动吧旅游项目

权限管理子系统

SSM技术整合

Spring+SpringMVC+MyBatis

(XML方式)

[1. SSM 技术整合 2](#_Toc30259)

[1.1. 环境准备 2](#_Toc258)

[1.2. 基本架构 6](#_Toc28084)

[1.2.1. 项目分层架构 6](#_Toc30828)

[1.2.2. API 应用架构 7](#_Toc15687)

[1.3. 准备工作 10](#_Toc31072)

[1.3.1. 创建并配置项目 10](#_Toc14348)

[1.3.2. 添加项目依赖 12](#_Toc10832)

[1.3.3. 创建配置文件 13](#_Toc29468)

[1.3.4. 创建测试基类 15](#_Toc31392)

[1.4. 资源整合 15](#_Toc15439)

[1.4.1. 整合DataSource 对象 16](#_Toc13503)

[1.4.2. 整合Mybatis框架 18](#_Toc103)

[1.4.3. 整合Spring MVC模块 20](#_Toc30257)

[1.4.4. 整合Jackson 实现 22](#_Toc25471)

[1.5. 首页初始化 22](#_Toc25794)

[1.5.1. 定义页面初始资源 22](#_Toc9699)

[1.5.2. 创建页面Controller 23](#_Toc29304)

[1.5.3. 启动项目进行测试 23](#_Toc190)

[2. SSM 整合总结 24](#_Toc19975)

[2.1. 重点难点分析 24](#_Toc16717)

[2.2. 问题分析 24](#_Toc19398)

[2.3. Bug分析 28](#_Toc24172)

[2.4. 原理分析 32](#_Toc13180)

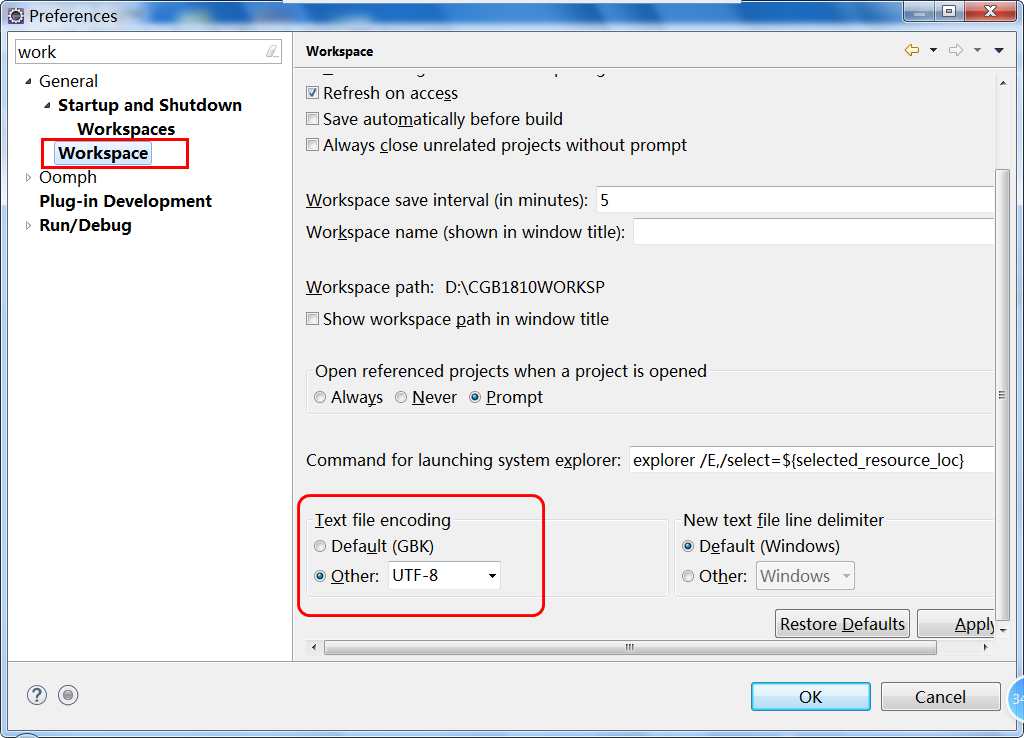
# SSM 技术整合

## 环境准备

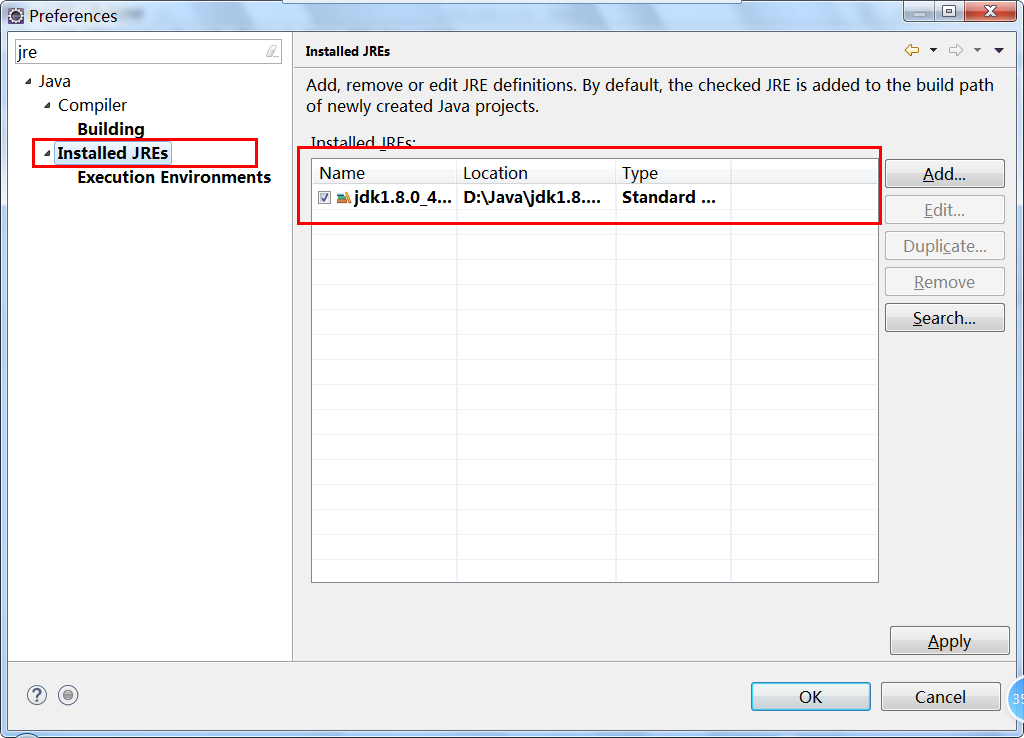
1. Eclipse 工作区设置
2. 统一工作区编码(UTF-8)
3. 统一JDK版本(JDK1.8)
4. 统一TOMCAT版本(Tomcat8)
5. 统一MAVEN配置(3.5.3)

图文演示：

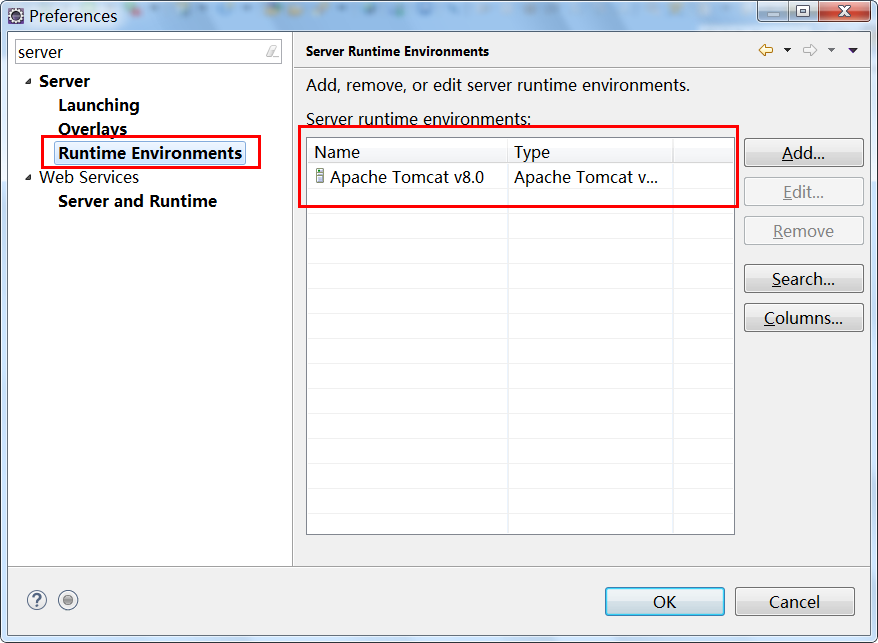
1. 统一工作区编码



1. 统一工作区编码(UTF-8)



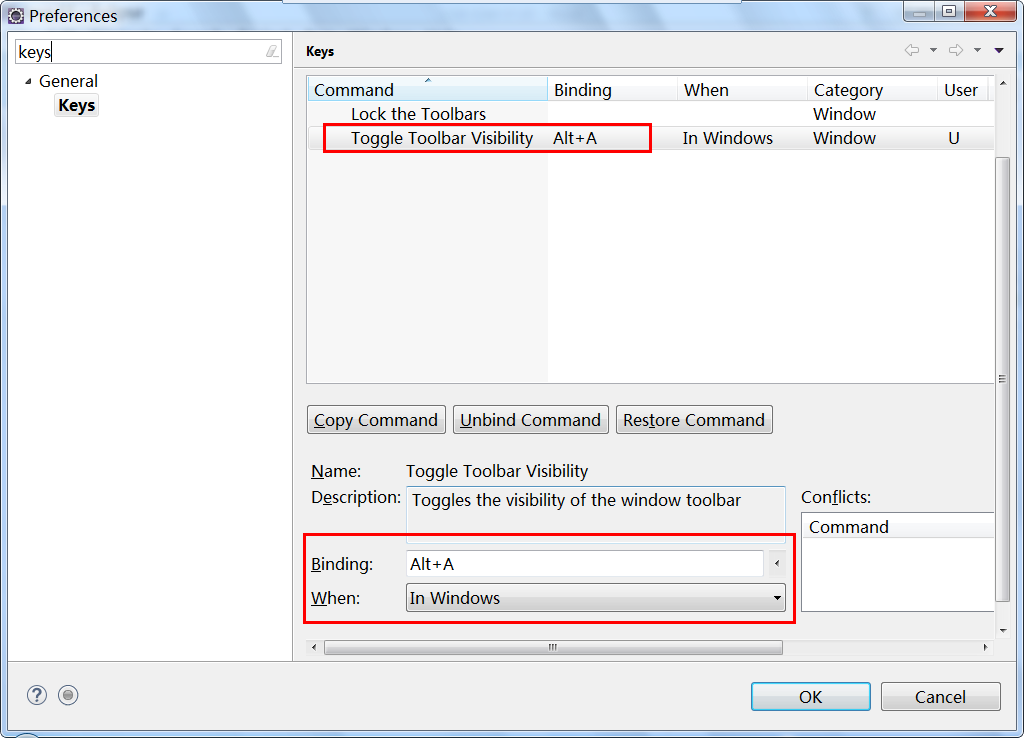
1. 统一TOMCAT版本(Tomcat8)



