FRAMEWORK进阶

一图胜百文

[1. MYBATIS 框架基础 1-2](#_Toc10642573)

[1.1. MYBATIS 架构分析 1-2](#_Toc10642574)

[1.1.1. Mybatis 应用架构(重点) 1-2](#_Toc10642575)

[1.1.2. Mybatis 产品架构 1-3](#_Toc10642576)

[1.1.3. MyBatis 技术架构 1-4](#_Toc10642577)

[1.2. MYBATIS 快速实践(脱离文档) 1-6](#_Toc10642578)

[1.2.1. 初始化数据环境 1-6](#_Toc10642579)

[1.2.2. 创建并配置项目 1-6](#_Toc10642580)

[1.2.3. 业务应用快速实践 1-8](#_Toc10642581)

[1.3. MYBATIS 应用原理进阶分析 1-11](#_Toc10642582)

[1.3.1. 会话工厂创建分析(了解) 1-11](#_Toc10642583)

[1.3.2. 会话对象应用分析 (了解) 1-11](#_Toc10642584)

[1.3.3. 基于Mapper接口会话(了解) 1-12](#_Toc10642585)

[1.3.4. 缓存应用实现过程分析 1-12](#_Toc10642586)

[2. SPRING 框架基础进阶 2-14](#_Toc10642587)

[3.1. SPRING 框架架构分析 2-14](#_Toc10642588)

[3.1.1. Spring 框架应用架构 2-14](#_Toc10642589)

[3.1.2. Spring 框架产品架构 2-14](#_Toc10642590)

[3.1.3. Spring 框架技术架构 2-15](#_Toc10642591)

[3.2. SPRING 框架入门重构(注解方式-脱离文档) 2-16](#_Toc10642592)

[3.2.1. 创建并配置项目 2-16](#_Toc10642593)

[3.2.2. 业务实践与增强分析 2-17](#_Toc10642594)

[3.3. SPRING IOC 模块核心 2-19](#_Toc10642595)

[3.3.1. Spring Bean 容器的初始化 2-20](#_Toc10642596)

[3.3.2. Spring 中的两大map对象 2-20](#_Toc10642597)

[3.3.3. Spring 中两大bean对象 2-21](#_Toc10642598)

[3.3.4. Spring 中两大bean对象描述方式 2-22](#_Toc10642599)

[3.3.5. Spring 中Bean对象的依赖注入 2-22](#_Toc10642600)

[3.4. SPRING MVC 模块核心 2-24](#_Toc10642601)

[3.4.1. Spring MVC 核心架构 2-24](#_Toc10642602)

[3.4.2. Spring MVC 快速入门实现(xml方式-脱离文档) 2-24](#_Toc10642603)

[3.4.3. Spring MVC 快速入门实现(注解方式-脱离文档-了解) 2-26](#_Toc10642604)

[3.4.4. Spring MVC 请求响应处理 2-27](#_Toc10642605)

[4. FRAMEWORK总结分析 2-29](#_Toc10642606)

[4.1. 重点难点 2-29](#_Toc10642607)

[4.2. FAQ分析 2-29](#_Toc10642608)

[4.3. 作业实现 2-29](#_Toc10642609)

[4.4. BUG 分析 2-30](#_Toc10642610)

# MYBATIS 框架基础

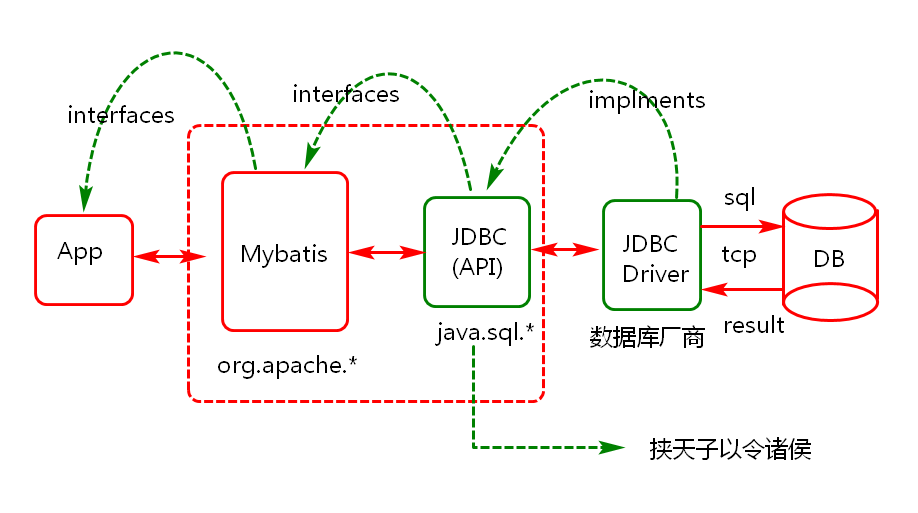
## MYBATIS 架构分析

### Mybatis 应用架构(重点)

谈谈对mybatis的应用架构的理解？

MyBatis 是一个持久层框架，实现了对JDBC操作的封装，主要用于简化JDBC

操作中的一些相对繁琐的步骤，例如参数的映射，结果集的映射等。



为何使用mybatis实现数据持久层应用？

1. 第一稳定，灵活（动态SQL），功能强大(池，日志，缓存)
2. 学习以及使用成本低
3. 解耦，SQL的可维护性，可复用性比较高。

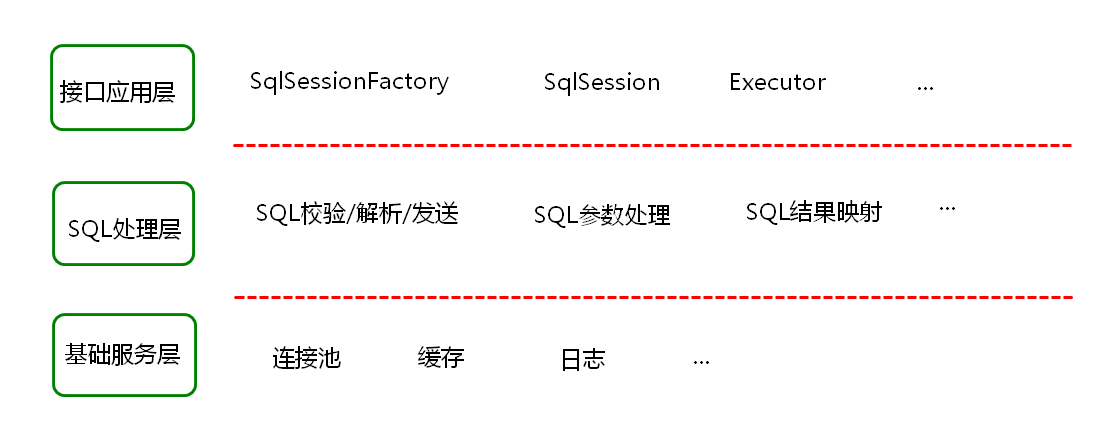
说明:mybatis官网

1. [mybatis.org/mybatis-3](http://www.mybatis.org/mybatis-3)
2. github.com/mybatis
3. <http://www.mybatis.org/spring/>

### Mybatis 产品架构

说说对mybatis的产品架构的理解？

所有框架都要解决一些共性问题（持久化），都是一种半成品，mybatis也不例外，它作为一种框架，它要解决相关问题，如何解决问题？采用怎样的架构解决问题，这是我们要学习的一个点。

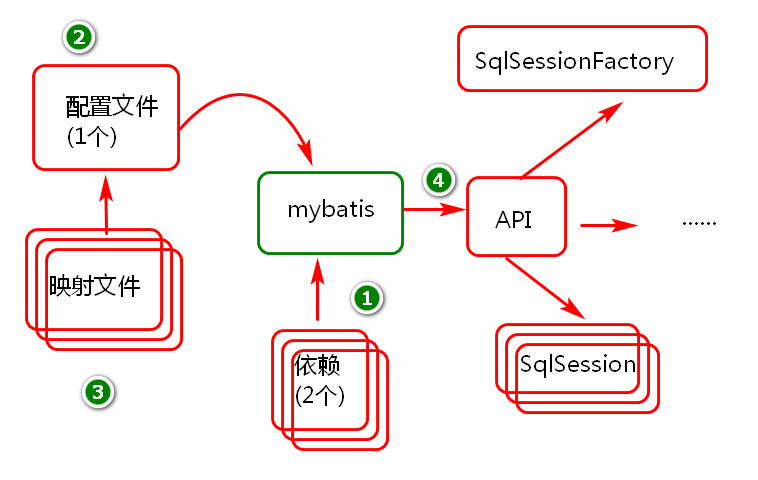


思考:mybatis作为一个持久层框架,应该解决哪些功能性问题?

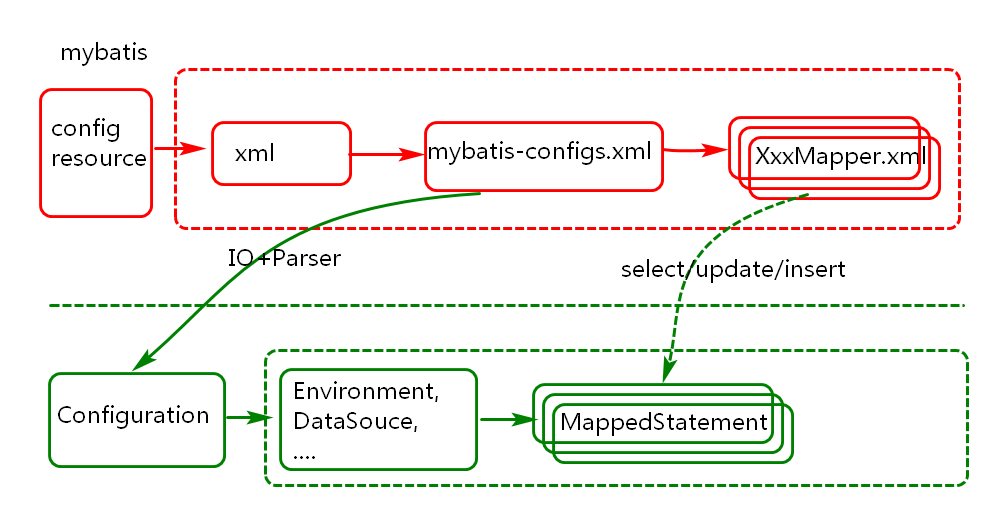
1. 会话功能 (SqlSession)
2. 会话语言 (SQL)
3. 用户体验 (池,日志)

