Java高级工程师

项目实战(上)

Java高级工程师

spring boot及其自动装配原理

技术选型问题

我们用spring-boot来解决一个实际中的场景

技术选型问题

Spring-boot快速上手

什么是spring boot

Spring boot是把原来的spring主流框架进行了产品化,让开发人员可以快速使用spring 全家桶的能力

Spring家族中的主要成员

Spring
Spring mvc
Spring Jpa

Spring

托管项目框架中Java Bean的实例化与引用的问题

Spring-mvc

Spring 家族中一个优秀的mvc框架

什么是mvc框架

Model View Controller，即模型-视图-控制器

Model（模型）

是Web应用中用于处理数据逻辑的部分，包括Service层和Dao层；
Service层用于和数据库联动，放置业务逻辑代码，处理数据库的增删改查，
Dao层用于放各种接口，以备调用；

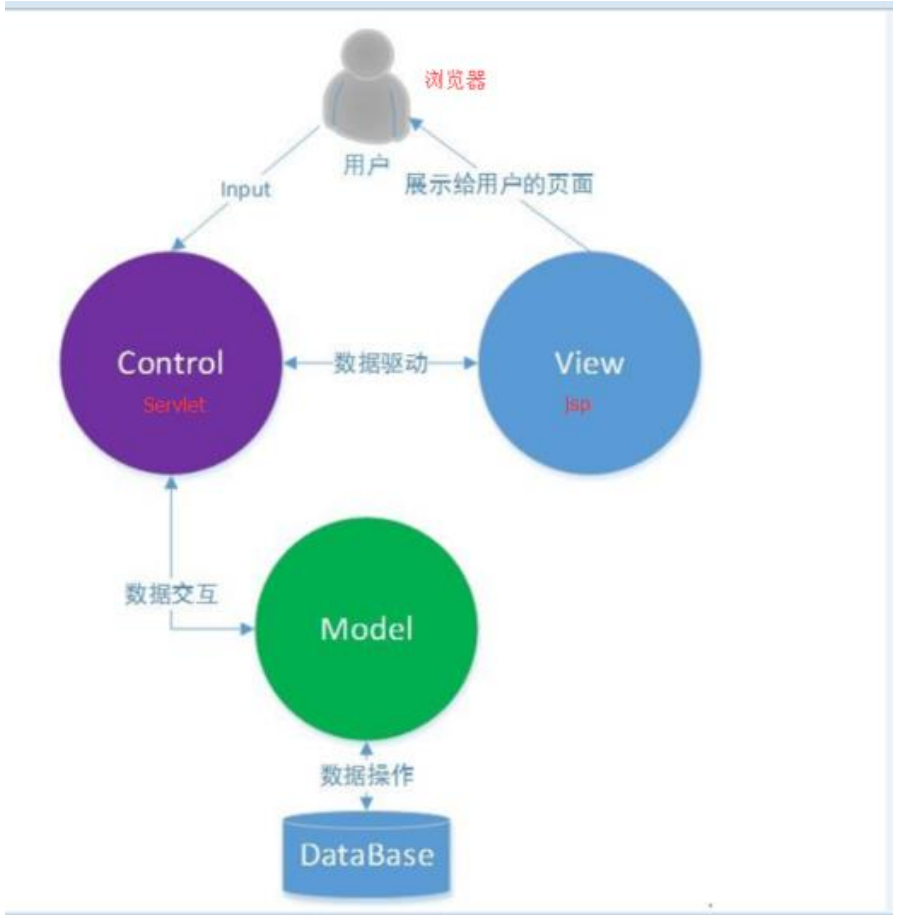
View（视图）

是Web应用中处理响应给客户的页面的部分，
例如我们写的html静态页面，jsp动态页面，这些最终响应给浏览器的页面都是视图；
通常视图是依据模型数据来创建的；

Controller（控制器）

在Web应用中，简而言之，就是Servlet，
SpringMVC框架中加了注解@Controller的方法
(实际上一个方法就相当于一个对应的Servlet)

