京淘项目

权限管理子系统

SSM技术整合

Spring+SpringMVC+MyBatis

(XML方式)

[1. SSM 技术整合 1-2](#_Toc531103793)

[1.1. 基本架构 1-2](#_Toc531103794)

[1.1.1. 项目分层架构 1-2](#_Toc531103795)

[1.1.2. API 应用架构 1-4](#_Toc531103796)

[1.2. 准备工作 1-4](#_Toc531103797)

[1.2.1. 数据库初始化 1-4](#_Toc531103798)

[1.2.2. 创建并配置项目 1-4](#_Toc531103799)

[1.2.3. 添加项目依赖 1-6](#_Toc531103800)

[1.2.4. 创建配置文件 1-7](#_Toc531103801)

[1.2.5. 创建测试基类 1-8](#_Toc531103802)

[1.3. 资源整合 1-9](#_Toc531103803)

[1.3.1. 整合DataSource 对象 1-9](#_Toc531103804)

[1.3.2. 整合Mybatis框架 1-11](#_Toc531103805)

[1.3.3. 整合Spring MVC模块 1-13](#_Toc531103806)

[1.3.4. 整合Jackson 实现 1-15](#_Toc531103807)

[1.4. 首页初始化 1-15](#_Toc531103808)

[1.4.1. 定义页面初始资源 1-15](#_Toc531103809)

[1.4.2. 创建页面Controller 1-15](#_Toc531103810)

[1.4.3. 启动项目进行测试 1-16](#_Toc531103811)

[2. SSM 整合总结 2-17](#_Toc531103812)

[2.1. 重点难点分析 2-17](#_Toc531103813)

[2.2. 问题分析 2-17](#_Toc531103814)

[2.3. Bug分析 2-21](#_Toc531103815)

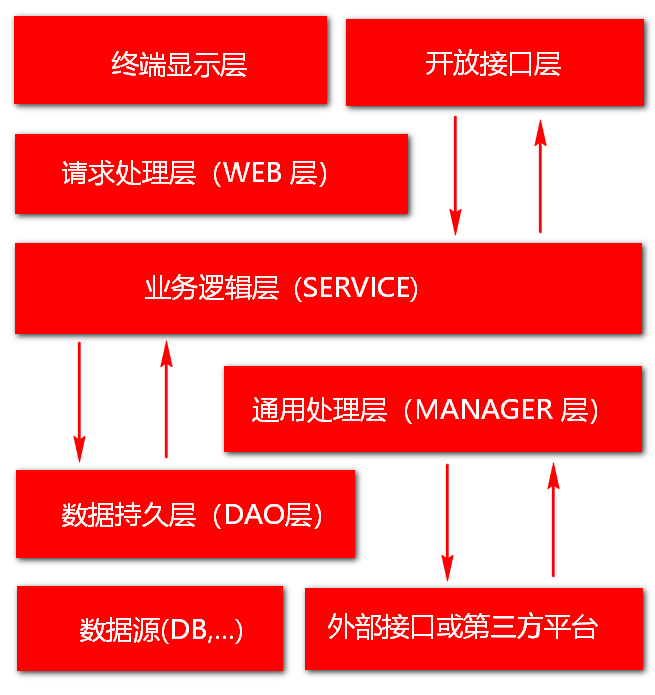
[2.4. 原理分析 2-22](#_Toc531103816)

