

1. Spring MVC 框架逻辑

M-Model, POJO 的包装类, 便于 POJO 在前后端之间传递。POJO, plain and old java object, 就是系统处理的数据的面向对象表示 ;

V-View, JSP 或 HTML 页面, 前端把接收到的 Model 通过页面组织语言 HTML 渲染成一个视图 ;

C-Control, 处理请求, 定义了控制页面跳转的逻辑, 相当于传统 web 项目当中处理请求的自定义 servlet。

请求进入 DispatcherServlet, 并被其转发到相应的 controller 上, 调用注册的 handler()。Handler 处理完业务逻辑后返回 model, 再由 View 将 model 渲染成页面。

通过依赖注入, 在 MVC 框架中应用到的 java beans 能有较高的灵活度, beans 之间的组装关系不需要在定义类时显示地用 new 来添加, 而是通过 xml 配置或者注解配置的方式从外部添加。这种做法使得组件之间的耦合度降低, 同一个组件可以被多个 bean 同时引用, 还能在配置文件中定义引用的优先级、对某些 bean 的可视度等其他属性。需要重构系统时, 只需要修改配置文件, 而不用修改代码, 一定程度上减少了“改了这个 bug 又出现新的 bug”的风险。依赖注入的配置可以通过 constructor 注入, 或者通过 setter 方法注入。

另一方面, 依赖注入也支持了 Spring 的 AOP, Aspect Oriented Programming。系统中往往有需要多次重用的模块, 但如果每次使用时都需要显示地通过相同的代码来调用并不方便, 也会产生很多重复代码。面向切面的编程方式能解决这个问题。通过外部配置, 把程序执行过程中的某些时刻定义为切点, 并在此处插入相应的调用方法, 就能在不修改代码的情况下, 增加或改变程序的行为。

2. 我做的一个 Spring MVC 小实验 -----MySpringMVCDemo

大概地扫了一遍《Spring 实战》这本书之后, 了解了依赖注入和面向切面编程的概念, 但对于实际编程还是无从下手。再查阅了网上相关的实践 demo 之后, 我动手写了一个用户登录/注册的小应用。亲身体会了 MVC 全栈之后的感受是, 1) MVC 全栈确实结构清晰, 2) 主要靠理解环境配置, 调好 xml 是关键 = =

项目结构如下 :

Model- User 类, 包装了用户的用户名和密码 ;

View- 主页面 index.jsp, 登录后页面 goodLogin.jsp, 注册后页面 goodRegister.jsp;

Controller- 响应登录请求的 loginController, 响应注册请求的 registerController;

Service- 数据库操作的接口

Dao- 数据库操作实现 (考虑到实验规模过于小, 直接使用 HashMap 在内存中存储数据)

web.xml- 项目主配置文件, 配置了名字叫 “default” 的 DispatcherServlet。

default-servlet.xml- Spring 项目的上下文配置文件, 定义了 beans 和 bean 之间的包含关系。该文件中一部分 beans 的装配也可以通过注解的方式完成。

在 form 定义时的 action 路径与 controller 中的@RequestMapping 路径一致：

```
<script type="text/javascript">
  //userRegister: addUser
  function addUser() {
    var form = document.forms[0];
    form.action = "${pageContext.request.contextPath}/default/userRegister";
    form.method = "get";
    form.submit();
  }
  //userLogin
  function userLogin() {
    var form = document.forms[0];
    form.action = "${pageContext.request.contextPath}/default/userLogin";
    form.method = "get";
    form.submit();
  }
</script>
```

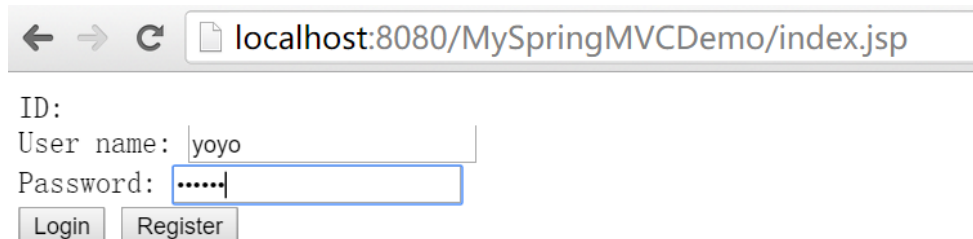
像这样，以 “/default/userLogin” 结尾的请求就会被 dispatcherServlet 分配到 loginController 中处理：

```
@Controller
public class loginController {
    @Autowired
    private IUserService userService;

    @RequestMapping(value = "/default/userLogin", method = RequestMethod.GET)
    public String doLogin(String userName, String password, ModelMap model) {
```