简单一张图说明mvc中的交互关系



SpringData JPA

spring基于ORM框架、JPA规范的基础上封装的一套JPA应用框架，可以使开发者使用极简的代码实现对数据库的访问和操作。它提供了包括增删改查等在内的基本功能，且易于扩展。

面试题

你知道spring data jpa和其他orm框架之间有什么关系吗?
频度:中
难度:低
通过率:低

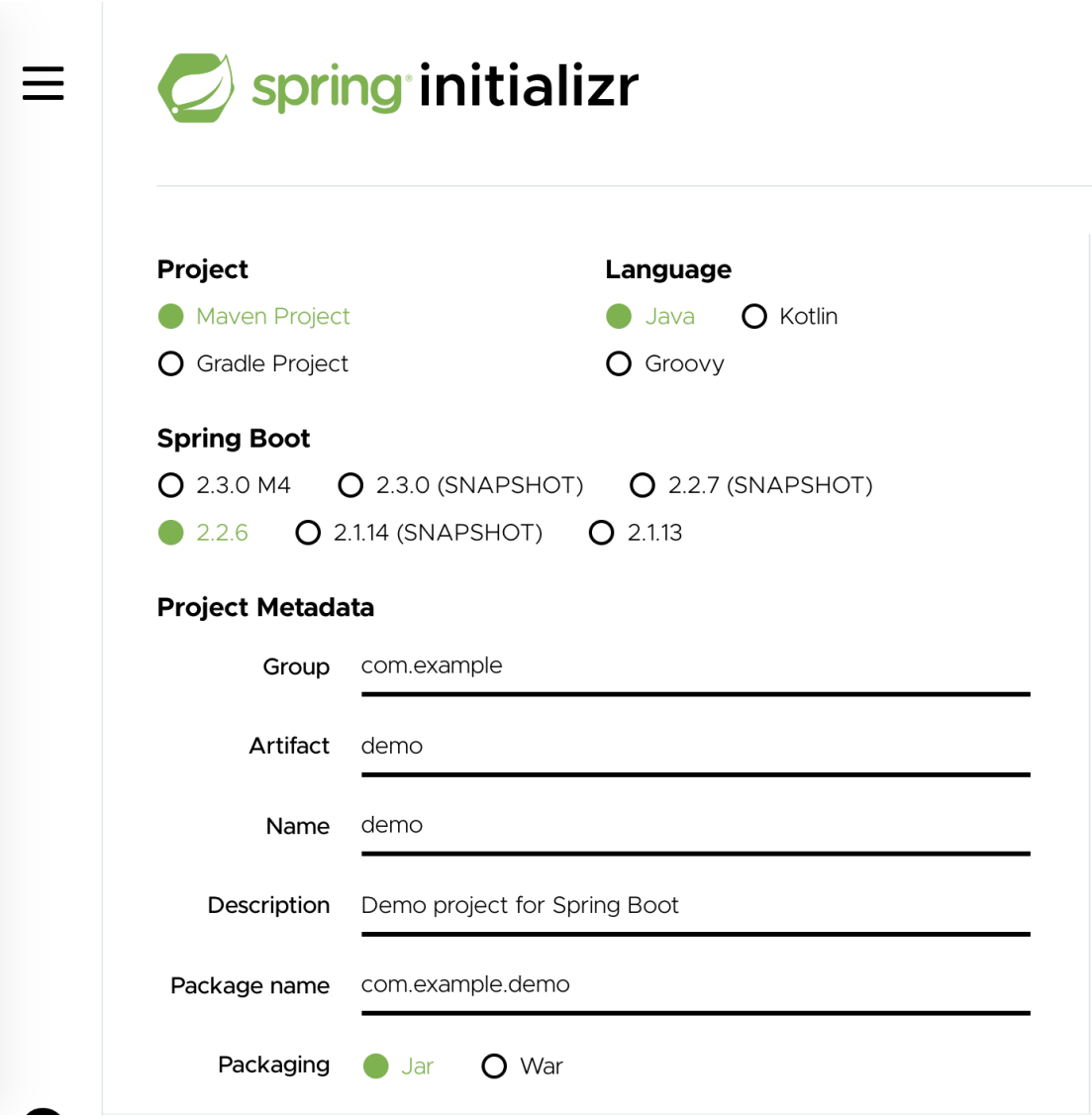
答案

springdata jpa是对jpa(java 持久化数据接口规范)规范的一层封装，hibernate实现了jpa规范。
java代码----->springdata jpa ------>jpa规范------>hibernate------>jdbc ----->mysql数据库