# SSM 技术整合

## 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；

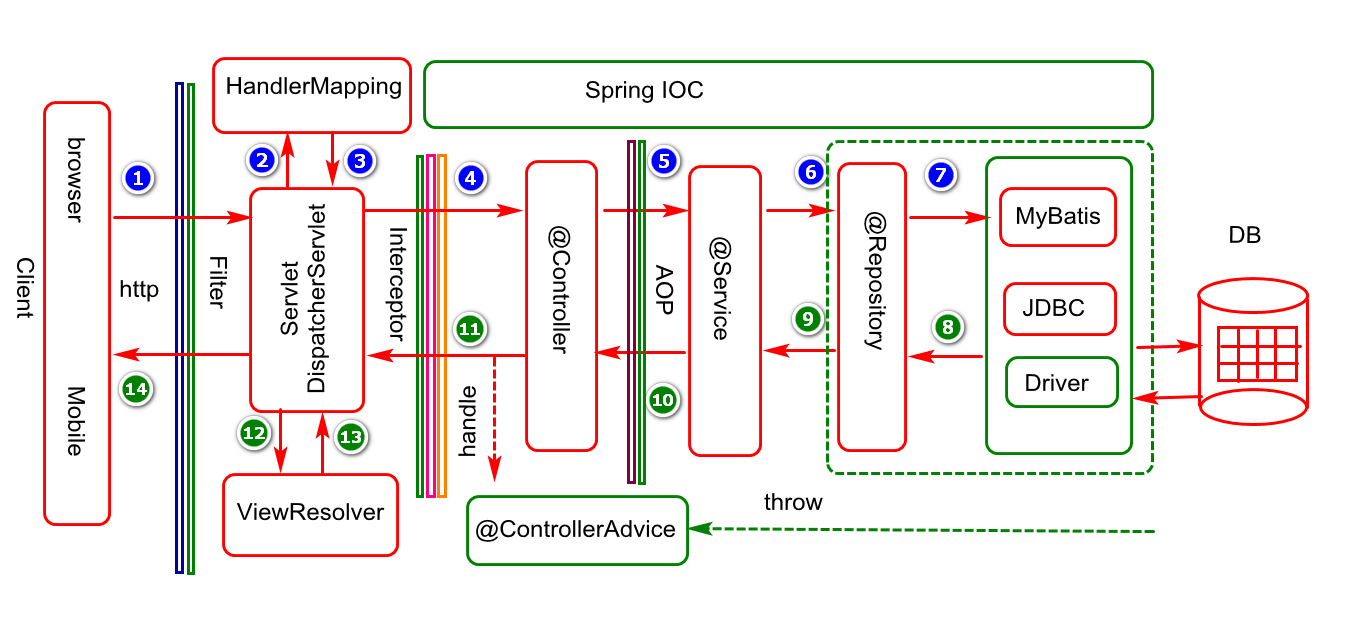
3） 与 DAO 层交互，对多个 DAO 的组合复用。

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。

### API 应用架构



## 准备工作

### 数据库初始化

在mysql客户端中执行jtsys.sql文件。

命令行执行步骤：

1. 登陆mysql
2. 执行set names utf8
3. 基于source指令运行jtsys.sql文件

说明：执行成功以后，系统中会自动创建jtsys数据库以及若干张表。

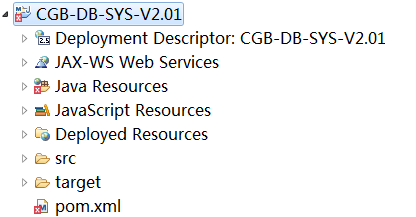
### 创建并配置项目

基本步骤

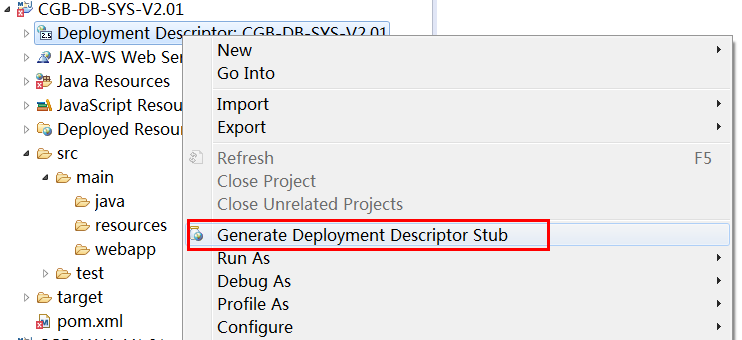
1. 创建maven web项目，名字为CGB-DB-SYS-V2.01
2. 生成部署描述文件 web.xml(全注解方式不需要)
3. 修改项目编码为utf-8（假如工作区编码为UTF-8,则无需修改）
4. 设置项目targeted runtimes (Tomcat)：暂时不用tomcat插件
5. 修改项目project facets (jdk1.8)

项目问题分析：

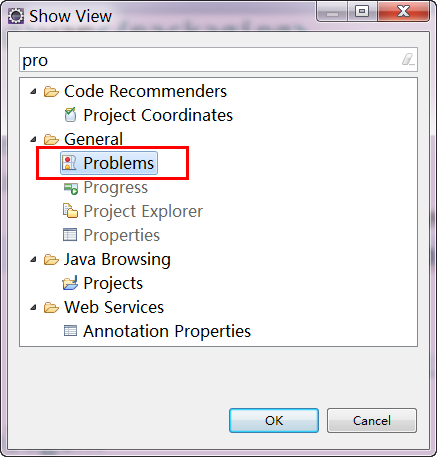
* + - 1. 项目创建完成没有web.xml?(生成web.xml)



生成web.xml



* + - 1. 项目中有错但不知什么错？（看problems）

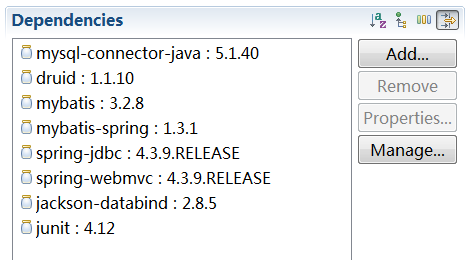




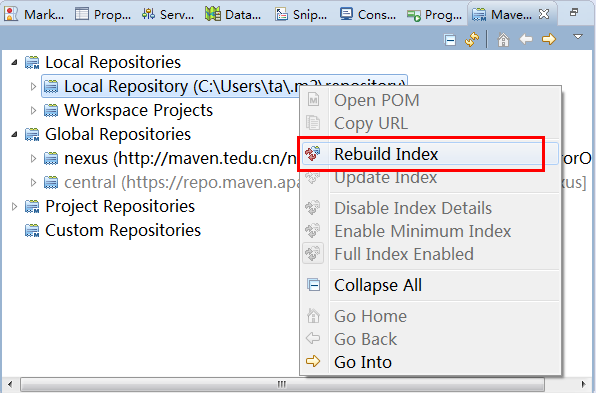
### 添加项目依赖

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（一个依赖fastjson,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：