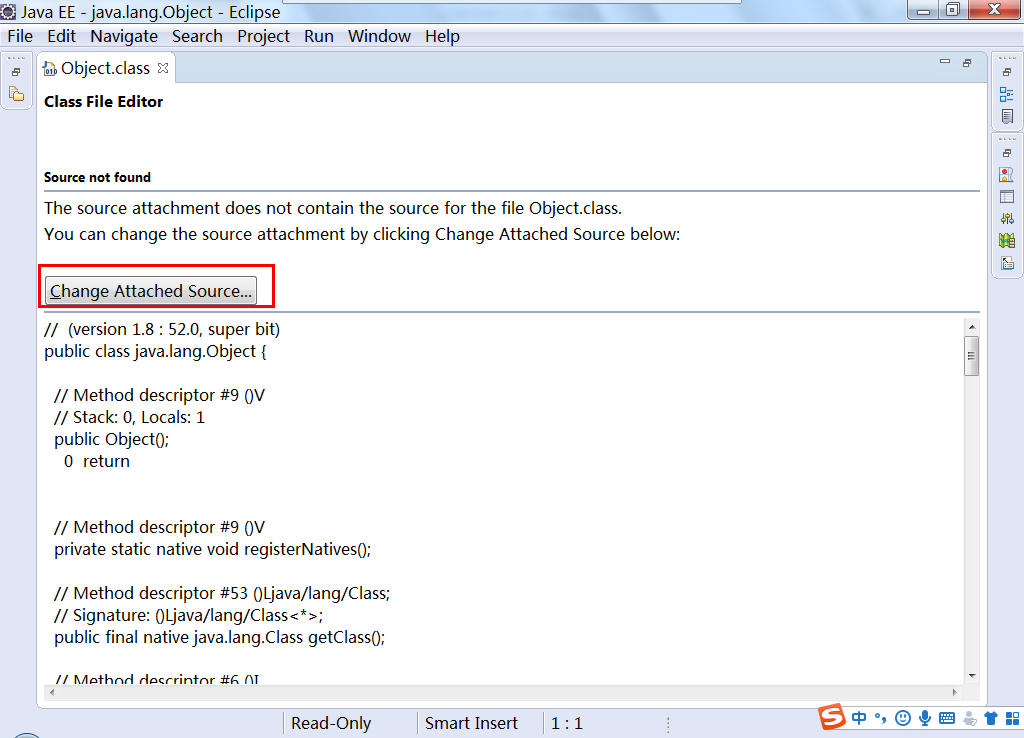
1. Eclipse 快捷键配置
2. 隐藏工具栏(window/preferences/kes/toolbar )
3. 查找类或接口(ctrl+shift+t)
4. 查看本方法（ctrl+o/ctrl+o）
5. 查看类的继承体系(ctrl+t)
6. 快速引入包中类:ctrl+shift+o

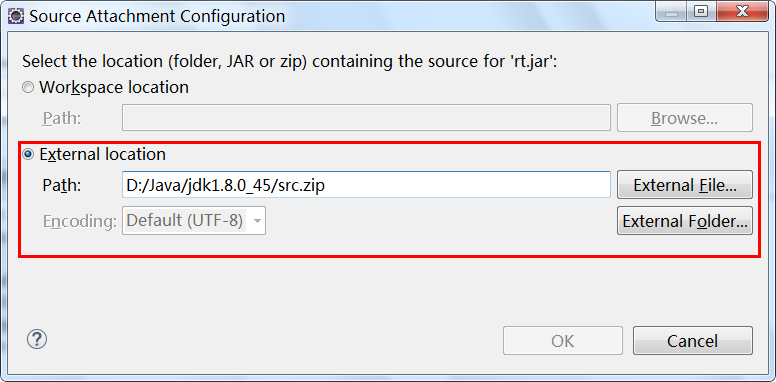
图文演示：

1. 隐藏工具栏(window/preferences/kes/toolbar )



1. 关联jdk源码





5.项目描述：

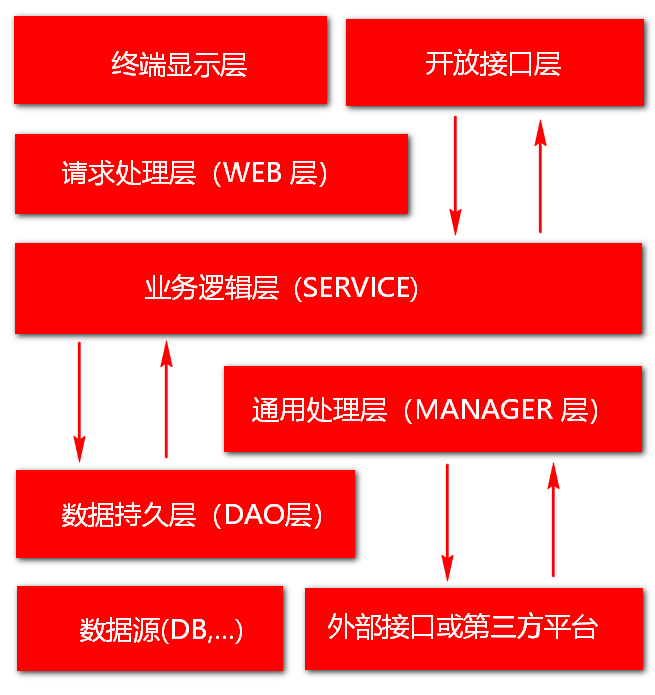
动吧旅游生态系统：应市场高端用户需求，公司决定开发这样的一套旅游系统，

此系统包含旅游电商系统（广告子系统，推荐子系统，评价子系统，商品子系统，订单子系统，…），旅游分销系统(分销商的管理)，旅游业务系统(产品研发，计调服务，系统管理，..)，，。。。

## 基本架构

### 项目分层架构

本项目基于MVC设计思想，进行分层架构设计，其目的主要基于系统“高内聚低耦合”的设计思想，提高系统的可维护性，可扩展性。



其中：

1.开放接口层：可直接封装 Service 方法暴露成 RPC (远程过程调用)接口；也可通过 Web 封装成 http 接口；同时也可进行网关安全控制、流量控制等。

2.终端显示层：负责各个端的模板渲染并显示。当前主要是 velocity 渲染，JS 渲染， JSP 渲染，移动端展示等。

3.Web请求处理层：主要是对访问控制进行转发，请求参数校验，响应结果处理等

4.Service 层：相对具体的业务逻辑服务层（核心业务，扩展业务）。

5.Manager 层：通用业务处理层，它有如下特征：

1） 对第三方平台封装的层，预处理返回结果及转化异常信息；

2） 对 Service 层通用能力的下沉，如缓存方案、中间件通用处理；

3） 与 DAO 层交互，对多个 DAO 的组合复用。

6.DAO 层：数据访问层，与底层 MySQL、Oracle、Hbase 等进行数据交互。

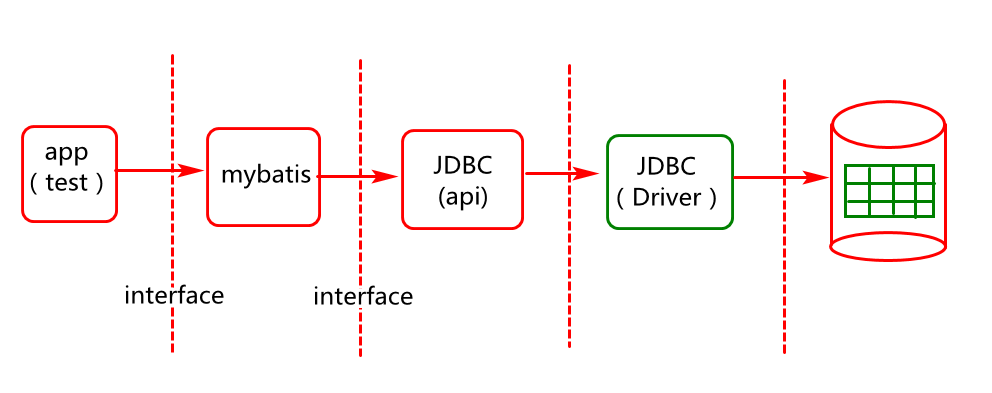
7.外部接口或第三方平台：包括其它部门 RPC 开放接口，基础平台，其它公司的 HTTP 接口

