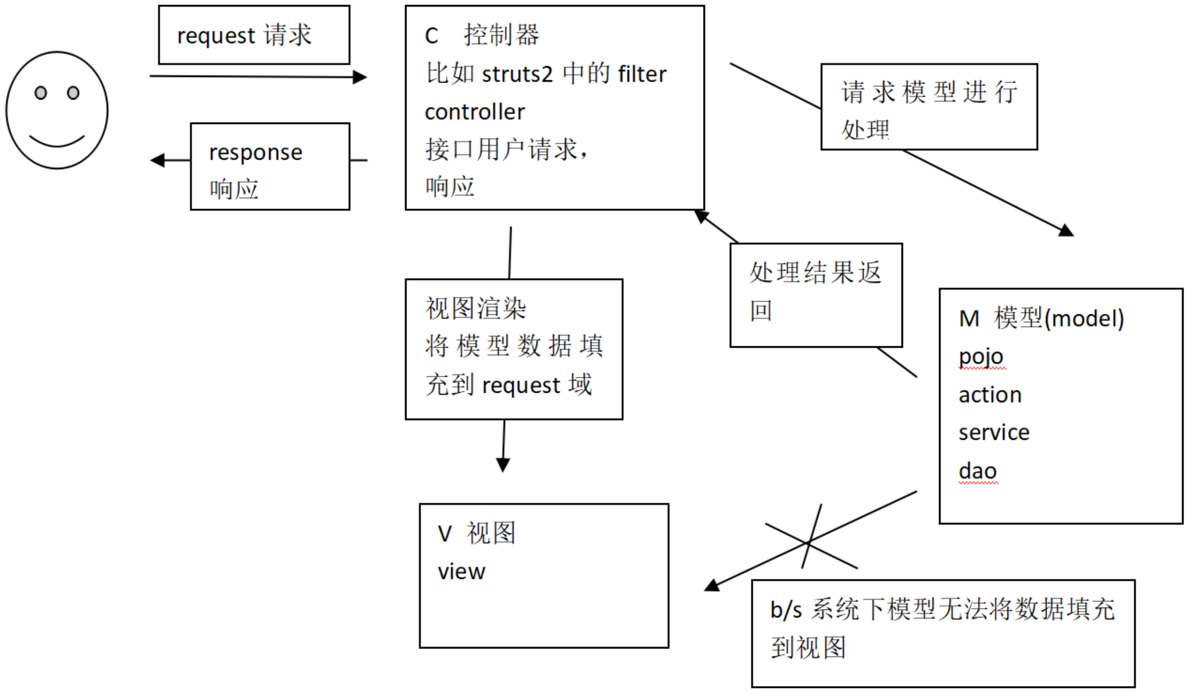
SpringMVC是spring框架的一个模块，springmvc和spring无需通过中间整合层进行整合。

springmvc是一个基于mvc的web框架。

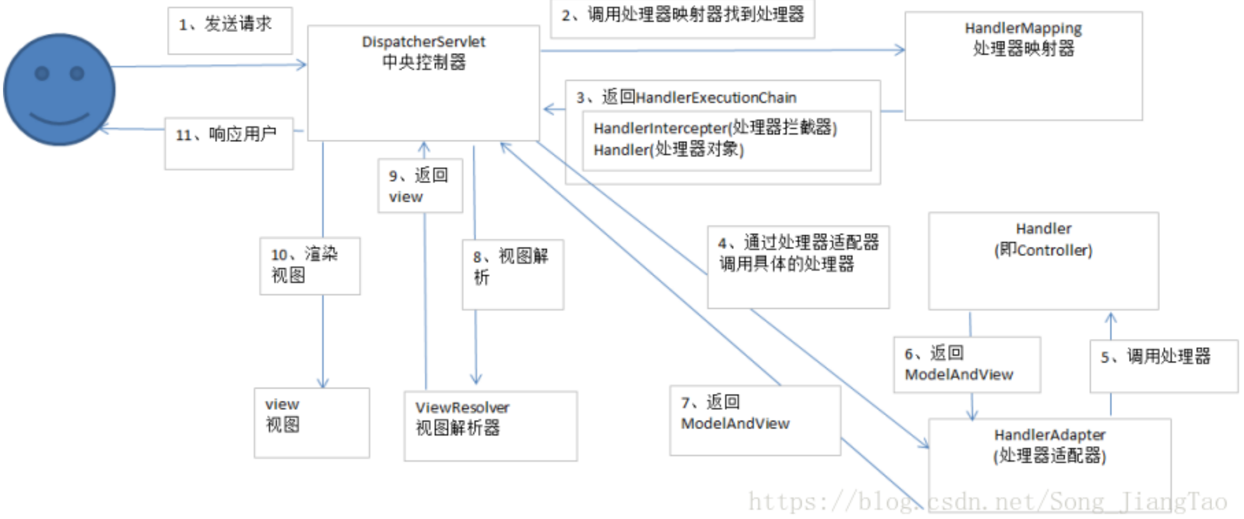
提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用Spring进行WEB开发时，可以选择使用Spring的Spring MVC框架或集成



MVC在b/s系统下的应用：



SpringMVC框架



Model（模型）

- 封装应用状态（封装应用数据）
- 响应状态查询（对数据进行增删改查）
- 暴露应用的功能（暴露接口<public>）

Controller（控制器）

- 验证HTTP请求的数据（收集组织数据）
- 将用户数据与模型的更新相映射（调用逻辑层）
- 选择用于响应的视图（选择下一个界面）

View（视图）

- 产生HTML响应（展示数据）
- 请求模型的更新（人机交互）
- 提供HTML form用于用户请求（收集参数，调用逻辑层api）

MVC的优点：

- 低耦合性：视图层和业务层分离
- 高重用性和可适用性
- 可维护性
- 有利于软件工程化管理
- 提高软件的健壮性

MVC的缺点：

工作量大，增加工作的复杂性，MVC不适合小型甚至中等规模的应用，程序将状态查询封装在Model中，使Model过于臃肿。

如果没有MVC设计模式。程序间的各层之间依赖非常强，耦合度高。严重违背了高内聚低耦合的设计原则。而WebMVC将控制逻辑和功能处理，模型和视图进行了分离。降低耦合**但是WebMVC也有严重的缺点：**

控制器(controller)

1.控制逻辑较为复杂，而且每个模块都需要一个控制器，（可能每一个页面都需要一个控制器）

2.请求参数到模型的封装麻烦。

3.视图的选择严重依赖于Servlet API ，这样很能甚至不可能更换视图

4.给视图传入模型数据 也依赖于Servlet API.如果要更换视图那么技术也要改变。

模型(model)

使用JavaBean组件，（域模型层+业务层+持久层）导致JavaBean组件类庞大。不利于开发人员管理

视图(view)

被绑定于Jsp,很难更换视图

执行流程：

1、用户发送请求至前端控制器DispatcherServlet

2、DispatcherServlet收到请求调用HandlerMapping处理器映射器。

3、处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。

4、DispatcherServlet通过HandlerAdapter处理器适配器调用处理器

5、执行处理器(Controller，也叫后端控制器)。

6、Controller执行完成返回ModelAndView

7、HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet

8、DispatcherServlet将ModelAndView传给ViewReslover视图解析器

9、ViewReslover解析后返回具体View

10、DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）。

11、DispatcherServlet响应用户

Spring MVC运行原理

整个处理过程从一个HTTP请求开始：

1.Tomcat在启动时加载解析web.xml,找到spring mvc的前端总控制器DispatcherServlet,并且通过DispatcherServlet来加载相关的配置文件信息。

2.DispatcherServlet接收到客户端请求，找到对应HandlerMapping，根据映射规则，找到对应的处理器（Handler）。

3.调用相应处理器中的处理方法，处理该请求后，会返回一个ModelAndView。

4.DispatcherServlet根据得到的ModelAndView中的视图对象，找到一个合适的ViewResolver（视图解析器），根据视图解析器的配置，DispatcherServlet将要显示的数据传给对应的视图，最后显示给用户。

SpringMVC框架原理

前端控制器DispatcherServlet（不需要程序员开发）

作用：

接收请求，响应结果，相当于转发器，中央处理器。有了DispatcherServlet减少了其它组件之间的耦合度。

处理器映射器HandlerMapping(不需要程序员开发)

作用：根据请求的url查找Handler

处理器适配器HandlerAdapter

作用：按照特定规则（HandlerAdapter要求的规则）去执行Handler

处理器Handler (需要程序员开发)

注意：编写Handler时按照HandlerAdapter的要求去做，这样适配器才可以去正确执行Handler

视图解析器View resolver(不需要程序员开发)

作用：进行视图解析，根据逻辑视图名解析成真正的视图（view）

视图View (需要程序员开发)

View是一个接口，实现类支持不同的View类型（jsp、freemarker、pdf...）

如何解决WebMVC的缺点

从服务到工作者模式：

服务到工作者：Front Controller + Application Controller + Page Controller + Context即，前端控制器+应用控制器+页面控制器（也有称其为动作）+上下文

前端控制器:

负责为表现层提供统一访问点，从而避免WebMVC中出现的重复的控制逻辑,可以为多个请求提供共用的逻辑（如准备上下文等等），
将选择具体视图和具体的功能处理分离。

应用控制器:

前端控制器分离选择具体视图和具体的功能处理之后，需要有人来管理，应用控制器就是用来选择具体视图技术（视图的管理）和具体的功能处理（页面控制器/命令对象/动作管理），一种策略设计模式的应用，可以很容易的切换视图/页面控制器，相互不产生影响

页面控制器/动作/处理器：

功能处理代码，收集参数、封装参数到模型，转调业务对象处理模型，返回逻辑视图名交给前端控制器（和具体的视图技术解耦），由前端控制器委托给应用控制器选择具体的视图来展示，可以是命令设计模式的实现。页面控制器也被称为处理器或动作。

上下文

WebMVC中为视图准备要展示的模型数据，我们直接放在request中（Servlet API相关），有了上下文之后，我们就可以将相关数据放置在上下文，从而与协议无关（如Servlet API）的访问/设置模型数据，一般通过ThreadLocal模式实现。

SpringMVC常用注解：

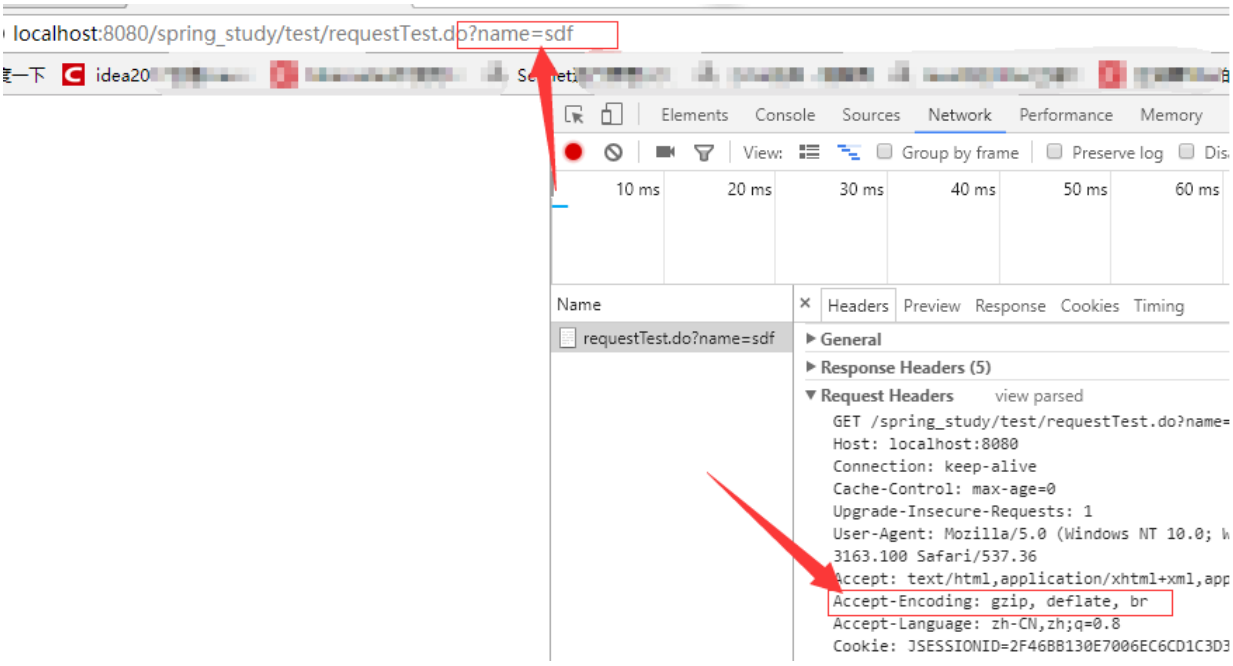
@Controller：注解一个类表示控制器，Spring MVC会自动扫描标注了这个注解的类。

@RequestMapping：用于处理请求地址映射，可以作用于类和方法上。

- value：定义request请求的映射地址
- method：定义request地址请求的方式，包括【GET, POST, HEAD, OPTIONS, PUT, PATCH, DELETE, TRACE.】默认接受get请求，如果请求方式和定义的方式不一样则请求无法成功。
- params：定义request请求中必须包含的参数值。
- headers：定义request请求中必须包含某些指定的请求头，如：RequestMapping(value = "/something", headers = "content-type=text/*")说明请求中必须要包含"text/html", "text/plain"这中类型的Content-type头，才是一个匹配的请求。
- consumes：定义请求提交内容的类型。
- produces：指定返回的内容类型，仅当request请求头中的(Accept)类型中包含该指定类型才返回

```
@RequestMapping(value="/requestTest.do",params = {"name=sdf"},headers = {"Accept-Encoding=gzip, deflate, br"},method = RequestMethod.GET)
2 public String getIndex(){
3 System.out.println("请求成功");
4 return "index";
5 }
```

> 上面代码表示请求的方式为GET请求，请求参数必须包含name=sdf这一参数，然后请求头中必须有 Accept-Encoding=gzip, deflate, br这个类型头。



```
GET /spring_study/test/requestTest.do?name=
Host: localhost:8080
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; W
3163.100 Safari/537.36
Accept: text/html,application/xhtml+xml,app
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=2F46BB130E7006EC6CD1C3D3
```

这样通过注解就能对一个请求进行约束了。

@RequestParam：用于获取传入参数的值

- value：参数的名称
- required：定义该传入参数是否必须，默认为true，（和@RequestMapping的params属性有点类似）

```
@RequestMapping("/requestParams1.do")
public String requestParams1(@RequestParam(required = false) St
ring name){
System.out.println("name = "+name);
return "index";
}
@RequestMapping("/requestParams2.do")
public String requestParams2(@RequestParam(value = "name",requi
red = false) String names){
System.out.println("name = "+names);
return "index";
 }
```

> 两种请入参方式是一样的，显示声明value的名称时，入参参数名和value一样，没有显示声明的话，像第一种方式声明的，入参参数名和函数参数变量名一样。

@PathViriable：用于定义路径参数值

- value：参数的名称

- required：定义传入参数是否为必须值

```
@RequestMapping("/{myname}/pathVariable2.do")
 public String pathVariable2(@PathVariable(value = "myname") String name){
 System.out.println("myname = "+name);
 return "index";
 }
```

> 这个路径声明了{myname}作为路径参数，那么这一段路径将为任意值，@PathVariable将可以根据value获取路径的值。

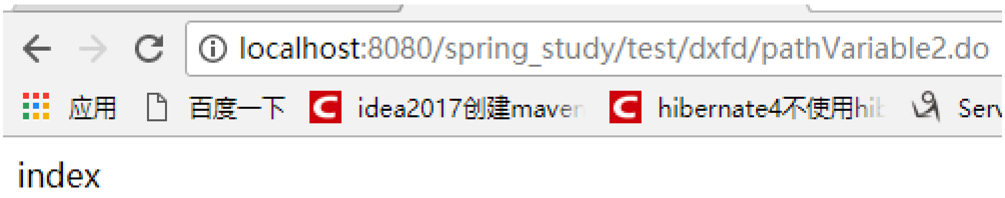
@RequestBody

放在参数前，表示参数从request body中获取，而不是从地址栏获取，所以这肯定是接收一个POST请求的非a=b格式的数据，即 Content-Type不为application/x-www-form-urlencoded类型的内容。

@ResponseBody：作用于方法上，可以将整个返回结果以某种格式返回，如json或xml格式。

```
@RequestMapping("/{myname}/pathVariable2.do")
 @ResponseBody
 public String pathVariable2(@PathVariable(value = "myname") String name){
 System.out.println("myname = "+name);
 return "index";
 }
```

和上一段例子一样，只是添加了@ResponseBody标签，然后返回结果为：



localhost:8080/spring_study/test/dxfd/pathVariable2.do

index

它返回的不是一个页面，而是把字符串“index”直接在页面打印出来了，这其实和如下代码时类似的。

```
PrintWriter out = resp.getWriter();
 out.print("index");
 out.flush();
```

@CookieValue：用于获取请求的Cookie值

```
@RequestMapping("/requestParams.do")
 public String requestParams(@CookieValue("JSESSIONID") String cookie){
 return "index";
 }
```

@ModelAttribute：用于把参数保存到model中

@ModelAttribute：用于把参数保存到model中，可以注解方法或参数，注解在方法上的时候，该方法将在处理器方法执行之前执行，然后把返回的对象存放在 session（前提时要有@SessionAttributes注解） 或模型属性中，@ModelAttribute(“attributeName”) 在标记方法的时候指定，若未指定，则使用返回类型的类名称（首字母小写）作为属性名称。

```
@ModelAttribute("user")
 public UserEntity getUser(){
 UserEntity userEntityr = new UserEntity();
 userEntityr.setUsername("asdf");
 return userEntityr;
 }

 @RequestMapping("/modelTest.do")
 public String getUsers(@ModelAttribute("user") UserEntity user){
  System.out.println(user.getUsername());
  return "/index";
  }
```

如上代码中，使用了@ModelAttribute("user")注解，在执行控制器前执行，然后将生成一个名称为user的model数据，在控制器中我们通过注解在参数上的@ModelAttribute获取参数，然后将model应用到控制器中，在jsp页面中我们同样可以使用它，

```
<body>
 ${user.username}
</body>
```

@SessionAttributes

默认情况下Spring MVC将模型中的数据存储到request域中。当一个请求结束后，数据就失效了。如果要跨页面使用。那么需要使用到session。而@SessionAttributes注解就可以使得模型中的数据存储一份到session域中。配合@ModelAttribute("user")使用的时候,会将对应的名称的model值存到session中，

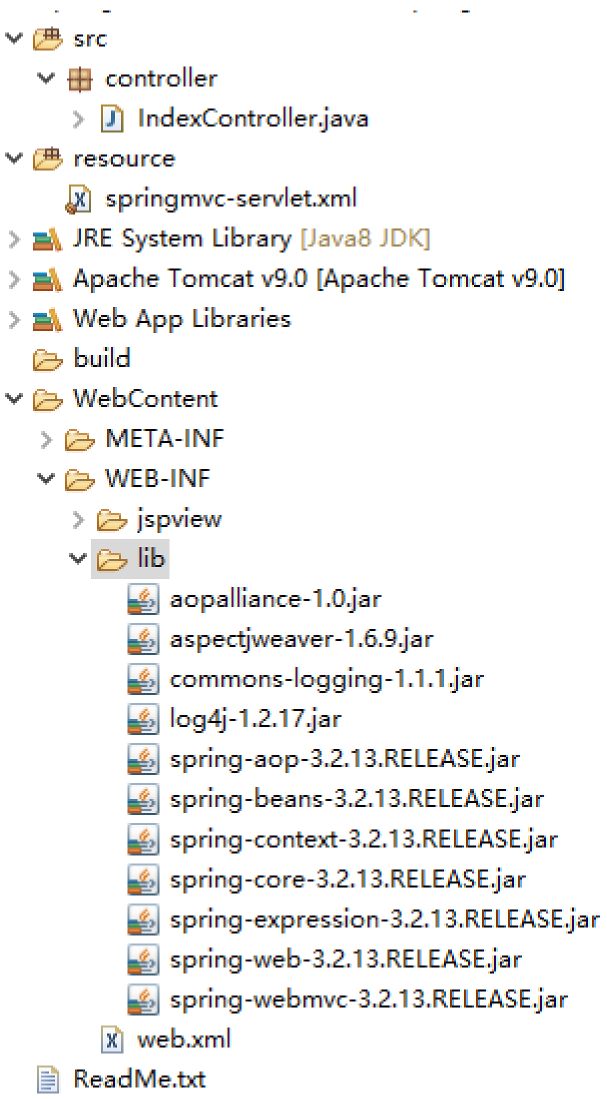
```
@Controller
@RequestMapping("/test")
@SessionAttributes(value = {"user","test1"})
public class LoginController{
 @ModelAttribute("user")
 public UserEntity getUser(){
 UserEntity userEntityr = new UserEntity();
 userEntityr.setUsername("asdf");
 return userEntityr;
  }

  @RequestMapping("/modelTest.do")
  public String getUsers(@ModelAttribute("user") UserEntity user
,HttpSession session){
  System.out.println(user.getUsername());
  System.out.println(session.getAttribute("user"));
  return "/index";
  }
}
```

> 结合上一个例子的代码，加了@SessionAttributes注解，然后请求了两次，第一次session中不存在属性名为user的值，第二次请求的时候发现session中又有了，这是因为，这是因为第一次请求时，model数据还未保存到session中请求结束返回的时候才保存，在第二次请求的时候已经可以获取上一次的model了，

注意：@ModelAttribute("user") UserEntity user获取注解内容的时候，会先查询session中是否有对应的属性值，没有才去查询Model。

案例-HelloSpringMVC使用BeanNameUrlHandlerMapping

项目结构：



```
src
  controller
    IndexController.java
resource
  springmvc-servlet.xml
JRE System Library [Java8 JDK]
Apache Tomcat v9.0 [Apache Tomcat v9.0]
Web App Libraries
build
WebContent
  META-INF
  WEB-INF
    jspview
    lib
      aopalliance-1.0.jar
      aspectjweaver-1.6.9.jar
      commons-logging-1.1.1.jar
      log4j-1.2.17.jar
      spring-aop-3.2.13.RELEASE.jar
      spring-beans-3.2.13.RELEASE.jar
      spring-context-3.2.13.RELEASE.jar
      spring-core-3.2.13.RELEASE.jar
      spring-expression-3.2.13.RELEASE.jar
      spring-web-3.2.13.RELEASE.jar
      spring-webmvc-3.2.13.RELEASE.jar
    web.xml
ReadMe.txt
```

IndexController.java

```java
package controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.AbstractController;

public class IndexController extends AbstractController {

    @Override
    protected ModelAndView handleRequestInternal(HttpServletRequest arg0, HttpServletResponse arg1) throws Exception {
    System.out.println("hello.springmvc的controller");
    //返回视图模型，名字为index的
```

```
  return new ModelAndView("index");
  }

}

```

springmvc-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:mvc="http://www.springframework.org/schema/mvc"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:task="http://www.springframework.org/schema/task"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.2.xsd
  http://www.springframework.org/schema/mvc
  http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
  http://www.springframework.org/schema/aop
  http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
  http://www.springframework.org/schema/tx
  http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
  http://www.springframework.org/schema/task
  http://www.springframework.org/schema/task/spring-task-3.2.xsd">