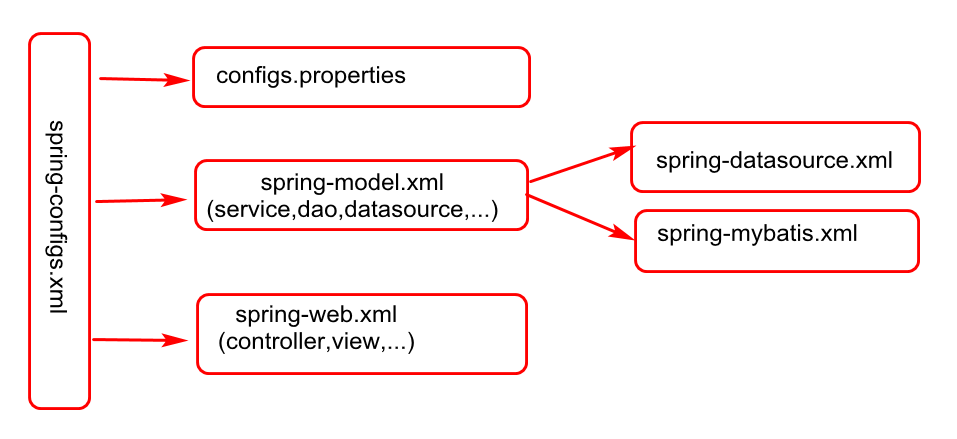


说明：假如依赖是从本地库添加可以在本地库创建索引：



### 创建配置文件

配置文件结构如下：



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configs.properties文件内容如下：

jdbcDriver=com.mysql.jdbc.Driver

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

jdbcUser=root

jdbcPassword=root

### 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** ClassPathApplicationContext(“spring-configs.xml”);

}

@After

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

ctx.close();

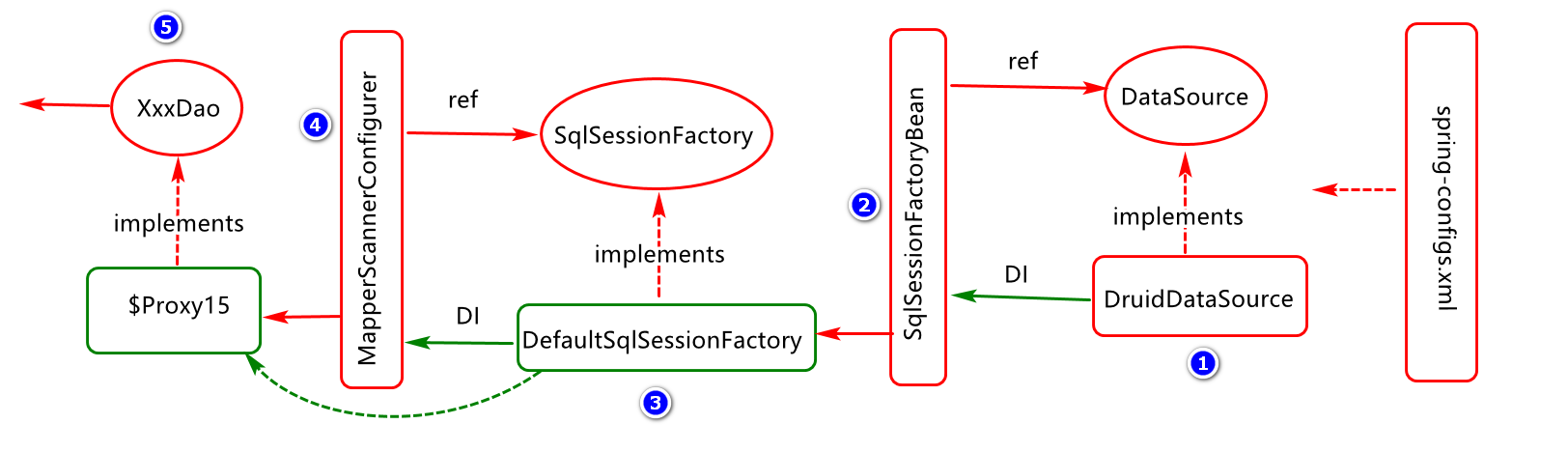
}

}

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configs.properties文件，并定义key/value,其内容如下：

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

jdbcDriver=com.mysql.jdbc.Driver

jdbcUrl=jdbc:mysql:///jtsys

jdbcUser=root

jdbcPassword=root

Step03:创建AppDataSourceConfig配置类，并对数据源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**);