说明：对如上分层中涉及到知识的点，逐步加强。

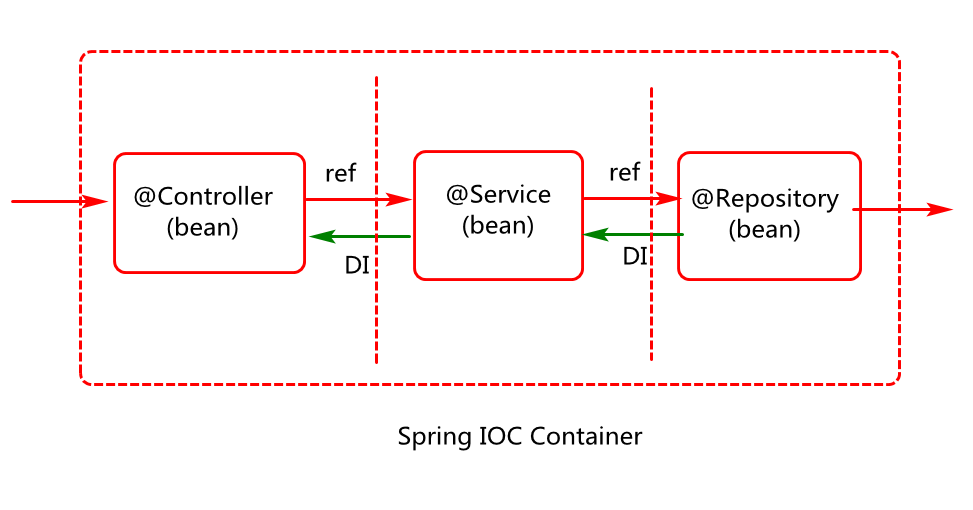
总之：分层的目的就是将复杂问题进行拆解，然后分而治，进而提高系统的可扩展性以及可维护性。

### API 应用架构

知识点回顾：mybatis 的应用架构

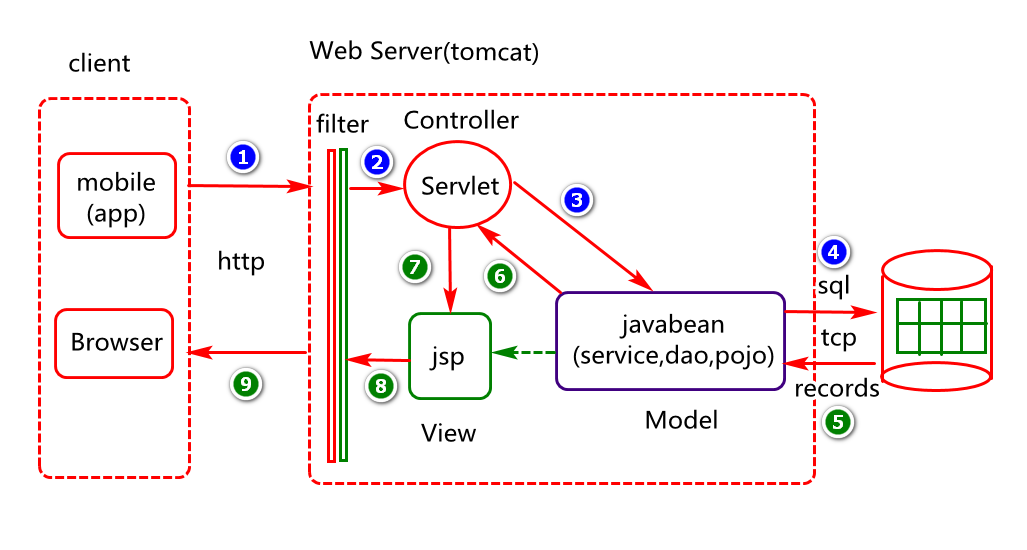


知识点回顾：spring 中的IOC

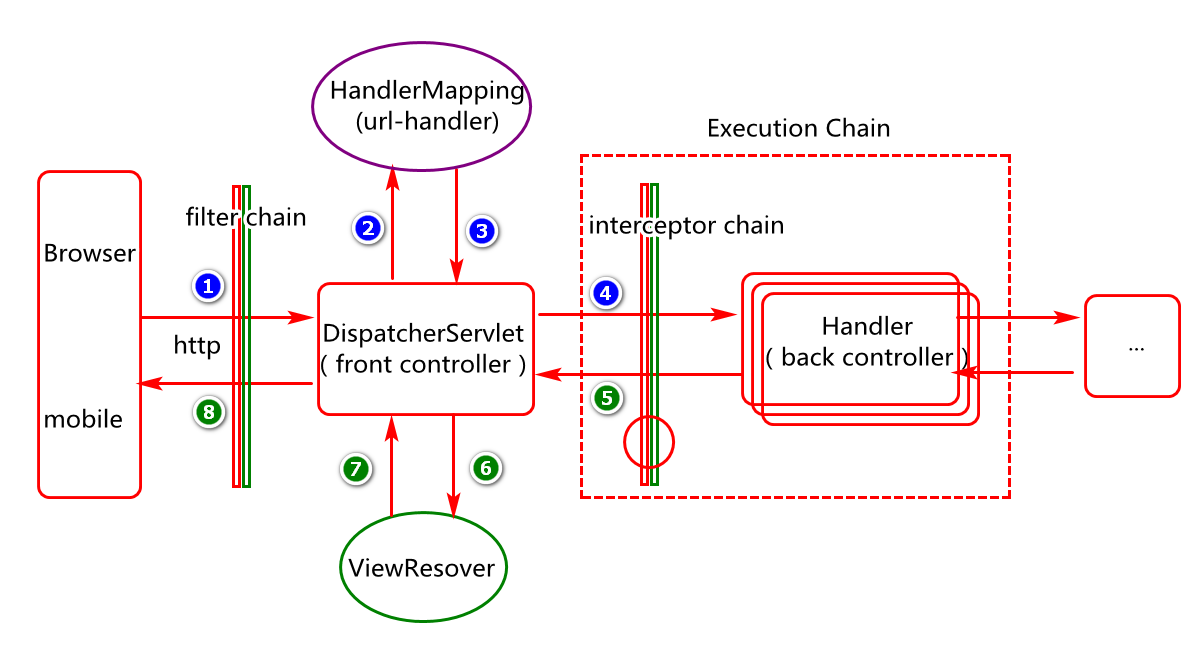


知识点回顾：MVC 设计思想

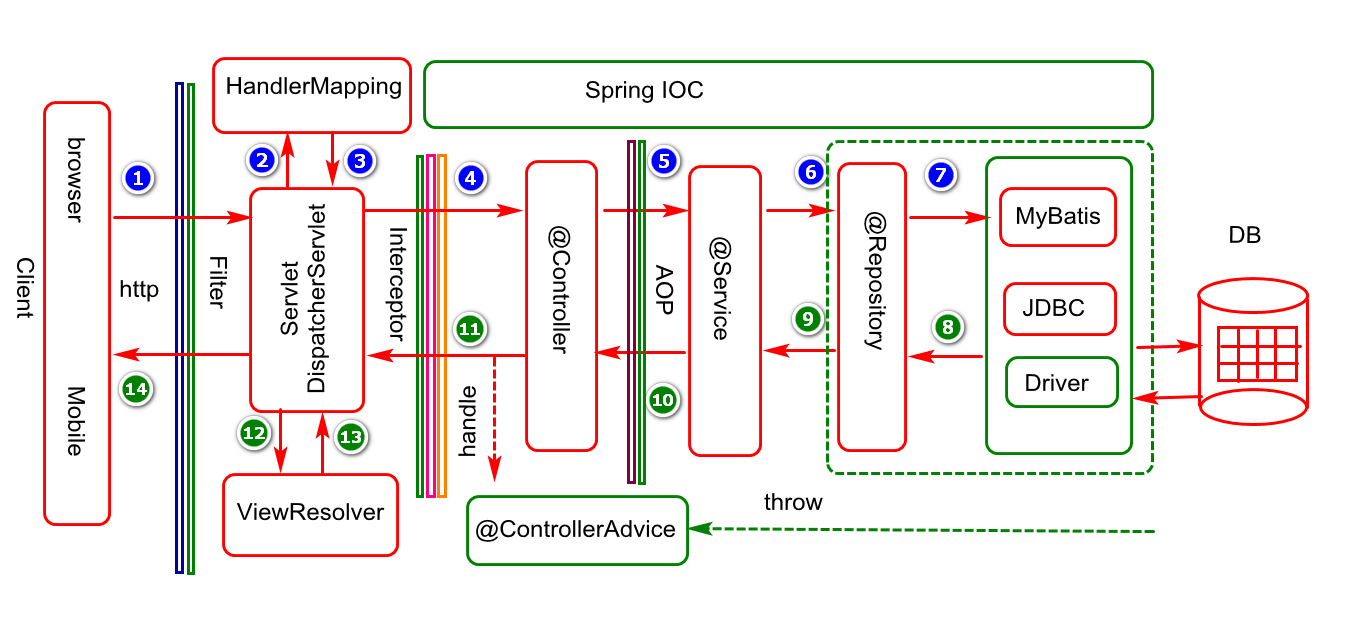
基于servlet技术的MVC实现



Spring 中MVC实现



本次项目的总体业务架构：



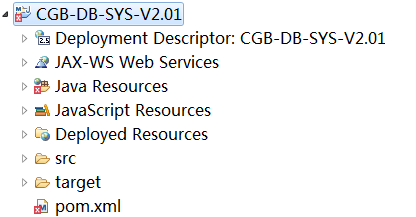
## 准备工作

基本步骤

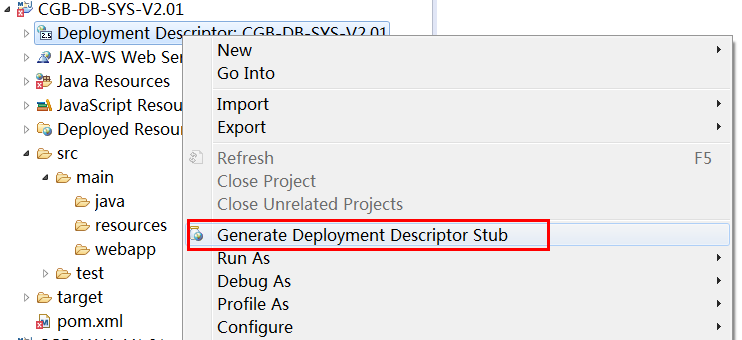
1. 创建maven web项目（打包方式war包）
2. group id：com.db
3. 项目名：CGB-DB-SYS-V3.01
4. 生成部署描述文件 web.xml(全注解方式不需要)
5. 修改项目编码为utf-8（假如工作区编码为UTF-8,则无需修改）
6. 设置项目targeted runtimes (Tomcat)：暂时不用tomcat插件
7. 修改项目project facets (jdk1.8)

