Eclipse手动搭建SSM框架

项目Github地址：<https://github.com/pifutan/PifutanSSM>

**总结：**

经常搭框架的人应该都清楚，框架搭建的核心就是配置文件。

### 环境：

Windows10+JDK1.8+Tomcat8+Eclipse Oxygen JavaEE+MySQL57

**Spring框架（反转控制IOC也称依赖注入DI，AOP）**

* 概念区分：Spring 是一个“引擎” ，Spring MVC 是基于 Spring 的一个 MVC 框架，Spring Boot 是基于 Spring4 的条件注册的一套快速开发整合包。Spring 最初利用“工厂模式”（ DI ）和“代理模式”（ AOP ）解耦应用组件。大家觉得挺好用，于是按照这种模式搞了一个 MVC 框架（一些用 Spring 解耦的组件），用开发 web 应用（ SpringMVC ）。然后有发现每次开发都要搞很多依赖，写很多样板代码很麻烦，于是搞了一些懒人整合包（ starter ），这套就是 Spring Boot。Spring Boot extends SpringMVC extends Spring。
* 下载：spring.io中的Spring FrameWork便是spring，官网有说怎么下载：<http://repo.spring.io/release/org/springframework/spring/> 下载-dist.zip即可。
* 教程：<http://www.yiibai.com/spring/>
* 需要的包：

commons-logging-1.2.jar（下载：http://commons.apache.org/proper/commons-logging/index.html）

spring-aop-4.3.9.RELEASE.jar

spring-aspects-4.3.9.RELEASE.jar

spring-beans-4.3.9.RELEASE.jar

spring-context-4.3.9.RELEASE.jar

spring-context-support-4.3.9.RELEASE.jar

spring-core-4.3.9.RELEASE.jar

spring-expression-4.3.9.RELEASE.jar

spring-instrument-4.3.9.RELEASE.jar

spring-instrument-tomcat-4.3.9.RELEASE.jar

spring-jdbc-4.3.9.RELEASE.jar

spring-jms-4.3.9.RELEASE.jar

spring-orm-4.3.9.RELEASE.jar

spring-oxm-4.3.9.RELEASE.jar

spring-test-4.3.9.RELEASE.jar

spring-tx-4.3.9.RELEASE.jar

spring-web-4.3.9.RELEASE.jar

spring-webmvc-4.3.9.RELEASE.jar

spring-webmvc-portlet-4.3.9.RELEASE.jar

spring-websocket-4.3.9.RELEASE.jar

* Spring的控制反转实际上就是Class用配置成bean的方式来管理，否则每次我们都得自己写清楚，改动时很麻烦且容易出错还耗时。
* 声明为bean的Class可以有两种方式，一种是在applicationContext.xml中配置：<bean id="helloWorld" class="com.pifutan.spring.HelloWorld"></bean>，另一种是使用组件自动扫描。
* 组件自动扫描：使用@Component需要配置applicationContext.xml：

<!-- 启用自动组件扫描 -->

<context:component-scan base-package="com.pifutan.bean" />

实际上有四种方式：

@Component – 指示自动扫描组件。

@Repository – 表示在持久层DAO组件。

@Service – 表示在业务层服务组件。

@Controller – 表示在表示层控制器组件。

但是这四种方式实际上是一样的，且都是开启component-scan就可以了，只是为了方便阅读弄出了四种而已。

* 过滤器组件自动扫描，即不使用如@Component就可以用了。

可以使用包含和不包含两种过滤。

<context:component-scan base-package="com.pifutan" >

<context:include-filter type="regex"

expression="com.pifutan.dao.\*DAO.\*" />

<context:include-filter type="regex"

expression="com.pifutan.service.\*Service.\*" />

<context:exclude-filter type="regex"

expression="com.pifutan.service.\*Service.\*" />

</context:component-scan>

* List，Set，Map，Properties和Date注入，都有相应支持和使用。
* 在Spring中，bean作用域用于确定哪种类型的 bean 实例应该从Spring容器中如何返回给调用者。bean支持的5种范围域scope：