### MyBatis 技术架构

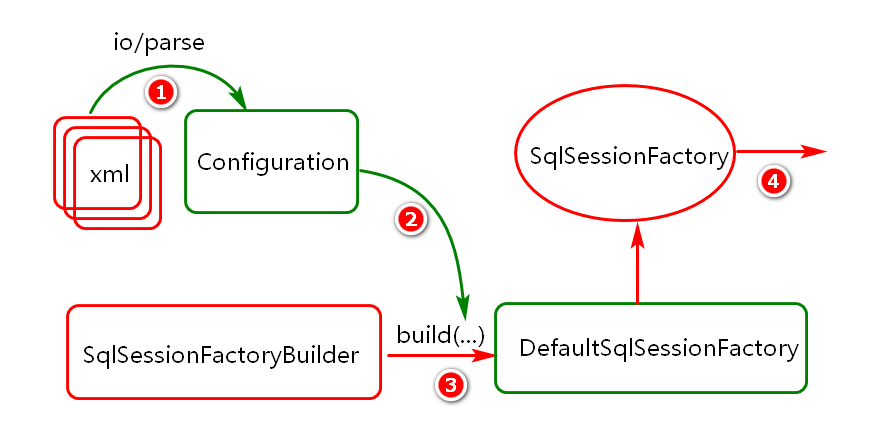
Mybatis 框架"构成"分析



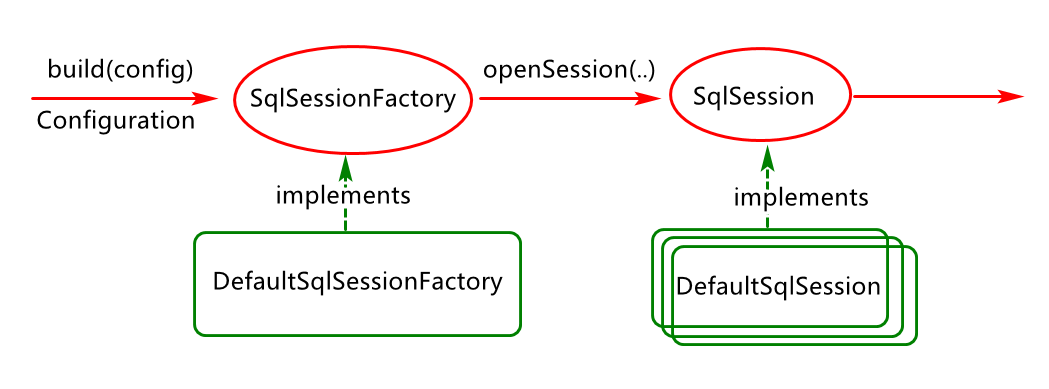
资源配置技术架构



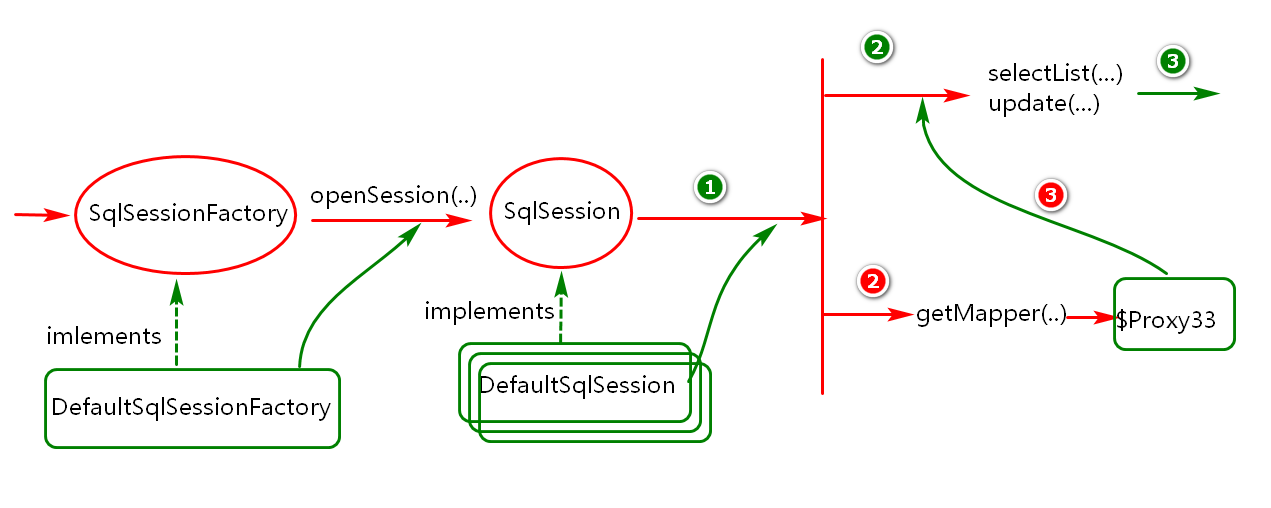
会话工厂对象创建(说说SqlSessionFactory对象的创建过程)



会话对象创建？(说说sqlsession对象创建过程)



会话对象应用方式？



## MYBATIS 快速实践(脱离文档)

### 初始化数据环境

假如此环境已有则无需执行:

1. 启动系统命令行控制台
2. 登陆数据库:mysql -u root -p
3. 设置客户端编码：set names utf8
4. 导入数据：source d:/jtsys.sql

说明：查询时，假如有中文要显示，可先执行set names gbk.

### 创建并配置项目

1. 创建maven项目并添加依赖

项目名称:CGB-MYBATIS-V1.01

打包方式:jar包

组id:com.company

MySQL驱动依赖(假如驱动版本与当前数据库不一致可能会有问题)

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.40</version>

</dependency>

Mybatis 框架依赖（参考官方 mybatis.org/mybatis-3）

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.6</version>

</dependency>

Junit单元测试依赖

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

说明:学了spring boot以后还有一种测试方式

1. 配置项目

在resources目录下创建mybatis-configs.xml配置文件如下：

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

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<!-- mybatis 核心配置 -->

<configuration>

<!-- 配置初始化环境(连接) -->

<environments default=*"development"*>

<environment id=*"development"*>

<transactionManager type=*"JDBC"*/>

<!-- 使用mybatis自带连接池 -->

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"url"* value=*"jdbc:mysql:///jtsys"*/>

<property name=*"username"* value=*"root"*/>

<property name=*"password"* value=*"root"*/>

</dataSource>

</environment>

</environments>

</configuration>

其模板：参考官方配置实现（Getting start ）

1. 项目环境测试

在测试包中，创建测试类，测试是否可以获取与数据库的连接

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

/\*\*

\* 借助此对象创建SqlSession(通过此对象

\* 实现与数据库之间的会话)

\*/

**protected** SqlSessionFactory factory;

/\*\*

\* 此方会在@Test注解修饰的方法之前执行,

\* 通常用于做一些初始化操作(方法名自己定义)

\*/

@Before

**public** **void** init()**throws** IOException{

InputStream in=Resources.*getResourceAsStream*("mybatis-configs.xml");

factory=**new** SqlSessionFactoryBuilder().build(in);

//系统底层建造者模式构建工厂对象(此对象构建过程相对复杂)

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

}

@Test

**public** **void** testSqlSessionConnection(){

SqlSession session=factory.openSession();

Connection conn=session.getConnection();

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

}

}

### 业务应用快速实践

参考官网：[www.mybatis.org/mybatis-3](http://www.mybatis.org/mybatis-3)

业务实现：基于xml方式实现相关业务实现.

1. 准备工作
2. 创建SysLogMapper.xml文件并添加到src/main/resources的mapper目录(目录不存在则创建)
3. 然后需要在mybatis-configs.xml中进行配置
4. 基于id删除数据库中日志数据
5. 在SysLogMapper.xml文件中添加删除映射元素
6. 创建单元测试类,对删除业务进行测试.
7. 执行分页查询日志信息（两种方式）
8. 在SysLogMapper.xml文件中添加查询元素
9. 创建单元测试类或方法,对查询操作进行测试

自己参考课上代码:参考cgb-mybatis-v1.01 项目，进行多次实现

重构业务：脱离xml文件配置

* + - 1. 定义用户业务接口,参考如下写法:

**package** com.mybatis.dao;

**import** java.util.Map;

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

**public** **interface** SysUserDao {

/\*\*基于id执行查询操作\*/

@Select("select \* from sys\_users where id=#{id}")

Map<String,Object> findObjectById(Integer id);

}

* + - 1. 定义测试基类TestBase02,脱离配置文件创建SqlSessionFactory对象

**public** **class** TestBase02 {

**protected** SqlSessionFactory factory;

@Before

**public** **void** init() {//参考官网

//1.构建数据源对象

DataSource dataSource=newDataSource();

//2.创建事务工厂

TransactionFactory transactionFactory=

**new** JdbcTransactionFactory();

//3.创建持久化环境对象

Environment en=**new** Environment(

"development",transactionFactory,dataSource);

//4.构建Configuration对象

Configuration cfg=**new** Configuration(en);

//5.添加映射文件

cfg.addMapper(SysUserDao.**class**);

//6.构建SqlSessionFactoy对象

factory = **new** SqlSessionFactoryBuilder()

.build(cfg);

}

/\*\*创建连接池对象\*/

**private** DataSource newDataSource() {

PooledDataSource ds=

**new** PooledDataSource();

ds.setDriver("com.mysql.jdbc.Driver");

ds.setUrl("jdbc:mysql:///jtsys");

ds.setUsername("root");

ds.setPassword("root");

**return** ds;

}

}

* + - 1. 定义业务测试类,基于用户id查询用户信息.

**package** com.mybatis.test;

**import** java.util.Map;

**import** org.apache.ibatis.session.SqlSession;

**import** org.junit.Test;

**import** com.mybatis.dao.SysUserDao;

**public** **class** TestSysUser02 **extends** TestBase02{

@Test

**public** **void** testFindObjectById() {

//1.创建sqlSession对象

SqlSession session=factory.openSession();

//2.执行查询操作

SysUserDao dao=session.getMapper(SysUserDao.**class**);

Map<String,Object> map=dao.findObjectById(16);

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

//3.释放资源

session.close();