  <!-- 通过BeanNameUrlHandlerMapping方式完成请求与Controller之间的映射关系，若有多个请求
  就在springmvc-servlet.xml文件中配置多个映射关系，同时要建立多个javaBean作为控制器进行处
  理的处理
  缺点：
  配置过多，controller过多，业务复杂的话，是不合理的
  -->
  <!-- 页面的转化对象，用来配置访问名字 -->
  <bean name="/index.html" class="controller.IndexController"></bean>

  <!-- 完成试图的对应处理-->
  <!-- 处理路径问题
  对转向页面的路径解析；
  prefix：前缀
  suffix：后缀
  -->
  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix" value="/WEB-INF/jspview/"></property>
  <property name="suffix" value=".jsp"></property>
  </bean>
 </beans>
```

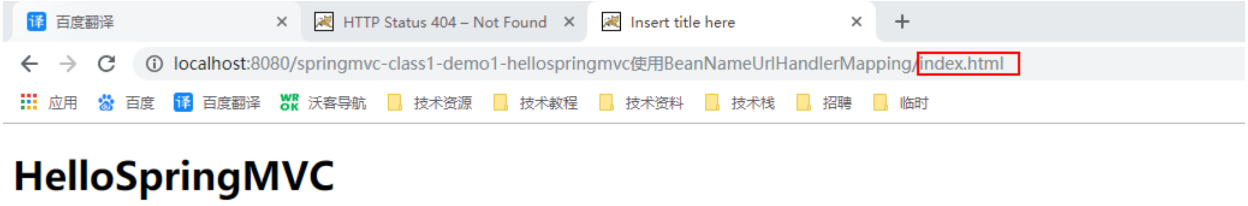
index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
 <h1>HelloSpringMVC</h1>
</body>
</html>
```

WEB-INF/web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID" version="4.0">
  <display-name>springmvc-class1-demo1-hellospringmvc使用BeanNameUrlHandlerMapping</display-name>

  <servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <!-- 初始参数 -->
  <init-param>
   <!-- 初始化springmvc上下文配置的路径 -->
   <param-name>contextConfigLocation</param-name>
   <param-value>classpath:springmvc-servlet.xml</param-value>
   </init-param>

   <!-- 是否应该在启动的时候就加载当前这个servlet，是一个整数；
   数值表示servlet应该被载入的顺序，值越小则该servlet优先级越高，应用启动是就越先加载
   -->
   <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
   <servlet-name>springmvc</servlet-name>
   <url-pattern>/</url-pattern>
   </servlet-mapping>

 </web-app>
```

程序运行结果：

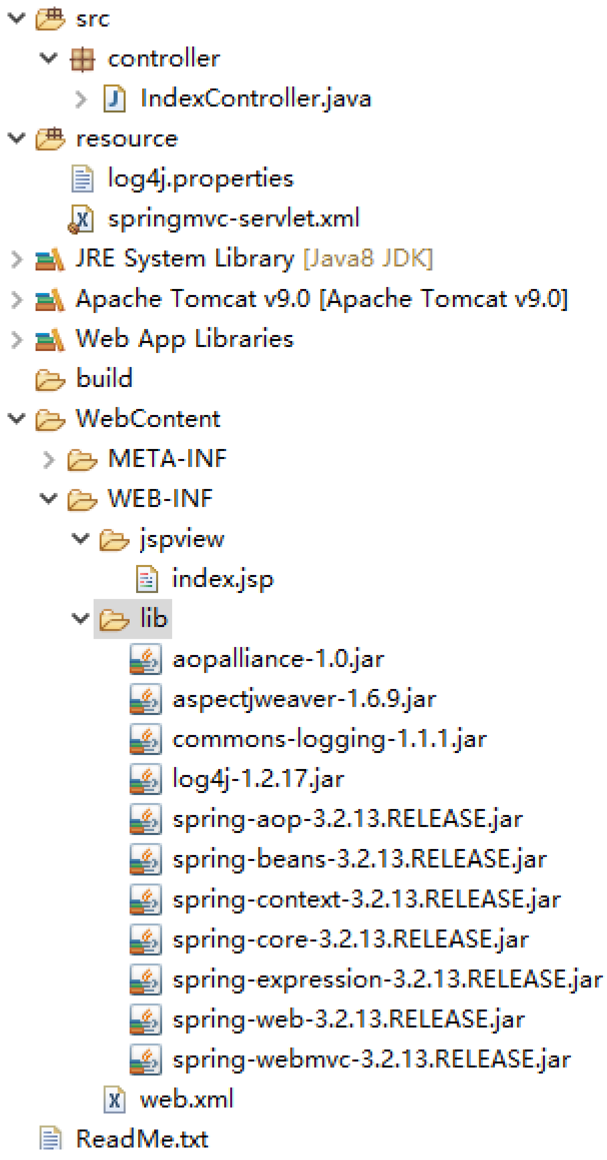


localhost:8080/springmvc-class1-demo1-hellospringmvc使用BeanNameUrlHandlerMapping/index.html

HelloSpringMVC

案例-HelloSpringMVC使用注解

项目结构：



```
src
  controller
    IndexController.java
resource
  log4j.properties
  springmvc-servlet.xml
JRE System Library [Java8 JDK]
Apache Tomcat v9.0 [Apache Tomcat v9.0]
Web App Libraries
build
WebContent
  META-INF
  WEB-INF
    jspview
      index.jsp
    lib
      aopalliance-1.0.jar
      aspectjweaver-1.6.9.jar
      commons-logging-1.1.1.jar
      log4j-1.2.17.jar
      spring-aop-3.2.13.RELEASE.jar
      spring-beans-3.2.13.RELEASE.jar
      spring-context-3.2.13.RELEASE.jar
      spring-core-3.2.13.RELEASE.jar
      spring-expression-3.2.13.RELEASE.jar
      spring-web-3.2.13.RELEASE.jar
      spring-webmvc-3.2.13.RELEASE.jar
    web.xml
ReadMe.txt
```

IndexController.java

```java
package controller;
```

```java

import org.apache.log4j.Logger;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

//当前类为控制器类（进行页面的映射）
@Controller
@RequestMapping("/login")
 public class IndexController{
  private Logger log=Logger.getLogger(IndexController.class);

  //@RequestMapping(请求的映射路径):表示用哪一个url来对应此方法执行
  @RequestMapping("/indexview")
  public String index() {
  log.info("hello.springmvc,注解形式");
  return "index";
  }
 }

```

springmvc-servlet.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:mvc="http://www.springframework.org/schema/mvc"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:task="http://www.springframework.org/schema/task"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
  http://www.springframework.org/schema/context
 http://www.springframework.org/schema/context/spring-context-3.2.xsd
 http://www.springframework.org/schema/mvc
 http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
 http://www.springframework.org/schema/aop
 http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
 http://www.springframework.org/schema/tx
 http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
 http://www.springframework.org/schema/task
 http://www.springframework.org/schema/task/spring-task-3.2.xsd">

 <!-- 扫描注解的控制器包 -->
 <context:component-scan base-package="controller"></context:component-scan>
 <!-- 一键式配置，通过注解方式进行spingmvc开发 -->
 <mvc:annotation-driven></mvc:annotation-driven>

 <!-- 完成试图的对应处理-->
 <!-- 处理路径问题
 对转向页面的路径解析；
 prefix：前缀
 suffix：后缀
 -->
 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
 <property name="prefix" value="/WEB-INF/jspview/"></property>
 <property name="suffix" value=".jsp"></property>
 </bean>
</beans>
```

WebContent/WEB-INF/jspview/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
```

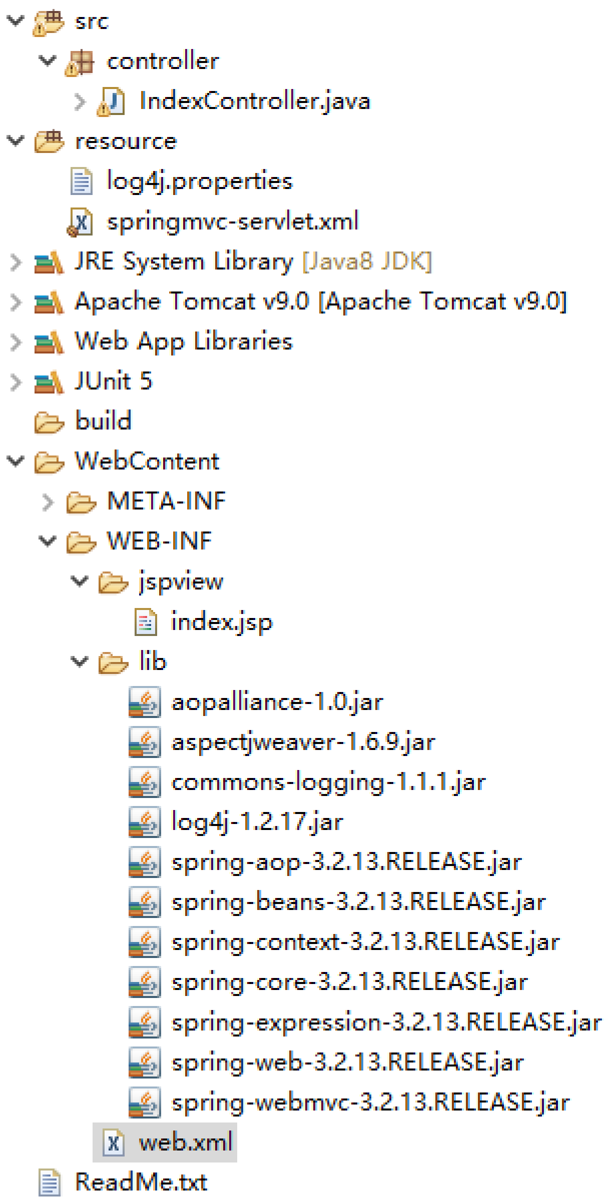
```
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
 <h1>HelloSpringMVC,注解形式</h1>
</body>
</html>
```

WebContent/WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" version="4.0">
 <display-name>springmvc-class1-demo2-hellospringmvc使用注解</display-name>
 <welcome-file-list>
 <welcome-file>index.html</welcome-file>
 <welcome-file>index.htm</welcome-file>
 <welcome-file>index.jsp</welcome-file>
 <welcome-file>default.html</welcome-file>
 <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <!-- 初始参数 -->
  <init-param>
  <!-- 初始化springmvc上下文配置的路径 -->
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:springmvc-servlet.xml</param-value>
  </init-param>

  <!-- 是否应该在启动的时候就加载当前这个servlet，是一个整数；
  数值表示servlet应该被载入的顺序，值越小则该servlet优先级越高，应用启动是就越先加载
  -->
  <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

案例-参数传递从视图到控制器

项目结构:



```
src
  controller
    IndexController.java
resource
  log4j.properties
  springmvc-servlet.xml
JRE System Library [Java8 JDK]
Apache Tomcat v9.0 [Apache Tomcat v9.0]
Web App Libraries
JUnit 5
build
WebContent
  META-INF
  WEB-INF
    jspview
      index.jsp
    lib
      aopalliance-1.0.jar
      aspectjweaver-1.6.9.jar
      commons-logging-1.1.1.jar
      log4j-1.2.17.jar
      spring-aop-3.2.13.RELEASE.jar
      spring-beans-3.2.13.RELEASE.jar
      spring-context-3.2.13.RELEASE.jar
      spring-core-3.2.13.RELEASE.jar
      spring-expression-3.2.13.RELEASE.jar
      spring-web-3.2.13.RELEASE.jar
      spring-webmvc-3.2.13.RELEASE.jar
    web.xml
ReadMe.txt
```

IndexController.java

```java
package controller;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

//当前类为控制器类（进行页面的映射）
@Controller
public class IndexController{
 private Logger log=Logger.getLogger(IndexController.class);

 //@RequestMapping(请求的映射路径):表示用哪一个url来对应此方法执行
 @RequestMapping("/indexview")
 public String index() {
 log.info("hello.springmvc,注解形式");
 return "index";
 }


 /**
 * 参数传递： view to controller
 * 视图层页面的请求路径向控制器中指定的方法跳转并携带参数
 * 控制器需要对应接收携带的信息
 */
// @RequestMapping("/welcome")
 //@RequestParam：请求中携带的参数信息
 //（如果页面携带的参数名称不是username，可以在@RequestParam("携带参数名称")指明携带参数的名称）
// public String welcome(@RequestParam("uname") String username) {
// log.info("接收的内容："+username);
// return "index";
// }

 //方法的请求映射结果，任意一个均可访问
 //@RequestMapping({"/welcome","/"})
// @RequestParam：请求中携带的参数信息
// （如果页面携带的参数名称不是username，可以在@RequestParam("携带参数名称")指明携带参数的名称）
 //value：参数的名称
 //required：定义该传入参数是否必须，默认为true
// public String welcome(@RequestParam(value="uname",required=false) String username) {
// log.info("接收的内容："+username);
// return "index";
// }

```

```java

    /**
    * 请求映射配置中，直接添加设置请求方法和所携带参数
    * value：定义request请求的映射地址
    * method：定义地request址请求的方式，包括
    *【GET, POST, HEAD, OPTIONS, PUT, PATCH, DELETE, TRACE.】默认
    * 接受get请求，如果请求方式和定义的方式不一样则请求无法成功。
    * params：定义request请求中必须包含的参数值。
    */

    @RequestMapping(value="/welcome",method=RequestMethod.GET)
    public String welcome(String username) {
    log.info("接收的内容："+username);
    return "index";
    }
}
```

springmvc-servlet.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:mvc="http://www.springframework.org/schema/mvc"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:task="http://www.springframework.org/schema/task"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.2.xsd
  http://www.springframework.org/schema/mvc
```

```
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-3.2.xsd">

<!-- 扫描注解的控制器包 -->
<context:component-scan base-package="controller"></context:component-scan>
<!-- 一键式配置，通过注解方式进行spingmvc开发 -->
<mvc:annotation-driven></mvc:annotation-driven>

<!-- 完成视图的对应处理-->
<!-- 处理路径问题
对转向页面的路径解析；
prefix：前缀
suffix：后缀
-->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="prefix" value="/WEB-INF/jspview/"></property>
<property name="suffix" value=".jsp"></property>
</bean>
</beans>
```

WebContent/WEB-INF/jspview/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
```

```
</head>
<body>
  <h1>HelloSpringMVC</h1>
 </body>
 </html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" id="WebApp_ID" version="4.0">
  <display-name>springmvc-class1-demo3-参数传递从视图到控制器</display-name>
  <welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
   <welcome-file>default.jsp</welcome-file>
   </welcome-file-list>

   <servlet>
   <servlet-name>springmvc</servlet-name>
   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
   <!-- 初始参数 -->
   <init-param>
   <!-- 初始化springmvc上下文配置的路径 -->
   <param-name>contextConfigLocation</param-name>
   <param-value>classpath:springmvc-servlet.xml</param-value>
   </init-param>

   <!-- 是否应该在启动的时候就加载当前这个servlet，是一个整数；
   数值表示servlet应该被载入的顺序，值越小则该servlet优先级越高，应用启动是就越先加载
```

```
 -->
 <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
 <servlet-name>springmvc</servlet-name>
 <url-pattern>/</url-pattern>
 </servlet-mapping>
</web-app>
```

案例-参数传递从控制器到视图-使用ModelAndView

> 此处仅展示部分代码，其他相关代码，见上一案例

IndexController.java

```java
package controller;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

//当前类为控制器类（进行页面的映射）
@Controller
public class IndexController{
 private Logger log=Logger.getLogger(IndexController.class);

 //@RequestMapping(请求的映射路径):表示用哪一个url来对应此方法执行
 @RequestMapping("/indexview")
 public String index() {
 log.info("hello.springmvc,注解形式");
 return "index";
 }

 /*
 * 参数传递：controller to view -ModelAndView
 * 从控制器向视图中传递数据，以返回后ModelAndView的方式
 * 既包含视图信息，又包含数据信息
 */
 @RequestMapping("/index1")
 public ModelAndView index1(String username) {
 log.info("控制器中的数据："+username);
 //创建视图信息的和数据信息的对象
 ModelAndView mView = new ModelAndView();
 //添加数据到mView中
 mView.addObject("username",username);
 //添加视图mView中
 mView.setViewName("index");
 //返回
 return mView;
 }
}

```

WebContent/WEB-INF/jspview/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <h1>HelloSpringMVC + ${username }</h1>
 </body>
 </html>
```

案例-参数传递从控制器向视图-使用Model对象

User.java

```java
package entity;

public class User {
 private Integer id;
 private String uName;
 private Integer uAge;

 public User(Integer id, String uName, Integer uAge) {
 super();
  this.id = id;
  this.uName = uName;
  this.uAge = uAge;
  }
  public User() {
  super();
  }

  public Integer getId() {
  return id;
  }
  public void setId(Integer id) {
  this.id = id;
  }
  public String getuName() {
  return uName;
  }
  public void setuName(String uName) {
  this.uName = uName;
  }
  public Integer getuAge() {
  return uAge;
  }
  public void setuAge(Integer uAge) {
  this.uAge = uAge;
 }
}

```

IndexController.java

```
package controller;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import entity.User;

//当前类为控制器类（进行页面的映射）
@Controller
public class IndexController{
 private Logger log=Logger.getLogger(IndexController.class);

 //@RequestMapping(请求的映射路径):表示用哪一个url来对应此方法执行
 @RequestMapping("/indexview")
 public String index() {
 log.info("hello.springmvc,注解形式");
 return "index";
 }

 /*
 * 参数传递：controller to view -ModelAndView
 * 从控制器向视图中传递数据，以返回后ModelAndView的方式
 * 既包含视图信息，又包含数据信息
 */
 @RequestMapping("/index1")
 public ModelAndView index1(String username) {
 log.info("控制器中的数据："+username);
```

```
//创建视图信息的和数据信息的对象
ModelAndView mView = new ModelAndView();
//添加数据到mView中
mView.addObject("username",username);
//添加视图mView中
mView.setViewName("index");
//返回
return mView;
}

/**
* 参数传递 controller to view
* 控制器处理方法返回的是String（逻辑视图名称），模型数据信息放入Model中；
*
* @param username
* @param model
* @return
*/
@RequestMapping("/index2")
public String index2(String username,Model model) {
log.info("控制器中的数据："+username);

//向Model中放入属性字符串
model.addAttribute("currentusername",username+"字符串");
//直接放入字符串
model.addAttribute(username);
//________________________________
User user = new User();
user.setuName(username);
//向Model中传递对象
model.addAttribute("currentuser",user);
//放入对象
model.addAttribute(user);

return "index";
```

```
 }
}

```

WebContent/WEB-INF/jspview/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
 <h1>名字（key为currentusername） + ${currentusername }</h1>
 <h1>名字（key为string） + ${string }</h1>
 <h1>名字（key为currentuser） + ${currentuser.uName }</h1>
 <h1>名字（key为user） + ${user.uName }</h1>
</body>
</html>
```

案例-传递参数从控制器到视图-使用Map

User.java

```
package entity;

public class User {
 private Integer id;
 private String uName;
 private Integer uAge;

 public User(Integer id, String uName, Integer uAge) {
 super();
 this.id = id;
 this.uName = uName;
 this.uAge = uAge;
 }
 public User() {
 super();
 }

 public Integer getId() {
 return id;
 }
 public void setId(Integer id) {
 this.id = id;
 }
 public String getuName() {
 return uName;
 }
 public void setuName(String uName) {
 this.uName = uName;
 }
 public Integer getuAge() {
 return uAge;
 }
 public void setuAge(Integer uAge) {
 this.uAge = uAge;
 }
}
```

IndexController.java

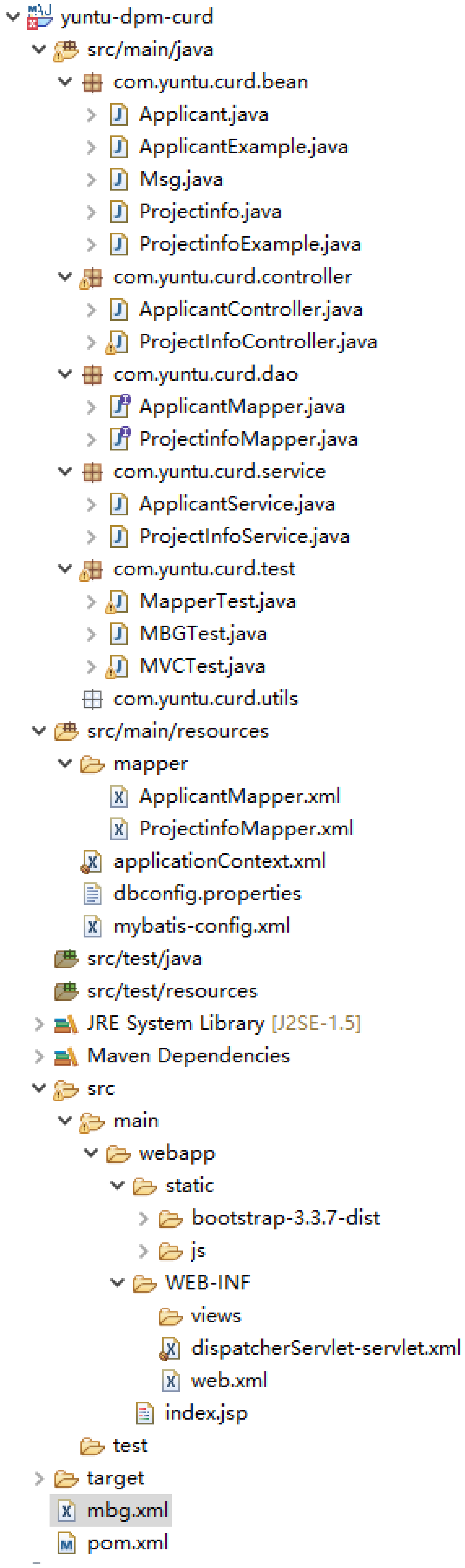
```
package controller;

import java.util.Map;

import org.apache.log4j.Logger;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import entity.User;

//当前类为控制器类（进行页面的映射）
@Controller
public class IndexController{
 private Logger log=Logger.getLogger(IndexController.class);

 /**
 * 参数传递采用的是 controller to view -Map<String, Object>
 * 从ModelAndView和Model中可以发现，SpringMVC的模板就是一个map
 * 的数据结构，故而可以使用Map直接作为处理方法入参
 *
 * SpringMVC的处理方法中若有map或model类型的入参，就会将请求中的隐含的模型对象传递给入参
 * 因此方法体内可以通过这个入参对模型中的数据进行读写
 * 注意：SpringMVC的标准用法中，强烈推荐使用Model
 * @param username
 * @param model
 * @return
 */
 @RequestMapping("/index3")
 public String index3 (String username, Map<String, Object> model) {
 log.info("控制器中的数据："+username);
 model.put("currentusername", username);

 User user = new User();
 user.setuName(username);
 model.put("currentuser", user);

 return "index";
 }
}

```

WebContent/WEB-INF/jspview/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
 <h1>名字（key为currentusername） + ${currentusername }</h1>
 <h1>名字（key为currentuser） + ${currentuser.uName }</h1>
</body>
</html>
```

