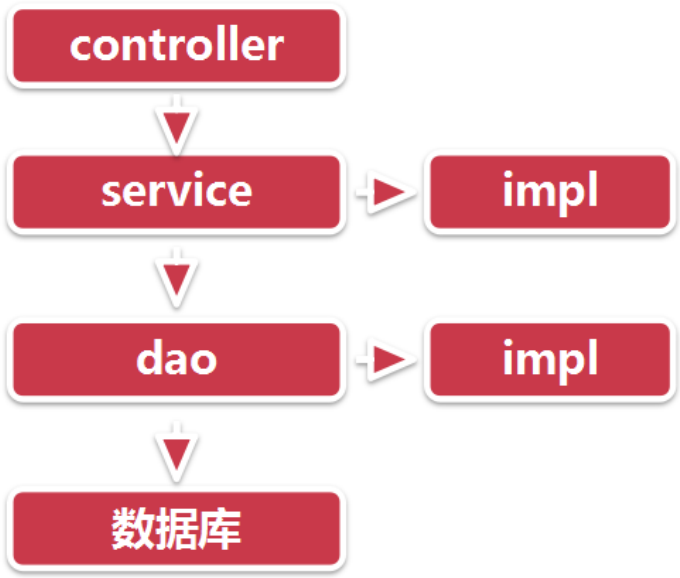
【Java框架型项目从入门到装逼】第十二节 项目分层

这一节我们开始对项目进行分层，一般来说，一个web项目的层次结构如下图所示：

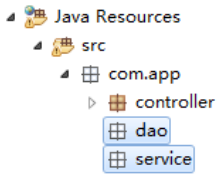


controller层为我们的控制层，用来接收用户的请求，比如新增一个学生的信息，新增的请求最先就是走到这一层。controller层只管接收用户的请求，不会涉及太多的业务处理操作。但凡涉及到业务处理，就交给service层来操作。所以，controller层中必然拥有某一个service层的引用。

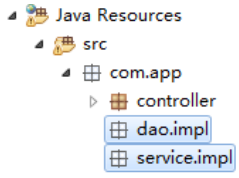
service层主要用来处理一些业务逻辑，不做任何的数据库操作。数据库的操作都交给dao层来做，因此，在service层中必然拥有一个dao层的引用。

一般来说，service层和dao层中，都是直接存放的接口类，然后专门有一个包放所有接口的具体实现类，impl就是指每个接口对应的实现类。

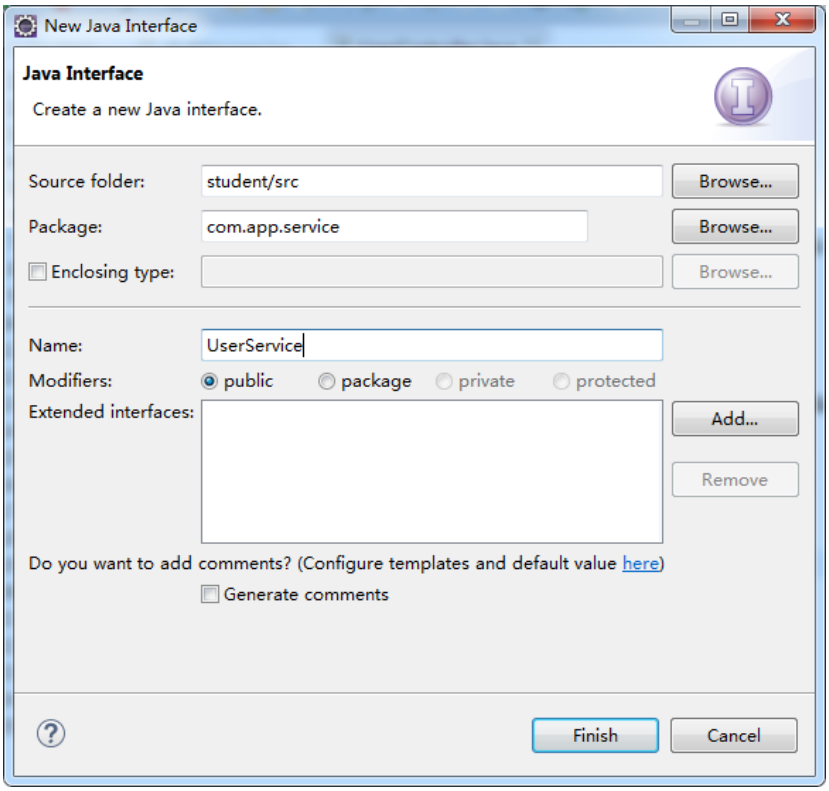
说了这么多概念，还是让我们一步一步来实际操作吧。首先，新建两个包，service包和dao包。