单例singleton - 每个Spring IoC 容器返回同一个bean实例

原型prototype - 当每次请求时返回一个新的bean实例

请求request - 返回每个HTTP请求的一个Bean实例

会话session - 返回每个HTTP会话的一个bean实例

全局会话global session - 返回全局HTTP会话的一个bean实例

在大多数情况下，可能只处理了 Spring 的核心作用域 - 单例和原型，默认作用域是单例。

注：意味着只有在一个基于web的Spring ApplicationContext情形下有效！

* 如果需要同一个类的两个单例，则可以声明不同的两个beanid，使用@Qualifier("personA")自动装配。
* 声明IUser接口然后User和Admin都实现IUser接口，并且User和Admin都在xml中配置bean或者在类@Component(“beanid”)，使用的时候声明IUser user则会匹配User，声明IUser admin则会匹配Admin，如果找不到beanid且接口只有一个实现时能匹配，如果有多个实现则会报错多个匹配。如果是使用过滤器自动扫描组件，则类名和变量名对应即可（大小写无区别）。

当然IUser需要加@Autowired自动装配，[使用@Autowired需要在applicationContext.xml](mailto:使用@Autowired需要在applicationContext.xml)中添加<context:annotation-config />

* 配置bean使用new方式实例化是无法填充内部的类变量的。需要：

ApplicationContext context = **new** ClassPathXmlApplicationContext( "applicationContext.xml");(User) context2.getBean("user");

* Spring EL表达式，就是可以定义按一定条件和形式注入值。用到的时候再详细学吧。
* @Required以及@Mandatory 定制 @Required-style 注解，相当于@Required注解。
* Spring提供了JDBC支持，提供了JdbcTemplate和JdbcDaoSupport以及SimpleJdbcTemplate。
* Spring AOP：Advice（指示之前或方法执行后采取的行动），Pointcut（指明哪些方法应该拦截，通过方法的名称或正则表达式模式），Advisor（分组"通知"和”切入点“成为一个单元，并把它传递到代理工厂对象）。
* Spring AOP+ AspectJ：注意Spring AOP 中没有 AspectJ 支持

常见AspectJ的注解：

@Before – 方法执行前运行

@After – 运行在方法返回结果后

@AfterReturning – 运行在方法返回一个结果后，在拦截器返回结果。

@AfterThrowing – 运行方法在抛出异常后，

@Around – 围绕方法执行运行，结合以上这三个通知。

### SpringMVC框架（Controller以及很多的注解）

* SpringMVC实际上就包含在SpringFramework中。
* 下载地址：<http://repo.spring.io/release/org/springframework/spring/>

教程参考（不是很好，所以还是看官网文档吧，有了SSH的基础的话，SSM就相对容易一些了）：<http://www.yiibai.com/spring_mvc/>

* 官网文档：<https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>
* 需要的包：

commons-logging-1.2.jar

spring-aop-4.3.9.RELEASE.jar

spring-beans-4.3.9.RELEASE.jar

spring-context-4.3.9.RELEASE.jar

spring-core-4.3.9.RELEASE.jar

spring-expression-4.3.9.RELEASE.jar

spring-tx-4.3.9.RELEASE.jar

spring-web-4.3.9.RELEASE.jar

spring-webmvc-4.3.9.RELEASE.jar

spring-websocket-4.3.9.RELEASE.jar

* 配置web.xml，添加并配置applicationContext.xml，且与web.xml中的路径对应。如果使用诸如mvc:resource的标签，记得要引入mvc和tx解析。
* 映射静态页面：<mvc:resources location="/\*.html" mapping="/\*\*.html" />
* Controller可以在\*context.xml中配置bean或者使用组件自动扫描且使用注解@Controller。
* 组件自动扫描：<context:component-scan base-package="com.pifutan.springmvc" />

使用注解需要：<mvc:annotation-driven />

* 请求：使用注解@RequestMapping映射请求，快捷方式有：@GetMapping，@PostMapping，@PutMapping，@DeleteMapping，@PatchMapping。以上可以组合使用。

URL路径支持glob模式和通配符映射请求（?：匹配一个字符，\*：匹配路径段中的零个或多个字符，\*\*：匹配零个或多个路径段）

或者声明URI变量并访问其值@PathVariable如：

@GetMapping("/owners/{ownerId}/pets/{petId}")

public Pet findPet(@PathVariable Long ownerId, @PathVariable Long petId) { }

或者在类和方法级别声明如：

@Controller

@RequestMapping("/owners/{ownerId}")

public class OwnerController {

@GetMapping("/pets/{petId}")

public Pet findPet(@PathVariable Long ownerId, @PathVariable Long petId) { }

}

或者使用正则表达式如@GetMapping(“{varName:regex}”)示例：

@GetMapping("/{name:[a-z-]+}-{version:\\d\\.\\d\\.\\d}{ext:\\.[a-z]+}")

public void handle(@PathVariable String version, @PathVariable String ext) {}

指定Content-type: @PostMapping(path = "/pets", consumes = "application/json")