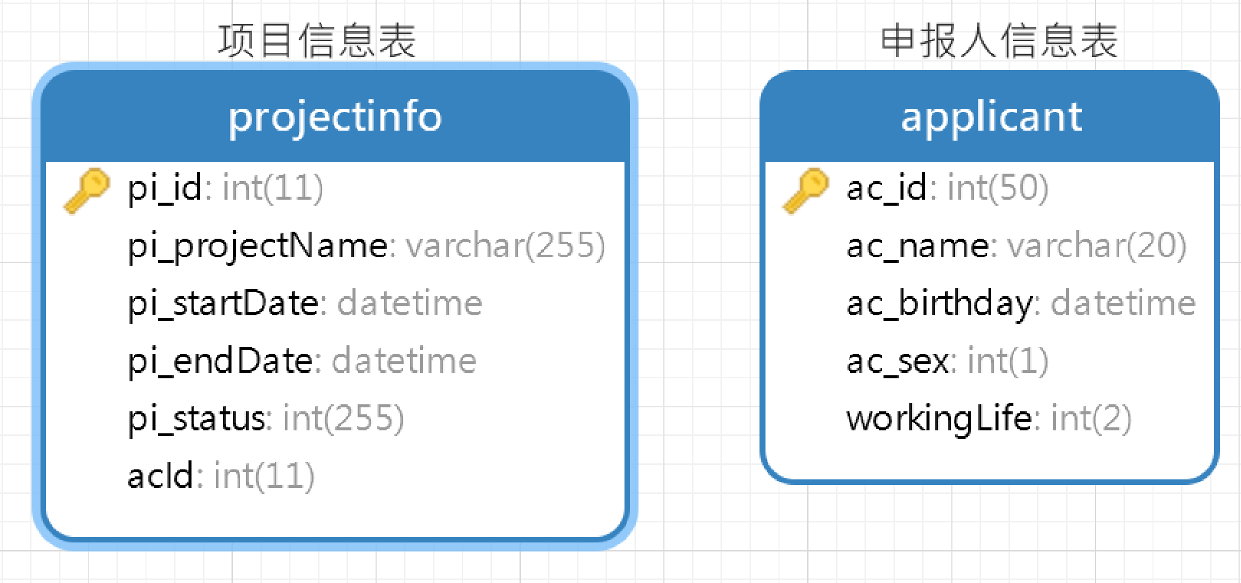
第二章、SSM整合-项目申报系统

项目结构：



```
yuntu-dpm-curd
  src/main/java
    com.yuntu.curd.bean
      Applicant.java
      ApplicantExample.java
      Msg.java
      Projectinfo.java
      ProjectinfoExample.java
    com.yuntu.curd.controller
      ApplicantController.java
      ProjectInfoController.java
    com.yuntu.curd.dao
      ApplicantMapper.java
      ProjectinfoMapper.java
    com.yuntu.curd.service
      ApplicantService.java
      ProjectInfoService.java
    com.yuntu.curd.test
      MapperTest.java
      MBGTest.java
      MVCTest.java
    com.yuntu.curd.utils
  src/main/resources
    mapper
      ApplicantMapper.xml
      ProjectinfoMapper.xml
    applicationContext.xml
    dbconfig.properties
    mybatis-config.xml
  src/test/java
  src/test/resources
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
    main
      webapp
        static
          bootstrap-3.3.7-dist
          js
        WEB-INF
          views
          dispatcherServlet-servlet.xml
          web.xml
        index.jsp
    test
  target
  mbg.xml
  pom.xml
```

数据库表：



项目信息表

projectinfo

- pi_id: int(11)
- pi_projectName: varchar(255)
- pi_startDate: datetime
- pi_endDate: datetime
- pi_status: int(255)
- acId: int(11)

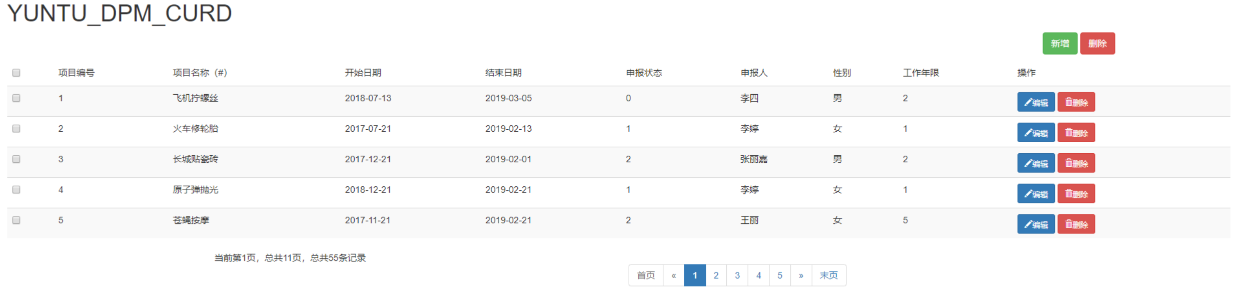
申报人信息表

applicant

- ac_id: int(50)
- ac_name: varchar(20)
- ac_birthday: datetime
- ac_sex: int(1)
- workingLife: int(2)

页面效果

首页

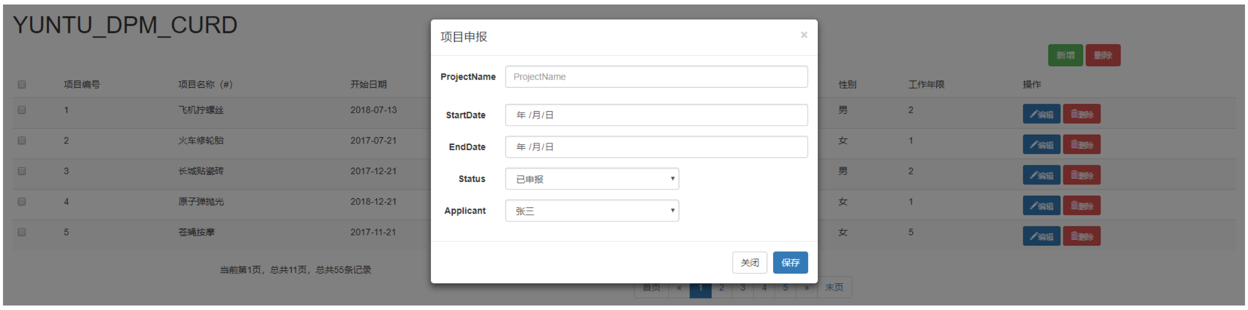


YUNTU_DPM_CURD

□	项目编号	项目名称（#）	开始日期	结束日期	申报状态	申报人	性别	工作年限	操作
□	1	飞机拧螺丝	2018-07-13	2019-03-05	0	李四	男	2	编辑 删除
□	2	火车修轮胎	2017-07-21	2019-02-13	1	李婷	女	1	编辑 删除
□	3	长城贴瓷砖	2017-12-21	2019-02-01	2	张丽嘉	男	2	编辑 删除
□	4	原子弹抛光	2018-12-21	2019-02-21	1	李婷	女	1	编辑 删除
□	5	苍蝇按摩	2017-11-21	2019-02-21	2	王丽	女	5	编辑 删除

当前第1页，总共11页，总共55条记录

新增



YUNTU_DPM_CURD

项目申报

ProjectName: ProjectName

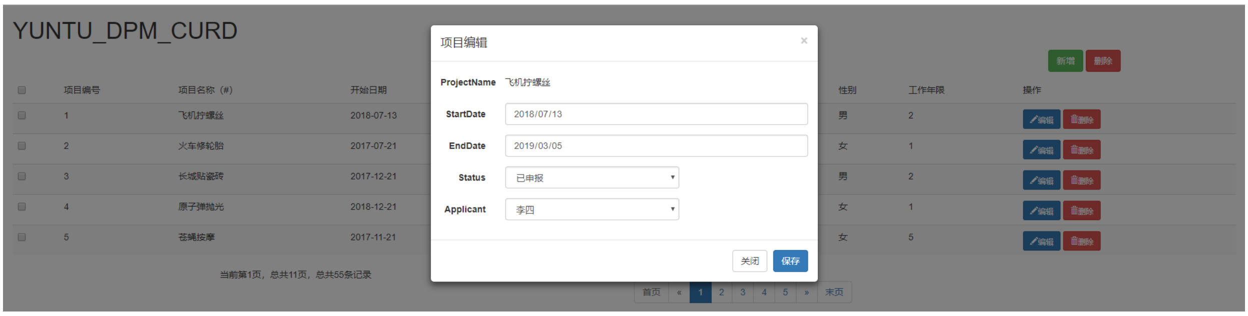
StartDate: 年 /月/日

EndDate: 年 /月/日

Status: 已申报

Applicant: 张三

编辑



YUNTU_DPM_CURD

□	项目编号	项目名称（#）	开始日期	性别	工作年限	操作
□	1	飞机拧螺丝	2018-07-13	男	2	编辑 删除
□	2	火车修轮胎	2017-07-21	女	1	编辑 删除
□	3	长城贴瓷砖	2017-12-21	男	2	编辑 删除
□	4	原子弹抛光	2018-12-21	女	1	编辑 删除
□	5	苍蝇按摩	2017-11-21	女	5	编辑 删除

当前第1页，总共11页，总共55条记录

项目编辑

ProjectName 飞机拧螺丝

StartDate 2018/07/13

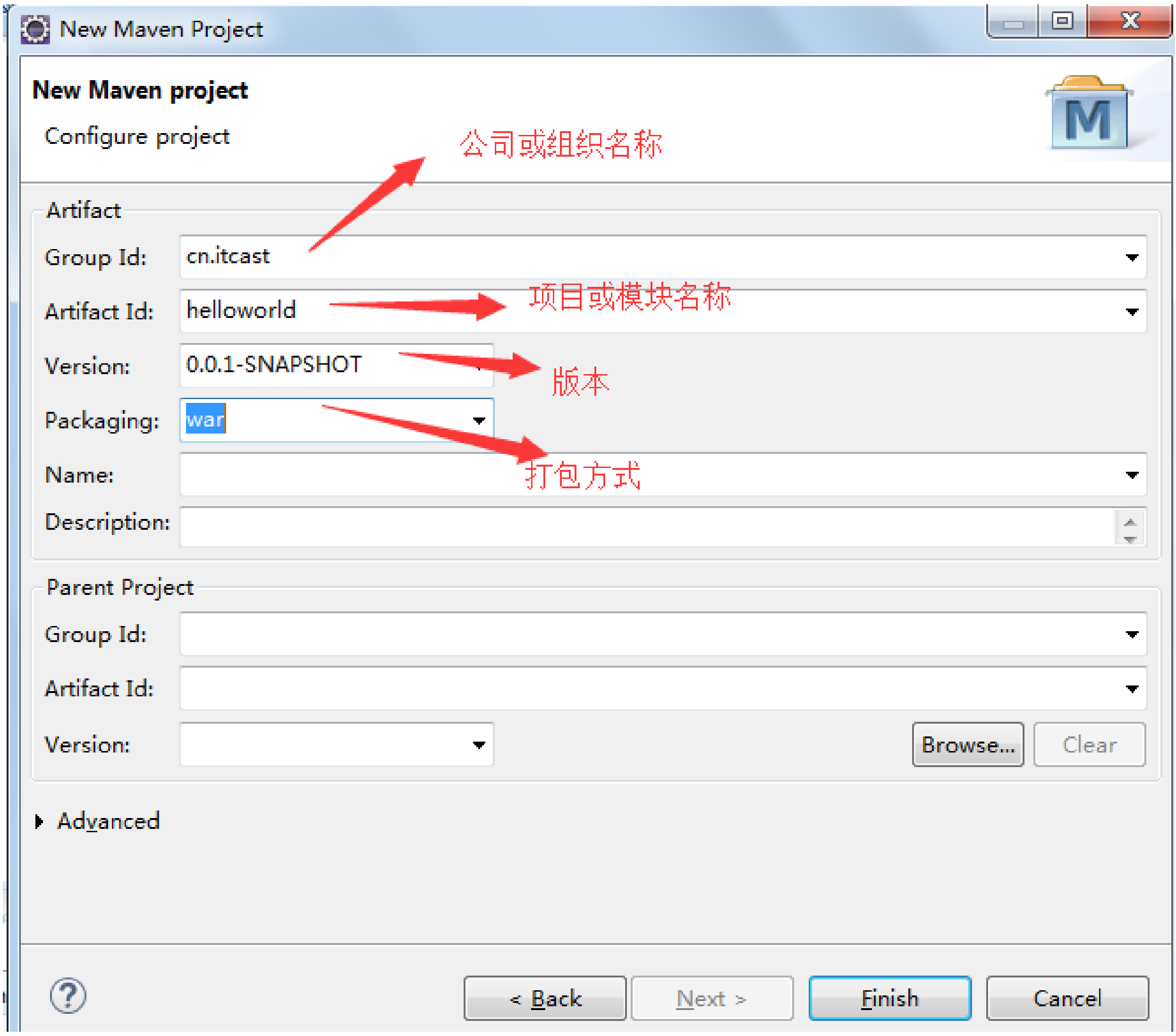
EndDate 2019/03/05

Status 已申报

Applicant 李四

环境搭建

1.创建Maven项目（打成war包）



New Maven Project

New Maven project

Configure project

Artifact

Group Id: cn.itcast

Artifact Id: helloworld

Version: 0.0.1-SNAPSHOT

Packaging: war

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

2.配置pom.xml引入项目依赖jar包

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <!-- -->
```

```xml
<modelVersion>4.0.0</modelVersion>
<!-- 公司或者组织域名倒序+项目名 -->
<groupId>com.yuntu</groupId>
<!-- 模块名 -->
<artifactId>yuntu-dpm-curd</artifactId>
<!-- 版本 -->
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>

<!-- 依赖组 -->
<!-- 引入项目依赖jar包 -->
<dependencies>
<!-- 引入SpringMVC，Spring -->
<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
<dependency>
<!-- jar包所在的package -->
<groupId>org.springframework</groupId>
<!-- jar包名称 -->
<artifactId>spring-webmvc</artifactId>
<!-- jar包版本 -->
<version>4.3.18.RELEASE</version>
</dependency>
<!-- 引入Spring支持的JDBC -->
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>4.3.18.RELEASE</version>
</dependency>
<!-- 引入Spring的面向切面 spring-aspects -->
<!-- https://mvnrepository.com/artifact/org.springframework/spring-aspects -->
<dependency>
<groupId>org.springframework</groupId>
```

```xml
<artifactId>spring-aspects</artifactId>
<version>4.3.18.RELEASE</version>
</dependency>
<!-- 引入Spring提供的测试Spring-test -->
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>4.3.18.RELEASE</version>
</dependency>



<!-- 引入mybatis -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis</artifactId>
<version>3.4.6</version>
</dependency>
<!-- 引入mybatis整合Spring -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis-spring</artifactId>
<version>1.3.1</version>
</dependency>
<!-- 引入数据库连接池，驱动 c3p0 -->
<!-- https://mvnrepository.com/artifact/com.mchange/c3p0 -->
<dependency>
<groupId>com.mchange</groupId>
<artifactId>c3p0</artifactId>
<version>0.9.5.2</version>
```

```xml
</dependency>
<!-- mysql驱动 -->
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.46</version>
</dependency>



<!-- 引入jstl -->
<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
<groupId>jstl</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
<!-- 引入servlet-api -->
<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.0.1</version>
<!--
scope依赖传递
这个标签有以下参数：
compile（编译）
 test（测试）
 runtime（运行时）
 provided（执行前面三个过程，但是不打包（解决Tomcat也有servlet-api的问题））
 system（与provided大致相同，但是依赖项从本地文件系统导入，不走maven仓库） -->
 <scope>provided</scope>
```

```
</dependency>
<!-- 引入junit测试 -->
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.12</version>
</dependency>


<!-- mybatis逆向工程 -->
<!-- https://mvnrepository.com/artifact/org.mybatis.generator/mybatis-generator-core -->
<dependency>
<groupId>org.mybatis.generator</groupId>
<artifactId>mybatis-generator-core</artifactId>
<version>1.3.5</version>
</dependency>


<!-- 引入PageHelper分页插件 -->
<!-- 使用pagehelper只需要在classpath中包含jsqlparser-x.x.x.jar和pagehelper-x.x.x.jar -->
<!-- https://mvnrepository.com/artifact/com.github.pagehelper/pagehelper -->
<dependency>
<groupId>com.github.pagehelper</groupId>
<artifactId>pagehelper</artifactId>
<version>5.0.0</version>
</dependency>

<!-- 引入jackson进行json数据解析 -->
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
<dependency>
```

```xml
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.5</version>
    </dependency>

    <!-- 引入jsr303进行后台数据校验
    说明:
    Tomcat7及以上的服务器时没有问题的，可以直接导入hibernate-validator
    Tomcat7以下的服务器（EL表达式版本过低），需要额外给服务器的lib包中替换新的El包 -->
    <!-- https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-validator -->
    <dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
    </dependency>
    </dependencies>
  </project>
```

实体类：com.yuntu.curd.bean

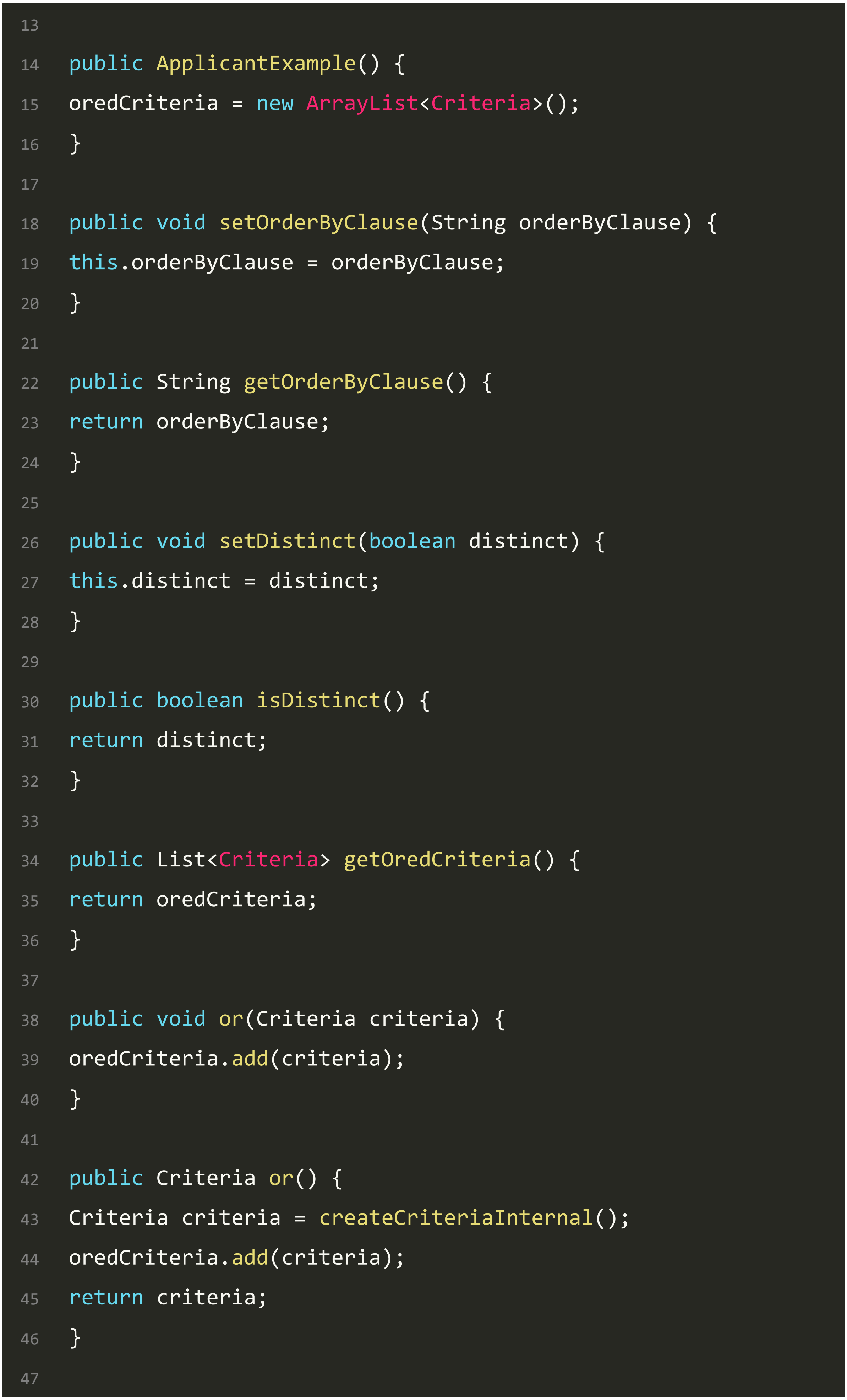
Applicant.java

```java
package com.yuntu.curd.bean;

import java.util.Date;

public class Applicant {
 private Integer acId;

 private String acName;

 private Date acBirthday;

 private Integer acSex;

private Integer workinglife;

public Applicant() {
super();
}

public Applicant(Integer acId, String acName, Date acBirthday, Integer acSex, Integer workinglife) {
super();
this.acId = acId;
this.acName = acName;
this.acBirthday = acBirthday;
this.acSex = acSex;
this.workinglife = workinglife;
}

public Integer getAcId() {
return acId;
}

public void setAcId(Integer acId) {
this.acId = acId;
}

public String getAcName() {
return acName;
}

public void setAcName(String acName) {
this.acName = acName == null ? null : acName.trim();
}

public Date getAcBirthday() {
return acBirthday;
}
```

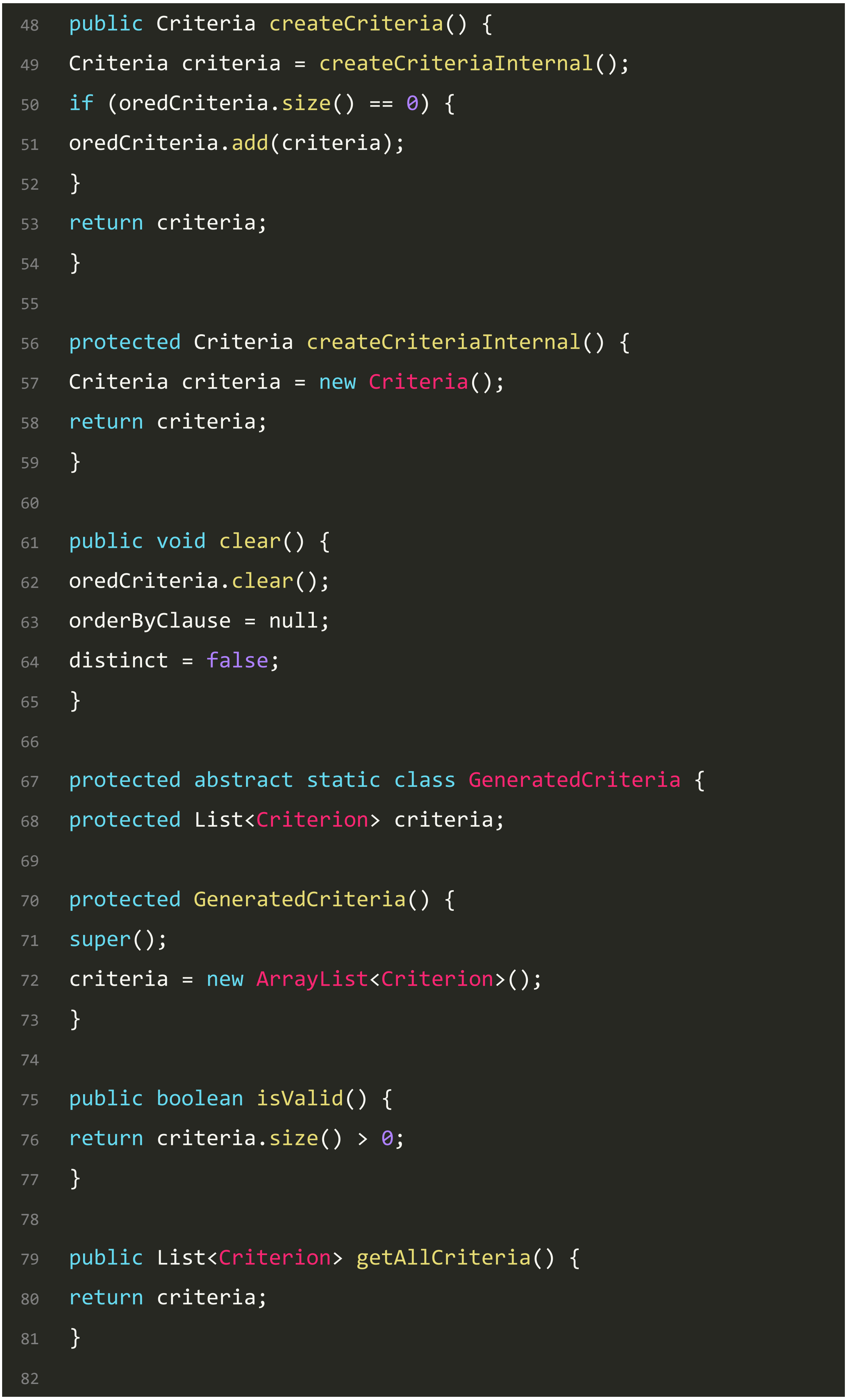
```java

 public void setAcBirthday(Date acBirthday) {
 this.acBirthday = acBirthday;
 }

 public Integer getAcSex() {
 return acSex;
 }

 public void setAcSex(Integer acSex) {
 this.acSex = acSex;
 }

 public Integer getWorkinglife() {
 return workinglife;
 }

 public void setWorkinglife(Integer workinglife) {
 this.workinglife = workinglife;
 }
}
```

ApplicantExample.java