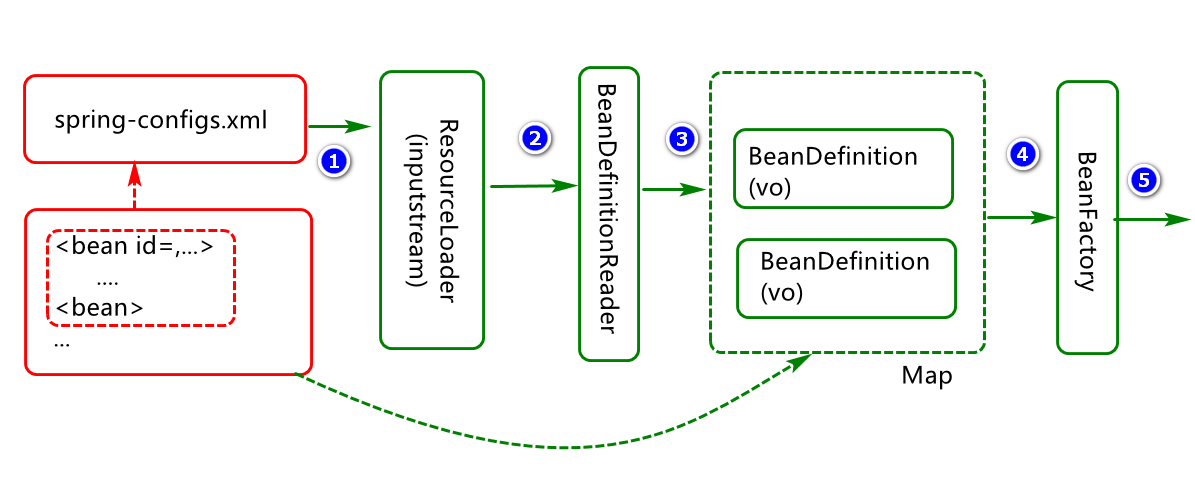
System.***out***.println(ds.getConnection());

}

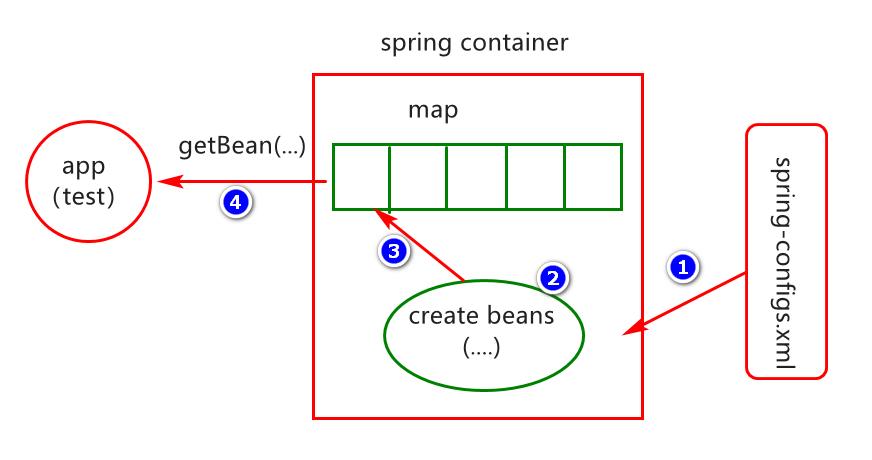
}

思考：

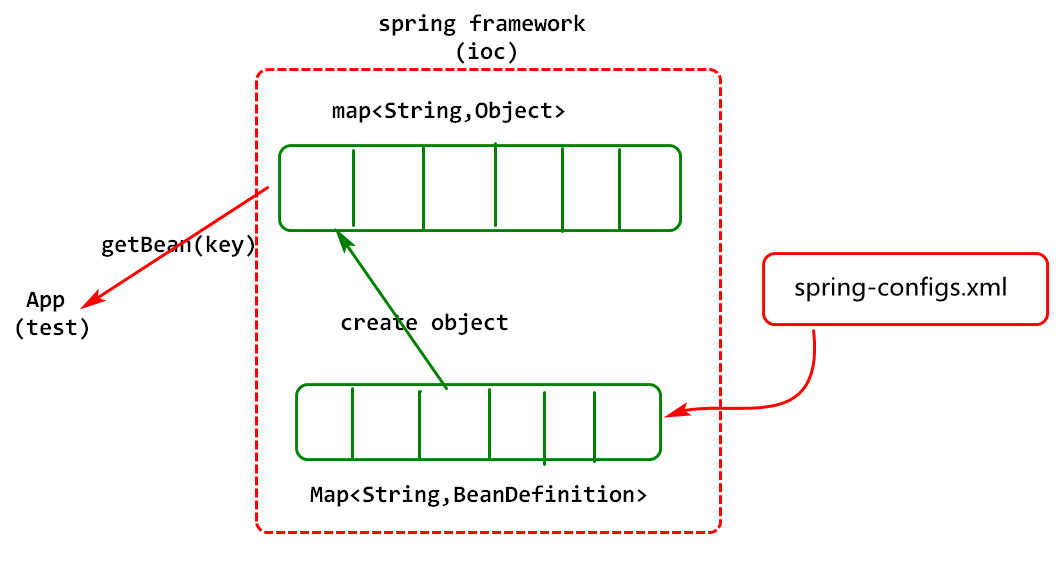
Spring 中容器初始化过程分析



Spring 中对象Bean创建过程分析：



Spring 中的两大MAP：



### 整合Mybatis框架

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

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.6</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.2</version>

</dependency>

<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-mybatis.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/\*Mapper.xml"*/>