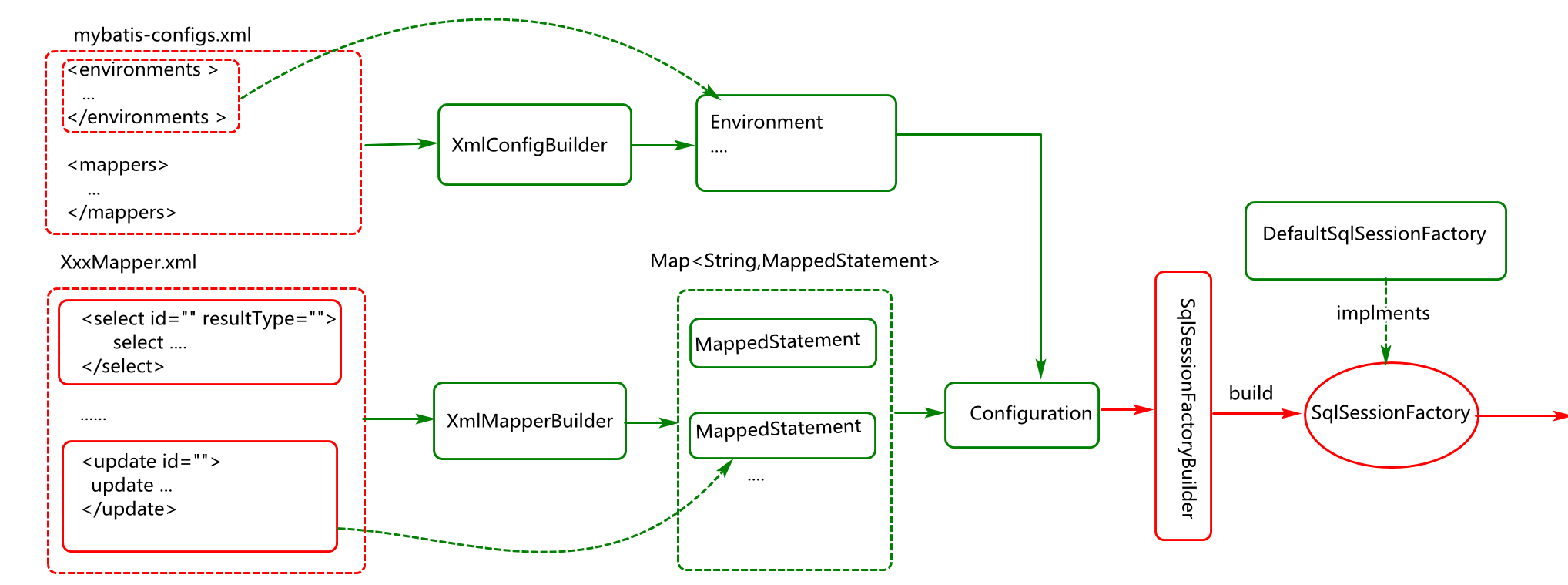
}

}

## MYBATIS 应用原理进阶分析

### 会话工厂创建分析(了解)

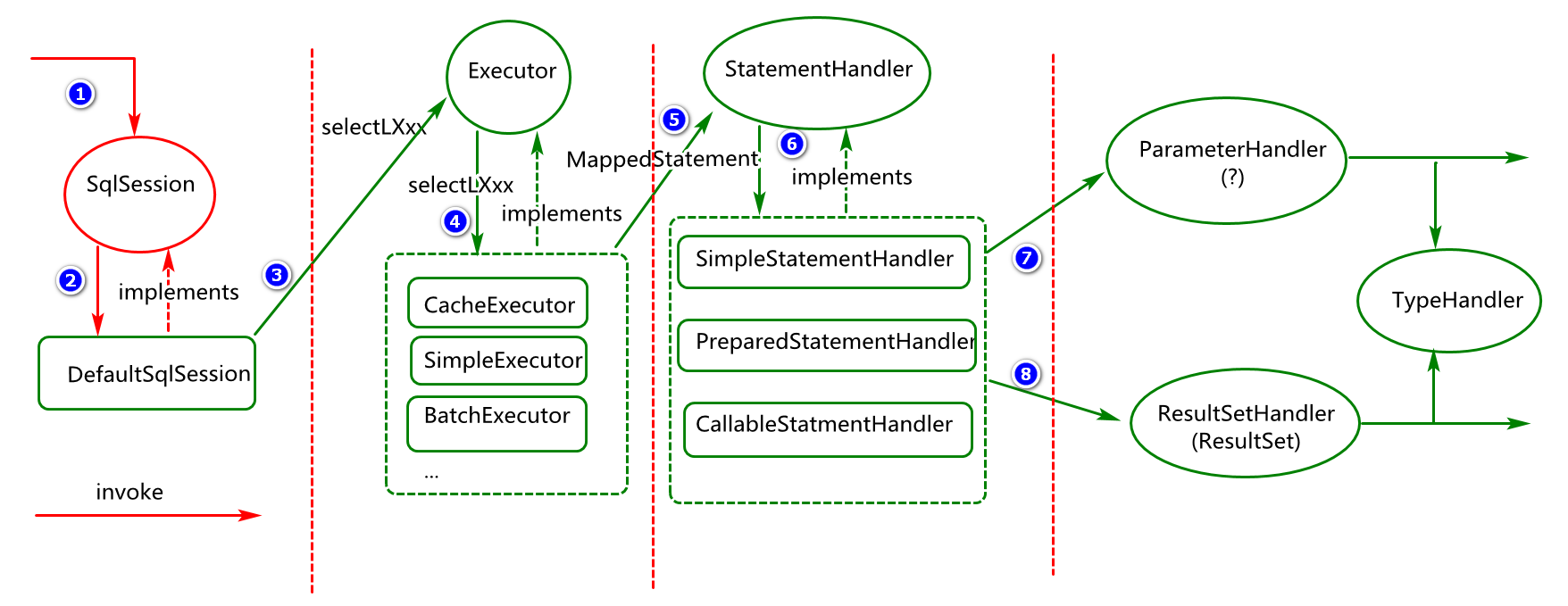
其创建过程分析



* + - 1. 通过IO读取配置文件
      2. 解析IO数据并进行封装(所有信息都会存储到Configuration对象)
      3. 基于配置对象创建SqlSessionFactory对象

### 会话对象应用分析 (了解)

说说mybatis对jdbc的封装过程（了解）？（SqlSession应用增强分析）



1)封装了连接获取过程？（Executor->JDBCTransaction🡪Connection）

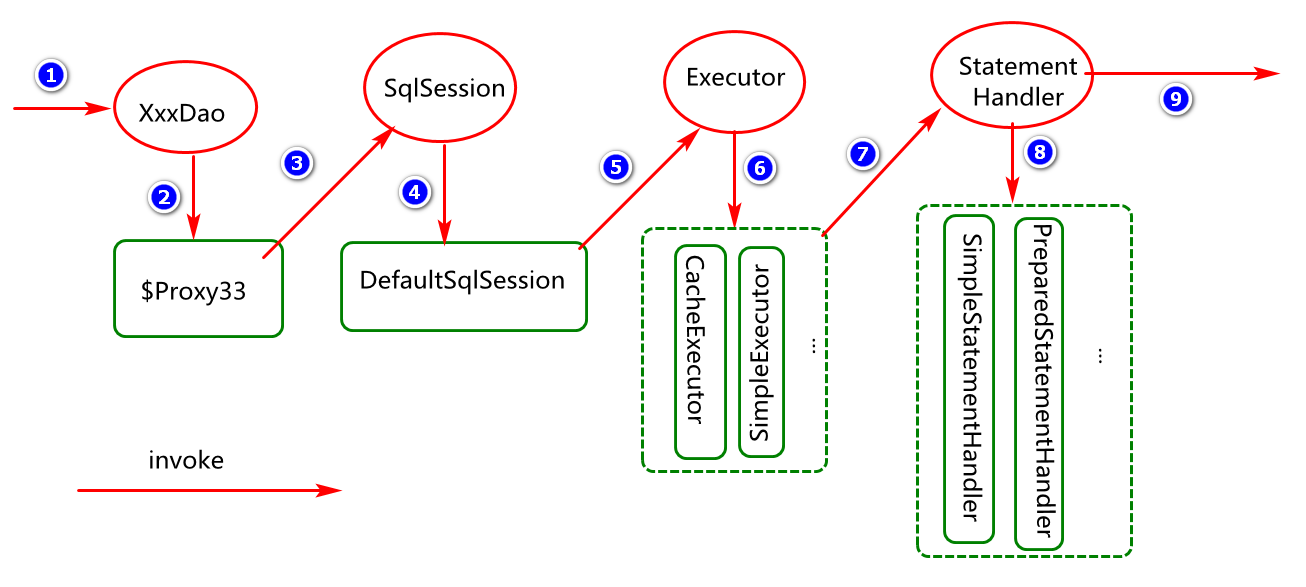
2)封装了Statement对象创建过程？(connection)

3)封装了sql的发送过程，参数的处理过程，结果的映射过程。

### 基于Mapper接口会话(了解)

例如：XxxMapper xm=Session.getMapper(XxxDao.class);

当获取到mapper接口对应的实现类对象以后可以基于实现类底层执行sql操作。

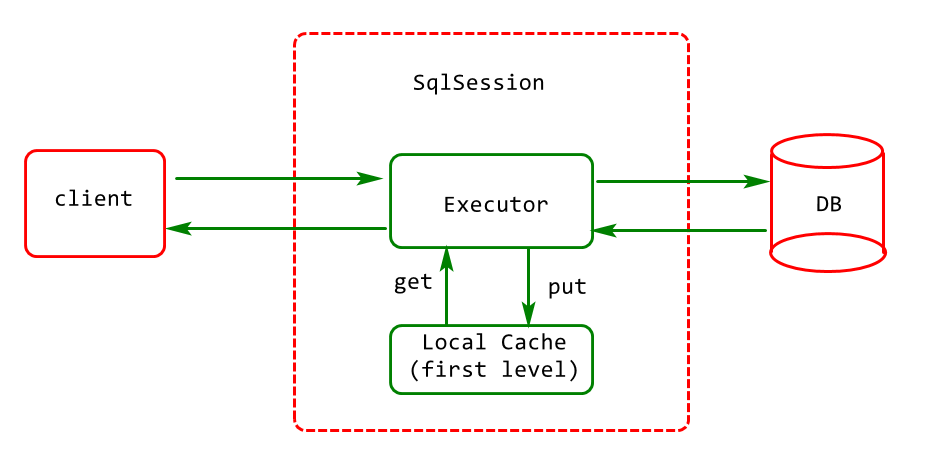


### 缓存应用实现过程分析

Mybatis 框架提供一种缓存机制，通过缓存的应用来提高查询性能,但可能会有一定的不一致(脏读)问题。Mybatis中的缓存提供一级缓存和二级缓存的实现,默认都是开启状态（可参考官网）。

1.Mybatis 一级缓存

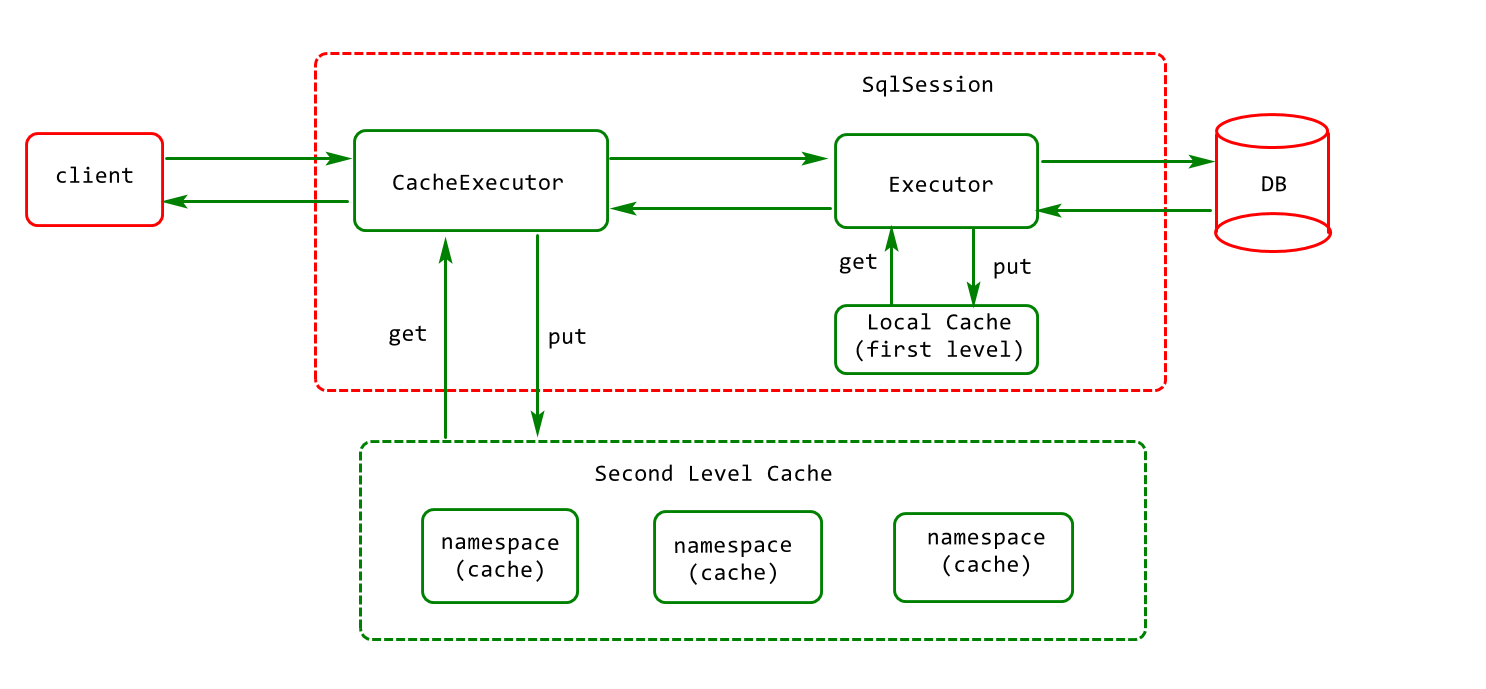
Mybatis中的一级缓存有时又称为SqlSession级缓存，SqlSession关闭时一级缓存失效。在同一个SqlSession内部多次执行同一个查询，后续的查询会从此缓存取数据，其查询架构如下图所示：



说明：一级缓存的实现可查看BaseExecutor类中的localCache属性。

2.MyBatis二级缓存

Mybatis中的二级缓存有时又称跨session缓存，可在多个sqlsession间共享数据,假如要使用二级缓存，可在对应的mapper文件中借助cache元素进行配置(可参考官方映射文件配置)。



# SPRING 框架基础进阶

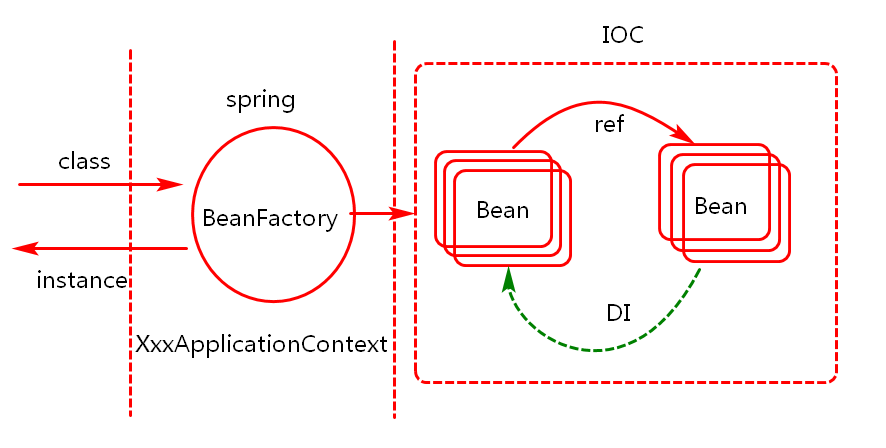
## SPRING 框架架构分析

### Spring 框架应用架构

Spring 官网资源:spring.io/projects

Spring 是一个资源整合框架,通过spring可将很多资源(例如连接池,

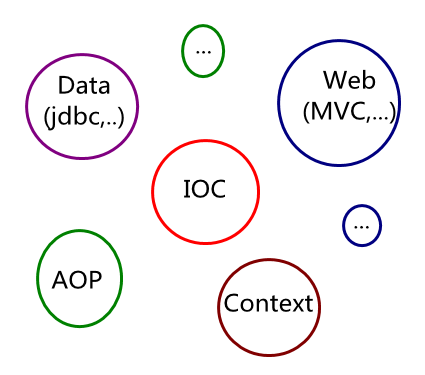
mybatis,...)等整合在一起,对外提供相关服务(例如,秒杀服务,支付服务,...)。