项目问题分析：

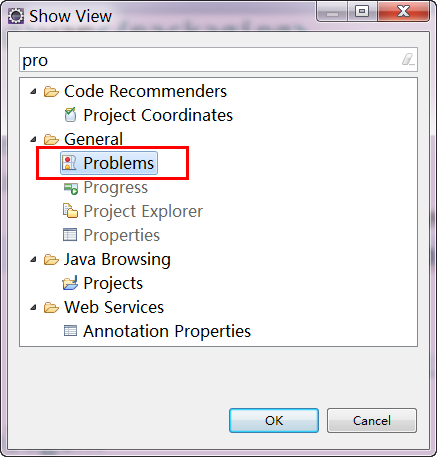
* + - 1. 项目创建完成没有web.xml?(生成web.xml)



生成web.xml （在项目(Project)视图中执行此操作）



* + - 1. 项目中有错但不知什么错？（看problems）

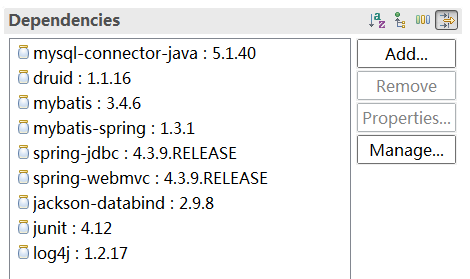




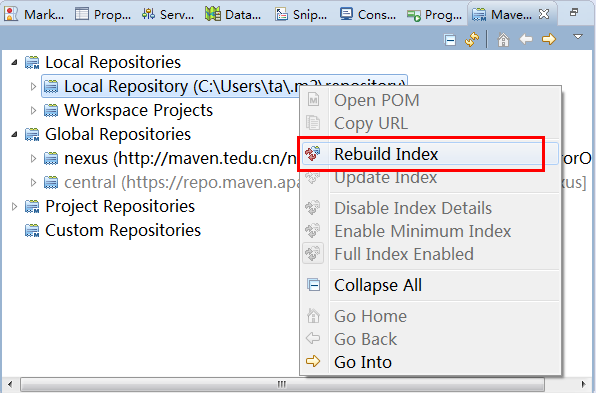
### 添加项目依赖

1. 添加Druid 连接池(两个依赖:mysql驱动，druid库文件)
2. 添加Mybatis 框架(三个依赖:mybatis,mybatis-spring,spring-jdbc)
3. 添加Spring MVC 模块环境(一个依赖spring-webmvc)
4. 添加 jackjson库（用于将对象转换为json串）
5. 添加Junit 单元测试依赖

相关版本如下：



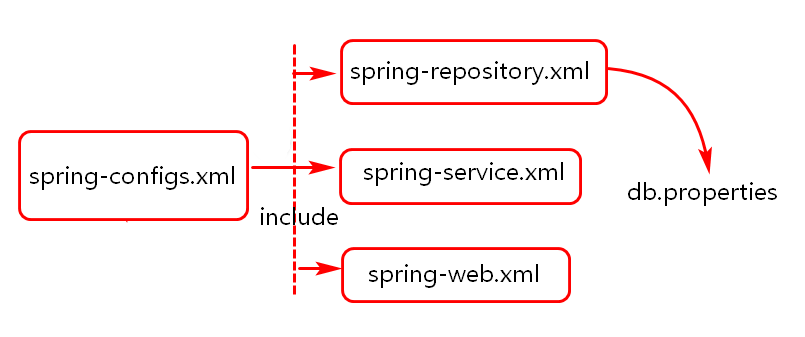
说明：假如依赖是从本地库添加可以在本地库创建索引：(window/show view/)



假如本地库没有这些依赖,可以直接从maven.tedu.cn 这个地址去搜索.

### 创建配置文件

配置文件结构如下：所在目录(src/main/resources)



业务说明：

基于配置文件结构创建配置文件，放在src/main/resources/目录下：

Xml配置文件的模板如下：

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

<beans default-lazy-init=*"true"*

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:util=*"http://www.springframework.org/schema/util"*

xmlns:jpa=*"http://www.springframework.org/schema/data/jpa"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-4.3.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-4.3.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-4.3.xsd*

*http://www.springframework.org/schema/util*

*http://www.springframework.org/schema/util/spring-util-4.3.xsd*

*http://www.springframework.org/schema/data/jpa*

*http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-4.3.xsd"*>

</beans>

其中:

db.properties文件初始内容如下：

jdbcDriver=com.mysql.jdbc.Driver

jdbcUrl=jdbc:mysql:///jtsys?useUnicode=true&characterEncoding=utf-8

jdbcUser=root

jdbcPassword=root

Spring-repository.xml文件中初始内容

<util:properties id=*"cfg"* location=*"classpath:db.properties"*/>

spring-service.xml文件初始中内容

<context:component-scan base-package=*"com.db.sys.service"*/>

spring-web.xml文件中初始内容

<context:component-scan base-package=*"com.db.sys.controller"*/>

spring-configs.xml文件中内容

<import resource=*"classpath:spring-repository.xml"*/>

<import resource=*"classpath:spring-service.xml"*/>

<import resource=*"classpath:spring-web.xml"*/>

在src/main/resources目录下创建log4j.properties文件，其内容如下

log4j.rootLogger=INFO,stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d [%-5p] %c - %m%n

log4j.logger.com.db=DEBUG

### 创建测试基类

业务说明：

定义测试基类，初始化spring相关资源。

代码实现：

**package** com.test;

**public** **class** TestBase {

**protected** ClassPathXmlApplicationContext ctx;

@Before

**public** **void** init() {

ctx=**new** ClassPathXmlApplicationContext(“spring-configs.xml”);

}

@Test

**public** **void** testCtx() {

System.***out***.println(ctx);

}

@After

**public** **void** destory() {

ctx.close();

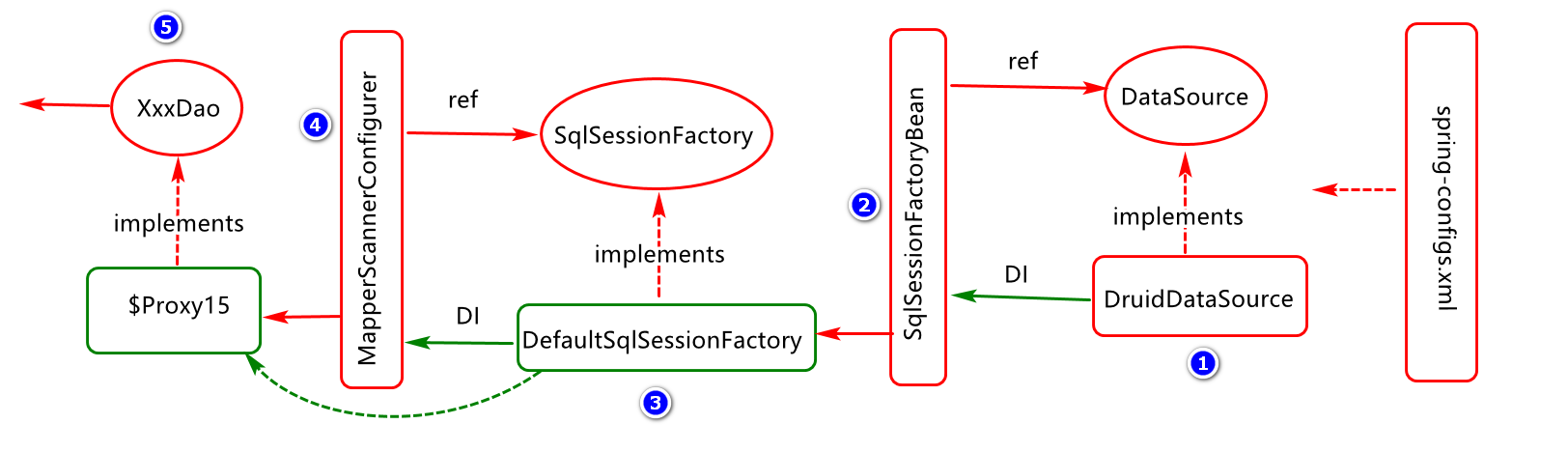
}

}

其中ClassPathXmlApplicationContext为基于xml方式的Context对象。

## 资源整合

整合的过程基本分析以及实现:参考具体数字



### 整合DataSource 对象

本项目采用阿里的druid作为数据源对象，实现与数据库的链接。

Step01: 添加依赖(mysql,druid)：已添加则无需添加

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.40</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.1.10</version>

</dependency>

Step02:添加db.properties文件，并定义key/value,其内容如下：

在src/main/resources 目录下添加db.properties文件.

jdbcDriver=com.mysql.jdbc.Driver

jdbcUrl=jdbc:mysql:///jtsys?useUnicode=true&characterEncoding=utf-8

jdbcUser=root

jdbcPassword=root

Step03:在spring-repository.xml中对数据源druid进行配置。

<bean id=*"dataSource"* class=*"com.alibaba.druid.pool.DruidDataSource"*

init-method=*"init"* destroy-method=*"close"* lazy-init=*"false"*>

<property name=*"DriverClassName"* value=*"#{cfg.jdbcDriver}"*/>

<property name=*"Url"* value=*"#{cfg.jdbcUrl}"*/>

<property name=*"Username"* value=*"#{cfg.jdbcUser}"*/>

<property name=*"Password"* value=*"#{cfg.jdbcPassword}"*/>

</bean>

Step04:定义测试类继承TestBase类：

**package** com.test;

**public** **class** TestDataSource **extends** TestBase{

@Test

**public** **void** testDruidDataSource()**throws** Exception {

DataSource ds=ctx.getBean("dataSource", DataSource.**class**);