下面我们开始我们的实践

用https://start.spring.io/ 的能力,省去我们搭建框架的过程

定义项目的基本信息


spring initializr
Project
Maven Project
Gradle Project
Language
Java
Kotlin
Groovy
Spring Boot
2.3.0 M4
2.3.0 (SNAPSHOT)
2.2.7 (SNAPSHOT)
2.2.6
2.1.14 (SNAPSHOT)
2.1.13
Project Metadata
Group
com.example
Artifact
demo
Name
demo
Description
Demo project for Spring Boot
Package name
com.example.demo
Packaging
Jar
War

增加项目中的能力

Dependencies

ADD DEPENDENCIES... ⌘ + B

No dependency selected

增加项目中的能力

九章算法
www.jiuzhang.com

DEVELOPER TOOLS

Spring Boot DevTools
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok
Java annotation library which helps to reduce boilerplate code.

Spring Configuration Processor
Generate metadata for developers to offer contextual help and "code completion" when working with custom configuration keys (ex.application.properties/.yml files).

WEB

Spring Web
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

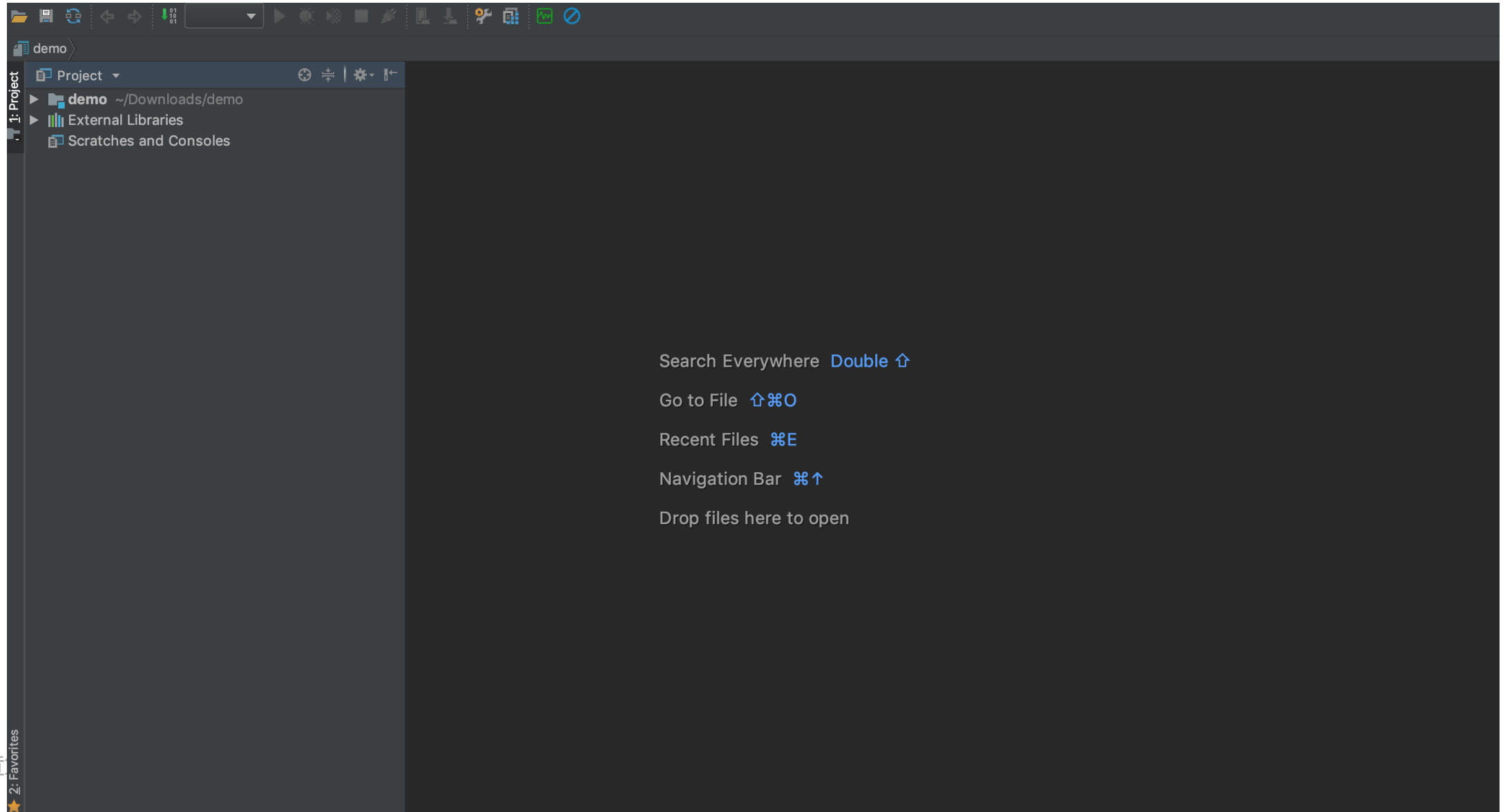
Spring Reactive Web
Build reactive web applications with Spring WebFlux and Netty.