* 获取请求参数方法：

通过HttpServletRequest获取；

直接用变量名获取；

使用bean对应变量名获取；

通过@PathVariable获取路径中的参数；

使用@ModelAttribute注解获取（method(@ModelAttribute("variable") String variable)）；

使用@RequestParam注解获取（类似上面）；

获取cookie参数@CookieValue("JSESSIONID")

@RequestHeader("Accept-Encoding")等等

* 返回：

返回页面ModelAndView，new ModelAndView("viewName")。指定文件夹和后缀配置在：

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/" />

<property name="suffix" value=".html" />

</bean>

返回值Map、Model和ModelMap。

返回Json（需要jackson包jackson-annotations-2.9.0.jar，jackson-core-2.9.0.jar，jackson-databind-2.9.0.jar，官网<https://github.com/FasterXML/jackson>只有源码没找到jar包，虽然可以自己打包，但是也可以到<http://mvnrepository.com/search?q=jackson> 下载）：注解@ResponseBody便能将对象转化为json返回。

跨域Json：jsonp请求方式返回json与Struts中一样需要返回一个callbackParameter参数，或者response.setHeader("Access-Control-Allow-Origin", "\*");，或者服务端设置支持跨域添加注解@CrossOrigin即可，或者使用CORS全局配置，或者使用AbstractJsonpResponseBodyAdvice以及@ControllerAdvice实现，或者其他方法（还有其他方法如MappingJackson2JsonView等等，未探究）。

* SpringMVC还有很多内容，需要的再上官网查询吧，百度上的一些东西有时候让人摸不着头脑，而且还不一定有效，还是系统地了解SpringMVC使用及其大概工作原理比较好，这样才能够对症下药。

### Mybatis框架

* MyBatis 是支持普通 SQL 查询,存储过程和高级映射的优秀持久层框架。MyBatis 消除 了几乎所有的 JDBC 代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML 或注解用于配置和原始映射,将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java 对象)映射成数据库中的记录。
* MyBatis 最强大的特性之一就是它的动态语句功能。
* 下载：<https://github.com/mybatis/> 应该比较容易找到，地址：<https://github.com/mybatis/mybatis-3/releases>
* 教程：<http://www.yiibai.com/mybatis/>

官方文档：<http://www.mybatis.org/mybatis-3/zh/index.html>

* 需要的包：

mybatis-3.4.5.jar

mysql-connector-java-5.1.44-bin.jar

* 需要mybatis配置文件如mybatis.xml。表对应的实体类配置为

<typeAliases>

<typeAlias alias="User" type="com.pifutan.mybatis.User" />

</typeAliases>

使用User.xml，实体类User，对应数据库查询的Wapper为User.xml，在mybatis.xml配置mappers。

使用接口编程，在User.xml定义sql，在mybatis.xml配置mappers。

使用接口与注释来写sql，不需要User.xml，且不需要mybatis.xml配置mappers。

* MyBatis表关联一对多、多对一、多对多。
* MyBatis分页：RowBounds实现的分页是逻辑分页,也就是先把数据记录全部查询出来,然在再根据 offset 和 limit 截断记录返回。为了在数据库层面上实现物理分页，又不改变原来 MyBatis 的函数逻辑,可以编写 plugin 截获 MyBatis Executor 的 statementhandler，重写SQL来执行查询。
* 动态 SQL：在xml文件中使用if, choose, when, otherwise, trim, where, set, foreach, bind等来做test之类的，实际上在代码中也是可以做的。
* 使用log4j2: <http://logging.apache.org/log4j/2.x/articles.html>