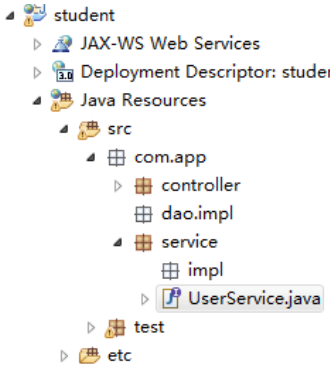


然后，在每个包里面再新建一个impl包，用来放所有的实现类。

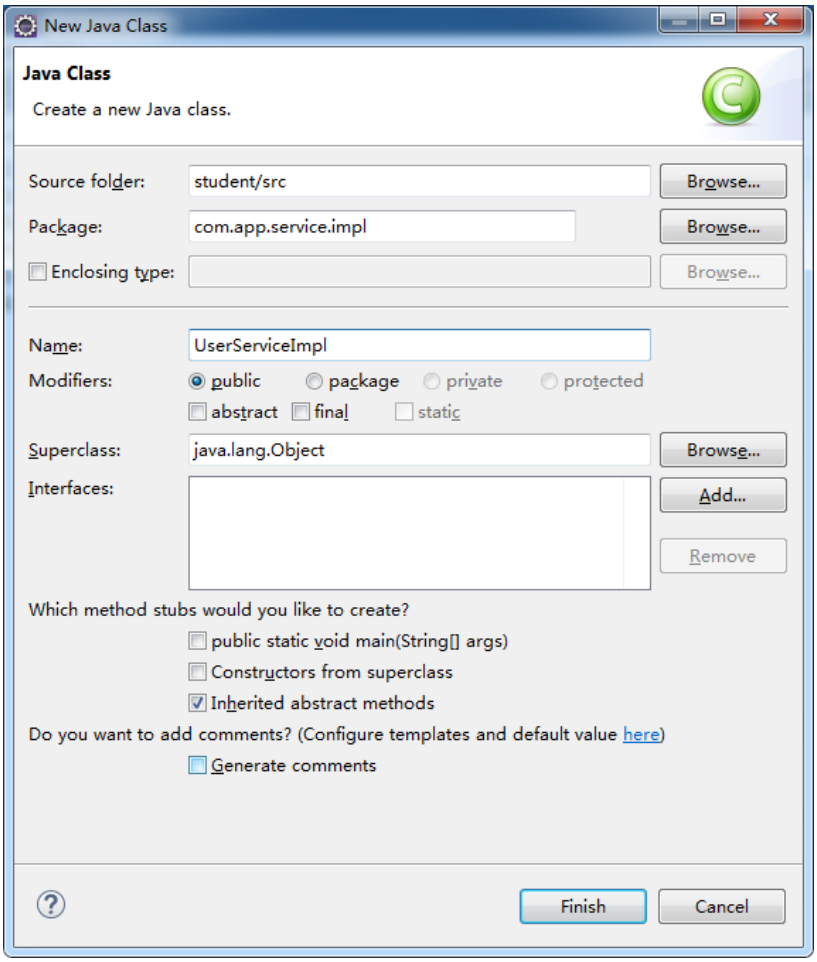


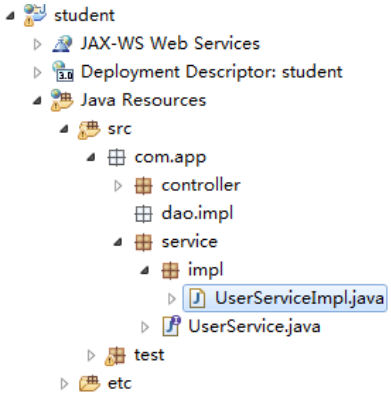
接下来，我们在service包里面新建一个UserService接口类，注意，是接口哦：