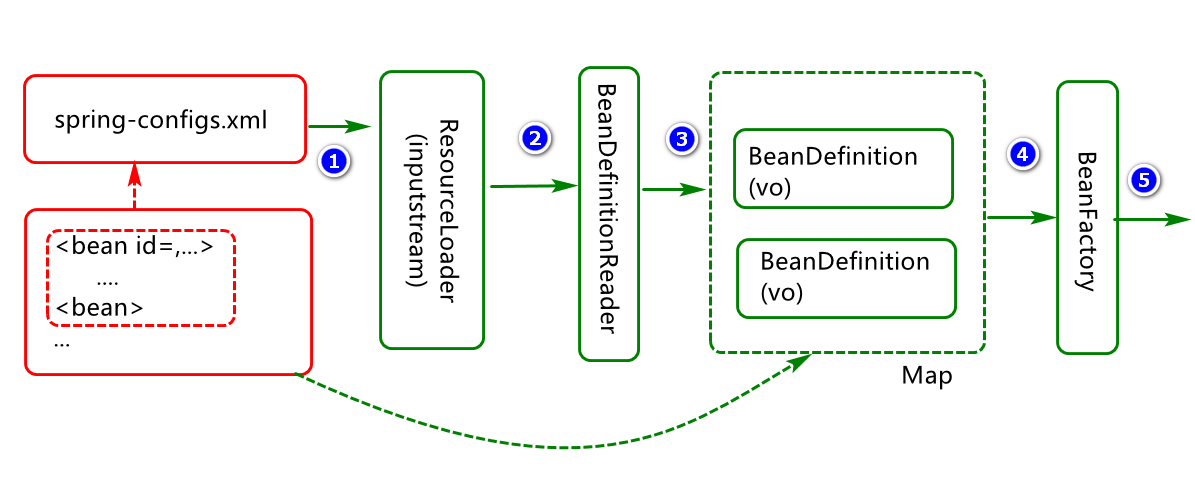
System.***out***.println(ds.getConnection());

}

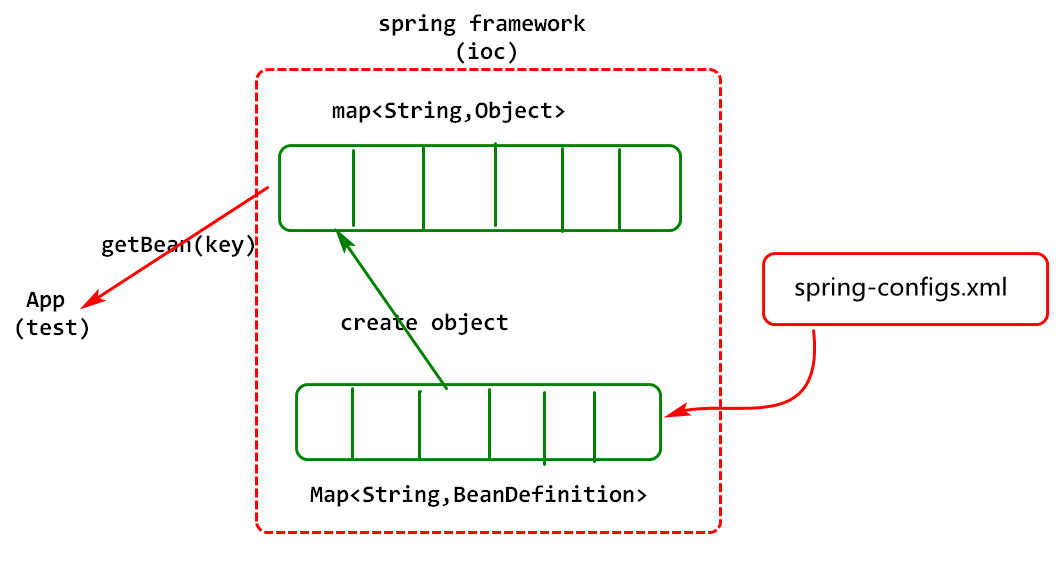
}

思考：

Spring 中容器初始化过程分析



Spring 中的两大MAP分析：



### 整合Mybatis框架

官网:http://www.mybatis.org/spring/getting-started.html

Step01：添加依赖（mybatis,mybatis-spring,spring-jdbc），已添加则无需添加

1)mybatis 核心依赖

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.6</version>

</dependency>

2)spring整合mybatis时需要的依赖

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.2</version>

</dependency>

3)spring整合mybatis时系统会将事务控制切换到spring，然后需要如下依赖。

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.3.9.RELEASE</version>

</dependency>

Step02: 将如下配置添加到spring-repository.xml配置文件中

<!-- 借助此Bean对象创建SqlSessionFactory对象

，当我们调用工厂对象的getBean方法获取id为sqlSessionFactory

的对象时，系统会返回SqlSessionFactoryBean对象的getObject

方法返回的对象。-->

<bean id=*"sqlSessionFactory"*

class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

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

<!-- 设置mapper文件的位置 -->

<property name=*"MapperLocations"*

value=*"classpath\*:mapper/sys/\*.xml"*/>

</bean>

Step03:定义测试类，对mybatis进行单元测试

**package** com.test;

**public** **class** TestMyBatis **extends** TestBase{

@Test

**public** **void** testSqlSessionFactory()**throws** Exception {

SqlSessionFactory ssf=

ctx.getBean("sqlSessionFactory", SqlSessionFactory.**class**);

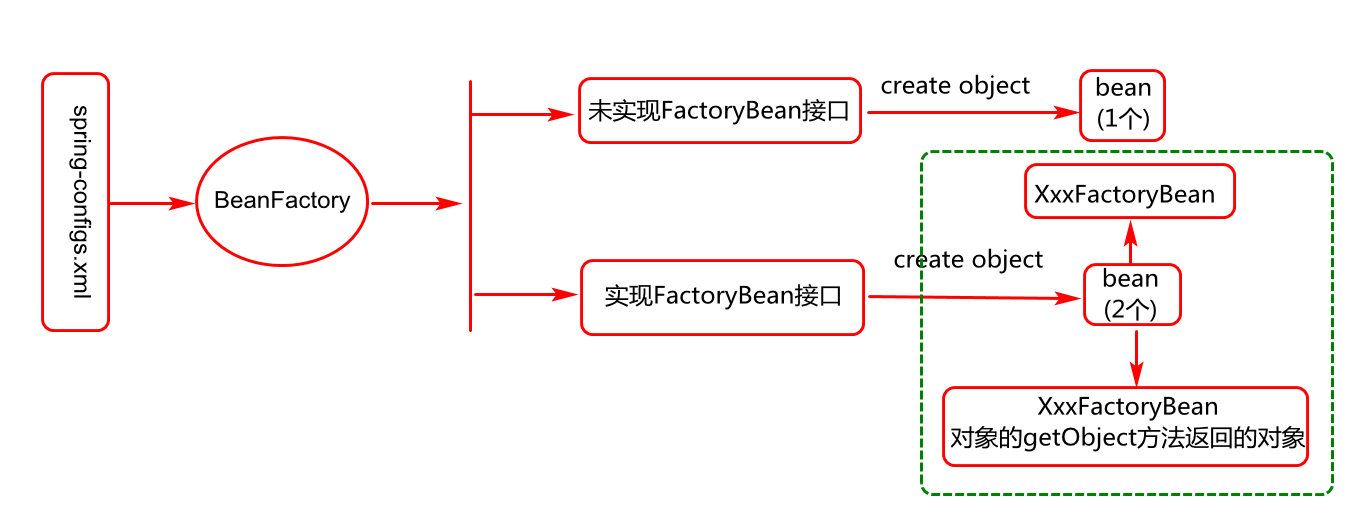
System.***out***.println(ssf);

}

}

假如测试ok,后续我们就可以通过SqlSessionFactory创建SqlSession对象,然后基于此SqlSession对象实现与数据库会话了.

Spring 中Bean对象构建方式增强分析：



Mybatis 增强配置:

<!-- 通过此bean对象扫描指定包下的dao接口，并基于接口创建其实现类对象(代理对象) -->

<bean id=*"daoScanner"*

class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<property name=*"BasePackage"* value=*"com.db.\*\*.dao"*/>

<!-- 当spring容器只有一个sqlSessionFactory时如下配置可以不写 -->

<property name=*"SqlSessionFactoryBeanName"* value=*"sqlSessionFactory"*/>

</bean>

课堂案例: 通过整合mybatis完成一个日志删除操作

1. 创建SysLogDao接口,并添加如下方法:

**package** com.db.sys.dao;

**import** java.util.List;

**import** org.apache.ibatis.annotations.Param;

**import** com.db.sys.entity.SysLog;

/\*\*

\* spring-respository.xml 此

\* 文件中定义了一个MapperScannerConfigurer

\* 对象,此对象会对BasePackage属性指定的包

\* 下接口进行扫描,然后会为接口创建实现类的对象,

\* 并将这个对象存储到bean池,其key为接口名(首字母小写).