```java
package com.yuntu.curd.bean;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class ApplicantExample {
 protected String orderByClause;

 protected boolean distinct;

 protected List<Criteria> oredCriteria;
```



```java

public ApplicantExample() {
oredCriteria = new ArrayList<Criteria>();
}

public void setOrderByClause(String orderByClause) {
this.orderByClause = orderByClause;
}

public String getOrderByClause() {
return orderByClause;
}

public void setDistinct(boolean distinct) {
this.distinct = distinct;
}

public boolean isDistinct() {
return distinct;
}

public List<Criteria> getOredCriteria() {
return oredCriteria;
}

public void or(Criteria criteria) {
oredCriteria.add(criteria);
}

public Criteria or() {
Criteria criteria = createCriteriaInternal();
oredCriteria.add(criteria);
return criteria;
}

```



```
public Criteria createCriteria() {
Criteria criteria = createCriteriaInternal();
if (oredCriteria.size() == 0) {
oredCriteria.add(criteria);
}
return criteria;
}

protected Criteria createCriteriaInternal() {
Criteria criteria = new Criteria();
return criteria;
}

public void clear() {
oredCriteria.clear();
orderByClause = null;
distinct = false;
}

protected abstract static class GeneratedCriteria {
protected List<Criterion> criteria;

protected GeneratedCriteria() {
super();
criteria = new ArrayList<Criterion>();
}

public boolean isValid() {
return criteria.size() > 0;
}

public List<Criterion> getAllCriteria() {
return criteria;
}

```

```
public List<Criterion> getCriteria() {
return criteria;
}

protected void addCriterion(String condition) {
if (condition == null) {
throw new RuntimeException("Value for condition cannot be null");
}
criteria.add(new Criterion(condition));
}

protected void addCriterion(String condition, Object value, String property) {
if (value == null) {
throw new RuntimeException("Value for " + property + " cannot be null");
}
criteria.add(new Criterion(condition, value));
}

protected void addCriterion(String condition, Object value1, Object value2, String property) {
if (value1 == null || value2 == null) {
throw new RuntimeException("Between values for " + property + " cannot be null");
}
criteria.add(new Criterion(condition, value1, value2));
}

public Criteria andAcIdIsNull() {
addCriterion("ac_id is null");
return (Criteria) this;
}

public Criteria andAcIdIsNotNull() {
```

```
addCriterion("ac_id is not null");
return (Criteria) this;
}

public Criteria andAcIdEqualTo(Integer value) {
addCriterion("ac_id =", value, "acId");
return (Criteria) this;
}

public Criteria andAcIdNotEqualTo(Integer value) {
addCriterion("ac_id <>", value, "acId");
return (Criteria) this;
}

public Criteria andAcIdGreaterThan(Integer value) {
addCriterion("ac_id >", value, "acId");
return (Criteria) this;
}

public Criteria andAcIdGreaterThanOrEqualTo(Integer value) {
addCriterion("ac_id >=", value, "acId");
return (Criteria) this;
}

public Criteria andAcIdLessThan(Integer value) {
addCriterion("ac_id <", value, "acId");
return (Criteria) this;
}

public Criteria andAcIdLessThanOrEqualTo(Integer value) {
addCriterion("ac_id <=", value, "acId");
return (Criteria) this;
}

public Criteria andAcIdIn(List<Integer> values) {
```

```java
addCriterion("ac_id in", values, "acId");
return (Criteria) this;
}

public Criteria andAcIdNotIn(List<Integer> values) {
addCriterion("ac_id not in", values, "acId");
return (Criteria) this;
}

public Criteria andAcIdBetween(Integer value1, Integer value2) {
addCriterion("ac_id between", value1, value2, "acId");
return (Criteria) this;
}

public Criteria andAcIdNotBetween(Integer value1, Integer value2) {
addCriterion("ac_id not between", value1, value2, "acId");
return (Criteria) this;
}

public Criteria andAcNameIsNull() {
addCriterion("ac_name is null");
return (Criteria) this;
}

public Criteria andAcNameIsNotNull() {
addCriterion("ac_name is not null");
return (Criteria) this;
}

public Criteria andAcNameEqualTo(String value) {
addCriterion("ac_name =", value, "acName");
return (Criteria) this;
}

```

```
public Criteria andAcNameNotEqualTo(String value) {
addCriterion("ac_name <>", value, "acName");
return (Criteria) this;
}

public Criteria andAcNameGreaterThan(String value) {
addCriterion("ac_name >", value, "acName");
return (Criteria) this;
}

public Criteria andAcNameGreaterThanOrEqualTo(String value) {
addCriterion("ac_name >=", value, "acName");
return (Criteria) this;
}

public Criteria andAcNameLessThan(String value) {
addCriterion("ac_name <", value, "acName");
return (Criteria) this;
}

public Criteria andAcNameLessThanOrEqualTo(String value) {
addCriterion("ac_name <=", value, "acName");
return (Criteria) this;
}

public Criteria andAcNameLike(String value) {
addCriterion("ac_name like", value, "acName");
return (Criteria) this;
}

public Criteria andAcNameNotLike(String value) {
addCriterion("ac_name not like", value, "acName");
return (Criteria) this;
}

```

```
public Criteria andAcNameIn(List<String> values) {
addCriterion("ac_name in", values, "acName");
return (Criteria) this;
}

public Criteria andAcNameNotIn(List<String> values) {
addCriterion("ac_name not in", values, "acName");
return (Criteria) this;
}

public Criteria andAcNameBetween(String value1, String value2
{
addCriterion("ac_name between", value1, value2, "acName");
return (Criteria) this;
}

public Criteria andAcNameNotBetween(String value1, String valu
e2) {
addCriterion("ac_name not between", value1, value2, "acName")
return (Criteria) this;
}

public Criteria andAcBirthdayIsNull() {
addCriterion("ac_birthday is null");
return (Criteria) this;
}

public Criteria andAcBirthdayIsNotNull() {
addCriterion("ac_birthday is not null");
return (Criteria) this;
}

public Criteria andAcBirthdayEqualTo(Date value) {
addCriterion("ac_birthday =", value, "acBirthday");
return (Criteria) this;
}
```

```java

public Criteria andAcBirthdayNotEqualTo(Date value) {
addCriterion("ac_birthday <>", value, "acBirthday");
return (Criteria) this;
}

public Criteria andAcBirthdayGreaterThan(Date value) {
addCriterion("ac_birthday >", value, "acBirthday");
return (Criteria) this;
}

public Criteria andAcBirthdayGreaterThanOrEqualTo(Date value) {
addCriterion("ac_birthday >=", value, "acBirthday");
return (Criteria) this;
}

public Criteria andAcBirthdayLessThan(Date value) {
addCriterion("ac_birthday <", value, "acBirthday");
return (Criteria) this;
}

public Criteria andAcBirthdayLessThanOrEqualTo(Date value) {
addCriterion("ac_birthday <=", value, "acBirthday");
return (Criteria) this;
}

public Criteria andAcBirthdayIn(List<Date> values) {
addCriterion("ac_birthday in", values, "acBirthday");
return (Criteria) this;
}

public Criteria andAcBirthdayNotIn(List<Date> values) {
addCriterion("ac_birthday not in", values, "acBirthday");
return (Criteria) this;
}

public Criteria andAcBirthdayBetween(Date value1, Date value2)
{
addCriterion("ac_birthday between", value1, value2, "acBirthda
y");
return (Criteria) this;
}

public Criteria andAcBirthdayNotBetween(Date value1, Date valu
e2) {
addCriterion("ac_birthday not between", value1, value2, "acBir
thday");
return (Criteria) this;
}

public Criteria andAcSexIsNull() {
addCriterion("ac_sex is null");
return (Criteria) this;
}

public Criteria andAcSexIsNotNull() {
addCriterion("ac_sex is not null");
return (Criteria) this;
}

public Criteria andAcSexEqualTo(Integer value) {
addCriterion("ac_sex =", value, "acSex");
return (Criteria) this;
}

public Criteria andAcSexNotEqualTo(Integer value) {
addCriterion("ac_sex <>", value, "acSex");
return (Criteria) this;
}

```

```java
public Criteria andAcSexGreaterThan(Integer value) {
addCriterion("ac_sex >", value, "acSex");
return (Criteria) this;
}

public Criteria andAcSexGreaterThanOrEqualTo(Integer value) {
addCriterion("ac_sex >=", value, "acSex");
return (Criteria) this;
}

public Criteria andAcSexLessThan(Integer value) {
addCriterion("ac_sex <", value, "acSex");
return (Criteria) this;
}

public Criteria andAcSexLessThanOrEqualTo(Integer value) {
addCriterion("ac_sex <=", value, "acSex");
return (Criteria) this;
}

public Criteria andAcSexIn(List<Integer> values) {
addCriterion("ac_sex in", values, "acSex");
return (Criteria) this;
}

public Criteria andAcSexNotIn(List<Integer> values) {
addCriterion("ac_sex not in", values, "acSex");
return (Criteria) this;
}

public Criteria andAcSexBetween(Integer value1, Integer value2) {
addCriterion("ac_sex between", value1, value2, "acSex");
return (Criteria) this;
}
```

```
public Criteria andAcSexNotBetween(Integer value1, Integer value2) {
addCriterion("ac_sex not between", value1, value2, "acSex");
return (Criteria) this;
}

public Criteria andWorkinglifeIsNull() {
addCriterion("workingLife is null");
return (Criteria) this;
}

public Criteria andWorkinglifeIsNotNull() {
addCriterion("workingLife is not null");
return (Criteria) this;
}

public Criteria andWorkinglifeEqualTo(Integer value) {
addCriterion("workingLife =", value, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeNotEqualTo(Integer value) {
addCriterion("workingLife <>", value, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeGreaterThan(Integer value) {
addCriterion("workingLife >", value, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeGreaterThanOrEqualTo(Integer value) {
addCriterion("workingLife >=", value, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeLessThan(Integer value) {
addCriterion("workingLife <", value, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeLessThanOrEqualTo(Integer value) {
addCriterion("workingLife <=", value, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeIn(List<Integer> values) {
addCriterion("workingLife in", values, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeNotIn(List<Integer> values) {
addCriterion("workingLife not in", values, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeBetween(Integer value1, Integer value2) {
addCriterion("workingLife between", value1, value2, "workinglife");
return (Criteria) this;
}

public Criteria andWorkinglifeNotBetween(Integer value1, Integer value2) {
addCriterion("workingLife not between", value1, value2, "workinglife");
return (Criteria) this;
}
```

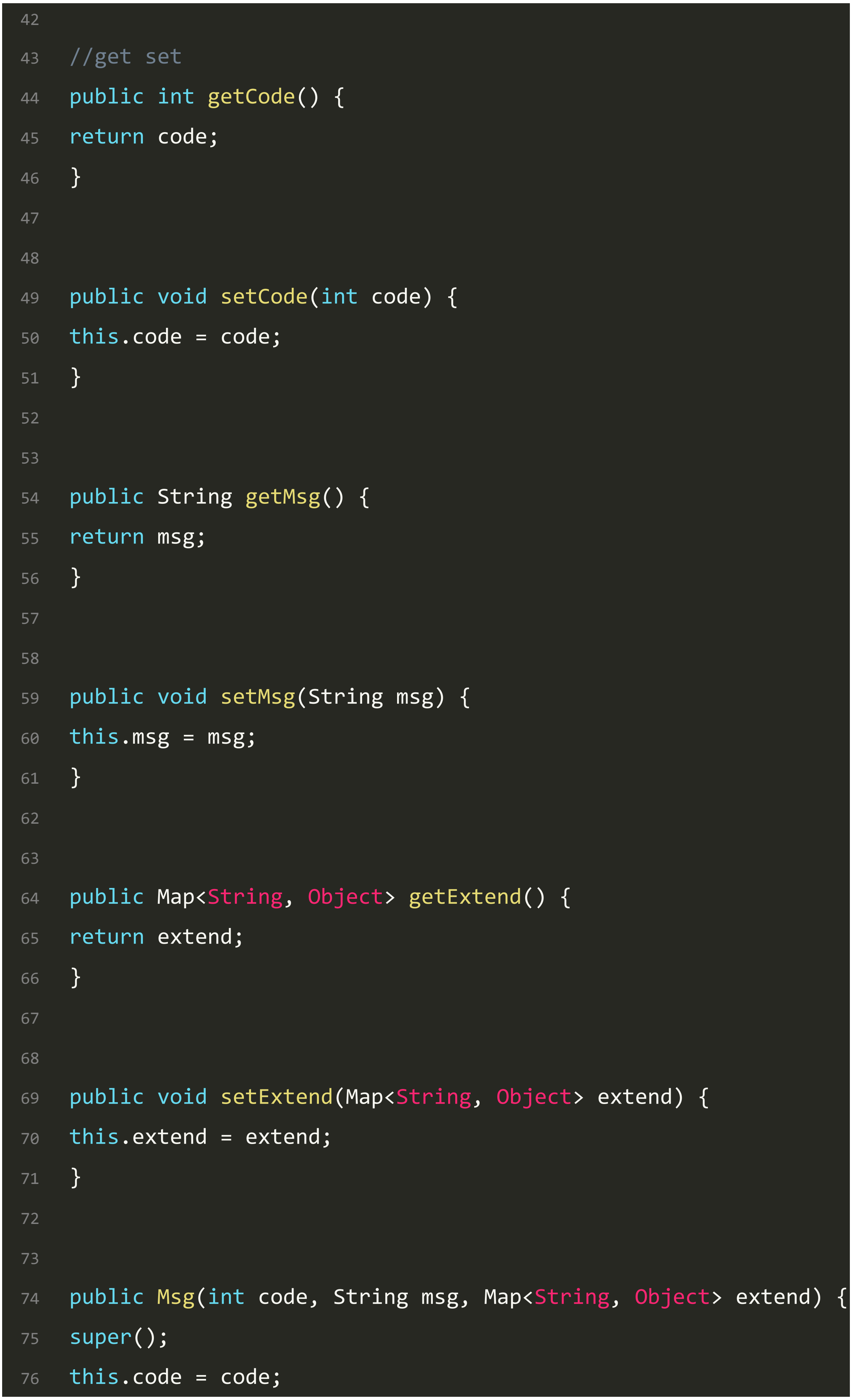
```
}

public static class Criteria extends GeneratedCriteria {

protected Criteria() {
super();
}
}

public static class Criterion {
private String condition;

private Object value;

private Object secondValue;

private boolean noValue;

private boolean singleValue;

private boolean betweenValue;

private boolean listValue;

private String typeHandler;

public String getCondition() {
return condition;
}

public Object getValue() {
return value;
}

public Object getSecondValue() {
```

```java
return secondValue;
}

public boolean isNoValue() {
return noValue;
}

public boolean isSingleValue() {
return singleValue;
}

public boolean isBetweenValue() {
return betweenValue;
}

public boolean isListValue() {
return listValue;
}

public String getTypeHandler() {
return typeHandler;
}

protected Criterion(String condition) {
super();
this.condition = condition;
this.typeHandler = null;
this.noValue = true;
}

protected Criterion(String condition, Object value, String typeHandler) {
super();
this.condition = condition;
this.value = value;
this.typeHandler = typeHandler;
if (value instanceof List<?>) {
this.listValue = true;
} else {
this.singleValue = true;
}
}

protected Criterion(String condition, Object value) {
this(condition, value, null);
}

protected Criterion(String condition, Object value, Object secondValue, String typeHandler) {
super();
this.condition = condition;
this.value = value;
this.secondValue = secondValue;
this.typeHandler = typeHandler;
this.betweenValue = true;
}

protected Criterion(String condition, Object value, Object secondValue) {
this(condition, value, secondValue, null);
}
}
}
```

Msg.java

```
package com.yuntu.curd.bean;
/**
 * 用来返回通用的类数据
 * @author 阿华
 *
 */
```

```java

import java.util.HashMap;
import java.util.Map;

 public class Msg {
  //状态码 200成功 400失败
  private int code;
  //提示信息（"操作成功"，"操作失败".......）
  private String msg;
  //用户要返回给浏览器的数据
  private Map<String, Object> extend=new HashMap<String, Object>();


  //成功，同时返回Msg对象
  public static Msg success() {
  Msg result = new Msg();
  result.setCode(200);
  result.setMsg("成功");
  return result;
  }

  //失败，同时返回Msg对象
  public static Msg fail() {
  Msg result = new Msg();
  result.setCode(400);
  result.setMsg("失败");
  return result;
  }

  //添加数据并返回
  public Msg add(String key, Object value) {
  this.getExtend().put(key,value);
  return this;
  }

```



```java

//get set
public int getCode() {
return code;
}


public void setCode(int code) {
this.code = code;
}


public String getMsg() {
return msg;
}


public void setMsg(String msg) {
this.msg = msg;
}


public Map<String, Object> getExtend() {
return extend;
}


public void setExtend(Map<String, Object> extend) {
this.extend = extend;
}


public Msg(int code, String msg, Map<String, Object> extend) {
super();
this.code = code;
```

```
  this.msg = msg;
  this.extend = extend;
  }


  public Msg() {
  super();
  }





}

```

Projectinfo.java

```
package com.yuntu.curd.bean;

import java.util.Date;

import javax.validation.constraints.Pattern;

import org.springframework.format.annotation.DateTimeFormat;

public class Projectinfo {
  private Integer piId;

  //正则表达式中\用来转义，但是在java中属于特殊字符，所以在\前再加一个\
  @Pattern(regexp="(^[a-zA-Z0-9_-]{6,64}$)|(^[\\u2E80-\\u9FFF]{2,32}$)",message="后台：论文名为2~32为中文，或者6~64为大小写字母、数字、下划线、横杠的组合")
  private String piProjectname;

  @DateTimeFormat(pattern="yyyy-MM-dd")
  private Date piStartdate;
```

```java

@DateTimeFormat(pattern="yyyy-MM-dd")
private Date piEnddate;

private Integer piStatus;

private Integer acid;

//用于查询出项目的同时，查询出申报人对象信息
private Applicant applicant;

public Projectinfo(Integer piId, String piProjectname, Date piStartdate, Date piEnddate, Integer piStatus,
Integer acid, Applicant applicant) {
super();
this.piId = piId;
this.piProjectname = piProjectname;
this.piStartdate = piStartdate;
this.piEnddate = piEnddate;
this.piStatus = piStatus;
this.acid = acid;
this.applicant = applicant;
}

public Projectinfo() {
}

public Applicant getApplicant() {
return applicant;
}

public void setApplicant(Applicant applicant) {
this.applicant = applicant;
}

public Integer getPiId() {
```

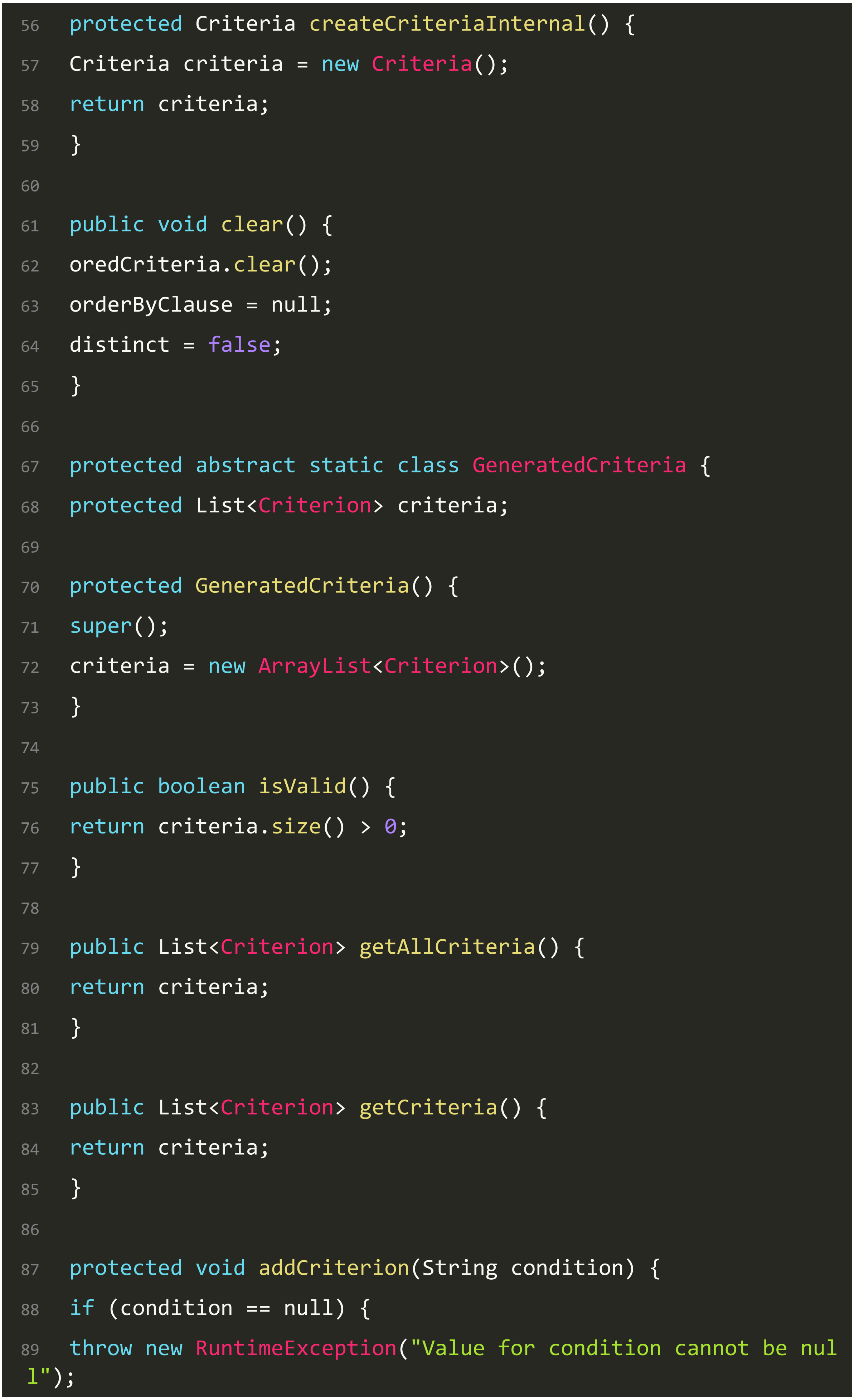
```
return piId;
}

public void setPiId(Integer piId) {
this.piId = piId;
}

public String getPiProjectname() {
return piProjectname;
}

public void setPiProjectname(String piProjectname) {
this.piProjectname = piProjectname == null ? null : piProjectname.trim();
}

public Date getPiStartdate() {
return piStartdate;
}

public void setPiStartdate(Date piStartdate) {
this.piStartdate = piStartdate;
}

public Date getPiEnddate() {
return piEnddate;
}

public void setPiEnddate(Date piEnddate) {
this.piEnddate = piEnddate;
}

public Integer getPiStatus() {
return piStatus;
}
```

```java

 public void setPiStatus(Integer piStatus) {
 this.piStatus = piStatus;
 }

 public Integer getAcid() {
 return acid;
 }

 public void setAcid(Integer acid) {
 this.acid = acid;
 }
}
```

ProjectinfoExample.java