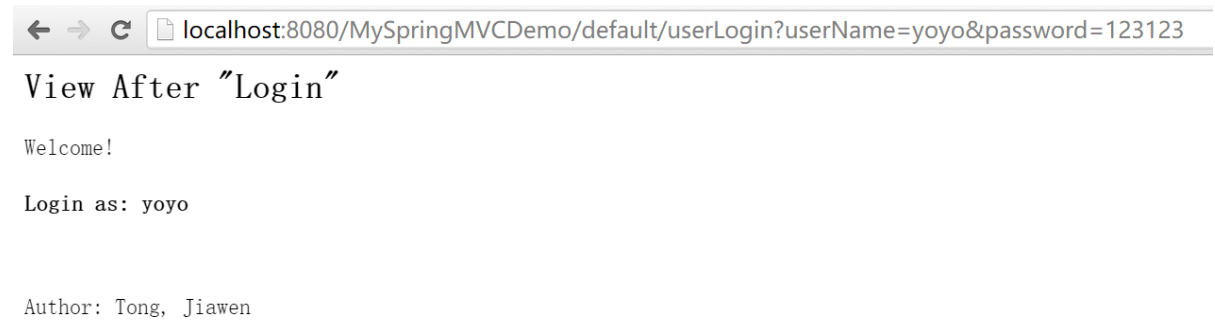
简陋的页面如下：

1) 主页面：



2) 注册后页面：

3) 登陆后页面：



3. Spring MVC 与 Spring-Struts-Hibernate 框架对比

我另外学习了一个 SSH 完成的用户登录/注册 Demo，感觉 SSH 三者的结合和 Spring 一样，提供了一个 MVC 框架，比较清楚地划分了前端控制器处理请求，后端相应，服务层访问数据库，再由控制器返回视图见面。Model 仍然是对 POJO 的封装，View 仍然是界面渲染，差别在于 Controller 的逻辑由 Struts 实现。在 xml 配置文件中定义好 action bean 的配置（相当于 Spring 的 requestMapping），由相应的 action 类继承 Struts 的 ActionSupport 类来实现控制器逻辑（相当于 Spring 的 controller 类）。

applicationContext.xml(下左) mapping: action 类---action bean

struts.xml(下右) mapping: 请求 url---action bean

```
<bean id="userLoginAction" class="action.UserLogin"
      scope="prototype">
  <property name="userService" ref="UserService"></property>
</bean>

<bean id="userRegisterAction" class="action.UserRegister"
      scope="prototype">
  <property name="userService" ref="UserService"></property>
</bean>

<action name="userLogin" class="userLoginAction">
  <result name="success" type="json">
    <param name="includeProperties">
      error_type,error_message,user.*
    </param>
  </result>
</action>

<action name="userRegister" class="userRegisterAction">
  <result name="success" type="json">
    <param name="includeProperties">
      error_type,error_message,user.*
    </param>
  </result>
</action>
```

再实现相应的 action 类就可以了，比如像这样：

- ▼ Java Resources
 - ▼ src
 - ▼ action
 - > UserLogin.java
 - > UserRegister.java
 - > WarnReminderOutput.java
 - > bean
 - > dao
 - > exception
 - > service
 - > service.impl
 - applicationContext.xml
 - hibernate.cfg.xml
 - log4j.properties
 - struts.xml

完成映射的类会从 `execute()` 入口开始执行：

```
public class UserLogin extends ActionSupport{
    private static final long serialVersionUID = 1L;

    private String userName;
    private String password;

    private User user;
    private IUserService userService;

    private int error_type = 100;
    private String error_message = "success";

    @SuppressWarnings({ "unchecked", "rawtypes" })
    public String execute() {
        Set params = new HashSet();
        params.add(userName);
        params.add(password);
        //要求params中内容不为空
        ValidateService.ValidateNecessaryArguments(params);

        User u = new User();
```

```

public class UserRegister extends ActionSupport{

    private String userName;
    private String password;
    private int gender;

    private User user;
    private IUserService userService;

    private int error_type = 100;
    private String error_message = "success";

    @SuppressWarnings({ "unchecked", "rawtypes" })
    public String execute() {
        Set params = new HashSet();
        params.add(userName);
        //params.add(password);
        //要求params中内容不为空
        ValidateService.ValidateNecessaryArguments(params);

        User u = new User();
        u.setUserName(userName);
        u.setPassword(password);
    }
}

```

另外一点是，SSH 在数据存储-H-方面使用的是 Hibernate 框架，Hibernate 框架有一套存储、查找、更新数据库等一系列模板操作。举个例子，Hibernate 把 User 的各个属性封装成一个用 User.hbm.xml 表示的对象，便于应用程序的访问。具体实现时，在 service 层包含一个 Dao 对象，service 层定义了使用 Dao 对象对数据的业务逻辑实现。

4. 一点点感想和说明

代码部分只提交了 SpringMVC 的实践，一个名字叫 MySpringMVCDemo 的包。

其实 MVC 框架挺好学的，重点是搞清楚配置是怎么完成的，依赖注入是在哪里出现的，是通过配置文件还是注解，bean 之间的关系，bean 和 servlet 之间的关系，request 和 handler 之间的映射，大概就差不多了。

PS：调 xml 和拖 lib 搞了一天；404 解决之后的其他 bug 倒是半天都不用 23333