</bean>

其中MapperLocations属性对应的映射要创建，存储到src/main/resources目中

例如SysLogMapper.xml,其内容如下：

<?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"*>

</mapper>

<!-- 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>

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=

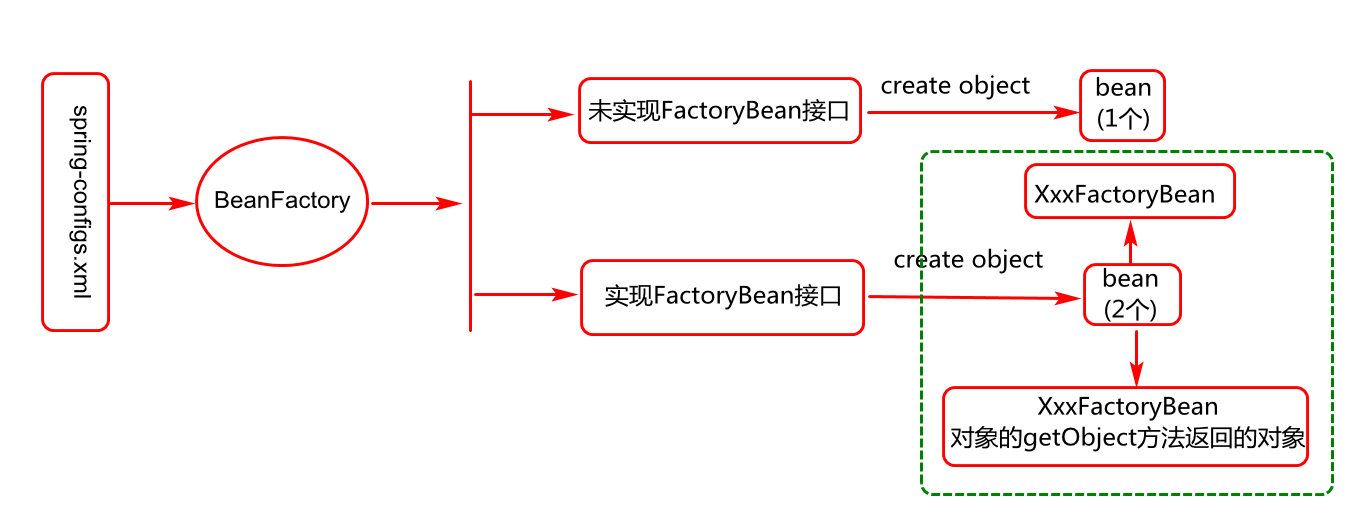
ctx.getBean("sqlSessionFactory", SqlSessionFactory.**class**);