\*/

**public** **interface** SysLogDao {//SysLogDao.class

/\*\*

\* 执行日志删除操作

\* **@param** id

\* **@return**

\*/

**int** deleteObjects(@Param("ids")Integer... id);

}

2.在指定目录创建在SysLogMapper文件,存储到src/main/resources目中

<?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.db.sys.dao.SysLogDao"*>

<delete id=*"deleteObjects"*>

delete from sys\_logs

where id in <!-- (1,2,3,4,5) -->

<foreach collection=*"ids"*

open=*"("*

close=*")"*

separator=*","*

item=*"id"*>

#{id}

</foreach>

</delete>

</mapper>

1. 创建单元测试类,执行删除操作

**package** com.test;

**import** java.util.List;

**import** org.junit.Test;

**import** com.db.sys.dao.SysLogDao;

**import** com.db.sys.entity.SysLog;

**public** **class** TestSysLogDao **extends** TestBase {

@Test

**public** **void** testDeleteObjects() {

SysLogDao dao=

ctx.getBean("sysLogDao",SysLogDao.**class**);

**int** rows=dao.deleteObjects(11,12);

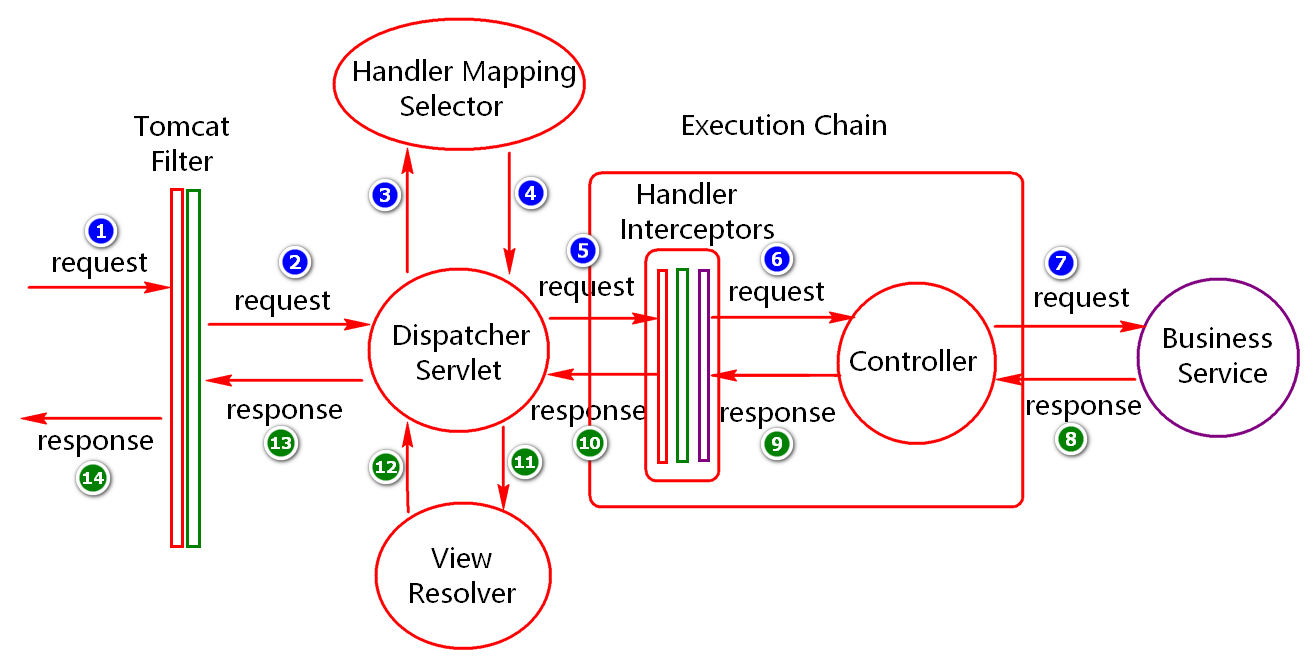
System.***out***.println("delete ok,rows="+rows);

}

}

### 整合Spring MVC模块

Spring MVC 基本架构回顾：



Step01:添加依赖(spring-webmvc)，已添加则无需添加

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.9.RELEASE</version>

</dependency>

Step02:将如下配置添加到spring-web.xml文件

<!--启用mvc默认配置(内置一部分bean对象的定义)-->

<mvc:annotation-driven/>

<!--注册视图解析对象 -->

<bean id=*"viewResovler"* class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"Prefix"* value=*"/WEB-INF/pages/"*/>

<property name=*"Suffix"* value=*".html"*/>

</bean>

Step03:打开web.xml配置前端控制器

<!-- 配置spring mvc前端控制器 -->

<servlet>

<servlet-name>frontController</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<!-- 通过此参数的配置加载spring配置文件，然后初始化资源 -->

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-configs.xml</param-value>

</init-param>

<!--让tomcat启动则加载此servlet对象，数值越小优先级越高 -->

<load-on-startup>1</load-on-startup>

</servlet>

<!-- 配置Spring MVC前端控制器映射(servlet映射) -->

<servlet-mapping>

<servlet-name>frontController</servlet-name>

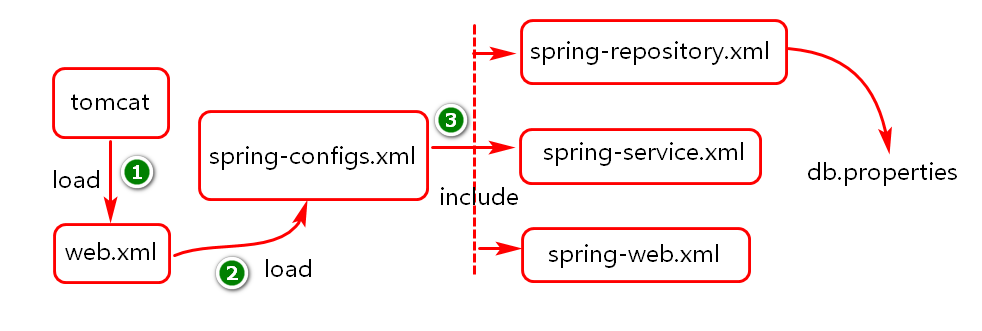
<url-pattern>\*.do</url-pattern>

</servlet-mapping>

Step04:部署项目启动tomcat进行测试（假如启动OK则测试通过）。

Step05:启动分析:

tomcat 启动时资源加载分析,如下图所示:



### 整合Jackson 实现

Spring mvc 模块默认支持Jackson，只需要在项目中添加jackson依赖即可

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.7</version>

</dependency>

## 首页初始化

### 定义页面初始资源

* + - 1. 将AdminLTE相关资源拷贝到项目webapps目录
      2. 将pages页面拷贝到项目的WEB-INF目录

### 创建页面Controller

创建呈现首页页面的controller对象。

**package** com.db.sys.controller;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

@RequestMapping("/")

@Controller

**public** **class** PageController {

@RequestMapping("doIndexUI")

**public** String doIndexUI(){

**return** "starter";

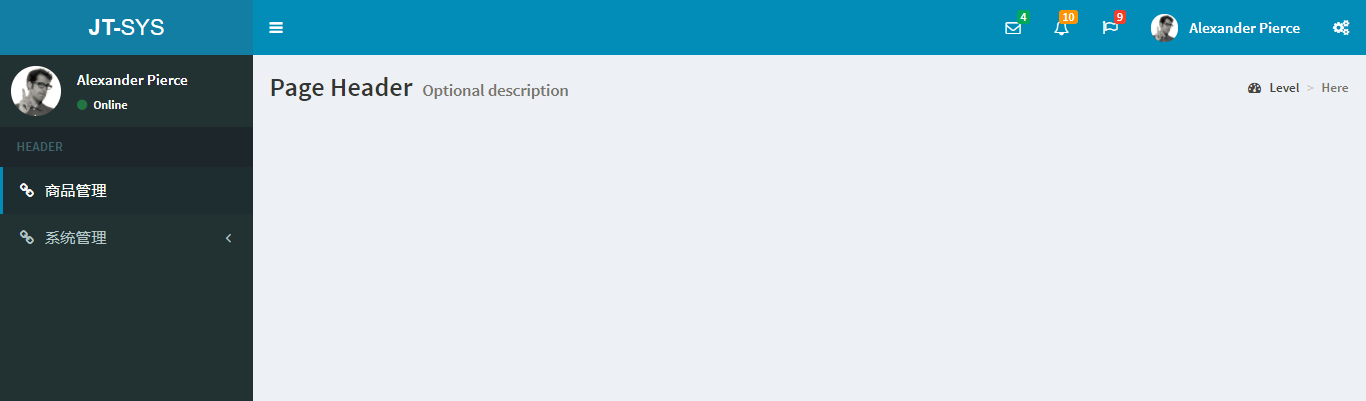
}

}

### 启动项目进行测试

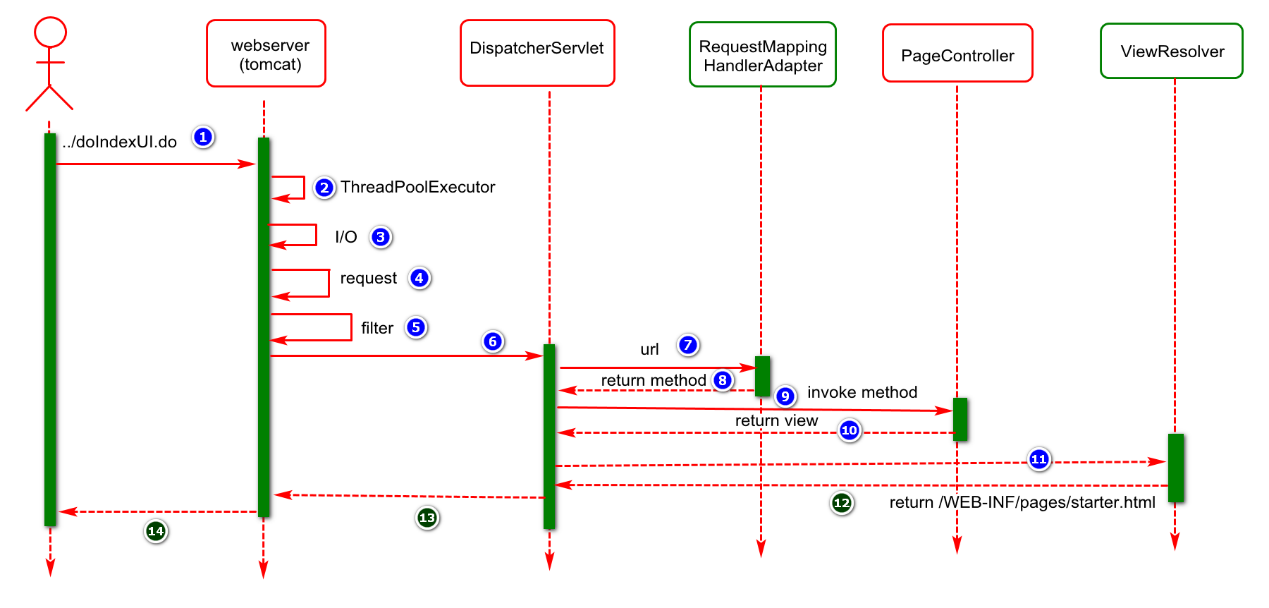
启动tomcat，在地址栏输入

http://localhost/CGB-DB-SYS-V3.01/doIndexUI.do



说明:地址中的端口号要参考自己tomcat启动端口.