有了接口以后，就得有对应的实现类，接着就在impl包下面建一个该接口的实现类，注意，是java类哦：

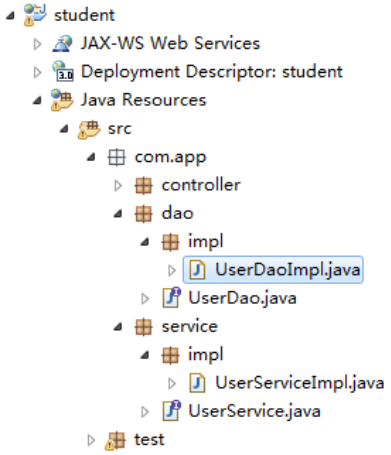




代码：

```
public class UserServiceImpl implements UserService{

}
```

同理，建立 UserDao 接口和 UserDaoImpl 类。

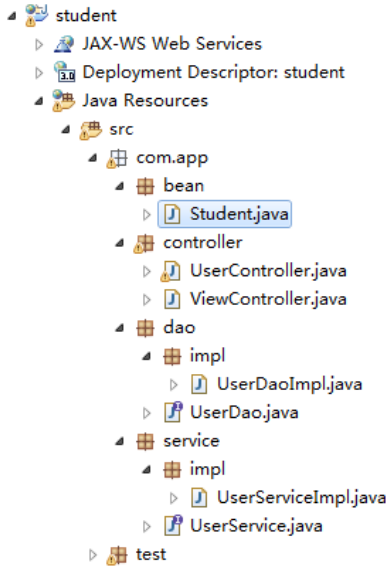


建好之后，考虑到我们的UserController中，从前台拿到了username，password，name和sex的值，思考一下如何才能把这些东西传到dao层呢？

我们可以在controller层中加入一个service层的引用：

```
@Controller
public class UserController {

    //用户业务类的引用
    private UserService userService = new UserServiceImpl();

    @RequestMapping("/addUser")
    public void addUser(HttpServletRequest request , HttpServletResponse response){
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String name = request.getParameter("name");
        String sex = request.getParameter("sex");

        System.out.println(username);
        System.out.println(password);
        System.out.println(name);
        System.out.println(sex);

    }
}
```

然后，我们需要在addUser方法里面调用service层的方法，为了避免产生过多的参数，我们先创建一个bean包，专门用来存放实体类。然后，新建一个student实体类：




```
package com.app.bean;

public class Student {

    private int id;
    private String username;
    private String password;
    private String name;
    private String sex;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
```

```
    }
    @Override
    public String toString() {
        return "Student [id=" + id + ", username=" + username + ", password=" + password + ", name=" + name + ", sex="
                + sex + "]";
    }



}
```

Spring框架支持数据动态绑定，所以我们直接这样写：

```
@Controller
public class UserController {

    //用户业务类的引用
    private UserService userService = new UserServiceImpl();

    @RequestMapping("/addUser")
    public void addUser(HttpServletRequest request , HttpServletResponse response,Student student){


        System.out.println(student);

    }
}
```


```
一月 29, 2018 3:57:02 下午 org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing WebApplicationContext for namespace 'spring-mvc': startup date [Mon Jan 29 15:57:02 CST 2018];
一月 29, 2018 3:57:02 下午 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from ServletContext resource [/WEB-INF/spring-mvc.xml]
一月 29, 2018 3:57:03 下午 org.springframework.web.servlet.handler.AbstractHandlerMethodMapping registerHandlerMeth
INFO: Mapped "{[/addUser],methods=[],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public void
一月 29, 2018 3:57:03 下午 org.springframework.web.servlet.handler.AbstractHandlerMethodMapping registerHandlerMeth
INFO: Mapped "{[/test],methods=[],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public void com
一月 29, 2018 3:57:03 下午 org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter initC
INFO: Looking for @ControllerAdvice: WebApplicationContext for namespace 'spring-mvc': startup date [Mon Jan 29
一月 29, 2018 3:57:03 下午 org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter initC
INFO: Looking for @ControllerAdvice: WebApplicationContext for namespace 'spring-mvc': startup date [Mon Jan 29
一月 29, 2018 3:57:03 下午 org.springframework.web.servlet.FrameworkServlet initServletBean
INFO: FrameworkServlet 'springmvc': initialization completed in 1214 ms
Student [id=0, username=aaa, password=123, name=啊啊啊, sex=男]
```