```java
package com.yuntu.curd.bean;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class ProjectinfoExample {
 protected String orderByClause;

 protected boolean distinct;

 protected List<Criteria> oredCriteria;

 public ProjectinfoExample() {
 oredCriteria = new ArrayList<Criteria>();
 }

 public void setOrderByClause(String orderByClause) {
 this.orderByClause = orderByClause;
 }
```

```java

public String getOrderByClause() {
return orderByClause;
}

public void setDistinct(boolean distinct) {
this.distinct = distinct;
}

public boolean isDistinct() {
return distinct;
}

public List<Criteria> getOredCriteria() {
return oredCriteria;
}

public void or(Criteria criteria) {
oredCriteria.add(criteria);
}

public Criteria or() {
Criteria criteria = createCriteriaInternal();
oredCriteria.add(criteria);
return criteria;
}

public Criteria createCriteria() {
Criteria criteria = createCriteriaInternal();
if (oredCriteria.size() == 0) {
oredCriteria.add(criteria);
}
return criteria;
}

```



```java
protected Criteria createCriteriaInternal() {
Criteria criteria = new Criteria();
return criteria;
}

public void clear() {
oredCriteria.clear();
orderByClause = null;
distinct = false;
}

protected abstract static class GeneratedCriteria {
protected List<Criterion> criteria;

protected GeneratedCriteria() {
super();
criteria = new ArrayList<Criterion>();
}

public boolean isValid() {
return criteria.size() > 0;
}

public List<Criterion> getAllCriteria() {
return criteria;
}

public List<Criterion> getCriteria() {
return criteria;
}

protected void addCriterion(String condition) {
if (condition == null) {
throw new RuntimeException("Value for condition cannot be null");
}
criteria.add(new Criterion(condition));
}

protected void addCriterion(String condition, Object value, String property) {
if (value == null) {
throw new RuntimeException("Value for " + property + " cannot be null");
}
criteria.add(new Criterion(condition, value));
}

protected void addCriterion(String condition, Object value1, Object value2, String property) {
if (value1 == null || value2 == null) {
throw new RuntimeException("Between values for " + property + " cannot be null");
}
criteria.add(new Criterion(condition, value1, value2));
}

public Criteria andPiIdIsNull() {
addCriterion("pi_id is null");
return (Criteria) this;
}

public Criteria andPiIdIsNotNull() {
addCriterion("pi_id is not null");
return (Criteria) this;
}

public Criteria andPiIdEqualTo(Integer value) {
addCriterion("pi_id =", value, "piId");
return (Criteria) this;
}
```

```java

public Criteria andPiIdNotEqualTo(Integer value) {
addCriterion("pi_id <>", value, "piId");
return (Criteria) this;
}

public Criteria andPiIdGreaterThan(Integer value) {
addCriterion("pi_id >", value, "piId");
return (Criteria) this;
}

public Criteria andPiIdGreaterThanOrEqualTo(Integer value) {
addCriterion("pi_id >=", value, "piId");
return (Criteria) this;
}

public Criteria andPiIdLessThan(Integer value) {
addCriterion("pi_id <", value, "piId");
return (Criteria) this;
}

public Criteria andPiIdLessThanOrEqualTo(Integer value) {
addCriterion("pi_id <=", value, "piId");
return (Criteria) this;
}

public Criteria andPiIdIn(List<Integer> values) {
addCriterion("pi_id in", values, "piId");
return (Criteria) this;
}

public Criteria andPiIdNotIn(List<Integer> values) {
addCriterion("pi_id not in", values, "piId");
return (Criteria) this;
}
```

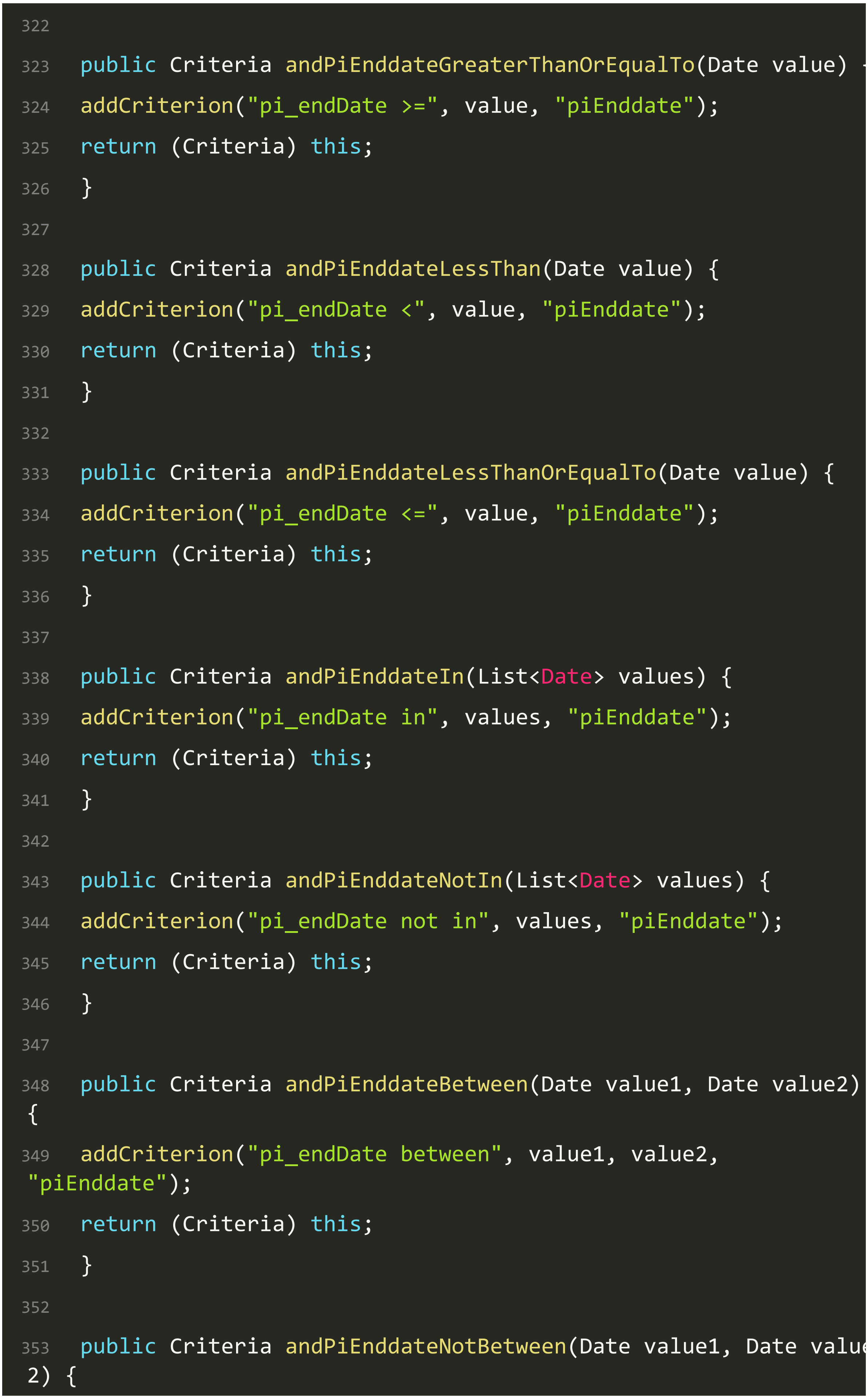
```java

public Criteria andPiIdBetween(Integer value1, Integer value2) {
addCriterion("pi_id between", value1, value2, "piId");
return (Criteria) this;
}

public Criteria andPiIdNotBetween(Integer value1, Integer value2) {
addCriterion("pi_id not between", value1, value2, "piId");
return (Criteria) this;
}

public Criteria andPiProjectnameIsNull() {
addCriterion("pi_projectName is null");
return (Criteria) this;
}

public Criteria andPiProjectnameIsNotNull() {
addCriterion("pi_projectName is not null");
return (Criteria) this;
}

public Criteria andPiProjectnameEqualTo(String value) {
addCriterion("pi_projectName =", value, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameNotEqualTo(String value) {
addCriterion("pi_projectName <>", value, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameGreaterThan(String value) {
addCriterion("pi_projectName >", value, "piProjectname");
return (Criteria) this;
```

```
}

public Criteria andPiProjectnameGreaterThanOrEqualTo(String value) {
addCriterion("pi_projectName >=", value, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameLessThan(String value) {
addCriterion("pi_projectName <", value, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameLessThanOrEqualTo(String value) {
addCriterion("pi_projectName <=", value, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameLike(String value) {
addCriterion("pi_projectName like", value, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameNotLike(String value) {
addCriterion("pi_projectName not like", value, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameIn(List<String> values) {
addCriterion("pi_projectName in", values, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameNotIn(List<String> values) {
```

```
addCriterion("pi_projectName not in", values, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameBetween(String value1, String value2) {
addCriterion("pi_projectName between", value1, value2, "piProjectname");
return (Criteria) this;
}

public Criteria andPiProjectnameNotBetween(String value1, String value2) {
addCriterion("pi_projectName not between", value1, value2, "piProjectname");
return (Criteria) this;
}

public Criteria andPiStartdateIsNull() {
addCriterion("pi_startDate is null");
return (Criteria) this;
}

public Criteria andPiStartdateIsNotNull() {
addCriterion("pi_startDate is not null");
return (Criteria) this;
}

public Criteria andPiStartdateEqualTo(Date value) {
addCriterion("pi_startDate =", value, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateNotEqualTo(Date value) {
addCriterion("pi_startDate <>", value, "piStartdate");
```

```
return (Criteria) this;
}

public Criteria andPiStartdateGreaterThan(Date value) {
addCriterion("pi_startDate >", value, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateGreaterThanOrEqualTo(Date value) {
addCriterion("pi_startDate >=", value, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateLessThan(Date value) {
addCriterion("pi_startDate <", value, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateLessThanOrEqualTo(Date value) {
addCriterion("pi_startDate <=", value, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateIn(List<Date> values) {
addCriterion("pi_startDate in", values, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateNotIn(List<Date> values) {
addCriterion("pi_startDate not in", values, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateBetween(Date value1, Date value2) {
```

```
addCriterion("pi_startDate between", value1, value2, "piStartdate");
return (Criteria) this;
}

public Criteria andPiStartdateNotBetween(Date value1, Date value2) {
addCriterion("pi_startDate not between", value1, value2, "piStartdate");
return (Criteria) this;
}

public Criteria andPiEnddateIsNull() {
addCriterion("pi_endDate is null");
return (Criteria) this;
}

public Criteria andPiEnddateIsNotNull() {
addCriterion("pi_endDate is not null");
return (Criteria) this;
}

public Criteria andPiEnddateEqualTo(Date value) {
addCriterion("pi_endDate =", value, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateNotEqualTo(Date value) {
addCriterion("pi_endDate <>", value, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateGreaterThan(Date value) {
addCriterion("pi_endDate >", value, "piEnddate");
return (Criteria) this;
}
```



```
public Criteria andPiEnddateGreaterThanOrEqualTo(Date value)
addCriterion("pi_endDate >=", value, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateLessThan(Date value) {
addCriterion("pi_endDate <", value, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateLessThanOrEqualTo(Date value) {
addCriterion("pi_endDate <=", value, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateIn(List<Date> values) {
addCriterion("pi_endDate in", values, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateNotIn(List<Date> values) {
addCriterion("pi_endDate not in", values, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateBetween(Date value1, Date value2) {
addCriterion("pi_endDate between", value1, value2, "piEnddate");
return (Criteria) this;
}

public Criteria andPiEnddateNotBetween(Date value1, Date value2) {
```

```
addCriterion("pi_endDate not between", value1, value2, "piEnd
ate");
return (Criteria) this;
}

public Criteria andPiStatusIsNull() {
addCriterion("pi_status is null");
return (Criteria) this;
}

public Criteria andPiStatusIsNotNull() {
addCriterion("pi_status is not null");
return (Criteria) this;
}

public Criteria andPiStatusEqualTo(Integer value) {
addCriterion("pi_status =", value, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusNotEqualTo(Integer value) {
addCriterion("pi_status <>", value, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusGreaterThan(Integer value) {
addCriterion("pi_status >", value, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusGreaterThanOrEqualTo(Integer value
{
addCriterion("pi_status >=", value, "piStatus");
return (Criteria) this;
}

```

```java
public Criteria andPiStatusLessThan(Integer value) {
addCriterion("pi_status <", value, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusLessThanOrEqualTo(Integer value) {
addCriterion("pi_status <=", value, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusIn(List<Integer> values) {
addCriterion("pi_status in", values, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusNotIn(List<Integer> values) {
addCriterion("pi_status not in", values, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusBetween(Integer value1, Integer value2) {
addCriterion("pi_status between", value1, value2, "piStatus");
return (Criteria) this;
}

public Criteria andPiStatusNotBetween(Integer value1, Integer value2) {
addCriterion("pi_status not between", value1, value2, "piStatus");
return (Criteria) this;
}

public Criteria andAcidIsNull() {
addCriterion("acId is null");
return (Criteria) this;
```

```
}

public Criteria andAcidIsNotNull() {
addCriterion("acId is not null");
return (Criteria) this;
}

public Criteria andAcidEqualTo(Integer value) {
addCriterion("acId =", value, "acid");
return (Criteria) this;
}

public Criteria andAcidNotEqualTo(Integer value) {
addCriterion("acId <>", value, "acid");
return (Criteria) this;
}

public Criteria andAcidGreaterThan(Integer value) {
addCriterion("acId >", value, "acid");
return (Criteria) this;
}

public Criteria andAcidGreaterThanOrEqualTo(Integer value) {
addCriterion("acId >=", value, "acid");
return (Criteria) this;
}

public Criteria andAcidLessThan(Integer value) {
addCriterion("acId <", value, "acid");
return (Criteria) this;
}

public Criteria andAcidLessThanOrEqualTo(Integer value) {
addCriterion("acId <=", value, "acid");
return (Criteria) this;
```

```java
}

public Criteria andAcidIn(List<Integer> values) {
addCriterion("acId in", values, "acid");
return (Criteria) this;
}

public Criteria andAcidNotIn(List<Integer> values) {
addCriterion("acId not in", values, "acid");
return (Criteria) this;
}

public Criteria andAcidBetween(Integer value1, Integer value2) {
addCriterion("acId between", value1, value2, "acid");
return (Criteria) this;
}

public Criteria andAcidNotBetween(Integer value1, Integer value2) {
addCriterion("acId not between", value1, value2, "acid");
return (Criteria) this;
}
}

public static class Criteria extends GeneratedCriteria {

protected Criteria() {
super();
}
}

public static class Criterion {
private String condition;

private Object value;

private Object secondValue;

private boolean noValue;

private boolean singleValue;

private boolean betweenValue;

private boolean listValue;

private String typeHandler;

public String getCondition() {
return condition;
}

public Object getValue() {
return value;
}

public Object getSecondValue() {
return secondValue;
}

public boolean isNoValue() {
return noValue;
}

public boolean isSingleValue() {
return singleValue;
}

public boolean isBetweenValue() {
return betweenValue;
```

```java
}

public boolean isListValue() {
return listValue;
}

public String getTypeHandler() {
return typeHandler;
}

protected Criterion(String condition) {
super();
this.condition = condition;
this.typeHandler = null;
this.noValue = true;
}

protected Criterion(String condition, Object value, String typeHandler) {
super();
this.condition = condition;
this.value = value;
this.typeHandler = typeHandler;
if (value instanceof List<?>) {
this.listValue = true;
} else {
this.singleValue = true;
}
}

protected Criterion(String condition, Object value) {
this(condition, value, null);
}

protected Criterion(String condition, Object value, Object secondValue, String typeHandler) {
  super();
  this.condition = condition;
  this.value = value;
  this.secondValue = secondValue;
  this.typeHandler = typeHandler;
  this.betweenValue = true;
  }

  protected Criterion(String condition, Object value, Object secondValue) {
  this(condition, value, secondValue, null);
  }
  }
}
```

控制器：com.yuntu.curd.controller

ApplicantController.java

```
package com.yuntu.curd.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import com.yuntu.curd.bean.Applicant;
import com.yuntu.curd.bean.Msg;
import com.yuntu.curd.service.ApplicantService;

/**
 * 申报人控制器
 * @author Administrator
 *
 */
```

```
@Controller
public class ApplicantController {
 @Autowired
 private ApplicantService applicantService;

 /**
 * 获取所有申报人的信息，并返回JSON数据
 */
 @RequestMapping("/applicants")
 @ResponseBody //响应的数据是一个JSON数据
 public Msg getApplicants() {
 List<Applicant> list = applicantService.getApplicants();
 return Msg.success().add("applicants", list);
 }
}

```

ProjectInfoController.java

```
package com.yuntu.curd.controller;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
```

```java
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;
import com.yuntu.curd.bean.Msg;
import com.yuntu.curd.bean.Projectinfo;
import com.yuntu.curd.service.ProjectInfoService;
/**
 * 项目信息控制器
 * @author Administrator
 *
 */
@Controller
public class ProjectInfoController {
 @Autowired
 private ProjectInfoService projectInfoService;

 /**
 * 将单个删除变成既能单个删除，又能多个删除
 * 传递的参数为ids
 *
 * 如果是单个值 ids:1
 * 如果是多个值 ids：1-2-3
 * @param ids
 * @return
 */
 @RequestMapping(value="/projectInfo/{ids}",method=RequestMethod.DELETE)
 @ResponseBody
 public Msg deleteProjectInfoById(@PathVariable("ids") String ids) {
 //判断是否包含-
 if (ids.contains("-")) {//多个删除
 List<Integer> int_ids = new ArrayList<Integer>();
String[] str_ids=ids.split("-");
//循环将字符串数组中的数据转换为Integer，并放到集合中
for (String str_id : str_ids) {
int_ids.add(Integer.valueOf(str_id));
}
projectInfoService.deletePrijectInfos(int_ids);;
} else {//单个删除
Integer id=Integer.valueOf(ids);
projectInfoService.deleteProjectInfo(id);
}
return Msg.success();
}

/**
* 更改数据
* @return
*/
//如果要把piId传入对象，piId必须是Projectinfo对象中的属性且名字必须一致
@RequestMapping(value="/projectInfo/{piId}", method=RequestMethod.PUT)
@ResponseBody
public Msg saveProjectInfo(Projectinfo projectInfo) {
/**
* PUT方式说明
* Tomcat在进行应用处理时，值判断是否包含POST请求，如果不是POST直接结束数据获取
* 所以PUT请求时无法拿到想要的数据，默认获得的全部是空
* 解决方案：
* 必须在web.xml文件添加HttpPutFormContentFilter过滤器
* 将请求中的数据解析包装成集合，同request被重新包装
* 这样的话，request.getParmenter("")被重写，这样讲就能通过PUT请求那数据
*
*/
projectInfoService.updateProjectInfo(projectInfo);
```

```java
return Msg.success();
}


/**
* 根据pi_id查询项目的详细信息
* @param pi_Id
* @return
*/
@RequestMapping(value="/projectInfo/{id}",method=RequestMethod.GET
@ResponseBody
public Msg getProjectInfo(@PathVariable("id") Integer pi_Id) {
//@PathVariable("id"): 代表从路径中获取id
Projectinfo projectInfo = projectInfoService.getProjectInfo(pi_Id);
return Msg.success().add("projectInfo",projectInfo);
}

 /**
 * 验证项目名是否存在
 * @param projectName
 * @return
 */
 @RequestMapping("/checkProjectInfo")
 @ResponseBody
 public Msg checkProjectInfo (@RequestParam("projectName") String projectName) {
 //判断项目名是否合法
 String regProjectName="(^[a-zA-Z0-9_-]{6,64}$)|(^[\\u2E80-\\u9FFF]{2,32}$)";
 if (regProjectName.matches(regProjectName)) {
 return Msg.fail().add("validate_msg", "重复验证：论文名为2~32为中文，或者6~64为大小写字母、数字、下划线、横杠的组合");
 } else {
 //执行查询
```

```java
boolean flag = projectInfoService.checkProjectInfo(projectName);
if(flag) {
return Msg.success();
} else {
return Msg.fail().add("validate_msg","× 不可用");
}
}
}

/**
* 保存，返回josn数据
* 请求方式（规定）：
* URI：
* /projectInfo/{id} GET方式 查询
* /projectInfo/ POST方式 添加
* /projectInfo/{id} PUT方式 修改
* /projectInfo/{id} DELETE方式 删除
*
* @param projectInfoo
* @param result
* @return
*/
@RequestMapping(value="/projectInfo",method=RequestMethod.POST)
@ResponseBody
public Msg saveProjectInfo(@Valid Projectinfo projectInfo, BindingResult result) {
//传递参数前添加@Valid 进行JSR303数据校验；BindingResult：绑定的验证结果
//判断验证结果是否有错误信息
if (result.hasErrors()) {//如果有错误信息
//创建用于存放错误的map集合
Map<String, Object> map = new HashMap<String, Object>();
//返回所有的字段错误
List<FieldError> errors = result.getFieldErrors();
```

```
//将所有得到的错误信息，，都放在map集合中
for (FieldError fieldError : errors) {
System.out.println("错误的字段："+fieldError.getField());
System.out.println("错误的提示信息："+fieldError.getDefaultMessage());
map.put(fieldError.getField(),
fieldError.getDefaultMessage());
}
//返回到页面处理上
return Msg.fail().add("errorFields", map);
} else {
projectInfoService.saveProjectInfo(projectInfo);
return Msg.success();
}
}