说明:spring 框架一切资源的整合都源于IOC模块，IOC要实现对象生命周期的管理，对象依赖关系的管理。

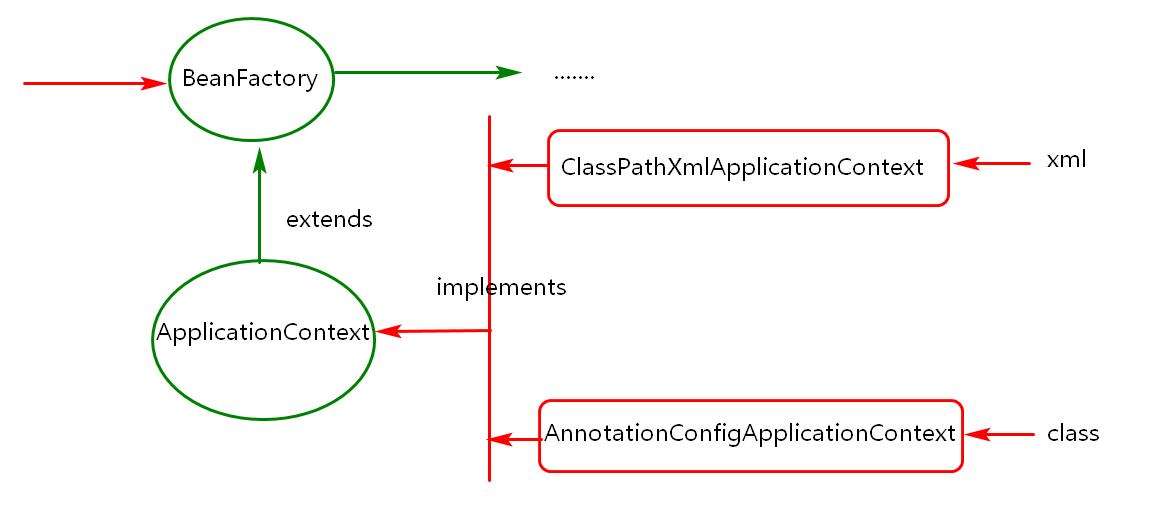
### Spring 框架产品架构

产品架构主要从这个框架对外提供的服务（功能）进行理解.

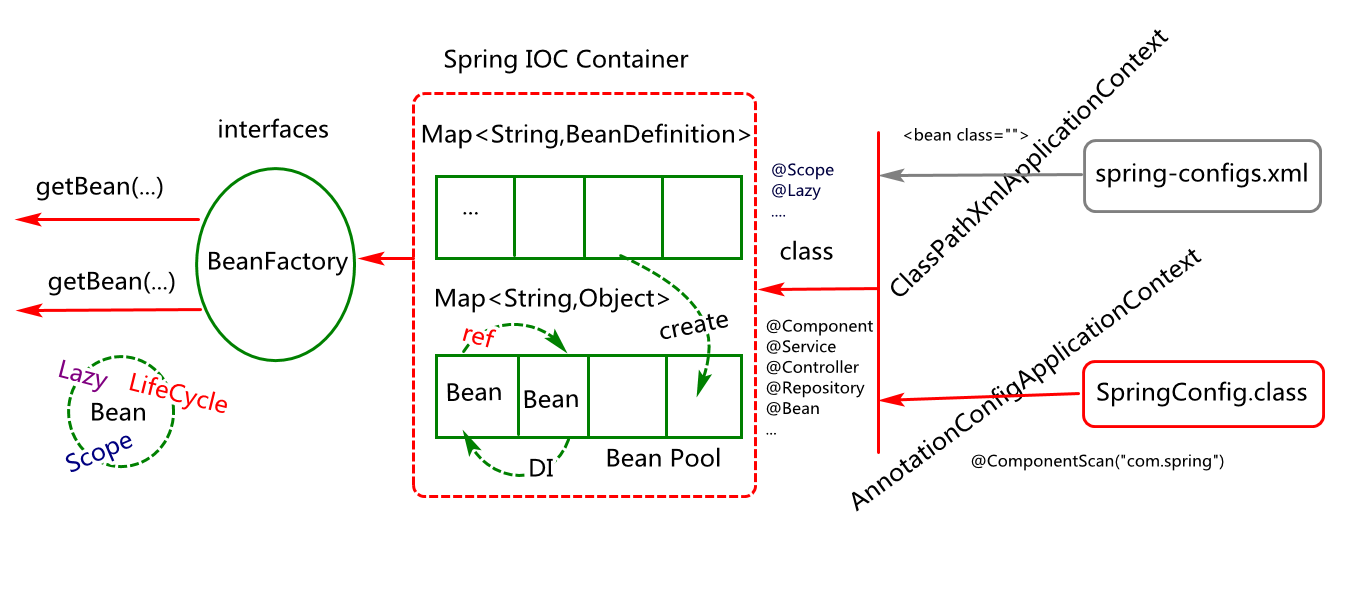


### Spring 框架技术架构

IOC API基础架构



IOC 容器初始化过程分析



## SPRING IOC 快速实践 (注解方式-脱离文档)

业务描述,创建一个DefaultCache对象然后将此对象交给Spring容器管理.

### 创建并配置项目

1. 创建maven项目(jar包项目)
2. 项目名称 CGB-SPRING-V1.01
3. 组id:com.company
4. 打包方式:jar
5. 添加项目依赖(spring-context)

添加spring依赖

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.9.RELEASE</version>

</dependency>

添加junit依赖

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

添加spring配置类(SpringConfig)

**package** com.spring;

**import** org.springframework.context.annotation.ComponentScan;

/\*\*

\* **@ComponentScan** 用于告诉spring容器从

\* 从指定包进行bean的扫描

\*/

@ComponentScan("com.spring.beans")

**public** **class** SpringConfig {//spring-configs.xml

}

定义测试基类

**package** com.spring;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

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

**protected** AnnotationConfigApplicationContext ctx;

@Before

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

ctx=**new** AnnotationConfigApplicationContext(

SpringConfig.**class**);

}

@After

**public** **void** close() {

ctx.close();

}

@Test

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

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

}

}

### 基本业务实现

创建DefaultCache类,并明确此类交给spring管理.

**package** com.spring.beans;

**import** javax.annotation.PostConstruct;

**import** javax.annotation.PreDestroy;

**import** org.springframework.context.annotation.Lazy;

**import** org.springframework.context.annotation.Scope;

**import** org.springframework.stereotype.Component;

/\*\*

**@Component** 注解用于告诉spring容器

请将这个类交给spring管理.

**@Lazy** 用于告诉spring容器此对象要延迟加载

**@Scope** 用于告诉spring容器此bean的作用域

1)singleton (单例作用域-默认,会存储到池中)

2)prototype (多例作用域,每次获取都创建新对象)

\*/

@Lazy

@Scope("singleton")

@Component //@Controller,@Service,...

**public** **class** DefaultCache {

**public** DefaultCache() {

System.***out***.println("DefaultCache()");

}

@PostConstruct //告诉spring 此对象初始化时执行init方法

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

System.***out***.println("init()");

}

@PreDestroy//告诉spring 此对象销毁时执行close方法

**public** **void** close() {

System.***out***.println("close()");

}

}

定义测试类,从spring容器中获取bean对象

**package** com.spring;

**import** org.junit.Test;

**import** com.spring.beans.DefaultCache;

**public** **class** TestBean01 **extends** TestBase {

@Test

**public** **void** testDefaultCache() {

//从spring容器获取bean对象

DefaultCache cache01=ctx.getBean("defaultCache",DefaultCache.**class**);

DefaultCache cache02=ctx.getBean("defaultCache",DefaultCache.**class**);

System.***out***.println(cache01==cache02);

}

}

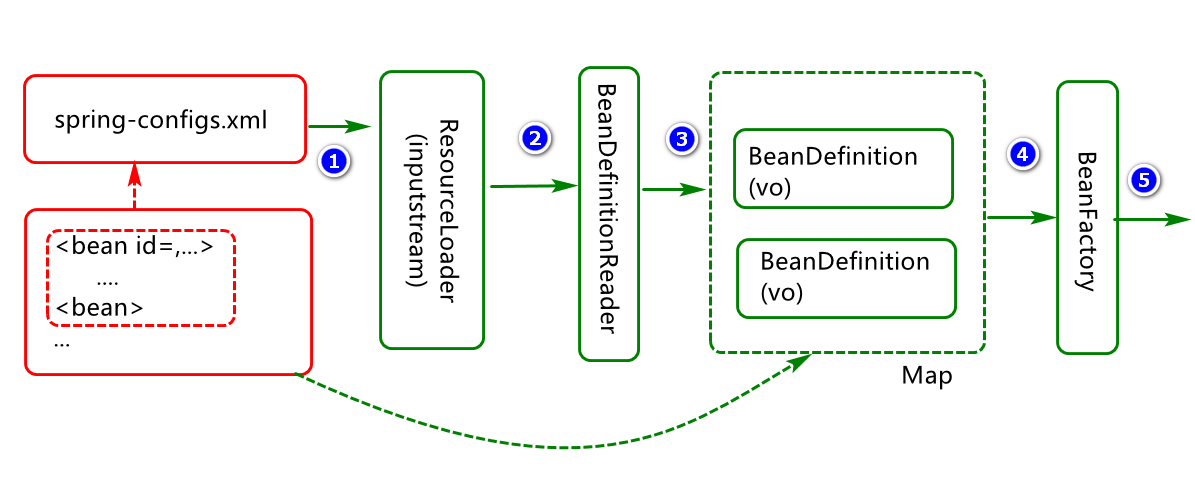
## SPRING IOC 应用原理分析

IOC 是一种设计思想，称之为控制反转。基于这种思想实现对象创建，对象的科学管理以及应用时的解耦(借助DI机制实现)。Spring框架核心就是基于这种机制进行了完美实现。