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

}

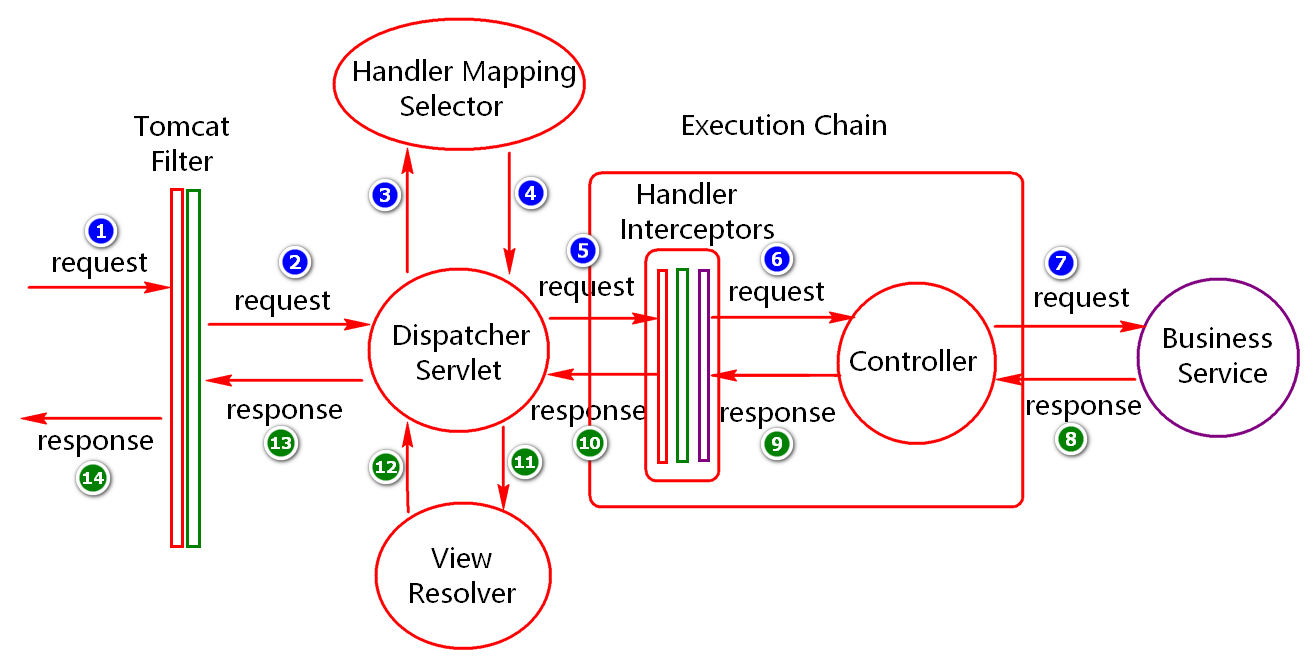
}

Spring 中Bean对象构建方式分析：



### 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>

Step06:部署项目启动tomcat进行测试（假如启动OK则测试通过）。

### 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>

Step04: 启动tomcat测试，是否启动OK

## 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";

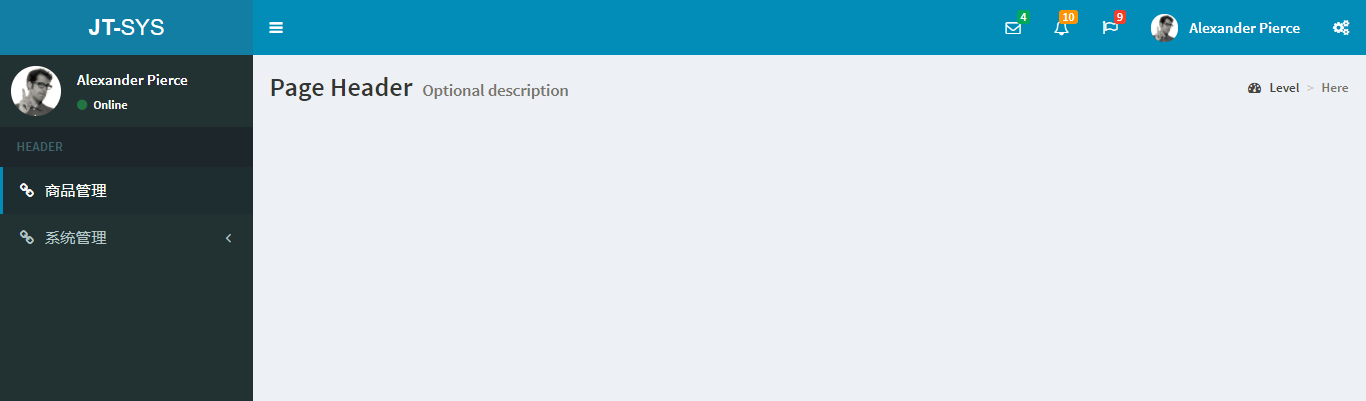