访问流程分析：



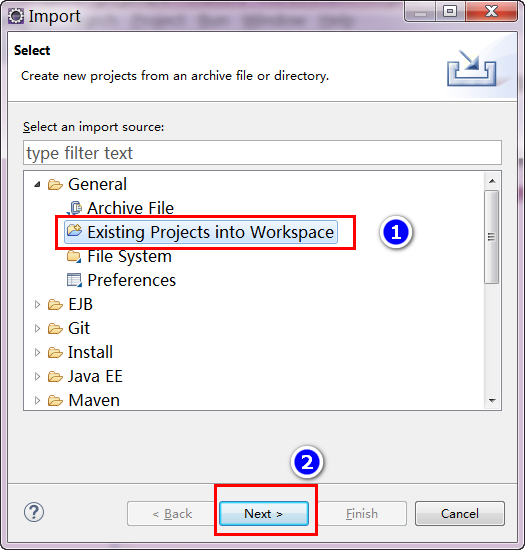
# SSM 整合总结

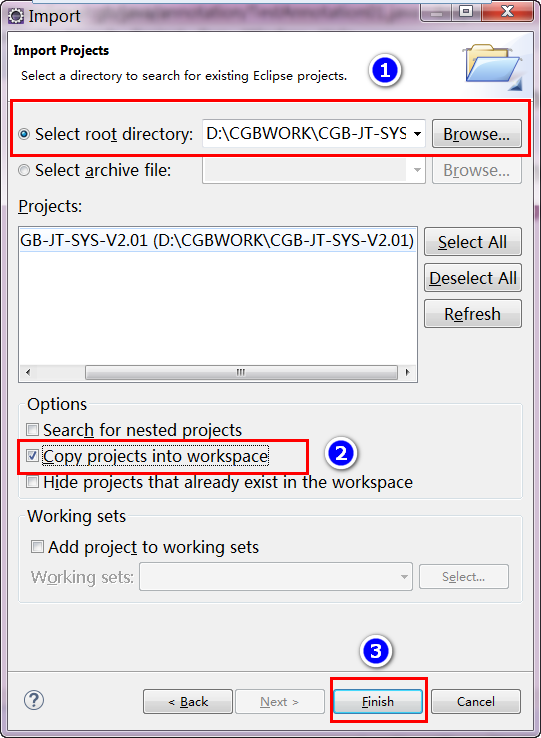
## 重点难点分析

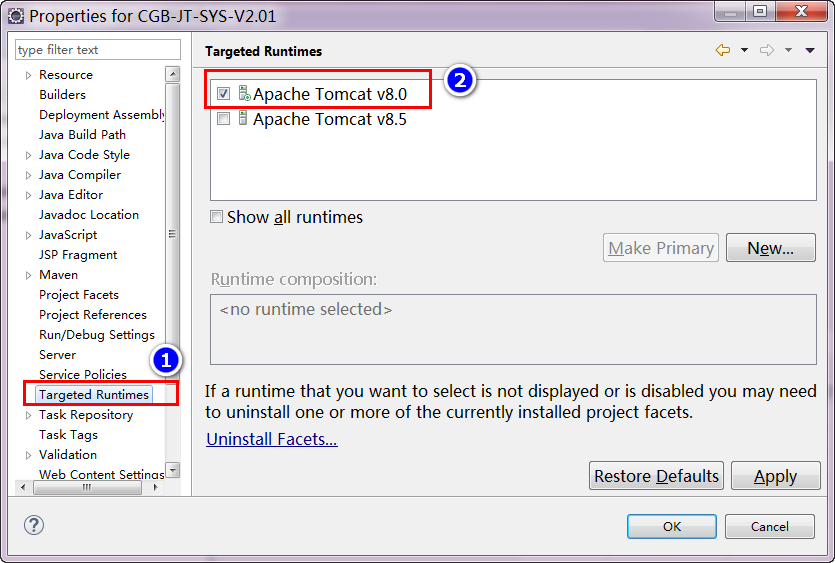
* + - 1. 整合Druid 数据源
      2. 整合mybatis 框架
      3. 整合Spring mvc 模块

## 问题分析

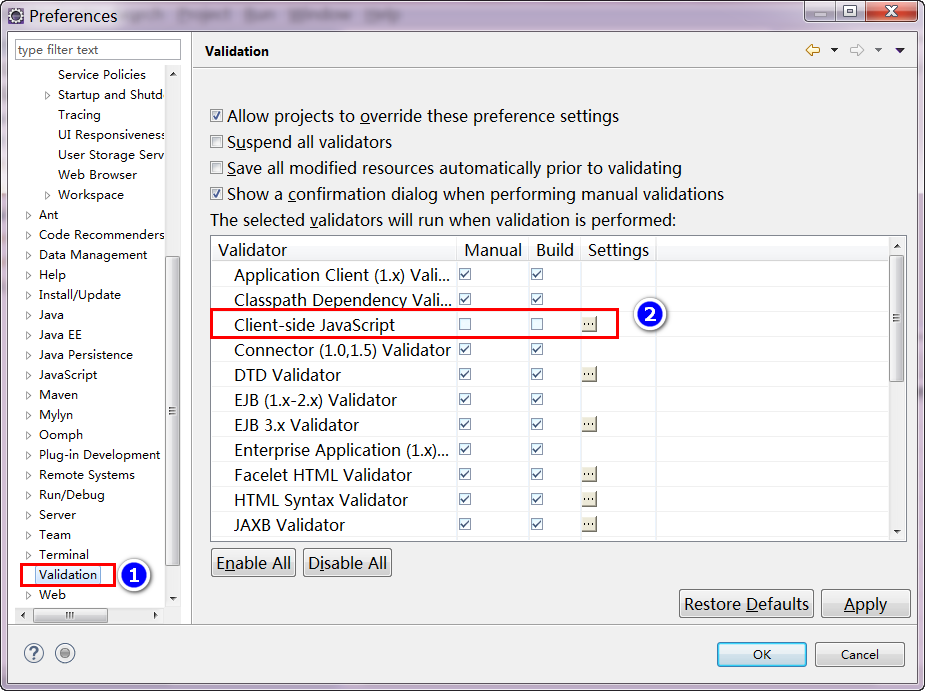
项目导入







项目中JS问题



数据库连不上？

1. 服务是否开启？（命令行开启服务：mysqld --console）
2. 密码不正确？
3. 驱动不匹配？
4. 数据库名字，url等是否正确？
5. …
   * + 1. Spring中的Bean对象何时创建？(看配置，是否配置了延迟加载)
       2. Spring中的的两大容器如何理解？
6. 存储bean配置信息的容器（Map<String,BeanDefinition>）
7. 存储bean实例信息的容器 (Map<String,Object>)
   * + 1. Spring中bean对象构造方法，set方法，init方法执行顺序？
8. 构造方法
9. Set方法
10. Init方法
    * + 1. Spring 中的核心工厂？（BeanFactory）
        2. Spring 中的两大Map？（存储bean定义，存储bean的实例）
        3. Spring 整合SqlSessionFactoryBean的目的？

基于SqlSessionFactoryBean对象创建SqlSessionFactory，在此

过程中其实创建了两个Bean对象，还有一个是SqlSessionFactoryBean对象，

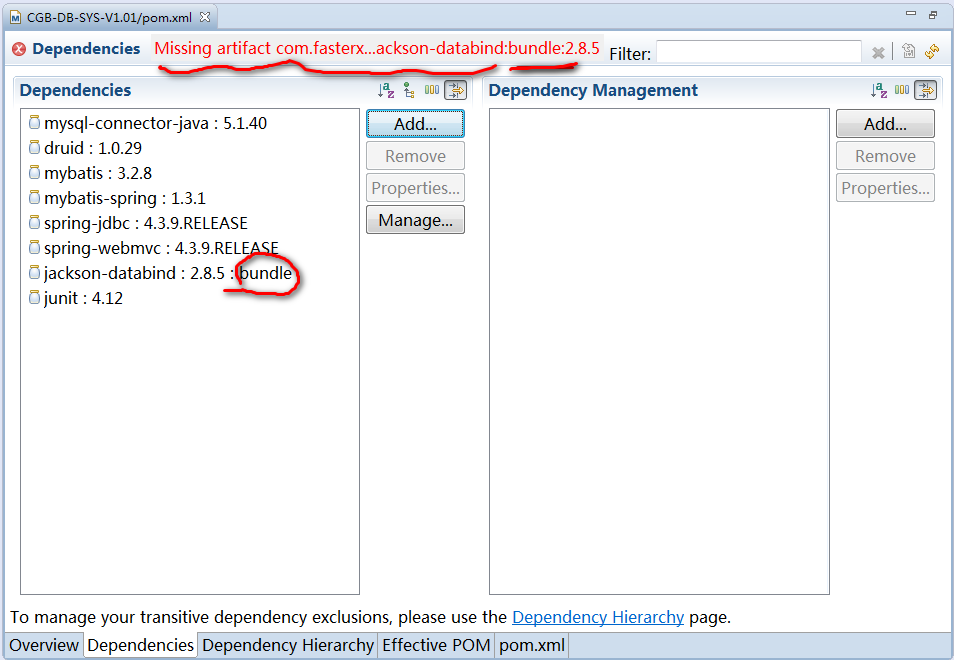
要想获取此bean对象，需要在对应key的前面加上”&”符号。

* + - 1. 如何理解Spring框架规范中的两大Bean类型？

1. 实现了FactoryBean接口的Bean类型。
2. 未实现FactoryBean接口的Bean类型。
   * + 1. 整合SqlSessionFactoryBean时是否依赖于DataSource接口对象？依赖
       2. 整合MapperScannerConfiger对象的作用是什么？基于此对象扫描指定包中的接口然后基于接口创建接口的实现类对象(代理对象)，目的是简化我们自己写实现类的过程。