/**
* ajax请求,分页查询页面数据，返回JSON格式
* @param pn
* @return
*/
@RequestMapping("/projectInfos")
@ResponseBody
public Msg getProjectInfosWithJson(@RequestParam(value="pn",defaultValue="1") Integer pn) {
//查询所有员工
//引入PageHelper分页插件
//查询之前值需要调用，传入页码及第一页页容量
PageHelper.startPage(pn, 5);
List<Projectinfo> infos = projectInfoService.getAll();
//使用Pageinfo进行包装查询后的结果，只需要将PageInfo交给页面
//PageInfo(封装了详细的分页信息，包括查询出来的信息，和连续查询显示的页数)
PageInfo page = new PageInfo(infos,5);

```

```
//直接将pageInfo对象所有的信息进行返回（JSON格式）
return Msg.success().add("pageInfo", page);
}


//@RequestMapping("/projectInfos")
public String getProjectInfos(@RequestParam(value="pn",defaultValue="1") Integer pn,
Model model) {
//查询所有员工
//引入PageHelper分页插件
//查询之前值需要调用，传入页码及第一页页容量
PageHelper.startPage(pn, 5);
List<Projectinfo> infos = projectInfoService.getAll();
//使用Pageinfo进行包装查询后的结果，只需要将PageInfo交给页面
//PageInfo(封装了详细的分页信息，包括查询出来的信息，和连续查询显示的页数)
PageInfo page = new PageInfo(infos,5);
model.addAttribute("PageInfo",page);
return "list";
}

}
```

数据访问层：com.yuntu.curd.dao

ApplicantMapper.java

```
package com.yuntu.curd.dao;

import com.yuntu.curd.bean.Applicant;
import com.yuntu.curd.bean.ApplicantExample;
import java.util.List;
import org.apache.ibatis.annotations.Param;

public interface ApplicantMapper {
 long countByExample(ApplicantExample example);
```

```java

 int deleteByExample(ApplicantExample example);

 int deleteByPrimaryKey(Integer acId);

 int insert(Applicant record);

 int insertSelective(Applicant record);

 List<Applicant> selectByExample(ApplicantExample example);

 Applicant selectByPrimaryKey(Integer acId);

 int updateByExampleSelective(@Param("record") Applicant record, @Param("example") ApplicantExample example);

 int updateByExample(@Param("record") Applicant record, @Param("example") ApplicantExample example);

 int updateByPrimaryKeySelective(Applicant record);

 int updateByPrimaryKey(Applicant record);
}
```

ProjectinfoMapper.java

```java
package com.yuntu.curd.dao;

import com.yuntu.curd.bean.Projectinfo;
import com.yuntu.curd.bean.ProjectinfoExample;
import java.util.List;
import org.apache.ibatis.annotations.Param;

public interface ProjectinfoMapper {
 long countByExample(ProjectinfoExample example);

 int deleteByExample(ProjectinfoExample example);
```

```
    
    int deleteByPrimaryKey(Integer piId);

    int insert(Projectinfo record);

    int insertSelective(Projectinfo record);

    /**
    * 根据条件查询所有，两表联查，同时查询出申报项目的项目和对应的申报人
    * @param example
    * @return
    */
    List<Projectinfo> selectByExampleWithApplicantion(ProjectinfoExample example);

    /**
    * 根据id查询所有，两表联查，同时查询出单个申报项目的项目和对应的申报人
    * @param piId
    * @return
    */
    Projectinfo selectByPrimaryKeyWithApplicantion(Integer piId);

    List<Projectinfo> selectByExample(ProjectinfoExample example);

    Projectinfo selectByPrimaryKey(Integer piId);

    int updateByExampleSelective(@Param("record") Projectinfo record, @Param("example") ProjectinfoExample example);

    int updateByExample(@Param("record") Projectinfo record, @Param("example") ProjectinfoExample example);

    int updateByPrimaryKeySelective(Projectinfo record);

    int updateByPrimaryKey(Projectinfo record);
```

```
}
```

业务逻辑层：com.yuntu.curd.service

ApplicantService.java

```
package com.yuntu.curd.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.yuntu.curd.bean.Applicant;
import com.yuntu.curd.dao.ApplicantMapper;

 /**
  * 申请人业务层
  * @author 阿华
  *
  */
 @Service
 public class ApplicantService {

  @Autowired
  private ApplicantMapper applicantMapper;

  /**
  * 业务查询所有的申报人
  * @return
  */
  public List<Applicant> getApplicants() {
  return applicantMapper.selectByExample(null);
  }
 }

```

ProjectInfoService.java

```java
package com.yuntu.curd.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.yuntu.curd.bean.Projectinfo;
import com.yuntu.curd.bean.ProjectinfoExample;
import com.yuntu.curd.bean.ProjectinfoExample.Criteria;
import com.yuntu.curd.dao.ProjectinfoMapper;

/**
 * 项目表业务处理层
 * @author 阿华
 *
 */
@Service
public class ProjectInfoService {
 @Autowired
 ProjectinfoMapper projectInfoMapper;

 /**
 * 查询所有项目信息
 * @return
 */
 public List<Projectinfo> getAll() {
 return projectInfoMapper.selectByExampleWithApplicantion(null);
 }

 /**
 * 保存项目信息
 * @param projectInfo
 * @return
```

```java
*/
public void saveProjectInfo(Projectinfo projectInfo) {
projectInfoMapper.insertSelective(projectInfo);
}


/**
*检查用户是否存在
*/
public boolean checkProjectInfo(String projectName) {
//创建查询条件
ProjectinfoExample example = new ProjectinfoExample();
//得到条件对象
Criteria Criteria = example.createCriteria();
Criteria.andPiProjectnameEqualTo(projectName);
//根据条件查询数量
long count = projectInfoMapper.countByExample(example);
return count==0;
}

/**
* 修改时，根据pi_Id查询出单个项目信息
* @param pi_Id
* @return
*/
public Projectinfo getProjectInfo(Integer pi_Id) {
return projectInfoMapper.selectByPrimaryKeyWithApplicantion(pi_Id);
}

/**
* 修改项目信息（根据id有选择的更新）
* @param projectInfo
*/
public void updateProjectInfo(Projectinfo projectInfo) {
projectInfoMapper.updateByPrimaryKeySelective(projectInfo);
```

```
 }

 /**
 * 根据单个id删除项目信息
 * @param id
 */
 public void deleteProjectInfo(Integer id) {
 projectInfoMapper.deleteByPrimaryKey(id);
 }

 /**
 * 根据多个id删除项目信息
 * @param ids
 */
 public void deletePrijectInfos(List<Integer> ids) {
 ProjectinfoExample example = new ProjectinfoExample();
 Criteria Criteria = example.createCriteria();
 Criteria.andPiIdIn(ids);
 projectInfoMapper.deleteByExample(example);
 }
}

```

测试：com.yuntu.curd.test

MapperTest.java

```
package com.yuntu.curd.test;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.UUID;

import org.apache.ibatis.session.SqlSession;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import com.yuntu.curd.bean.Applicant;
import com.yuntu.curd.bean.Projectinfo;
import com.yuntu.curd.dao.ApplicantMapper;
import com.yuntu.curd.dao.ProjectinfoMapper;

/**
 * 测试dao的任务
 * @author 阿华
 * 推荐spring项目就可以spring提供的单元测试，可以自动注入我们需要的组件
 * 步骤:
 * 1.导入spring测试模块
 * 2.@ContextConfiguration指定spring配置文件位置
 * 3.直接@Autowired自动使用组件
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations= {"classpath:applicationContext.xml"})
public class MapperTest {
 @Autowired
 ApplicantMapper applicantMapper;
 @Autowired
 ProjectinfoMapper projectinfoMapper;
 //批量操作
 @Autowired
 SqlSession sqlSession;

 @Test
 public void testCRUD() throws ParseException {
 /*
 * 原始版本的测试版本
```

```java
 * //1、创建IOC容器
 * ApplicationContext ioc = new ClassPathXmlApplicationContext();
 * //2、从容器中获取mapper
 * ApplicantExample bean = ioc.getBean(ApplicantExample.class);
 */
 System.out.println("-------------------------" + applicantMapper);
// Date date = new SimpleDateFormat("yyyy-MM-dd").parse("2000-11-11");
// //插入申报人
// applicantMapper.insertSelective(new Applicant(null,"andi",date,1,5));

 /*
 * 批量插入多个；使用批量操作的sqlSession
 * 在applicationContext.xml已经注入了SQLSession对象
 */
 //创建用于批量添加的mapper接口对象
 ProjectinfoMapper mapper = sqlSession.getMapper(ProjectinfoMapper.class);
 //循环操作
 for (int i = 0; i < 50; i++) {
 //自动生成字符串
 String proname=UUID.randomUUID().toString().substring(0, 5)+"原子弹抛光"+i;
 mapper.insertSelective(new Projectinfo(null, proname,
 new SimpleDateFormat("yyyy-MM-dd").parse((1990+i)+"-11-11"),
 new SimpleDateFormat("yyyy-MM-dd").parse((1999+i)+"-12-12"),
 i%3, i%7+1, null));
 }
 System.out.println("批量添加完成");
 }
}

```

MBGTest.java

```java
package com.yuntu.curd.test;

import java.io.File;
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.mybatis.generator.api.MyBatisGenerator;
import org.mybatis.generator.config.Configuration;
import org.mybatis.generator.config.xml.ConfigurationParser;
import org.mybatis.generator.exception.InvalidConfigurationException;
import org.mybatis.generator.exception.XMLParserException;
import org.mybatis.generator.internal.DefaultShellCallback;

public class MBGTest {
 public static void main(String[] args) throws IOException, XMLParserException, InvalidConfigurationException, SQLException, InterruptedException {
  List<String> warnings = new ArrayList<String>();
  boolean overwrite = true;
  File configFile = new File("mbg.xml");
  ConfigurationParser cp = new ConfigurationParser(warnings);
  Configuration config = cp.parseConfiguration(configFile);
  DefaultShellCallback callback = new DefaultShellCallback(overwrite);
  MyBatisGenerator myBatisGenerator = new MyBatisGenerator(config, callback, warnings);
  myBatisGenerator.generate(null);
 }
}

```

MVCTest.java

```java
package com.yuntu.curd.test;

```

```java
import java.util.List;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mock.web.MockHttpServletRequest;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;

import com.github.pagehelper.PageInfo;
import com.yuntu.curd.bean.Projectinfo;

/**
 * 使用spring提供的单元测试组件
 * 步骤:
 * 使用spring模块发送虚拟请求，测试页面的curd操作
 * @author 阿华
 *
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"classpath:applicationContext.xml",
 "file:src/main/webapp/WEB-INF/dispatcherServlet-servlet.xml"})
@WebAppConfiguration
public class MVCTest {
```

```
// 传入springmvc的容器
@Autowired
WebApplicationContext context;
// 虚拟mvc请求,获取处理结果
MockMvc mockMvc;

// 每次请求前都要初始化构建mockMvc
@Before
public void initMockMvc() {
mockMvc = MockMvcBuilders.webAppContextSetup(context).build();
}

@Test
public void testPage() throws Exception {
/*
* MockMvcRequestBuilders.get("/projectInfos"):使用springmvc发送
请求信息,而且是get请求
* param("pn", "7"): 携带一个参数 params(arg0): 携带多个参数
*/
MvcResult result =
mockMvc.perform(MockMvcRequestBuilders.get("/projectInfos").param(
n", "5")).andReturn();
// 请求成功之后,请求域就会有pageinfo,可以取出
MockHttpServletRequest request = result.getRequest();
// 从请求中拿到PageInfo信息
PageInfo pe = (PageInfo) request.getAttribute("PageInfo");
//输出验证
System.out.println("当前页" + pe.getPageNum());
System.out.println("页容量" + pe.getPageSize());
System.out.println("起始数" + pe.getStartRow());
System.out.println("结束数" + pe.getEndRow());
System.out.println("总条数" + pe.getTotal());
System.out.println("总页数" + pe.getPages());
System.out.println("当前应该连续展示的页码");
int[] nums = pe.getNavigatepageNums();
for (int i : nums) {
 System.out.print(" " + i);
 }
 System.out.println();
 // 获取当前页的项目信息
 List<Projectinfo> list = pe.getList();
 for (Projectinfo projectInfo : list) {
 System.out.println("ID:" + projectInfo.getPiId() + "\t项目名" + projectInfo.getPiProjectname() + "\t申报人"
 + projectInfo.getApplicant().getAcName());
 }
 }
}

```

src/main/resources

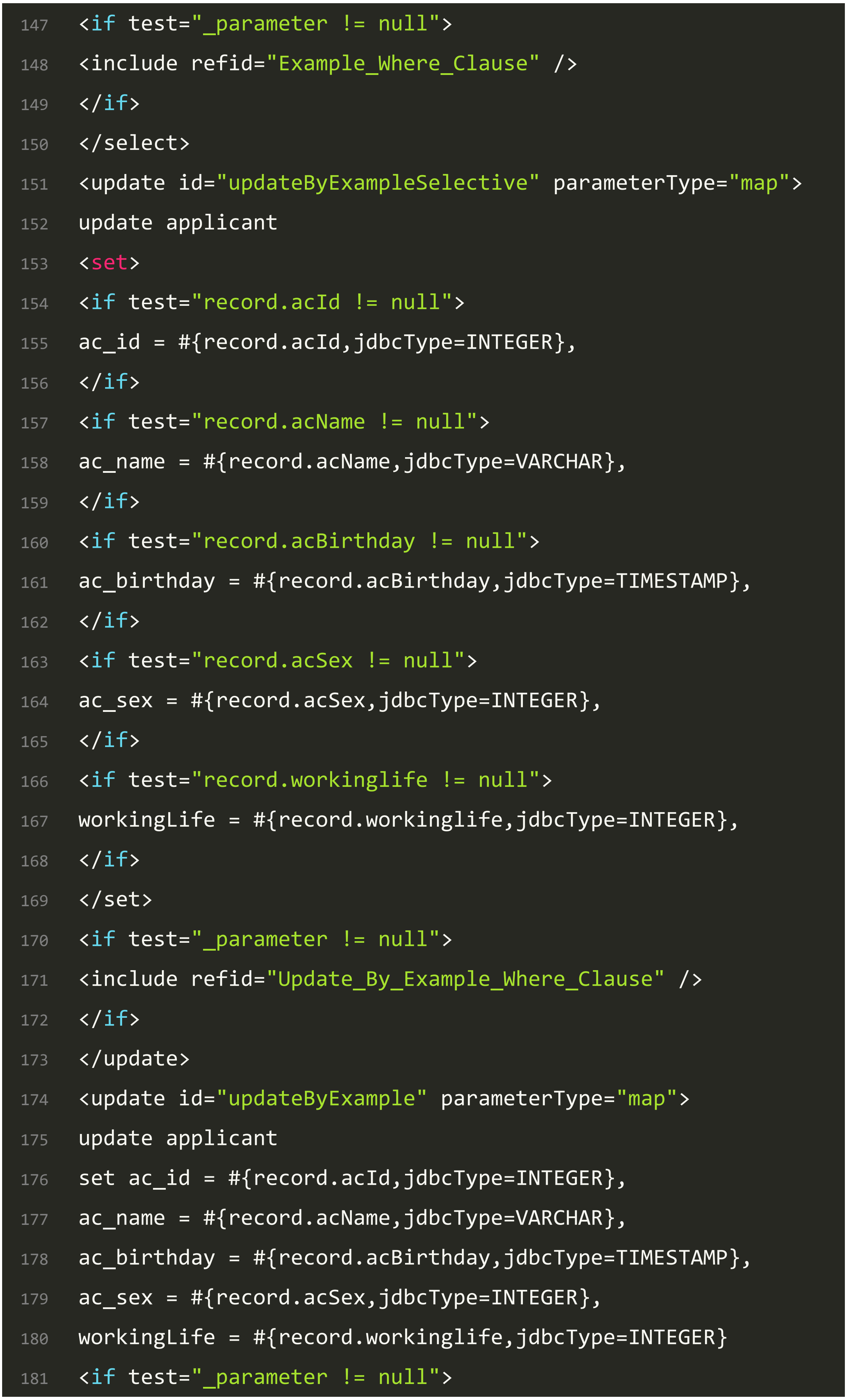
ApplicantMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.yuntu.curd.dao.ApplicantMapper">
 <resultMap id="BaseResultMap" type="com.yuntu.curd.bean.Applicant">
 <id column="ac_id" jdbcType="INTEGER" property="acId" />
 <result column="ac_name" jdbcType="VARCHAR" property="acName" />
 <result column="ac_birthday" jdbcType="TIMESTAMP" property="acBirthday" />
 <result column="ac_sex" jdbcType="INTEGER" property="acSex" />
 <result column="workingLife" jdbcType="INTEGER" property="workinglife" />
 </resultMap>
 <sql id="Example_Where_Clause">
 <where>
 <foreach collection="oredCriteria" item="criteria" separator="or">
 <if test="criteria.valid">
 <trim prefix="(" prefixOverrides="and" suffix=")">
```

```
<foreach collection="criteria.criteria" item="criterion">
<choose>
<when test="criterion.noValue">
and ${criterion.condition}
</when>
<when test="criterion.singleValue">
and ${criterion.condition} #{criterion.value}
</when>
<when test="criterion.betweenValue">
and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}
</when>
<when test="criterion.listValue">
and ${criterion.condition}
<foreach close=")" collection="criterion.value" item="listItem" open="(" separator=",">
#{listItem}
</foreach>
</when>
</choose>
</foreach>
</trim>
</if>
</foreach>
</where>
</sql>
<sql id="Update_By_Example_Where_Clause">
<where>
<foreach collection="example.oredCriteria" item="criteria" separator="or">
<if test="criteria.valid">
<trim prefix="(" prefixOverrides="and" suffix=")">
<foreach collection="criteria.criteria" item="criterion">
<choose>
<when test="criterion.noValue">
and ${criterion.condition}
```

```
</when>
<when test="criterion.singleValue">
and ${criterion.condition} #{criterion.value}
</when>
<when test="criterion.betweenValue">
and ${criterion.condition} #{criterion.value} and #{criterion.secondValue}
</when>
<when test="criterion.listValue">
and ${criterion.condition}
<foreach close=")" collection="criterion.value" item="listItem" open="(" separator=",">
#{listItem}
</foreach>
</when>
</choose>
</foreach>
</trim>
</if>
</foreach>
</where>
</sql>
<sql id="Base_Column_List">
ac_id, ac_name, ac_birthday, ac_sex, workingLife
</sql>
<select id="selectByExample" parameterType="com.yuntu.curd.bean.ApplicantExample" resultMap="BaseResultMap">
select
<if test="distinct">
distinct
</if>
<include refid="Base_Column_List" />
from applicant
<if test="_parameter != null">
<include refid="Example_Where_Clause" />
</if>
```

```
<if test="orderByClause != null">
order by ${orderByClause}
</if>
</select>
<select id="selectByPrimaryKey" parameterType="java.lang.Integer" resultMap="BaseResultMap">
select
<include refid="Base_Column_List" />
from applicant
where ac_id = #{acId,jdbcType=INTEGER}
</select>
<delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">
delete from applicant
where ac_id = #{acId,jdbcType=INTEGER}
</delete>
<delete id="deleteByExample" parameterType="com.yuntu.curd.bean.ApplicantExample">
delete from applicant
<if test="_parameter != null">
<include refid="Example_Where_Clause" />
</if>
</delete>
<insert id="insert" parameterType="com.yuntu.curd.bean.Applicant">
insert into applicant (ac_id, ac_name, ac_birthday,
ac_sex, workingLife)
values (#{acId,jdbcType=INTEGER}, #{acName,jdbcType=VARCHAR}, #{acBirthday,jdbcType=TIMESTAMP},
#{acSex,jdbcType=INTEGER}, #{workinglife,jdbcType=INTEGER})
</insert>
<insert id="insertSelective" parameterType="com.yuntu.curd.bean.Applicant">
insert into applicant
<trim prefix="(" suffix=")" suffixOverrides=",">
<if test="acId != null">
ac_id,
</if>
<if test="acName != null">
ac_name,
</if>
<if test="acBirthday != null">
ac_birthday,
</if>
<if test="acSex != null">
ac_sex,
</if>
<if test="workinglife != null">
workingLife,
</if>
</trim>
<trim prefix="values (" suffix=")" suffixOverrides=",">
<if test="acId != null">
#{acId,jdbcType=INTEGER},
</if>
<if test="acName != null">
#{acName,jdbcType=VARCHAR},
</if>
<if test="acBirthday != null">
#{acBirthday,jdbcType=TIMESTAMP},
</if>
<if test="acSex != null">
#{acSex,jdbcType=INTEGER},
</if>
<if test="workinglife != null">
#{workinglife,jdbcType=INTEGER},
</if>
</trim>
</insert>
<select id="countByExample" parameterType="com.yuntu.curd.bean.ApplicantExample" resultType="java.lang.Long">
select count(*) from applicant
```