}

}

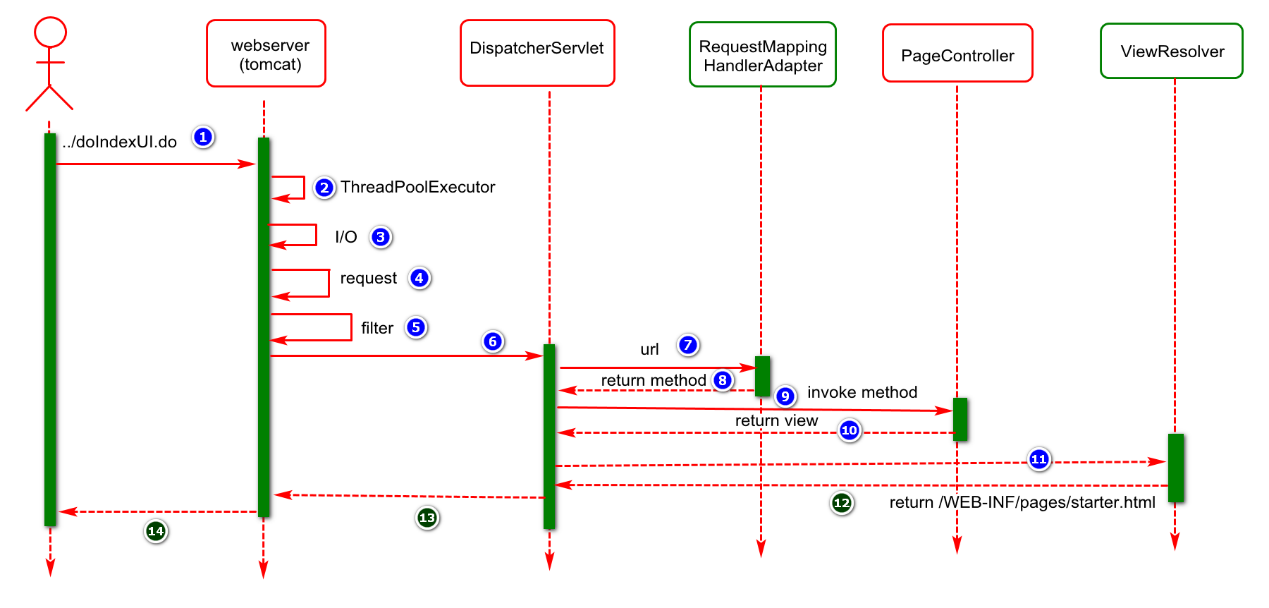
### 启动项目进行测试

启动tomcat，在地址栏输入

http://localhost:8080/CGB-DB-SYS-V2.01/doIndexUI.do



访问流程分析：



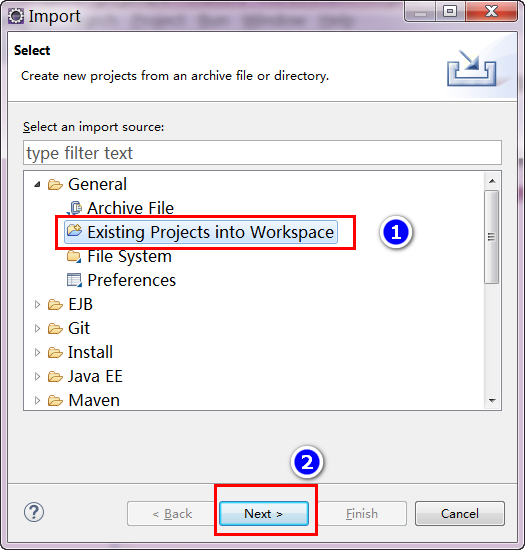
# SSM 整合总结

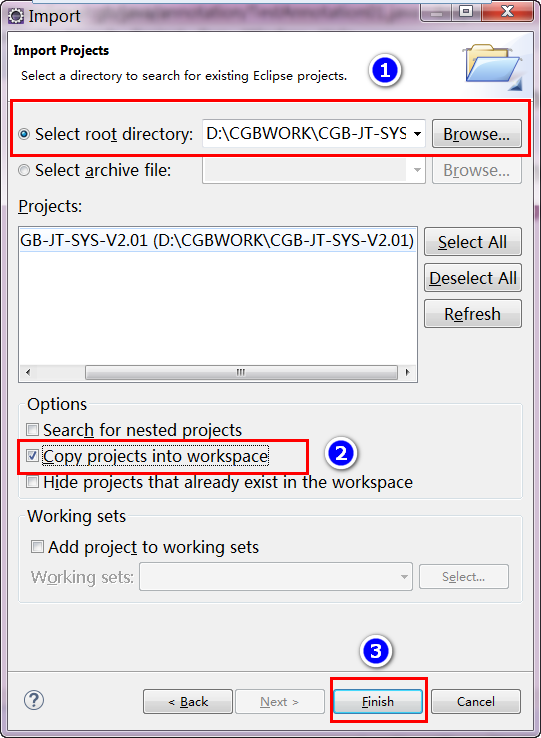
## 重点难点分析

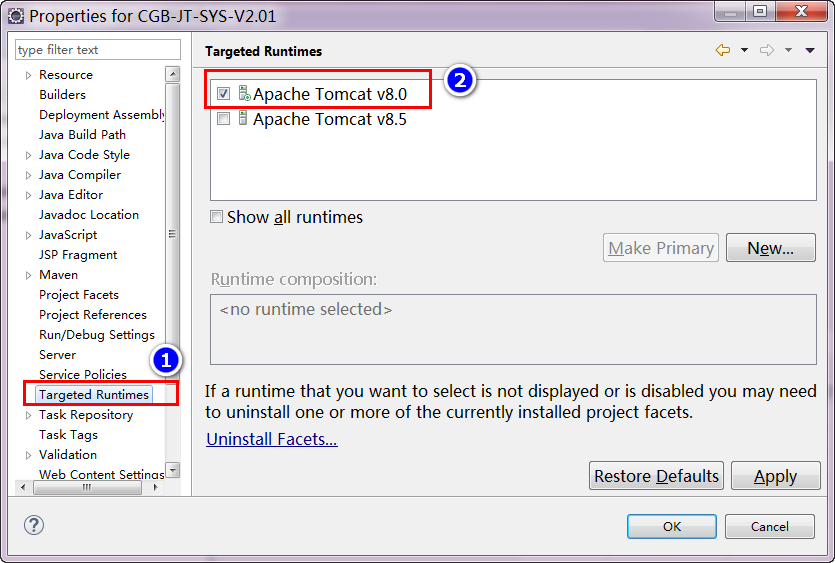
* + - 1. 整合mybatis 框架
      2. 整合Spring mvc 模块
      3. 整合Druid 数据源