说明：

1. 控制反转探讨的是什么？谁控制谁的问题（spring控制对象的创建管理）
2. 生活中的IOC的实现？(例如股票操盘手,父母包办婚姻）

### Spring Bean 容器的初始化

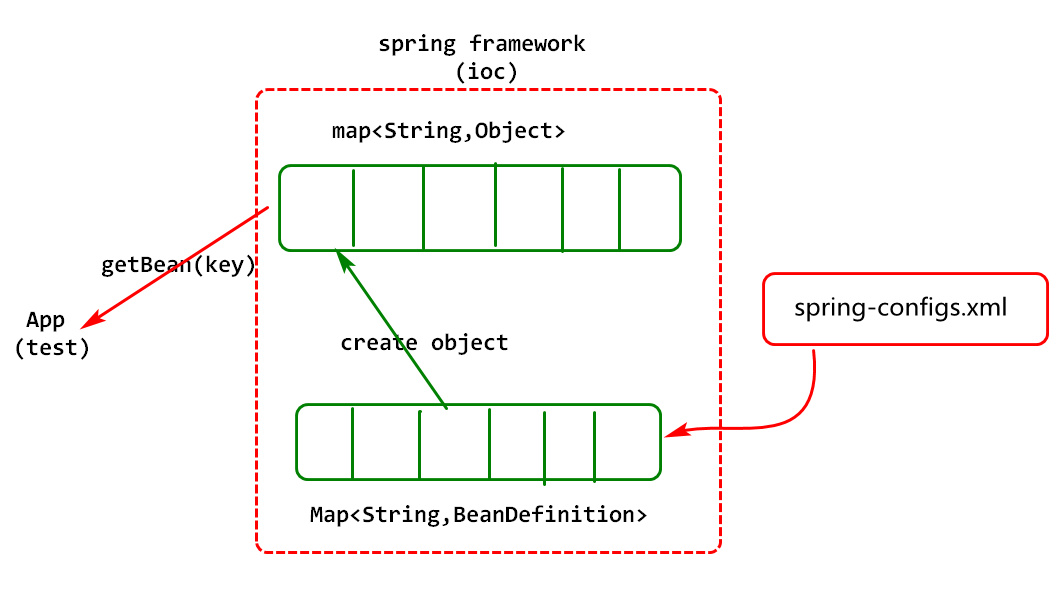


### Spring 中的两大map对象

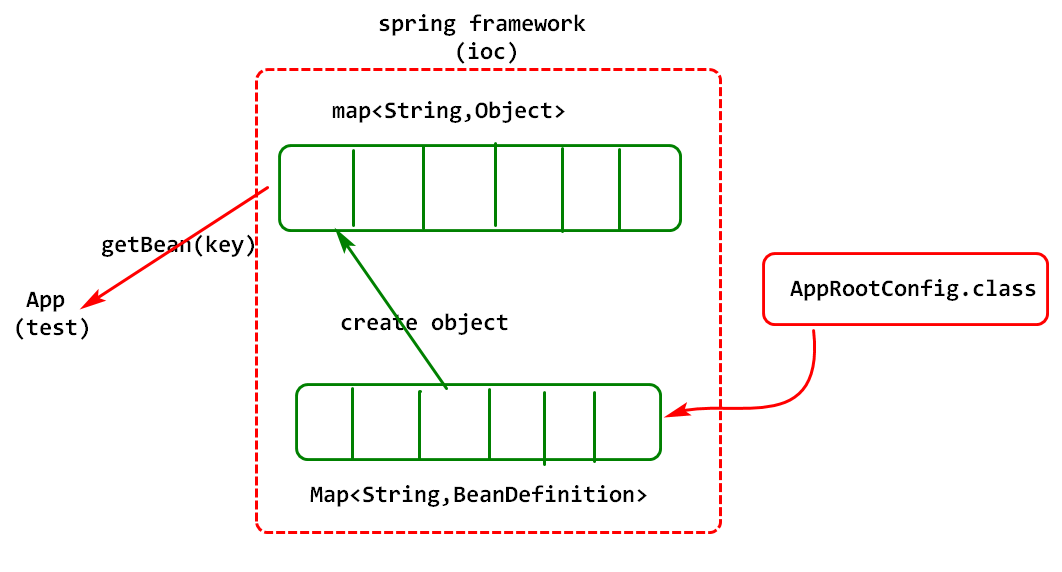
1.如何理解Spring中的两大map对象？

1. 一个map用于存储bean的配置信息(工厂的原材料)
2. 一个map用于存储bean的实例信息(工厂中的成品对象)

基于xml配置文件实现:



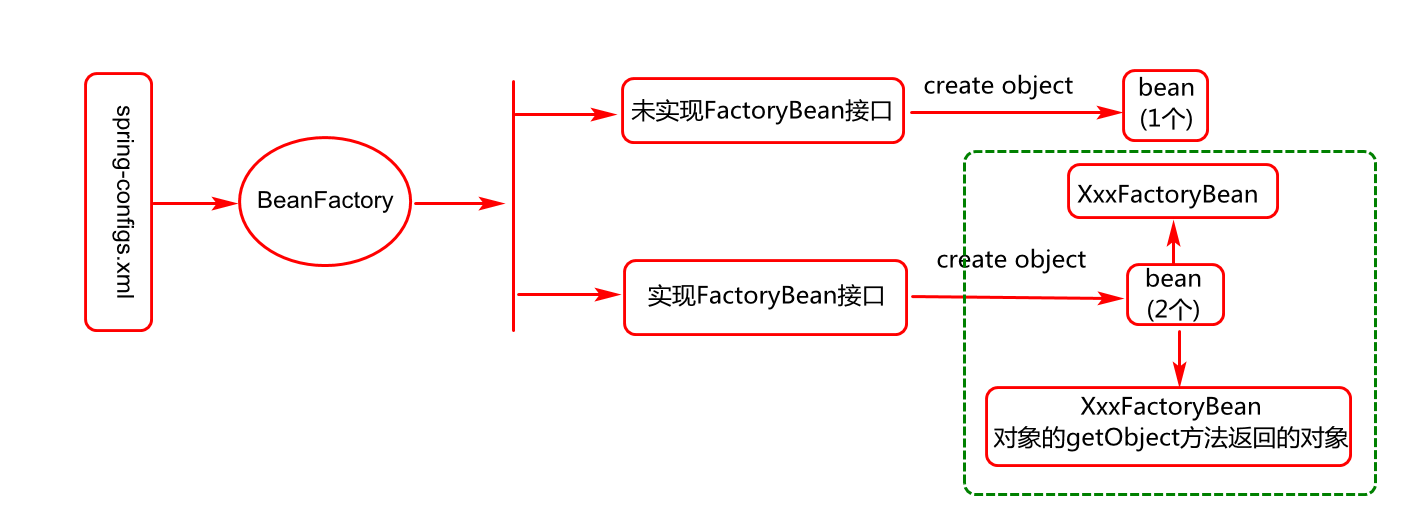
基于注解配置实现:



### Spring 中两大bean对象(了解)

Bean对象创建

1. 未实现FactoryBean接口(直接构造方法),
2. 实现FactoryBean接口（调用FactoryBean对象的getObject方法）



说明：一般在创建一些相对复杂的工厂对象时，通常会写一个工厂bean对象，

然后基于工厂bean对象创建具体的工厂对象，例如SqlSessionFactoryBean,

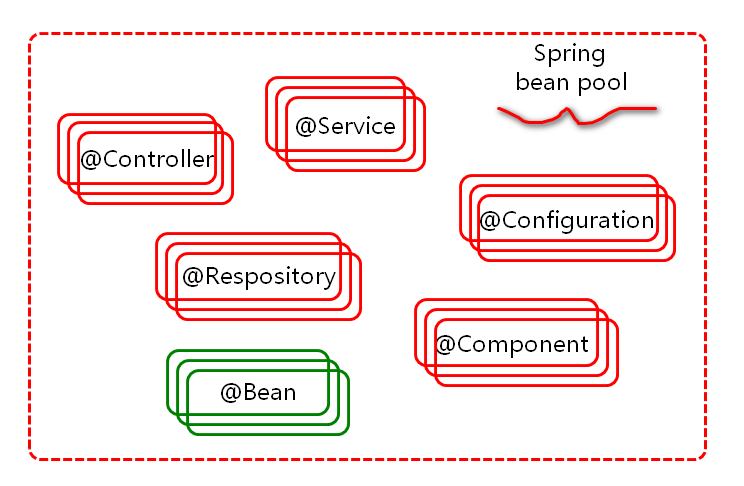
ShiroFilterFactoryBean，ProxyFactoryBean等。

### Spring 中两大bean对象描述方式

Bean 对象的描述

1. xml方式 (例如<bean id=”factory” class=”com.beans.Factory”>)
2. annotation方式（@Service,@Controller，@Configuration，@Bean，..）

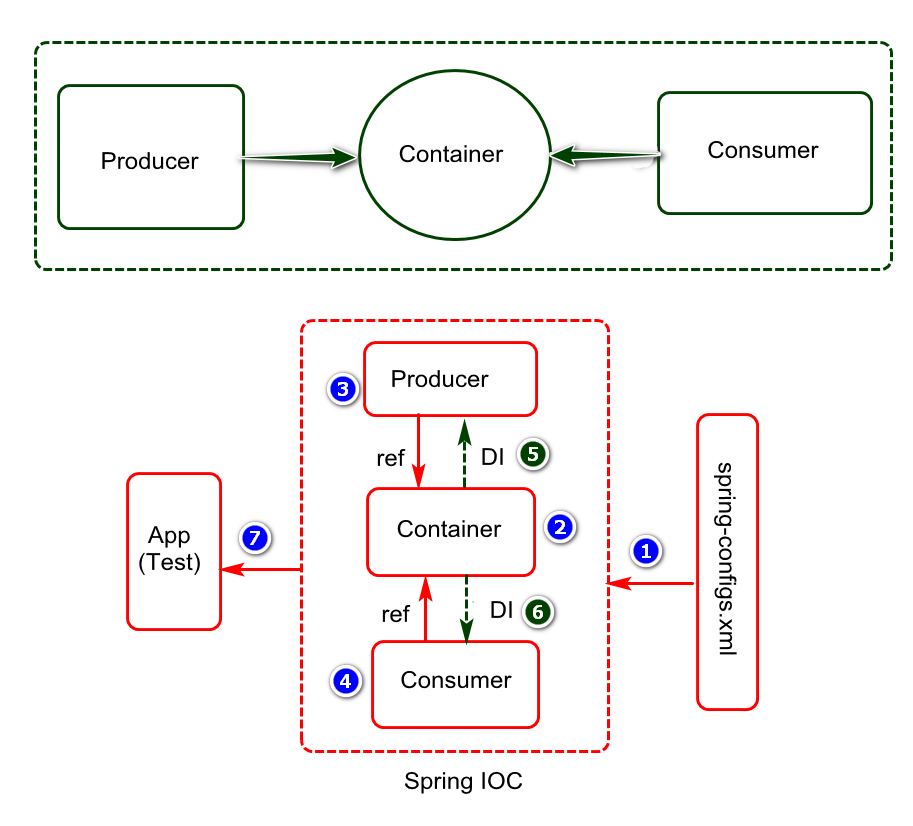
Spring 中用于描述这是一个Bean对象的相关注解如下:



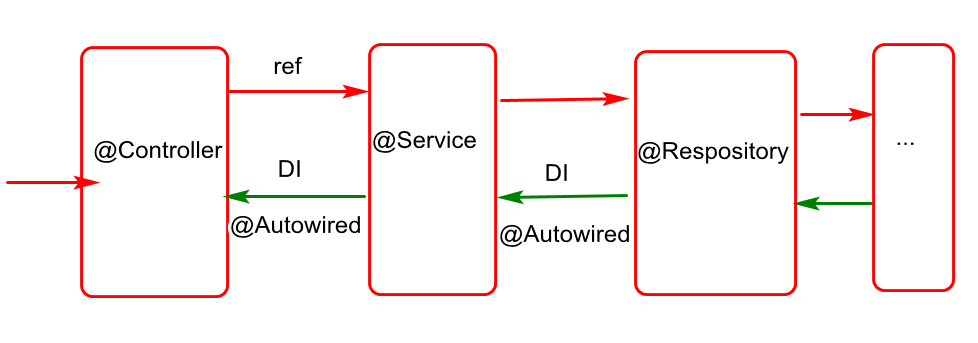
说明:无论使用如上图中的哪个注解对Bean进行描述,对Spring而言都认为是一样的Bean.