### Spring + SpringMVC + MyBatis

* Spring集成mybatis文档：<http://www.mybatis.org/spring/zh/index.html>
* 使用SqlSessionDaoSupport和SqlSessionTemplate等需要下载包mybatis-spring.jar，官网上没找到在哪里现在，只好去<http://mvnrepository.com/artifact/org.mybatis> 下载了。
* 使用c3p0除了需要c3p0-0.9.5.2.jar还需要mchange-commons-java-0.2.11.jar，可到<https://sourceforge.net/projects/c3p0/> 下载（官网<http://www.mchange.com/projects/c3p0/> 需要翻墙）（Struts的包里面有c3p0包以及Struts相配套c3p0的实现所以不需要自己再下载了）。
* 使用log4j2的话到<http://logging.apache.org/log4j/2.x/download.html>下载即可。
* SSM集成实际上就是配置及其使用，这边有一篇参考可以看看，但是并不一定都适合，而且可能会出错需要自己修改：<http://blog.csdn.net/qq598535550/article/details/51703190>。
* 这里采用上面那篇教程的配置方法结合自己的项目进行集成。
* 用到的包有：

c3p0-0.9.5.2.jar

commons-logging-1.2.jar

jackson-annotations-2.9.0.jar

jackson-core-2.9.0.jar

jackson-databind-2.9.0.jar

log4j-api-2.9.1.jar

log4j-core-2.9.1.jar

mchange-commons-java-0.2.11.jar

mybatis-3.4.5.jar

mybatis-spring-1.3.0.jar

mysql-connector-java-5.1.44-bin.jar

spring-aop-4.3.9.RELEASE.jar

spring-aspects-4.3.9.RELEASE.jar

spring-beans-4.3.9.RELEASE.jar

spring-context-4.3.9.RELEASE.jar

spring-context-support-4.3.9.RELEASE.jar

spring-core-4.3.9.RELEASE.jar

spring-expression-4.3.9.RELEASE.jar

spring-instrument-4.3.9.RELEASE.jar

spring-instrument-tomcat-4.3.9.RELEASE.jar

spring-jdbc-4.3.9.RELEASE.jar

spring-jms-4.3.9.RELEASE.jar

spring-orm-4.3.9.RELEASE.jar

spring-oxm-4.3.9.RELEASE.jar

spring-test-4.3.9.RELEASE.jar

spring-tx-4.3.9.RELEASE.jar

spring-web-4.3.9.RELEASE.jar

spring-webmvc-4.3.9.RELEASE.jar

spring-webmvc-portlet-4.3.9.RELEASE.jar

spring-websocket-4.3.9.RELEASE.jar

* 配置文件有：

jdbc.properties

log4j.xml

mybatis-config.xml

spring-dao.xml

spring-service.xml

spring-web.xml

* jdbc.properties配置mysql相关的驱动、连接地址、账号密码等，关于连接池的可以在spring配置文件中配置。
* log4j2.xml为日志的配置，要为mybatis开启日志，可在Loggers的Root前配置一个Logger对应的name应为mybatis放接口或者说mapper的包的全名，且在mybatis-config.xml中需要使用LOG4J2即<setting name="logImpl" value="LOG4J2" />。平常时候在代码中LogManager.getLogger(LogManager.ROOT\_LOGGER\_NAME).trace()等即可。
* mybatis-config.xml主要为mybatis配置一些settings即可。
* spring-dao.xml中需要引入jdbc配置：

<context:property-placeholder location="classpath:jdbc.properties" />

配置dataSource为c3p0数据源，配置sqlSessionFactory并在其中配置数据源：

<property name="dataSource" ref="dataSource" />

指定mybatis配置文件：

<property name="configLocation" value="classpath:mybatis-config.xml" />

指定entity包名：

<property name="typeAliasesPackage" value="com.pifutan.entity" />

以及mapper或者接口等：

<property name="mapperLocations" value="com.pifutan.entity.\*.xml,com.pifutan.ientity.\*.xml,com.pifutan.dao.\*.xml" />

* spring-service.xml配置一些相关与Service的东西。
* spring-web.xml主要配置和Controller相关的东西，如组件自动扫描，启用注解，静态资源配置，拦截器配置（拦截器三个方法preHandle，postHandle，afterCompletion）。
* 另，Log4j2有8个level：All < Trace < Debug < Info < Warn < Error < Fatal < OFF。
* 引入jackson加上@ResponseBody便可以将类自动转化为json返回，可以使用@CrossOrigin支持跨域访问。
* 如果想在返回的外面套一层固定格式，可以自定义一个Result类，然后把东西放里面返回。
* 详细见github项目：<https://github.com/pifutan/PifutanSSM>。