## 问题分析

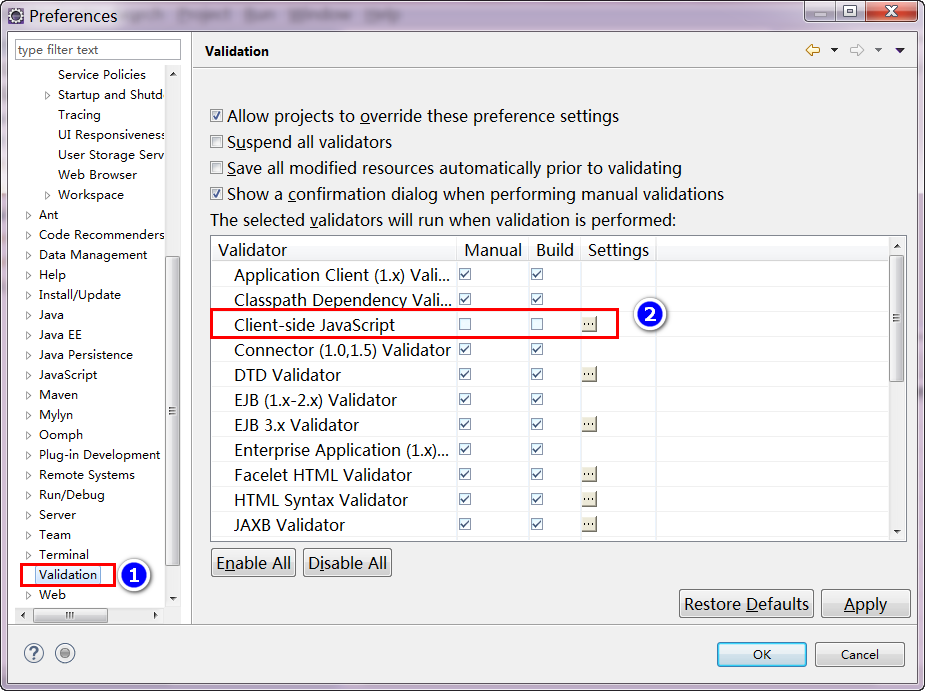
项目导入







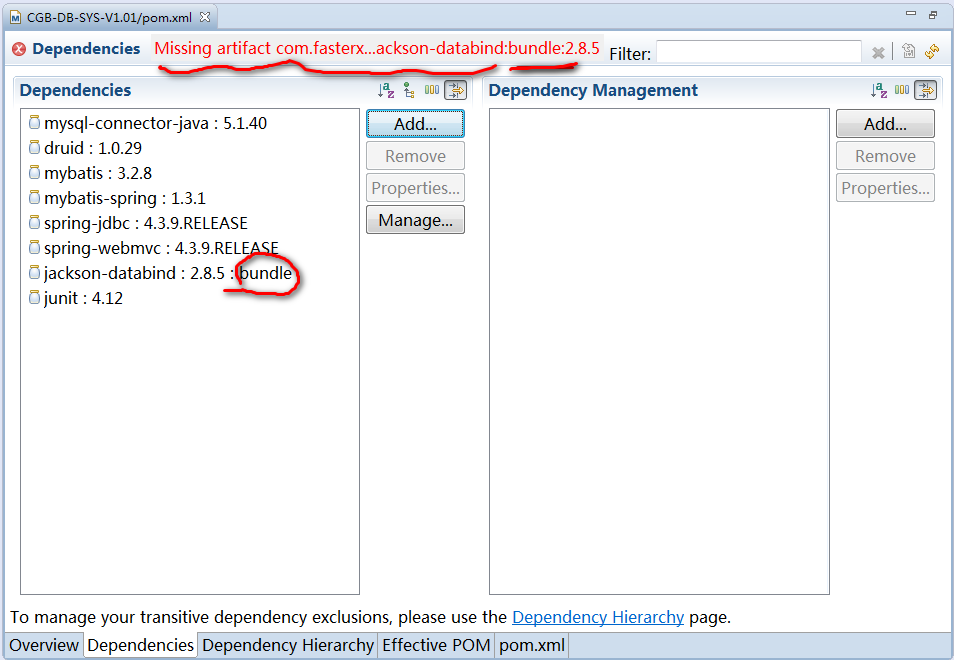
项目中JS问题



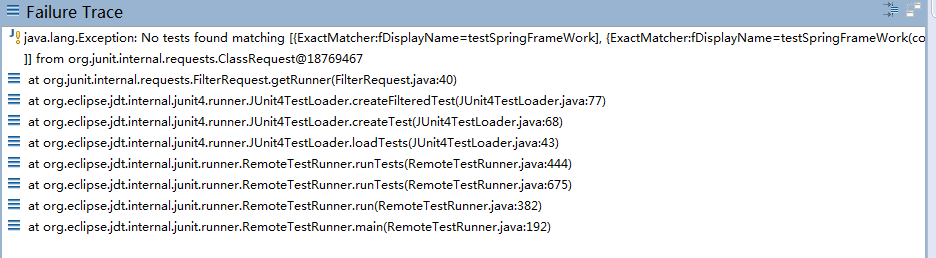
数据库连不上

## Bug分析

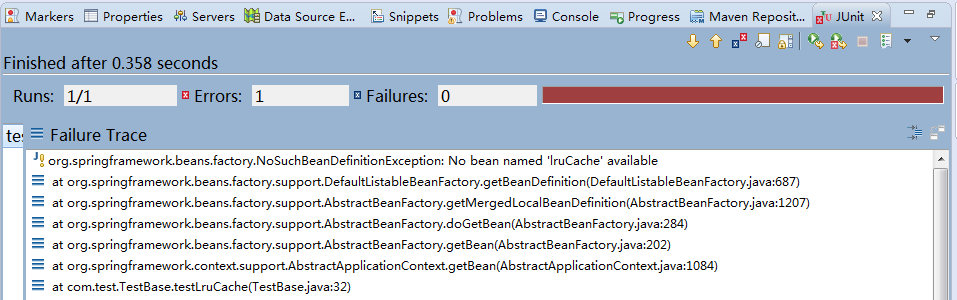
Bug-01



BUG-02



BUG-03



BUG-04:

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）

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)

2)maven clean （run as）

3)project clean （重新加载编译）

## 原理分析

理解泛型的作用及基本应用

加强反射技术理解