### Spring 中Bean对象的依赖注入

IOC 依赖注入(为对象属性赋值)基本概念

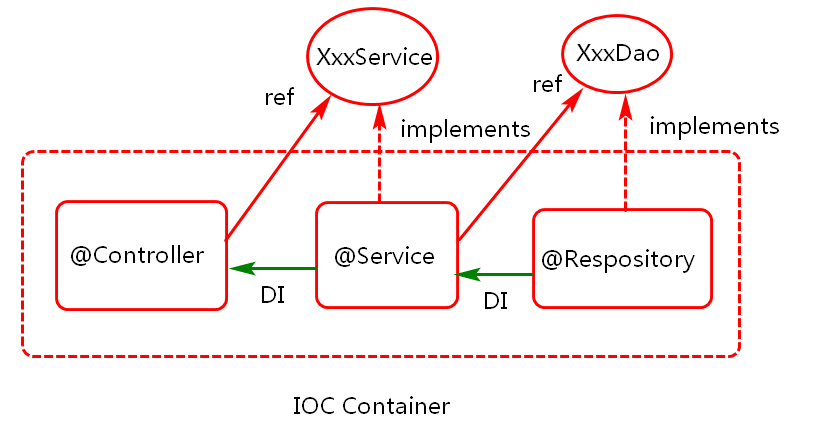


IOC 依赖注入在项目中的应用实现：



实际项目中为了解耦和,对象之间通常会通过接口进行通讯,也就是

说对象要耦合与接口,例如

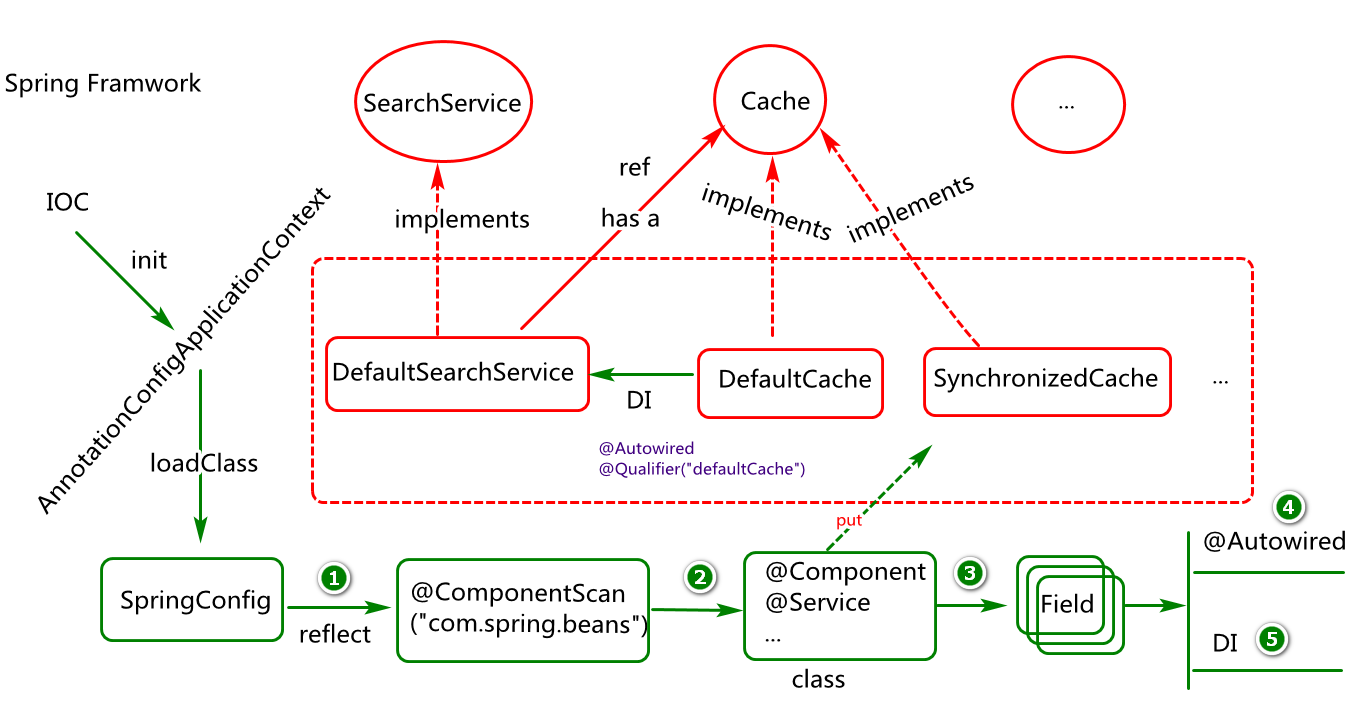


个人认为：IOC的核心是对象生命周期管理(资源管理)以及

依赖注入（资源协同）；

## SPRING IOC 实践进阶(课堂练习)

在3.2小节基础上,基于如下设计,进行代码实现:



具体代码参考: CGB-SPRING-V1.01

## SPRING MVC 模块核心

说说对MVC的理解？

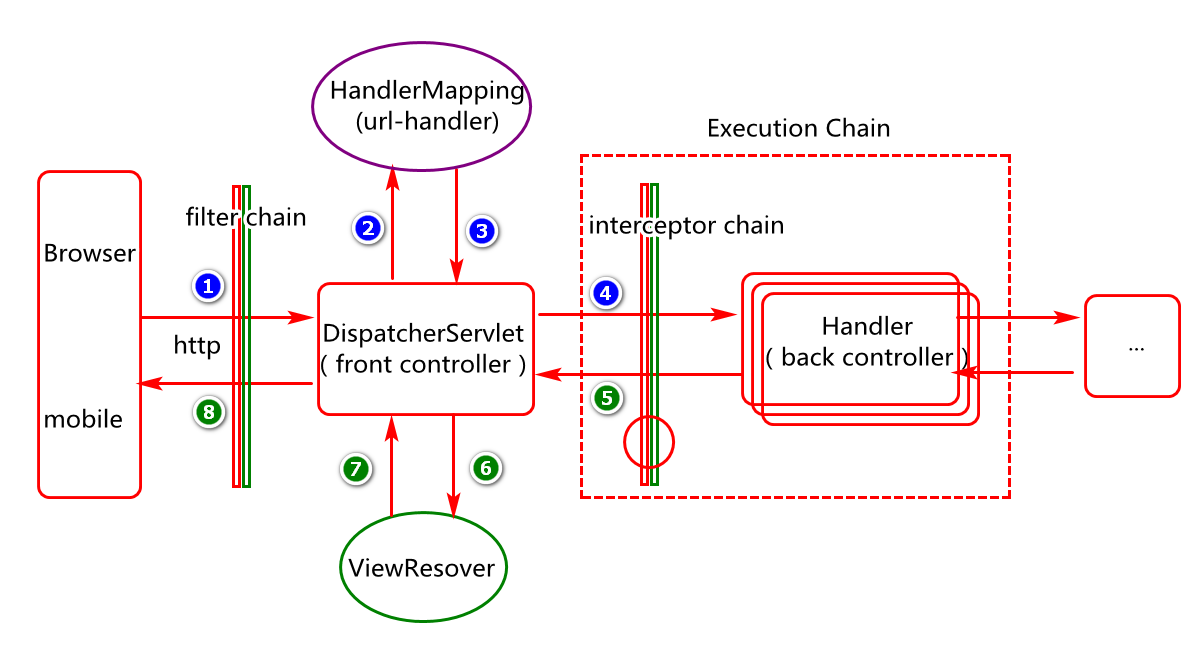
MVC是一种分层架构设计思想，目的是基于对象职责上的不同，进行分层设计，实现各司其职，各尽所能，以提高代码的可维护性，可扩展性。

1. 生活中的MVC:正规饭店(菜单,服务员,厨师)
2. 程序中的MVC:(html,jsp)/servlet/(service,dao))

Spring MVC中的MVC模块是基于MVC设计思想的一种完美实现。类似的还有struts2等。

### Spring MVC 核心架构

SPRING MVC模块中核心组件对象分析：



说说spring MVC中的核心组件？

### Spring MVC 快速入门实现(xml方式-脱离文档)

1. 创建maven项目
2. 项目名称: CGB-SPRINGMVC-V1.01
3. 组id: com.company
4. 打包方式: war包方式
5. 配置并初始化项目环境

1)生成web.xml

2)设置项目的运行时环境(选择tomcat)

3)设置项目编码方式 utf-8

4)设置统一编译环境 JDK8

5)添加项目依赖:spring-webmvc

6)添加spring mvc配置文件:spring-configs.xml

7)web.xml中配置spring mvc前端控制器(DispatcherServlet)

8)部署项目,启动tomcat测试 (假如tomcat正常启动,则没问题)

1. MVC基础业务实现
2. 定义Controller类
3. 包名:com.company.controller
4. 类名:SearchController
5. 创建Controller方法
6. 方法 public String doSearch(String key){..}
7. 配置controller类和方法.(基于注解方式)
8. 类上添加注解(@Controller,@RequestMapping(“/”))
9. 方法上添加注解(@RequestMapping(“doSearch”),@ResponseBody)
10. 配置spring-configs.xml文件
11. 配置包中类的扫描 :

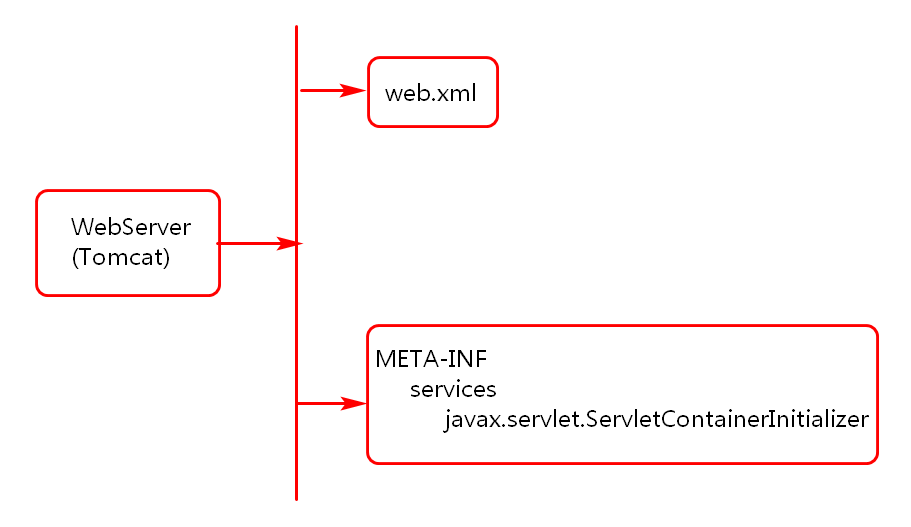
<context:component-scan base-package="com.company"/>

1. 配置MVC注解驱动:<mvc:annotation-driven/>
2. ...

说明：参考项目CGB-SPRINGMVC-V1.01代码

### Spring MVC 快速入门实现(注解方式-脱离文档-了解)

Tomcat 启动加载方式:



1. 创建maven项目
2. 项目名称: CGB-SPRINGMVC-V1.02
3. 组id: com.company
4. 打包方式: war包方式
5. 配置并初始化项目环境

1)配置maven war包插件(忽略web.xml)

2)设置项目的运行时环境(选择tomcat)

3)设置项目编码方式 utf-8

4)设置统一编译环境 JDK8

5)添加项目依赖:spring-webmvc

6)添加spring mvc配置类:SpringWebConfig类

7)创建AppWebApplicationInitializer类配置spring mvc。

8)部署项目,启动tomcat测试 (假如tomcat正常启动,则没问题)

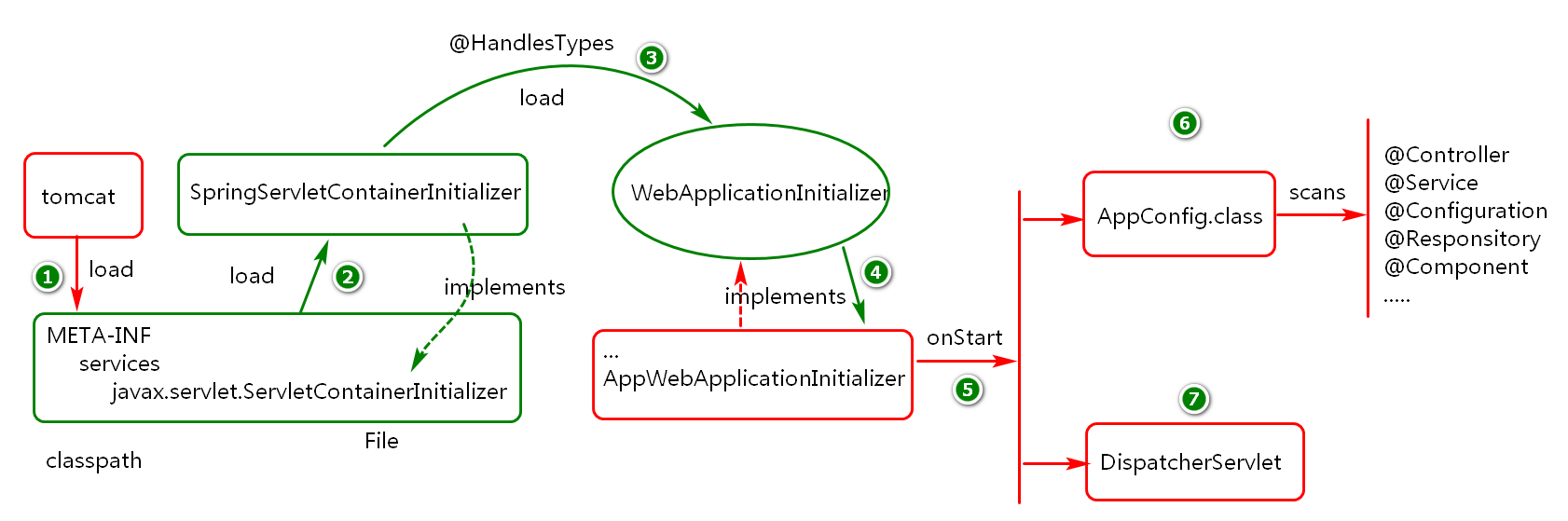
1. MVC基础业务实现
2. 定义Controller类
3. 包名:com.company.controller
4. 类名:SearchController
5. 创建Controller方法
6. 方法 public String doSearch(String key){..}
7. 配置controller类和方法.(基于注解方式)
8. 类上添加注解(@Controller,@RequestMapping(“/”))
9. 方法上添加注解(@RequestMapping(“doSearch”),@ResponseBody)
10. 配置SpringWebConfig类
11. 配置包中类的扫描 :

@ComponentScan("com.company.controller")

1. 配置MVC注解驱动: @EnableWebMvc
2. ...