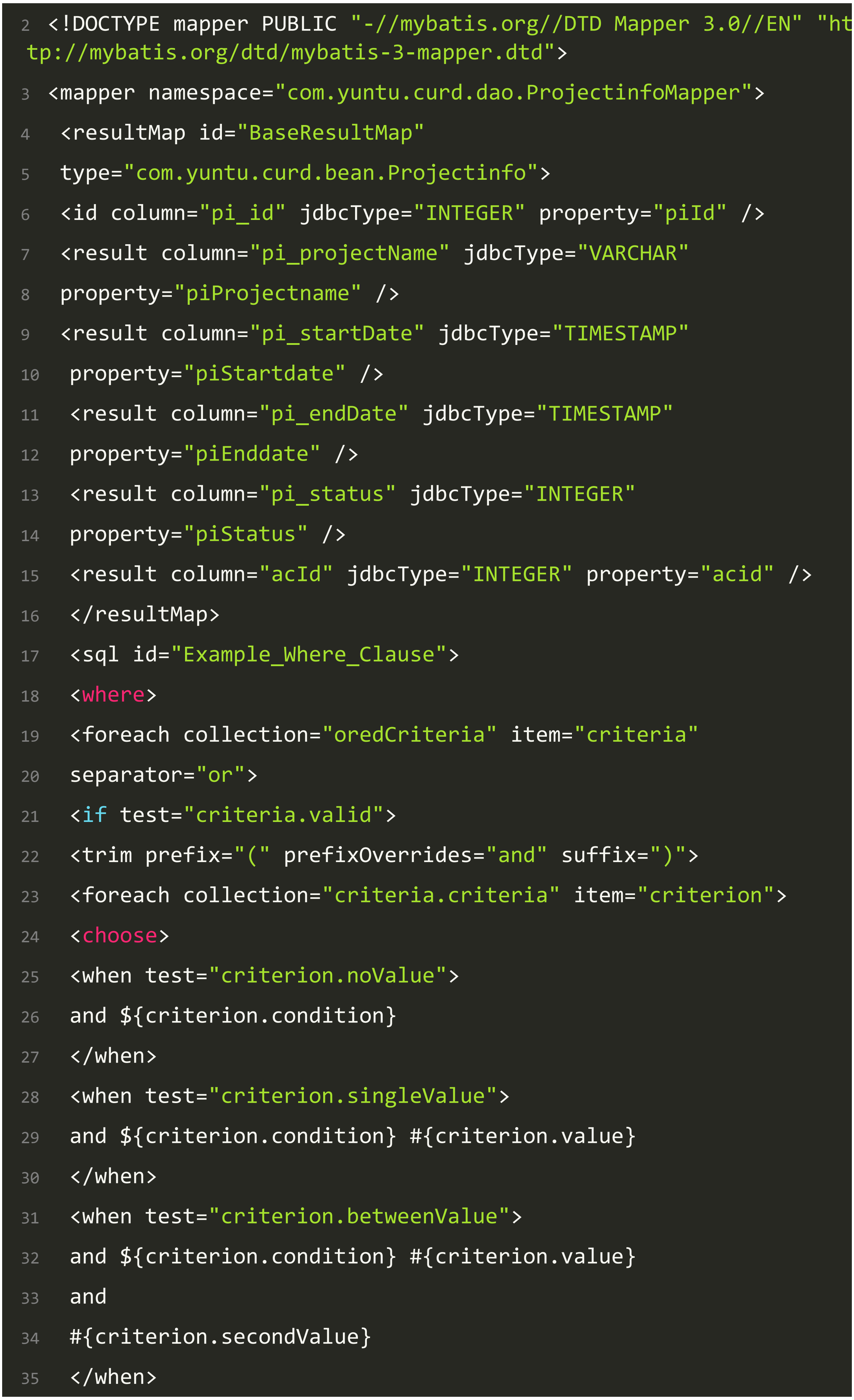


```
<if test="_parameter != null">
<include refid="Example_Where_Clause" />
</if>
</select>
<update id="updateByExampleSelective" parameterType="map">
update applicant
<set>
<if test="record.acId != null">
ac_id = #{record.acId,jdbcType=INTEGER},
</if>
<if test="record.acName != null">
ac_name = #{record.acName,jdbcType=VARCHAR},
</if>
<if test="record.acBirthday != null">
ac_birthday = #{record.acBirthday,jdbcType=TIMESTAMP},
</if>
<if test="record.acSex != null">
ac_sex = #{record.acSex,jdbcType=INTEGER},
</if>
<if test="record.workinglife != null">
workingLife = #{record.workinglife,jdbcType=INTEGER},
</if>
</set>
<if test="_parameter != null">
<include refid="Update_By_Example_Where_Clause" />
</if>
</update>
<update id="updateByExample" parameterType="map">
update applicant
set ac_id = #{record.acId,jdbcType=INTEGER},
ac_name = #{record.acName,jdbcType=VARCHAR},
ac_birthday = #{record.acBirthday,jdbcType=TIMESTAMP},
ac_sex = #{record.acSex,jdbcType=INTEGER},
workingLife = #{record.workinglife,jdbcType=INTEGER}
<if test="_parameter != null">
```

```
<include refid="Update_By_Example_Where_Clause" />
</if>
</update>
<update id="updateByPrimaryKeySelective" parameterType="com.yuntu.curd.bean.Applicant">
update applicant
<set>
<if test="acName != null">
ac_name = #{acName,jdbcType=VARCHAR},
</if>
<if test="acBirthday != null">
ac_birthday = #{acBirthday,jdbcType=TIMESTAMP},
</if>
<if test="acSex != null">
ac_sex = #{acSex,jdbcType=INTEGER},
</if>
<if test="workinglife != null">
workingLife = #{workinglife,jdbcType=INTEGER},
</if>
</set>
where ac_id = #{acId,jdbcType=INTEGER}
</update>
<update id="updateByPrimaryKey" parameterType="com.yuntu.curd.bean.Applicant">
update applicant
set ac_name = #{acName,jdbcType=VARCHAR},
ac_birthday = #{acBirthday,jdbcType=TIMESTAMP},
ac_sex = #{acSex,jdbcType=INTEGER},
workingLife = #{workinglife,jdbcType=INTEGER}
where ac_id = #{acId,jdbcType=INTEGER}
</update>
</mapper>
```

ProjectinfoMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```



```
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.yuntu.curd.dao.ProjectinfoMapper">
 <resultMap id="BaseResultMap"
 type="com.yuntu.curd.bean.Projectinfo">
 <id column="pi_id" jdbcType="INTEGER" property="piId" />
 <result column="pi_projectName" jdbcType="VARCHAR"
 property="piProjectname" />
 <result column="pi_startDate" jdbcType="TIMESTAMP"
  property="piStartdate" />
  <result column="pi_endDate" jdbcType="TIMESTAMP"
  property="piEnddate" />
  <result column="pi_status" jdbcType="INTEGER"
  property="piStatus" />
  <result column="acId" jdbcType="INTEGER" property="acid" />
  </resultMap>
  <sql id="Example_Where_Clause">
  <where>
  <foreach collection="oredCriteria" item="criteria"
  separator="or">
  <if test="criteria.valid">
  <trim prefix="(" prefixOverrides="and" suffix=")">
  <foreach collection="criteria.criteria" item="criterion">
  <choose>
  <when test="criterion.noValue">
  and ${criterion.condition}
  </when>
  <when test="criterion.singleValue">
  and ${criterion.condition} #{criterion.value}
  </when>
  <when test="criterion.betweenValue">
  and ${criterion.condition} #{criterion.value}
  and
  #{criterion.secondValue}
  </when>
```

```
<when test="criterion.listValue">
and ${criterion.condition}
<foreach close=")" collection="criterion.value"
item="listItem" open="(" separator=",">
#{listItem}
</foreach>
</when>
</choose>
</foreach>
</trim>
</if>
</foreach>
</where>
</sql>
<sql id="Update_By_Example_Where_Clause">
<where>
<foreach collection="example.oredCriteria" item="criteria"
separator="or">
<if test="criteria.valid">
<trim prefix="(" prefixOverrides="and" suffix=")">
<foreach collection="criteria.criteria" item="criterion">
<choose>
<when test="criterion.noValue">
and ${criterion.condition}
</when>
<when test="criterion.singleValue">
and ${criterion.condition} #{criterion.value}
</when>
<when test="criterion.betweenValue">
and ${criterion.condition} #{criterion.value}
and
#{criterion.secondValue}
</when>
<when test="criterion.listValue">
and ${criterion.condition}
```

```xml
<foreach close=")" collection="criterion.value"
item="listItem" open="(" separator=",">
#{listItem}
</foreach>
</when>
</choose>
</foreach>
</trim>
</if>
</foreach>
</where>
</sql>
<sql id="Base_Column_List">
pi_id, pi_projectName, pi_startDate, pi_endDate, pi_status,acId
</sql>



<!-- ***********自定义的sql查询开始********** -->
<!-- sql执行返回的结果对象 -->
<resultMap id="WithApplicantResultMap" type="com.yuntu.curd.bean.Projectinfo">
<id column="pi_id" jdbcType="INTEGER" property="piId" />
<result column="pi_projectName" jdbcType="VARCHAR" property="piProjectname" />
<result column="pi_startDate" jdbcType="TIMESTAMP" property="piStartdate" />
<result column="pi_endDate" jdbcType="TIMESTAMP" property="piEnddate" />
<result column="pi_status" jdbcType="INTEGER" property="piStatus" />
<result column="acId" jdbcType="INTEGER" property="acid" />
<!-- 关联申报人 -->
<association property="applicant" javaType="com.yuntu.curd.bean.Applicant">
 <id column="ac_id" jdbcType="INTEGER" property="acId" />
```

```
<result column="ac_name" jdbcType="VARCHAR" property="acName" />
<result column="ac_birthday" jdbcType="TIMESTAMP" property="acBirthday" />
<result column="ac_sex" jdbcType="INTEGER" property="acSex" />
<result column="workingLife" jdbcType="INTEGER" property="workinglife" />
</association>
</resultMap>
<!-- 联合查询所有的两表所有信息 -->
<sql id="WithApplicant_Column_List">
p.pi_id, p.pi_projectName, p.pi_startDate, p.pi_endDate, p.pi_status, p.acId , a.ac_id , a.ac_name , a.ac_birthday , a.ac_sex , a.workingLife
</sql>
<!-- 联合查询两张表 -->
<select id="selectByExampleWithApplicantion" parameterType="com.yuntu.curd.bean.ProjectinfoExample" resultMap="WithApplicantResultMap">
select
<if test="distinct">
distinct
</if>
<include refid="WithApplicant_Column_List" />
from projectinfo p
left join applicant a on p.acId=a.ac_id
<if test="_parameter != null">
<include refid="Example_Where_Clause" />
</if>
<if test="orderByClause != null">
order by ${orderByClause}
</if>
</select>

<select id="selectByPrimaryKeyWithApplicantion" parameterType="java.lang.Integer" resultMap="WithApplicantResultMap">
select
```

```
<include refid="WithApplicant_Column_List" />
from projectinfo p
left join applicant a on p.acId=a.ac_id
where pi_id = #{piId,jdbcType=INTEGER}
</select>
<!-- ***********自定义的sql查询结束********** -->



<select id="selectByExample"
parameterType="com.yuntu.curd.bean.ProjectinfoExample"
resultMap="BaseResultMap">
select
<if test="distinct">
distinct
</if>
<include refid="Base_Column_List" />
from projectinfo
<if test="_parameter != null">
<include refid="Example_Where_Clause" />
</if>
<if test="orderByClause != null">
order by ${orderByClause}
</if>
</select>
<select id="selectByPrimaryKey"
parameterType="java.lang.Integer" resultMap="BaseResultMap">
select
<include refid="Base_Column_List" />
from projectinfo
where pi_id = #{piId,jdbcType=INTEGER}
</select>
<delete id="deleteByPrimaryKey"
parameterType="java.lang.Integer">
delete from projectinfo
```

```
where pi_id =
#{piId,jdbcType=INTEGER}
</delete>
<delete id="deleteByExample"
parameterType="com.yuntu.curd.bean.ProjectinfoExample">
delete from projectinfo
<if test="_parameter != null">
<include refid="Example_Where_Clause" />
</if>
</delete>
<insert id="insert"
parameterType="com.yuntu.curd.bean.Projectinfo">
insert into projectinfo (pi_id, pi_projectName,
pi_startDate,
pi_endDate, pi_status, acId
)
values
(#{piId,jdbcType=INTEGER}, #{piProjectname,jdbcType=VARCHAR},
#{piStartdate,jdbcType=TIMESTAMP},
#{piEnddate,jdbcType=TIMESTAMP},
#{piStatus,jdbcType=INTEGER}, #{acid,jdbcType=INTEGER}
)
</insert>
<insert id="insertSelective"
parameterType="com.yuntu.curd.bean.Projectinfo">
insert into projectinfo
<trim prefix="(" suffix=")" suffixOverrides=",">
<if test="piId != null">
pi_id,
</if>
<if test="piProjectname != null">
pi_projectName,
</if>
<if test="piStartdate != null">
pi_startDate,
</if>
<if test="piEnddate != null">
pi_endDate,
</if>
<if test="piStatus != null">
pi_status,
</if>
<if test="acid != null">
acId,
</if>
</trim>
<trim prefix="values (" suffix=")" suffixOverrides=",">
<if test="piId != null">
#{piId,jdbcType=INTEGER},
</if>
<if test="piProjectname != null">
#{piProjectname,jdbcType=VARCHAR},
</if>
<if test="piStartdate != null">
#{piStartdate,jdbcType=TIMESTAMP},
</if>
<if test="piEnddate != null">
#{piEnddate,jdbcType=TIMESTAMP},
</if>
<if test="piStatus != null">
#{piStatus,jdbcType=INTEGER},
</if>
<if test="acid != null">
#{acid,jdbcType=INTEGER},
</if>
</trim>
</insert>
<select id="countByExample"
parameterType="com.yuntu.curd.bean.ProjectinfoExample"
resultType="java.lang.Long">
```

```
select count(*) from projectinfo
<if test="_parameter != null">
<include refid="Example_Where_Clause" />
</if>
</select>
<update id="updateByExampleSelective" parameterType="map">
update projectinfo
<set>
<if test="record.piId != null">
pi_id = #{record.piId,jdbcType=INTEGER},
</if>
<if test="record.piProjectname != null">
pi_projectName =
#{record.piProjectname,jdbcType=VARCHAR},
</if>
<if test="record.piStartdate != null">
pi_startDate = #{record.piStartdate,jdbcType=TIMESTAMP},
</if>
<if test="record.piEnddate != null">
pi_endDate = #{record.piEnddate,jdbcType=TIMESTAMP},
</if>
<if test="record.piStatus != null">
pi_status = #{record.piStatus,jdbcType=INTEGER},
</if>
<if test="record.acid != null">
acId = #{record.acid,jdbcType=INTEGER},
</if>
</set>
<if test="_parameter != null">
<include refid="Update_By_Example_Where_Clause" />
</if>
</update>
<update id="updateByExample" parameterType="map">
update projectinfo
set pi_id = #{record.piId,jdbcType=INTEGER},
pi_projectName = #{record.piProjectname,jdbcType=VARCHAR},
pi_startDate = #{record.piStartdate,jdbcType=TIMESTAMP},
pi_endDate =
#{record.piEnddate,jdbcType=TIMESTAMP},
pi_status =
#{record.piStatus,jdbcType=INTEGER},
acId =
#{record.acid,jdbcType=INTEGER}
<if test="_parameter != null">
<include refid="Update_By_Example_Where_Clause" />
</if>
</update>
<update id="updateByPrimaryKeySelective"
parameterType="com.yuntu.curd.bean.Projectinfo">
update projectinfo
<set>
<if test="piProjectname != null">
pi_projectName = #{piProjectname,jdbcType=VARCHAR},
</if>
<if test="piStartdate != null">
pi_startDate = #{piStartdate,jdbcType=TIMESTAMP},
</if>
<if test="piEnddate != null">
pi_endDate = #{piEnddate,jdbcType=TIMESTAMP},
</if>
<if test="piStatus != null">
pi_status = #{piStatus,jdbcType=INTEGER},
</if>
<if test="acid != null">
acId = #{acid,jdbcType=INTEGER},
</if>
</set>
where pi_id = #{piId,jdbcType=INTEGER}
</update>
<update id="updateByPrimaryKey"
```

```
 parameterType="com.yuntu.curd.bean.Projectinfo">
 update projectinfo
 set pi_projectName =
 #{piProjectname,jdbcType=VARCHAR},
 pi_startDate =
 #{piStartdate,jdbcType=TIMESTAMP},
 pi_endDate =
 #{piEnddate,jdbcType=TIMESTAMP},
 pi_status =
 #{piStatus,jdbcType=INTEGER},
 acId = #{acid,jdbcType=INTEGER}
 where
 pi_id = #{piId,jdbcType=INTEGER}
 </update>
</mapper>
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:mvc="http://www.springframework.org/schema/mvc"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:task="http://www.springframework.org/schema/task"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-4.0.xsd
  http://www.springframework.org/schema/mvc
  http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
  http://www.springframework.org/schema/aop
```

```xml
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-4.0.xsd">

<!-- Spring的配置文件,主要配置业务逻辑相关的内容 -->
<context:component-scan base-package="com.yuntu">
<!-- 除了控制器,其他都扫描 -->
<context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller" />
</context:component-scan>

<!-- ************数据源************ -->
<!-- 数据源(默认直接配置在applicationContext.xml文件中;放在database.properties文件中; JNDI) -->
<context:property-placeholder location="classpath:dbconfig.properties"/>
<bean id="pooledDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
<!-- URL -->
<property name="jdbcUrl" value="${jdbc.jdbcUrl}"></property>
<!-- 驱动 -->
<property name="driverClass" value="${jdbc.driverClass}"></property>
<!-- 账号 -->
<property name="user" value="${jdbc.user}"></property>
<!-- 密码 -->
<property name="password" value="${jdbc.password}"></property>
</bean>
<!-- ************数据源************ -->


<!-- ==============配置mybatis的整合============== -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
```

```xml
<!-- 指定mybatis全局配置文件的位置 -->
<property name="configLocation" value="classpath:mybatis-config.xml"></property>
<!-- 指定数据源的使用 -->
<property name="dataSource" ref="pooledDataSource"></property>
<!-- 指定mybatis，mapper文件所在的位置(类路径下的所有的映射文件) -->
<property name="mapperLocations" value="classpath:mapper/*.xml"></property>
</bean>

<!-- 配置扫描器,将mybatis接口的实现加入到Ioc容器中(接口实现类) -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
<property name="basePackage" value="com.yuntu.curd.dao"></property>
</bean>

<!-- 配置一个可以批量执行的sqlSession -->
<bean id="sSqlSession" class="org.mybatis.spring.SqlSessionTemplate">
<!-- 指定sqlSession会话工厂 -->
<constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"></constructor-arg>
<!-- mybatis默认使用单个更新（executorType默认为SIMPLE而不是BATCH）
SIMPLE：单个模式
BATCH：批量模式
-->
<constructor-arg name="executorType" value="BATCH"></constructor-arg>
</bean>
<!-- ==============配置mybatis的整合==============* -->

<!-- ************事务控制配置************ -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<!-- 添加数据源 -->
```

```xml
<property name="dataSource" ref="pooledDataSource"></property>
</bean>

<!-- 配置事务增强 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
<tx:attributes>
<!-- 所有的方法都是事务方法 -->
<tx:method name="*"/>
<!-- eg:以get开头的方法 -->
<tx:method name="get*" read-only="true"/>
</tx:attributes>
</tx:advice>

<!-- 1.开启基于注解的事务 2.使用xml配置的形式的事务(一般情况下，比较重要的都是使用配置的事务) -->
<aop:config>
<!-- 配置切入点 -->
<aop:pointcut expression="execution(* com.yuntu.curd.service.*.*(..))" id="txPoint"/>
<!-- 事务增强 -->
<aop:advisor advice-ref="txAdvice" pointcut-ref="txPoint"/>
</aop:config>

<!-- *************事务控制配置************* -->

</beans>
```

dbconfig.properties

```
jdbc.jdbcUrl=jdbc:mysql://127.0.0.1/dpm?useUnicode=true&characterEncoding=utf-8
jdbc.driverClass=com.mysql.jdbc.Driver
jdbc.user=root
jdbc.password=123456
```

mybatis-config.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE configuration
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
 "http://mybatis.org/dtd/mybatis-3-config.dtd">

 <!-- mybatis配置文件 -->
 <configuration>
 <!-- 设置（驼峰命名规则） -->
 <settings>
  <setting name="mapUnderscoreToCamelCase" value="true"/>
  </settings>

  <!-- 类型别名（实体类包） -->
  <typeAliases>
  <package name="com.yuntu.curd.bean"/>
  </typeAliases>

  <!-- 引入分页插件 -->
  <plugins>
  <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>
  </plugins>
  </configuration>
```

dispatcherServlet-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:mvc="http://www.springframework.org/schema/mvc"
 xmlns:aop="http://www.springframework.org/schema/aop"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:task="http://www.springframework.org/schema/task"
  xsi:schemaLocation="
  http://www.springframework.org/schema/beans
```

```xml
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-4.0.xsd">

<!-- SpringMVC的配置文件,包含网站的跳转逻辑的控制以及配置 -->
<!-- 注解扫描 -->
<context:component-scan base-package="com.yuntu"
use-default-filters="false">
<!-- 不用全部都扫描，只需要扫描控制器组件，所以要禁用掉默认的扫描规则 通过注解扫描Constroller组件，只扫描控制器 -->
<context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller" />
</context:component-scan>

<!-- 配置视图解析器，方便控制器处理结果之后进行页面的返回 -->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="prefix" value="/WEB-INF/views/"></property>
<property name="suffix" value=".jsp"></property>
</bean>

<!-- 进行辅助配置-->
<!-- 将SpringMVC不能处理的请求交给Tomcat处理 -->
<mvc:default-servlet-handler/>
<!-- 能支持springmvc更高级的注解功能(JSR303,快捷的AJAX，映射动态功能) -->
```

```xml
  <mvc:annotation-driven></mvc:annotation-driven>

</beans>
```

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
 <display-name>yuntu-dpm-curd</display-name>
 <welcome-file-list>
 <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>

 <!-- 1.启动spring容器，加载spring的配置文件 -->
 <context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:applicationContext.xml</param-value>
  </context-param>
  <listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <!-- 2.配置SpringMVC 前端控制器，拦截所有请求进行处理 -->
  <servlet>
  <servlet-name>dispatcherServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <!-- 指明SpringMVC的配置文件路径
  也可以不指定，需要跟web.xml文件处于同级目录，创建名为xxx-servlet.xml文件
  -->
  <!-- <init-param>
  <param-name></param-name>
  <param-value></param-value>
```

```
</init-param> -->
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>dispatcherServlet</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>

<!-- 3.字符编码过滤器（必须放在所有的过滤器之前） -->
<filter>
<filter-name>characterEncodingFilter</filter-name>
<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
<!-- 设置编码格式 -->
<init-param>
<param-name>encoding</param-name>
<param-value>utf-8</param-value>
</init-param>
<!-- 设置强制请求是否使用 -->
<init-param>
<param-name>forceRequestEncoding</param-name>
<param-value>true</param-value>
</init-param>
<!-- 设置强制响应是否使用 -->
<init-param>
<param-name>forceResponseEncoding</param-name>
<param-value>true</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>characterEncodingFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>

<!-- 4.使用REST风格的URI（将页面普通的post请求转为指定的delete或者put请求） -->
```

```xml
 <filter>
 <filter-name>HiddenHttpMethodFilter</filter-name>
 <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
 </filter>
 <filter-mapping>
 <filter-name>HiddenHttpMethodFilter</filter-name>
 <url-pattern>/*</url-pattern>
 </filter-mapping>

 <!-- 添加PUT请求的过滤器 -->
 <filter>
 <filter-name>HttpPutFormContentFilter</filter-name>
 <filter-class>org.springframework.web.filter.HttpPutFormContentFilter</filter-class>
 </filter>
 <filter-mapping>
 <filter-name>HttpPutFormContentFilter</filter-name>
 <url-pattern>/*</url-pattern>
 </filter-mapping>

</web-app>
```

index.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<%
 String path=request.getContextPath();
 String basePath=request.getScheme()+"//"+request.getServerName()+":"+
 request.getServerPort()+path+"/";
 //将path放到页作用域，方便使用el表达式调用
 pageContext.setAttribute("app_path", path);

%>
```

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>YUNTU_DPM_CURD</title>
<!-- 导入前端框架（jQuery的引入在bootstrap之前）-->
<link rel="stylesheet" type="text/css" href="${app_path}/static/bootstrap-3.3.7-dist/css/bootstrap.min.css">
<script type="text/javascript" src="${app_path}/static/js/jquery-1.12.3.js"></script>
<script type="text/javascript" src="${app_path}/static/bootstrap-3.3.7-dist/js/bootstrap.min.js"></script>
</head>
<body>

 <!-- 用于修改的模态框 -->
 <div class="modal fade" id="projectInfoUpdateModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
 <div class="modal-dialog" role="document">
 <div class="modal-content">
 <div class="modal-header">
 <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">×</span></button>
 <h4 class="modal-title" id="myModalLabel">项目编辑</h4>
 </div>
 <div class="modal-body">

 <!-- 模态框中的主体，表单构成 -->
 <form class="form-horizontal">
 <!-- 项目名称 -->
 <div class="form-group">
 <label for="ProjectName_add_input" class="col-sm-2 control-label">ProjectName</label>
 <div class="col-sm-10">
 <p class="form-control-static" id="projectName_update_static">
 </div>
 </div>
 <!-- 起始时间 -->
```