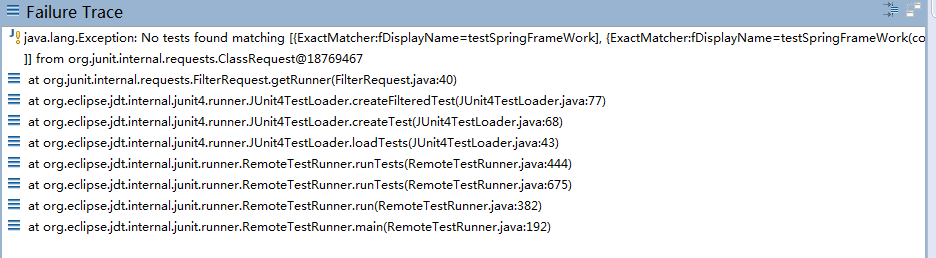
## Bug分析

Bug-01



问题分析：去掉用依赖对应的type属性即可。

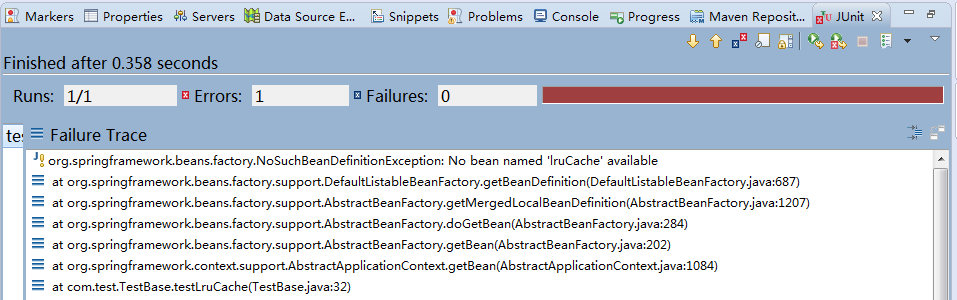
BUG-02



问题分析：

1. 测试方法是否私有了
2. 测试方法是否加了参数
3. 测试方法是否有返回值
4. 换个jar依赖试试。

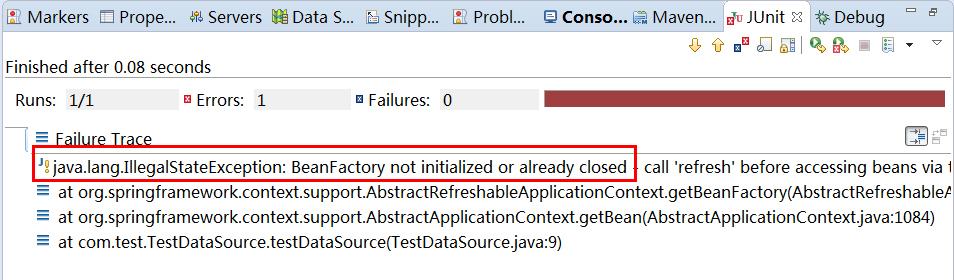
BUG-03



问题原因:

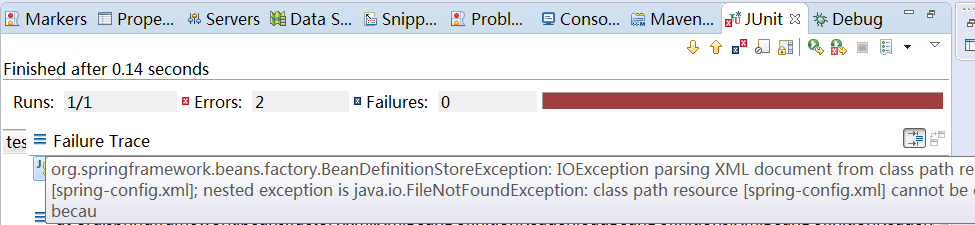
1. bean的名字不正确
2. bean的配置不正确
3. bean的扫描不正确
4. ....

BUG-04



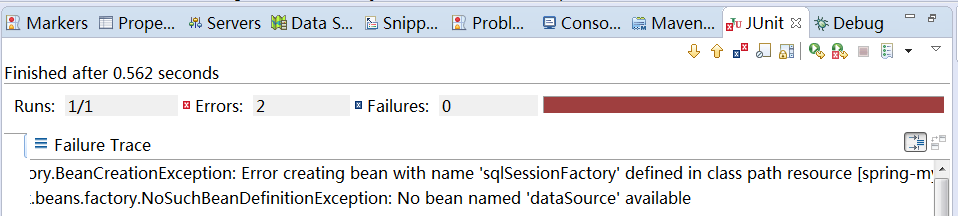
问题分析：单元测试父类中的的初始化方法中没有指定要加载的配置文件。

Bug-05



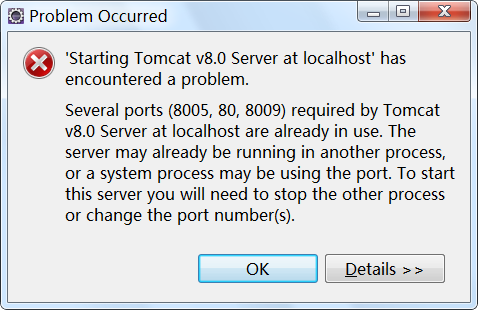
问题分析：spring-config.xml文件没找到，检查文件名是否正确。

Bug-04



问题分析：可能是id为dataSource的名字写错了

BUG-05



问题描述：tomcat端口被占用

1. 执行tomcat/bin目录中的shutdown指令。
2. 杀进程 (找进程,然后杀进程)
3. 改端口

BUG-06:

1.404 异常（请求资源404，响应资源404）

1)检测tomcat启动是否ok？

2)请求404要检测访问路径url是否正确？

3)请求404要检测对应的资源(对象)是否存在？（@Controller,扫描包对吗）

4)请求404要检测项目有没有编译到指定目录？（tomcat的加载目录）

5)项目是否有过重命名(检查eclipse/servers/tomcat/server.xml中的context)

2.项目不编译？(先重新发布项目publish，假如不起作用，然后四大clean)

1)tomcat clean (两个):清空项目部署目录中资源

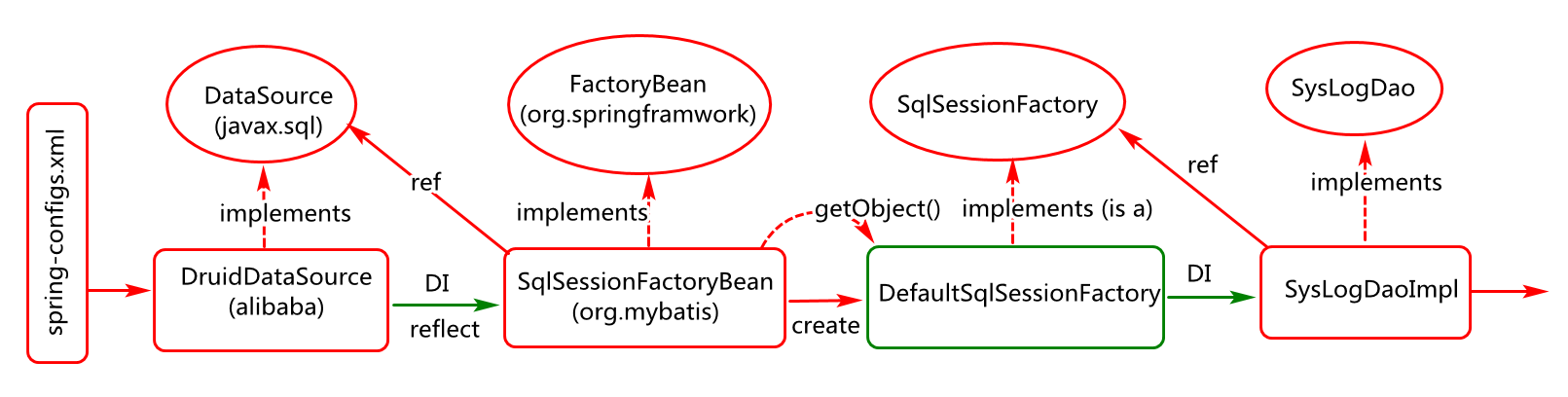
2)maven clean （run as）:清除项目target目录中的资源

3)project clean （重新加载编译）:让项目重新加载重新编译

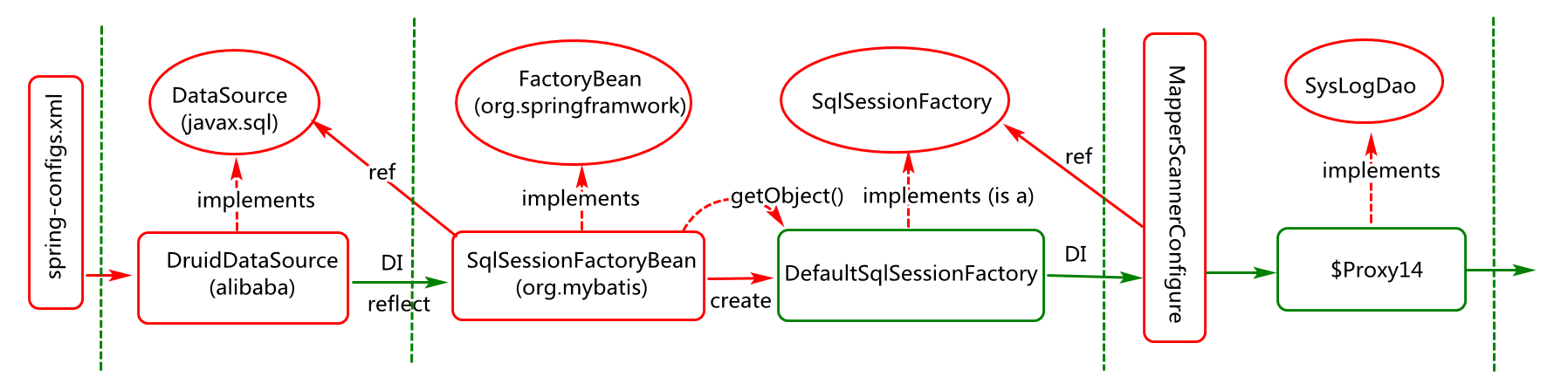
假如四大clean还解决不变了问题，重启eclipse.

## 原理分析

理解spring整合datasource,mybatis基本过程，自己写dao接口实现类



理解spring整合datasource,mybatis基本过程，系统自动生成dao接口实现类



理解mybatis中sqlsession对象的应用方式。