可见，我们使用Spring来做数据绑定是非常easy的。

然后，调用service层的方法，把这个学生数据传递到service中去。

UserController：

```
@Controller
public class UserController {

    //用户业务类的引用
    private UserService userService = new UserServiceImpl();

    @RequestMapping("/addUser")
    public void addUser(HttpServletRequest request , HttpServletResponse response,Student student){


        userService.addUser(student);


    }
}
```

UserService：

```
public interface UserService {

    void addUser(Student student);

}
```

UserServiceImpl：

```
public class UserServiceImpl implements UserService{

    @Override
    public void addUser(Student student) {


    }

}
```

同理，在service层中又需要加入一个dao层的引用，最终希望把student对象传递到dao层：

UserServiceImpl：

```
public class UserServiceImpl implements UserService{

    private UserDao userDao = new UserDaoImpl();

    @Override
    public void addUser(Student student) {

        userDao.addUser(student);

    }

}
```

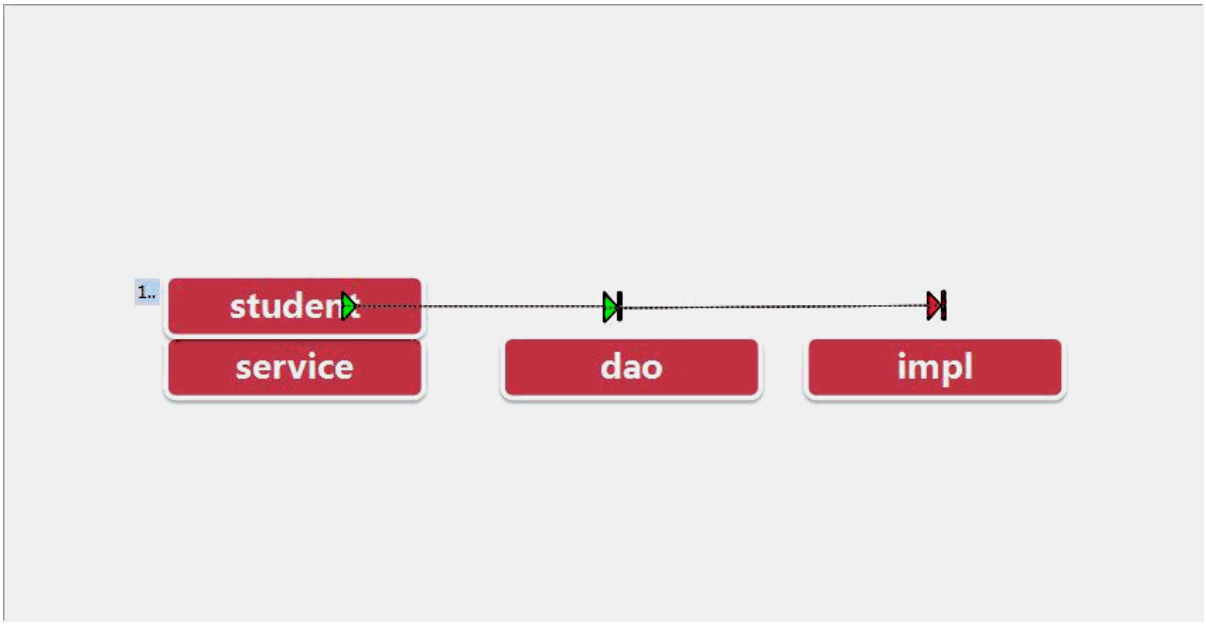
UserDao：

```
public interface UserDao {

    void addUser(Student student);

}
```

UserDaoImpl：

```
public class UserDaoImpl implements UserDao {

    @Override
    public void addUser(Student student) {

    }

}
```

图解：



以上就是整个的分层结构，然后，我们继续学生新增的业务流程，改写一下dao层的方法：

```
package com.app.dao.impl;

import java.util.HashMap;
```

```
import java.util.Map;

import com.app.bean.Student;
import com.app.dao.UserDao;
import com.simple.dao.SimpleDao;

public class UserDaoImpl implements UserDao {

    @Override
    public void addUser(Student student) {

        SimpleDao dao = new SimpleDao();

        Map map = new HashMap();

        map.put("id", null);
        map.put("username", student.getUsername());
        map.put("password", student.getPassword());
        map.put("name", student.getName());
        map.put("sex", student.getSex());

        dao.save("db_student", "t_student", map);

    }

}
```

因为只是一个简单的保存操作，我们直接调用simpleDao的save即可。测试过后，是完全没问题的。

[我要下载源码]