说明：参考项目CGB-SPRINGMVC-V1.02代码

Spring mvc 注解方式应用分析:

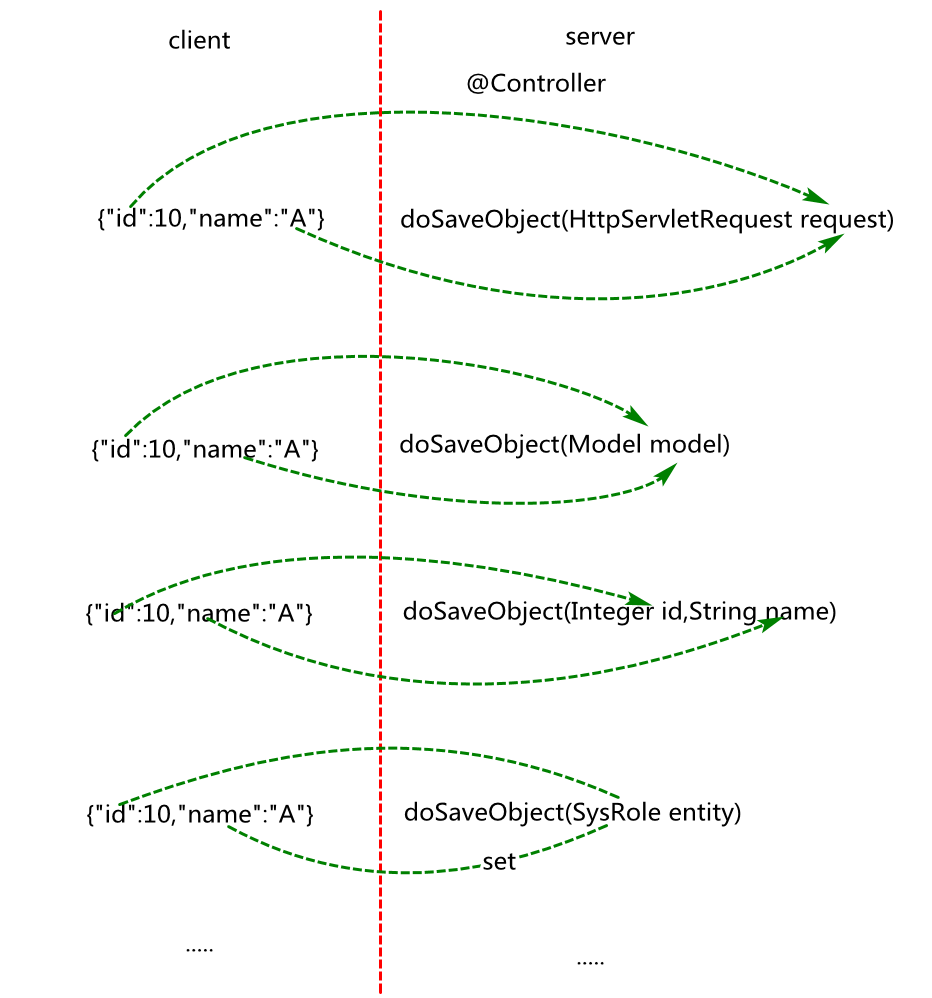


其中META-INF目录可在sping-web.jar中进行查看.

### Spring MVC 请求响应处理

所有MVC框架的重点都在请求和响应数据的处理上。

1. 请求处理
2. 请求路径(普通方式，rest方式) ：404
3. 请求方式(Get请求，Post请求):405
4. 请求参数(直接量，PO对象，MAP对象):400



1. 响应处理
2. 响应方式(转发，重定向redirect)
3. 响应数据封装(ModelAndView,Model,Map)
4. 响应数据转换（将对象序列化为JSON格式字符串）

# FRAMEWORK总结分析

## 重点难点

1. MyBatis基本架构及核心API应用
2. MyBatis 实现数据访问的两种基本方式
3. Spring 中的IOC模块设计原理及Bean对象管理
4. Spring 中MVC分层设计思想及核心组件作用.

## FAQ分析

1. 如何理解mybatis框架?
2. 如何理解spring框架?
3. 如何理解 @Autowired 这个注解？

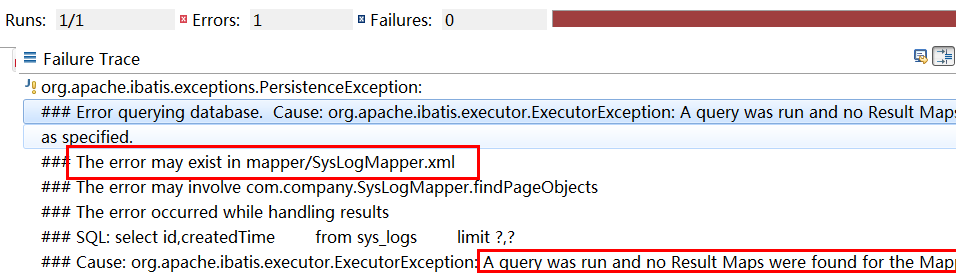
@Autowired注解用于修饰类中属性，构造方法，set方法，用于告诉spring框架请按照类型(属性类型，构造方法参数类型，set方法参数类型)为属性注入值。假如spring容器提供了多个相同类型对象，此时还可以按照名字（属性名，set方法参数名，构造方法参数名）查找匹配对象。

## 作业实现

1. 创建maven项目,基于mybatis实现对数据库数据的CRUD操作.(脱离文档)
2. 创建maven项目,基于Spring IOC实现对Bean对象的基本管理
3. 创建maven项目,基于Spring MVC 实现分层架构设计,以及请求响应处理.

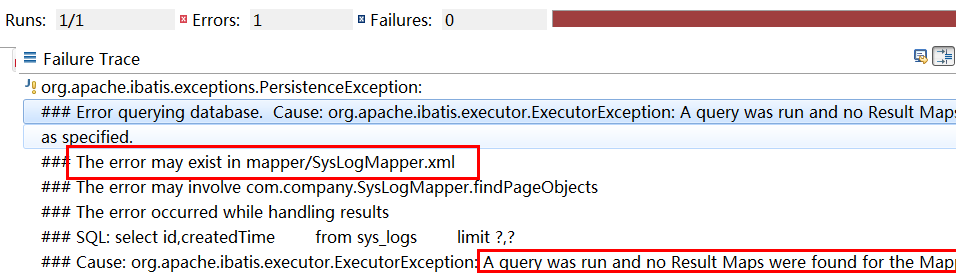
## BUG 分析

Bug-01:



问题分析：可能在SysLogMapper.xml文件中元素id为findPageObjects的元素没有写resultType。

Bug-02



Bug-03:

org.apache.ibatis.exceptions.PersistenceException:

### Error querying database.

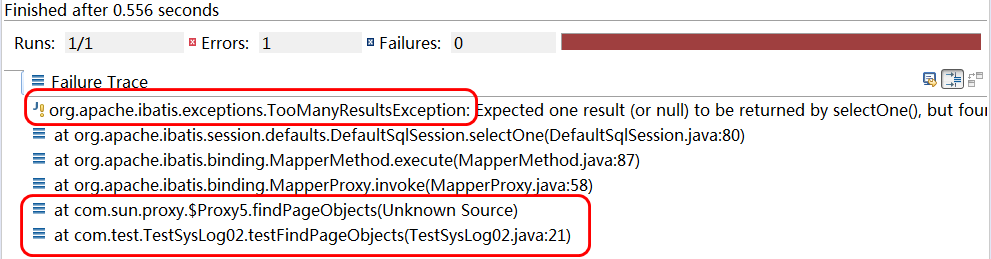
Cause: java.lang.IllegalArgumentException: Mapped Statements collection does not contain value for com.company.SysLogMapper.findPageObjects

问题分析：可能是你的查询statement “com.company.SysLogMapper.findPageObjects” \

与映射文件中定义的不致，可能的原因

1. 语句中的类全名与映射文件命名空间不一致。
2. 语句中的元素id与映射文件中的元素id不一致。
3. 映射文件没有添加到配置文件（mybatis-configs.xml）中。

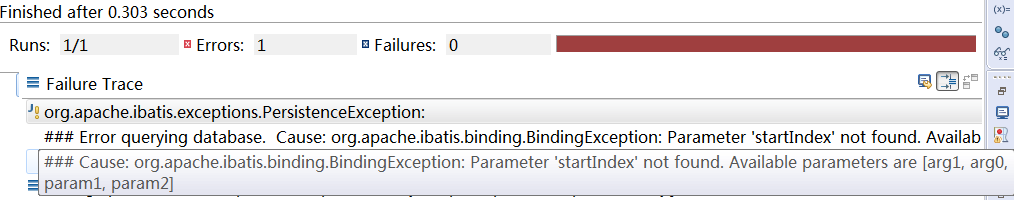
Bug-04



问题分析：查询出的结果是多行但是返回值对象只能处理一行。

Map findPageObjects()<--->select \* from sys\_logs

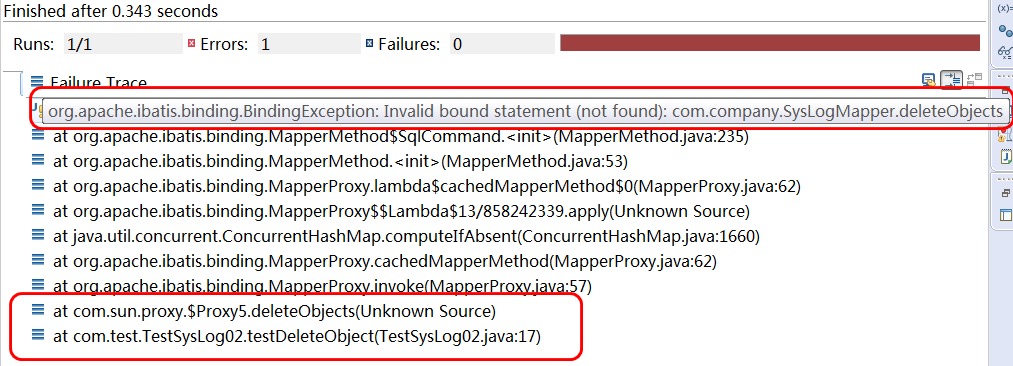
Bug-05



问题分析：

访问方式是基于接口方式，但是接口中的方法参数与映射文件中参数没有具体映射关系。

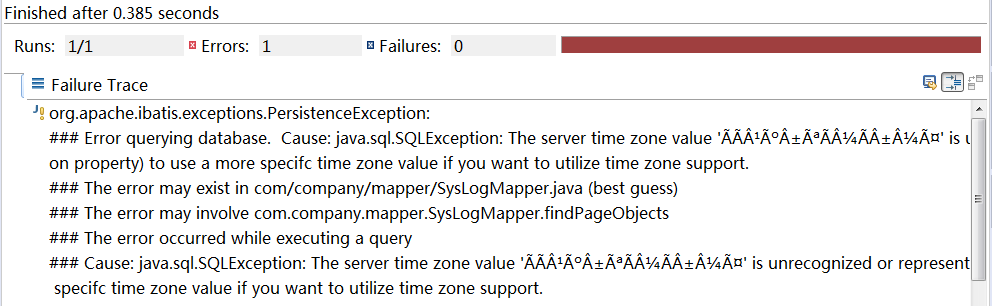
Bug-06



问题分析：基于接口访问方式

1. 接口名与映射文件命名空间(namespace)可能不同
2. 接口中的方法名与映射文件中的元素id不同。
3. 映射文件没有在配置文件(mybatis-configs.xml)进行注册