```html
<div class="form-group">
<label for="startDate_update_input" class="col-sm-2 control-label">StartDate</label>
<div class="col-sm-10">
<input type="date" name="piStartdate" class="form-control" id="startDate_update_input" placeholder="StartDate">
</div>
</div>
<!-- 结束时间 -->
<div class="form-group">
<label for="endDate_update_input" class="col-sm-2 control-label">EndDate</label>
<div class="col-sm-10">
<input type="date" name="piEnddate" class="form-control" id="endDate_update_input" placeholder="EndDate">
</div>
</div>
<!-- 申报状态 -->
<div class="form-group">
<label for="status_update_input" class="col-sm-2 control-label">Status</label>
<div class="col-sm-6">
<!-- 状态的显示和提交保存的值是不一样的 -->
<select class="form-control" name="piStatus">
<option value="0">已申报</option>
<option value="1">申报中</option>
<option value="2">已审核</option>
</select>
</div>
</div>
<!-- 申报人列表 -->
<div class="form-group">
<label for="acid_update_input" class="col-sm-2 control-label">Applicant</label>
<div class="col-sm-6">
<!-- 内容时查询出来名字，提交的值是申报人的acid编号 -->
<select class="form-control" name="acid">

</select>
```

```html
</div>
</div>
</form>
</div>
<div class="modal-footer">
<button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
<button type="button" class="btn btn-primary" id="projectInfo_update_btn">保存</button>
</div>
</div>
</div>
</div>





<!-- 用于添加的模态框 -->
<div class="modal fade" id="projectInfoAddModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
<div class="modal-dialog" role="document">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">×</span></button>
<h4 class="modal-title" id="myModalLabel">项目申报</h4>
</div>
<div class="modal-body">

<!-- 模态框中的主体，表单构成 -->
<form class="form-horizontal">
<!-- 项目名称 -->
<div class="form-group">
<label for="ProjectName_add_input" class="col-sm-2 control-label">ProjectName</label>
<div class="col-sm-10">
<input type="text" name="piProjectname" class="form-control" id="projectName_add_input" placeholder="ProjectName">
<!-- 显示错误提示信息 -->
```

```
<span class="help-block"></span>
</div>
</div>
<!-- 起始时间 -->
<div class="form-group">
<label for="startDate_add_input" class="col-sm-2 control-label">StartDate</label>
<div class="col-sm-10">
<input type="date" name="piStartdate" class="form-control" id="startDate_add_input" placeholder="StartDate">
</div>
</div>
<!-- 结束时间 -->
<div class="form-group">
<label for="endDate_add_input" class="col-sm-2 control-label">EndDate</label>
<div class="col-sm-10">
<input type="date" name="piEnddate" class="form-control" id="endDate_add_input" placeholder="EndDate">
</div>
</div>
<!-- 申报状态 -->
<div class="form-group">
<label for="status_add_input" class="col-sm-2 control-label">Status</label>
<div class="col-sm-6">
<!-- 状态的显示和提交保存的值是不一样的 -->
<select class="form-control" name="piStatus">
<option value="0">已申报</option>
<option value="1">申报中</option>
<option value="2">已审核</option>
</select>
</div>
</div>
<!-- 申报人列表 -->
<div class="form-group">
<label for="applicant_add_input" class="col-sm-2 control-label">Applicant</label>
<div class="col-sm-6">
```

```html
<!-- 内容时查询出来名字，提交的值是申报人的acid编号 -->
<select class="form-control" name="acid">

</select>
</div>
</div>
</form>
</div>
<div class="modal-footer">
<button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
<button type="button" class="btn btn-primary" id="projectInfo_save_btn">保存</button>
</div>
</div>
</div>
</div>


<!-- 主页内容 -->
<div class="container-fluid">
<!-- 标题行 -->
<div class="row">
<div class="col-md-12">
<h1>YUNTU_DPM_CURD</h1>
</div>
</div>
<!-- 按钮行 -->
<div class="row">
<!-- col-md-2:占两格
col-md-offset-10：向右偏移是个格子
-->
<div class="col-md-2 col-md-offset-10" style="margin-bottom:10px;">
<button type="button" class="btn btn-success" id="projectInfo_add_btn">新增</button>
<button type="button" class="btn btn-danger" id="projectInfo_delete_all_btn">删除</button>
</div>
```

```html
</div>
<!-- 显示表格数据 -->
<div class="row">
<div class="col-md-12">
<!-- 表格展示出数据 -->
<table class="table table-striped table-hover" id="projectInfos_table">
<!-- 列标题 -->
<thead>
<tr>
<td>
<input type="checkbox" id="check_all">
</td>
<td>项目编号</td>
<td>项目名称（#）</td>
<td>开始日期</td>
<td>结束日期</td>
<td>申报状态</td>
<td>申报人</td>
<td>性别</td>
<td>工作年限</td>
<td>操作</td>
</tr>
</thead>
<tbody>

</tbody>
</table>
</div>
</div>
<!-- 显示分页信息 -->
<div class="row">
<!-- 分页文字信息 -->
<div class="col-md-4 col-md-offset-2" id="page_info_area"></div>
<!-- 分页导航信息 -->
<div class="col-md-6" id="page_nav_area"></div>
</div>
</div>

```

```

</body>
<script type="text/javascript">
//全局变量
var totalCount,currentPage;
/*1.页面加载完成后直接发送ajax请求，拿到分页数据*/
$(function() {
 //页面进入直接访问第一页数据
 top_page(1);
})

function top_page(pn) {
 $.ajax({
 url:"${app_path}/projectInfos",
 data:"pn="+pn,
 type:"get",
 success:function(result){
 //console.log(result);
 //1.解析JSON数据并显示员工信息
 build_projectInfos_table(result);
 //2.解析并显示分页文字分页信息
 build_page_info(result);
 //3.解析并显示分页文字导航条
 build_page_nav(result);
 }
 });
}

function build_projectInfos_table(result) {
 //0.构建表格之前，将表格中数据全部清空，防止出现多个表格数据
 $("#projectInfos_table tbody").empty();

 //1.解析json数据，拿到其中的extend值中的PageInfo中的list
 var projectInfos=result.extend.pageInfo.list;
 //遍历
 $.each(projectInfos,function(index,item){
 //console.log(item.piProjectname,index)
 //复选框
```

```
var checkBoxTd=$("<td><input type='checkbox' class='check_item' /></td>");
//获取item中的每一个属性信息
var piIDTD=$("<td></td>").append(item.piId);
var piProjectnameTd=$("<td></td>").append(item.piProjectname);
var piStartdateTd=$("<td></td>").append(dateFarmat("yyyy-MM-dd",new Date(item.piStartdate)));
var piEnddateTd=$("<td></td>").append(dateFarmat("yyyy-MM-dd",new Date(item.piEnddate)));
var piStatusTd=$("<td></td>").append(item.piStatus);
var acNameTd=$("<td></td>").append(item.applicant.acName);
var acSexTd=$("<td></td>").append(item.applicant.acSex==1?'男':'女');
var workinglifeTd=$("<td></td>").append(item.applicant.workinglife);

//编辑按钮
var editBtn=$("<button></button>").addClass("btn btn-primary btn-sm edit_btn").
append($("<span></span>")).addClass("glyphicon glyphicon-pencil").append("编辑");
//为编辑按钮添加自定义属性，表示当前操作项的id
editBtn.attr("edit-id",item.piId);

//删除按钮
var delBtn=$("<button></button>").addClass("btn btn-danger btn-sm delete_btn").
append($("<span></span>")).addClass("glyphicon glyphicon-trash").append("删除");
//为编辑按钮添加自定义属性，表示当前操作项的id
delBtn.attr("del-id",item.piId);

//将两个按钮拼接在一起
var btnTd=$("<td></td>").append(editBtn).append(" ").append(delBtn);

//组合
$("<tr></tr>").append(checkBoxTd).append(piIDTD).append(piProjectnameTd).append(piStartdateTd).
```

```
 append(piEnddateTd).append(piStatusTd).append(acNameTd).append(acSexTd).
 append(workinglifeTd).append(btnTd).appendTo("#projectInfos_table tbody");
 });
}

//日期格式化操作
//dateFarmat("yyyy-MM-dd hh:mm:ss",new Date(item.piStartdate))
function dateFarmat(fmt,date)
{
 var o = {
 "M+" : date.getMonth()+1, //月份
 "d+" : date.getDate(), //日
 "h+" : date.getHours(), //小时
 "m+" : date.getMinutes(), //分
 "s+" : date.getSeconds(), //秒
 //季度
 "q+" : Math.floor((date.getMonth()+3)/3),
 "S" : date.getMilliseconds() //毫秒
 };
 if(/(y+)/.test(fmt))
 fmt=fmt.replace(RegExp.$1, (date.getFullYear()+"").substr(4 - RegExp.$1.length));
 for(var k in o)
 if(new RegExp("("+ k +")").test(fmt))
 fmt = fmt.replace(RegExp.$1, (RegExp.$1.length==1) ? (o[k]) : (("00"+ o[k]).substr((""+ o[k]).length)));
 return fmt;
}

//2.
function build_page_info(result) {
 //构建分页信息之前进行分页元素的清空
 $("#page_info_area").empty();
 //构建分页信息数据
 $("#page_info_area").append(
 "当前第"+result.extend.pageInfo.pageNum+
 "页，总共"+result.extend.pageInfo.pages+
 "页，总共"+result.extend.pageInfo.total+"条记录"
 );
 //全局变量totalCount和currentPage
 totalCount=result.extend.pageInfo.pages;
 currentPage=result.extend.pageInfo.pageNum;
}

//3.
function build_page_nav(result) {
 //构建分页导航之前进行分页导航元素的清空
 $("#page_nav_area").empty();

 //构建ul元素
 var ul=$("<ul></ul>").addClass("pagination");

 //首页元素
 var firstPageLi=$("<li></li>").append($("<a></a>").append("首页").attr("href","#"));
 //上一页元素
 var prePageLi=$("<li></li>").append($("<a></a>").append("«").attr("href","#"));
 //判断是否还有上一页数据
 if (result.extend.pageInfo.hasPreviousPage == false) {//没有上一页
 //禁用上一页和首页按钮，给首页元素添加类样式
 firstPageLi.addClass("disabled");
 prePageLi.addClass("disabled");
 } else {
 //为元素添加点击翻页事件
 firstPageLi.click(function() {
 top_page(1);
 });
 //为元素添加点击翻页事件
 prePageLi.click(function() {
 top_page(result.extend.pageInfo.pageNum-1);
 });
 }

 //下一页元素
```

```
var nextPageLi=$("<li></li>").append($("<a></a>").append("»").attr("href","#"));
//末页元素
var lastPageLi=$("<li></li>").append($("<a></a>").append("末页").attr("href","#"));
//判断是否还有下一页数据
if (result.extend.pageInfo.hasNextPage == false) {//没有上一页
//禁用下一页和末页按钮，给首页元素添加类样式
nextPageLi.addClass("disabled");
lastPageLi.addClass("disabled");
} else {
//为下一页添加点击翻页事件
nextPageLi.click(function() {
top_page(result.extend.pageInfo.pageNum+1);
});
//为末页点击翻页事件
lastPageLi.click(function() {
top_page(result.extend.pageInfo.pages);
});
}

//添加首页和上一页
ul.append(firstPageLi).append(prePageLi);
//遍历中间的数字
$.each(result.extend.pageInfo.navigatepageNums,function(index, item) {
//构建li的元素的对象
var numLi=$("<li></li>").append($("<a></a>").append(item).attr("href","#"));
//判断数字是否是当前显示的页码，添加bootstrap选中样式处理
if (result.extend.pageInfo.pageNum == item) {
numLi.addClass("active");
}
//每一个li的点击事件都是通过top_page函数进行ajax访问
numLi.click(function() {
top_page(item);
});

//循环将每个li添加ul中
```

```
 ul.append(numLi);
 });
 //添加下一页和末页
 ul.append(nextPageLi).append(lastPageLi);
 //将ul加到nav导航条中
 var navEle=$("<nav></nav>").append(ul);
 //显示
 navEle.appendTo("#page_nav_area");

}


//添加按钮的单击事件，弹出模态框
$("#projectInfo_add_btn").click(function() {
 //进行表单内容的重置（包括表单数据，表单样式）
 clear_add_from("#projectInfoAddModal form");
 //发送ajax请求，查询申报人信息显示在项目申报得下拉列表
 getApplicants("#projectInfoAddModal select[name=acid]");

 //模态框弹出
 $('#projectInfoAddModal').modal({
 backdrop:'static'
 });


});

//表单清空
function clear_add_from(ele) {
 //重置
 $(ele)[0].reset();
 //清空样式
 $(ele).find("*").removeClass("has-error has-success");
 //清空表单附加提示信息
 $(ele).find(".help-block").text("");
}

//获取申报人列表信息
function getApplicants(ele) {
 //先清除下拉列表中的原本的元素对象
 $(ele).empty();

 //发送ajax请求获取所有申报人
 $.ajax({
 url:"${app_path}/applicants",
 type:"GET",
 success:function(result){
 //console.log(result);
 //显示申报人信息在下拉列表中
 $.each(result.extend.applicants,function(){
 //console.log(result.extend.applicants);
 //构建列表项内容
 var optionEle=$("<option></option>").append(this.acName).attr("value",this.acId);
 //将构建的option列表添加在select里面
 optionEle.appendTo(ele);
 });
 }

 });
}

//显示验证之后的提示信息
function show_validate_msg(ele, status, msg) {//选择器，状态信息，展示信息
 //每次显示验证信息时，都将之前类样式清除
 $(ele).parent().removeClass("has-success has-error");
 //判断状态，确定状态对应的样式
 if ("success"==status) {
 //正确，添加正确样式
 $(ele).parent().addClass("has-success");
 } else if("error"==status) {
 //失败，添加失败样式
 $(ele).parent().addClass("has-error");
 }
 //展示正确或者错误的文字信息
 $(ele).next("span").text(msg);
}
```

```


//数据校验（添加表单）
function validate_add_form() {
 //主要校验项目名称格式
 //1.拿到校验数据，使用正则表达式
 var projectName=$("#projectName_add_input").val();
 var regName=/(^[a-zA-Z0-9_-]{6,64}$)|(^[\u2E80-\u9FFF]{2,32}$)/;
 //验证
 if (!regName.test(projectName)) {
 //调用验证信息的显示函数
 show_validate_msg("#projectName_add_input","error","论文名为2~32为中文，或者6~64为大小写字母、数字、下划线、横杠的组合");
 return false;
 } else {
 //调用验证信息的显示函数
 show_validate_msg("#projectName_add_input","success","");
 }
 return true;
}

//点击模态框中的保存，动态添加项目申报数据
$("#projectInfo_save_btn").click(function() {
 //0.先要对提交给服务器的数据校验
 if (!validate_add_form()) {//如果校验校验不成功，点击保存的事件就不再执行
 return false;
 }
 //1.通过ajax请求用户名校验是否可用；如果没有才能继续保存
 if ($(this).attr("ajax_validata_projectName")=="error") {
 return false;
 }

 //将模态框中的数据交给应用服务器，进行数据库服务器保存
 //发送ajax请求，保存员工
 /*
 通过jQuery提供的表单序列化进行表单数据提交（这用就不用通过id选择一个一个的获取一个一个值
 而是用serialize()得到信息）
 */
 $.ajax({
 url:"${app_path}/projectInfo",
 type:"POST",
 data:$("#projectInfoAddModal form").serialize(),
 success:function(result){
 //console.log(result);
 if (result.code == 200) {//成功
 //项目保存
 //1.关闭模态框
 $('#projectInfoAddModal').modal('toggle');
 //2.来到最后一页显示刚才数据
 top_page(totalCount);
 } else {//失败
 //判断时哪一个字段失败（假如失败中的值为undefined，证明代码有错误信息）
 if (result.extend.errorFields.piProjectname!=undefined) {
 show_validate_msg("#projectName_add_input","success",result.extend.errorFields.piProjectname);
 }
 }
 }
 });

});

//进行项目名重复验证，ajax异步请求
$("#projectName_add_input").change(function() {
 //获取输入框的值
 var projectName=this.value;
 //如果校验的方法返回不成功，用户名重复验证就不再执行
 if (!validate_add_form()) {//如果校验校验不成功，点击保存的事件就不再执行
 return false;
 }
 //发送ajax请求校验用户是否可用
 $.ajax({
 url:"${app_path}/checkProjectInfo",
 data:"projectName="+projectName,
 type:"POST",
 success:function(result){
 //判断返回的Msg对象中的状态码
 if (result.code == 200) {//成功
 show_validate_msg("#projectName_add_input","success","√ 可用");
 //通过jq对按钮进行携带信息（保证验证项目名重复时点击保存不能提交模态框）
 $("#projectInfo_save_btn").attr("ajax_validata_projectName","success");
 }else{
 show_validate_msg("#projectName_add_input","error",result.extend.validate_msg);
 $("#projectInfo_save_btn").attr("ajax_validata_projectName","error");
 }
 }
 });
});


//编辑按钮的单击事件
/*
 * 加载dom元素时，此时还没有编辑按钮直接绑定click函数无法绑定、
 解决方案：
 动态绑定：live()函数（已过去） on()函数（）
 */
//on()函数绑定，绑定整个文档中
$(document).on("click",".edit_btn",function(){
 //1.查询项目信息
 getProjectInfo($(this).attr("edit-id"));
 //2. 查询申报人的信息
 getApplicants("#projectInfoUpdateModal select[name=acid]");
 //3.将需要更新的员工id从编辑按钮上携带给模态框，用来修改时候使用；然后再弹出模态框
 $("#projectInfo_update_btn").attr("edit-id", $(this).attr("edit-id"));


 //然后模态框弹出
 $('#projectInfoUpdateModal').modal({
 backdrop:'static'
 });

});
//1.查询单个项目信息
function getProjectInfo(id) {
 $.ajax({
 url:"${app_path}/projectInfo/"+id,
 type:"GET",
 success:function(result){
 console.log(result);
 //获取员工对象
 var projectInfo=result.extend.projectInfo;
 //将员工对象放在模态框中对应的位置
 $("#projectName_update_static").text(projectInfo.piProjectname);
 $("#startDate_update_input").val(dateFarmat("yyyy-MM-dd",new Date(projectInfo.piStartdate)));
 $("#endDate_update_input").val(dateFarmat("yyyy-MM-dd",new Date(projectInfo.piEnddate)));
 $("#projectInfoUpdateModal select[name=piStatus]").val([parseInt(projectInfo.piStatus)]);
 $("#projectInfoUpdateModal select[name=acid]").val([parseInt(projectInfo.acid)]);

 }
 });
}

//修改模态框中保存按钮的点击事件
$("#projectInfo_update_btn").click(function() {
 //1.验证数据合法性，数据校验（根据需求）
 //2.发送ajax请求，进行员工的数据更新（PUT方式请求）
 $.ajax({
 url:"${app_path}/projectInfo/"+$(this).attr("edit-id"),
 type:"PUT",
 data:$("#projectInfoUpdateModal form").serialize(),
 success:function(result){
 //隐藏模态框
 $("#projectInfoUpdateModal").modal('hide');
 //跳转到当前页面
 top_page(currentPage);
 }
 });

});


//全选，全不选
$("#check_all").click(function() {
 //将当前操作复选框的checked的属性值，作为参数
 //设置为所有子项的复选框的checked的属性值
 $(".check_item").prop("checked",$(this).prop("checked"));
});

//每个选中，全选按钮也选中
$(document).on("click",".check_item",function(){
 //获取选中状态（选中状态是否等于复选框的总数量）
 var flag=$(".check_item:checked").length==$(".check_item").length;
 $("#check_all").prop("checked", flag);
});

//单个删除按钮的动态绑定点击事件
$(document).on("click",".delete_btn",function(){
 //1.获取删除项的名字
 var projectInfoName=$(this).parents("tr").find("td:eq(2)").text();
 //2.获取按钮上的del-id
 var delId=$(this).attr("del-id");
 //询问判断
 if (confirm("您确定要删除“"+ projectInfoName +"”吗？")) {
 //发送ajax请求
 $.ajax({
 url:"${app_path}/projectInfo/"+delId,
 type:"DELETE",
 success:function(result){
 //console.log(result);
 //跳转到当前页
 top_page(currentPage);
 }
 });
 }
});

//点击删除所选项
$("#projectInfo_delete_all_btn").click(function() {
 //创建用于存放删除的名字
 var projectInfoNames="";
 //创建用于存放项目id的字符串
 var del_ids_str="";
 //循环遍历
 $.each($(".check_item:checked"),function(){
 //将遍历的项目名称拼接到projectInfoNames中，并且用","隔开
 projectInfoNames+=$(this).parents("tr").find("td:eq(2)").text()+(",");
 //将遍历的id拼接到del_ids_str中，用"-"隔开
 del_ids_str+=$(this).parents("tr").find("td:eq(1)").text()+"-";
 });
 //去掉一个","
 projectInfoNames += projectInfoNames.substring(0, projectInfoNames.length-1);
 //去掉一个"-"
 del_ids_str += del_ids_str.substring(0, del_ids_str.length-1);
 if (confirm("您确定要删除 “"+ projectInfoNames +"” 吗？ ")) {
 //发送ajax请求
 $.ajax({
 url:"${app_path}/projectInfo/"+del_ids_str,
 type:"DELETE",
 success:function(result){
 console.log(result);
 top_page(currentPage);
 }
 });
 }
});
</script>
```

```
</html>
```

mbg.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
 PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
 "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>
 <context id="DB2Tables" targetRuntime="MyBatis3">
 <!-- 注释生成配置 -->
 <commentGenerator>
 <!-- 去掉所有自动生成的注释 -->
 <property name="suppressAllComments" value="true"/>
 </commentGenerator>

 <!-- 配置数据库连接信息 -->
 <jdbcConnection driverClass="com.mysql.jdbc.Driver"
 connectionURL="jdbc:mysql://127.0.0.1/dpm"
 userId="root"
 password="123456">
 </jdbcConnection>

 <javaTypeResolver >
 <property name="forceBigDecimals" value="false" />
 </javaTypeResolver>

 <!-- 指定JavaBean生成配置，位置
 targetPackage：目标包位置
 targetProject：所在项目位置
 -->
 <javaModelGenerator targetPackage="com.yuntu.curd.bean" targetProject=".\src\main\java">
 <property name="enableSubPackages" value="true" />
 <property name="trimStrings" value="true" />
```

```
</javaModelGenerator>

<!-- 指定sql映射文件生成配置，位置
targetPackage：目标包位置
targetProject：所在项目位置
-->
<sqlMapGenerator targetPackage="mapper" targetProject=".\src\main\resources">
<property name="enableSubPackages" value="true" />
</sqlMapGenerator>

<!-- 指定Dao接口生成的位置，mapper接口 -->
<javaClientGenerator type="XMLMAPPER" targetPackage="com.yuntu.curd.dao" targetProject=".\src\main\java">
<property name="enableSubPackages" value="true" />
</javaClientGenerator>

<!-- 每张表生成策略
tableName：表名
domainObjectName：实体类名
-->
<table tableName="applicant" domainObjectName="Applicant"></table>
<table tableName="projectinfo" domainObjectName="Projectinfo"></table>
<!-- <table schema="DB2ADMIN" tableName="ALLTYPES" domainObjectName="Customer" >
<property name="useActualColumnNames" value="true"/>
<generatedKey column="ID" sqlStatement="DB2" identity="true" />
<columnOverride column="DATE_FIELD" property="startDate" />
<ignoreColumn column="FRED" />
<columnOverride column="LONG_VARCHAR_FIELD" jdbcType="VARCHAR" />
</table> -->

</context>
```

```xml
</generatorConfiguration>
```