BUG-07

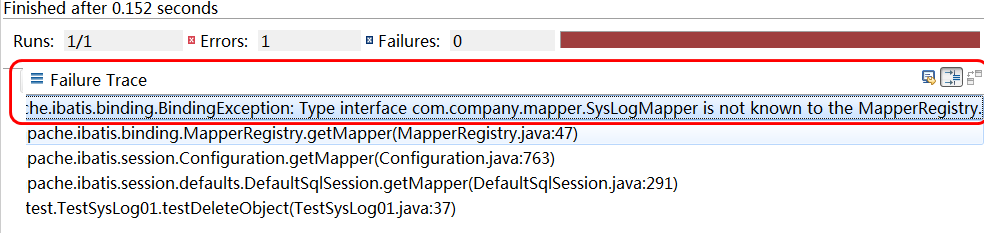


问题分析：mysql驱动程序与当前数据库版本可能存在兼容问题，解决方案

可在数据库的链接url中添加serverTimezone=GMT，例如：

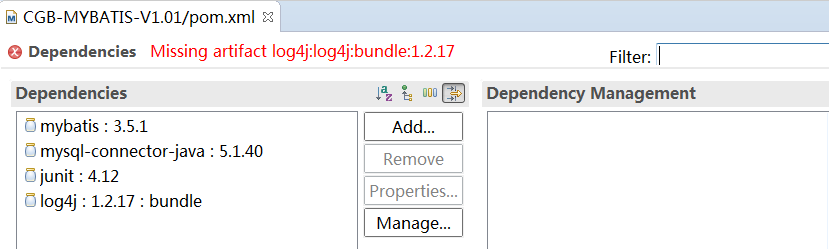
"jdbc:mysql:///jtsys?serverTimezone=GMT"

BUG-08

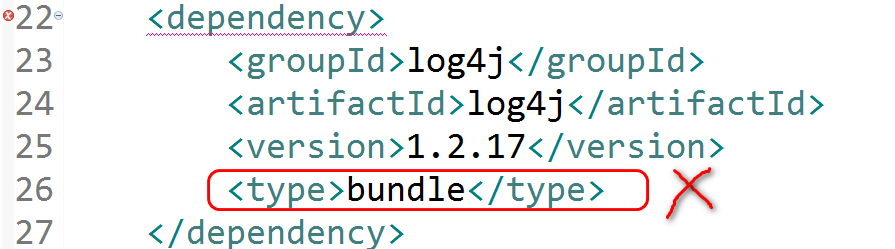


问题分析：可能映射文件或映射接口没有添加的配置文件或配置类中

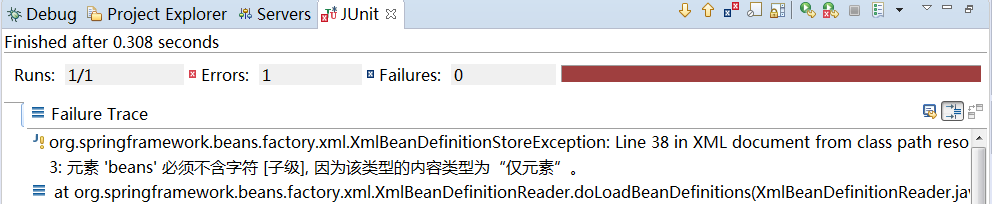
Bug-09



问题分析：在添加log4j依赖时自动，添加了bundle类型，去掉xml中的类型配置即可，如删除如下图中的红色框部分。

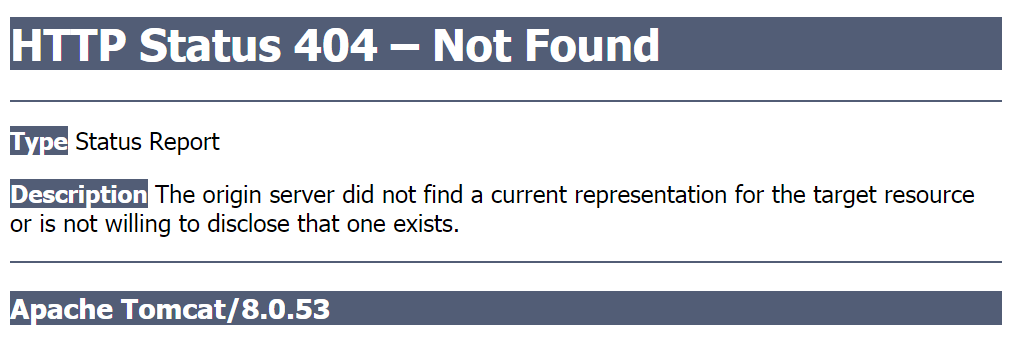


Bug-10



问题分析：应该是配置文件内部不小心多了一些特殊字符，并且写在了标签外面，例如分号，引号等

Bug-11 404



问题分析：404问题一般表示请求资源不存在，解决方案如下：

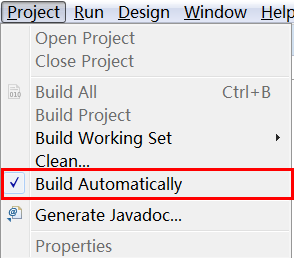
* + - 1. 确定tomcat服务启动ok
      2. 确定浏览器中的url资源访问路径是正确的。
      3. 确认要访问的资源配置是正确的(@Controller,包扫描)
      4. 确定部署目中有你对应的资源(资源不存在可能是编译有问题)
      5. 确定你的项目没有重命名。(重命名的项目可使用命名前方式访问)

如何确定你要访问的资源是否是存在的？查看tomcat启动日志？

-Dwtp.deploy=D:\CGB1902WORKSP\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps

资源不存在可能是项目没有正常编译，如何解决编译问题呢？

1)确保eclipse 中选中的自动编译



2）重新发布和部署项目，检测是否能够解决问题

3）尝试四大clean操作(先停止tomcat，从eclipse中移除项目)

3.1)clean tomcat work目录

3.2)clean tomcat 的部署文件

3.3）maven clean 删除原先编译的class

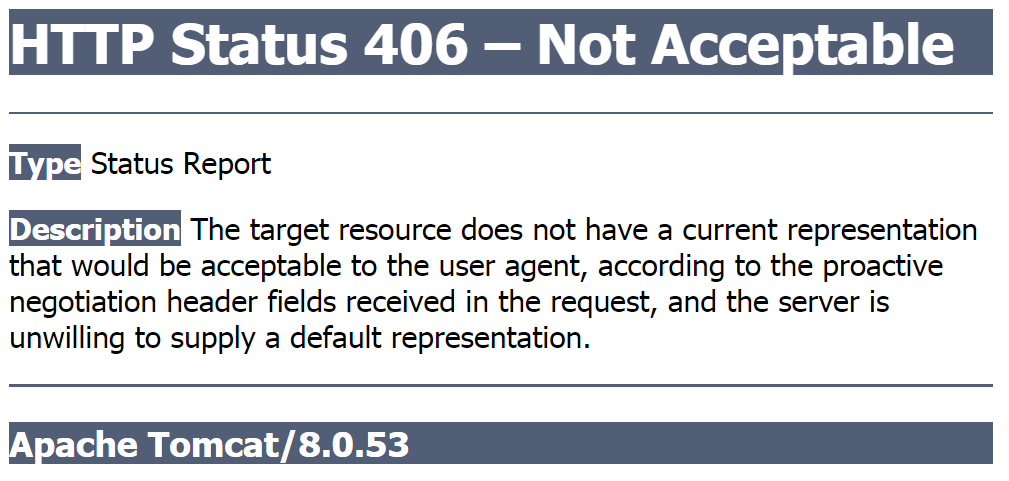
3.4）project clean 重新编译项目代码

BUG-12

Caused by: org.apache.catalina.LifecycleException: A child container failed during start at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:953

问题分析：web.xml中配置错误(例如servlet单词写错了)

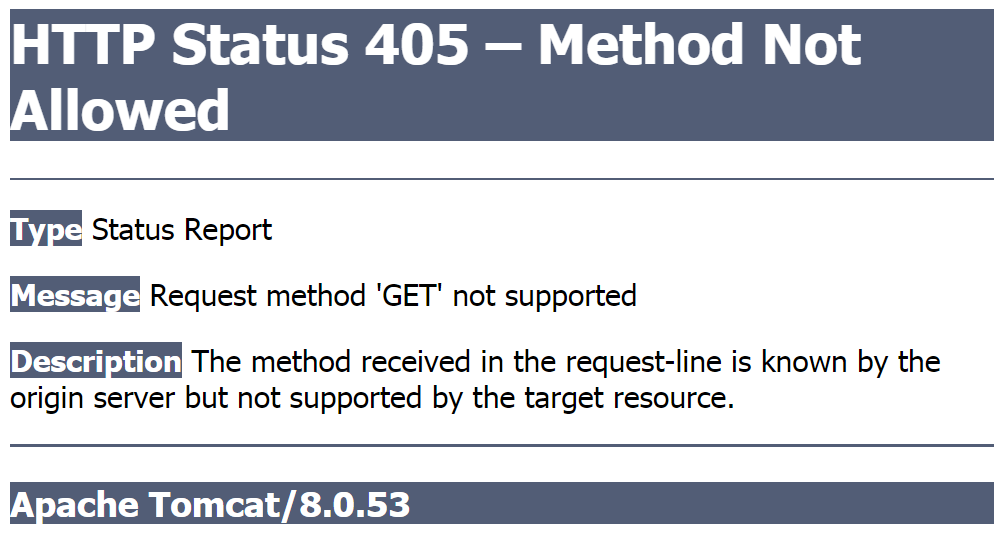
Bug-13



问题分析：spring 响应数据客户端无法处理

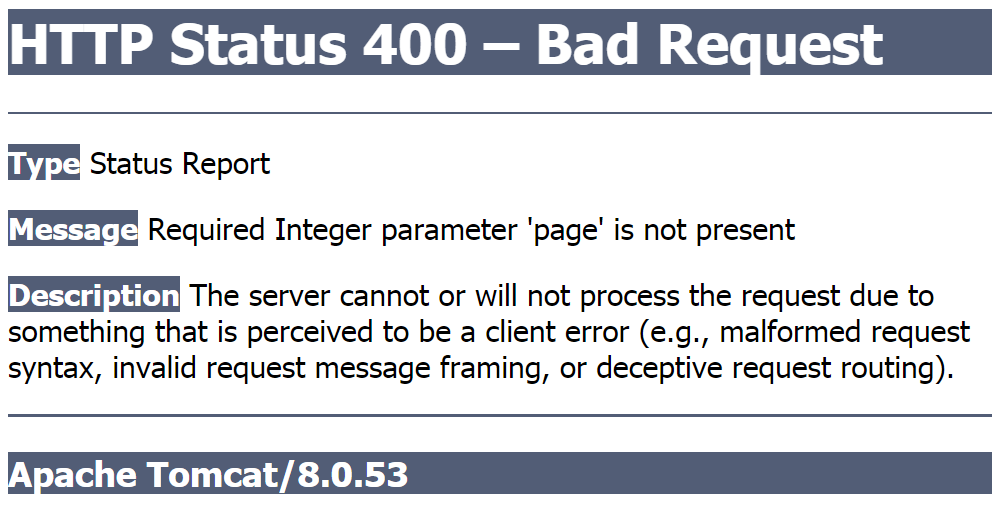
解决方案：检测spring 配置（例如jackson依赖，默认bean的配置等）

Bug-14



问题分析：请求方式与服务端的处理方式不匹配。

Bug-15



问题分析：客户端请求参数与服务端控制层方法参数不匹配(

参数个数，参数格式)