Rest Repositories
Exposing Spring Data repositories over REST via Spring Data REST.

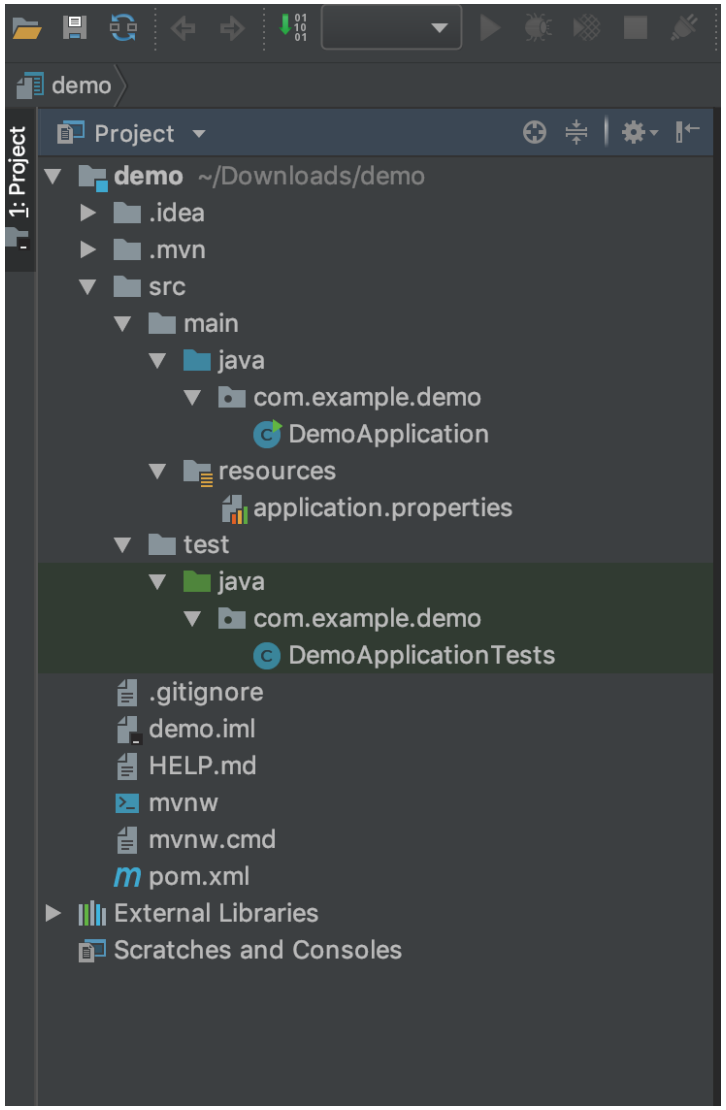
Spring Session

生成一个项目,自动下载,导入

刷新mvn后得到这样的一个项目框架

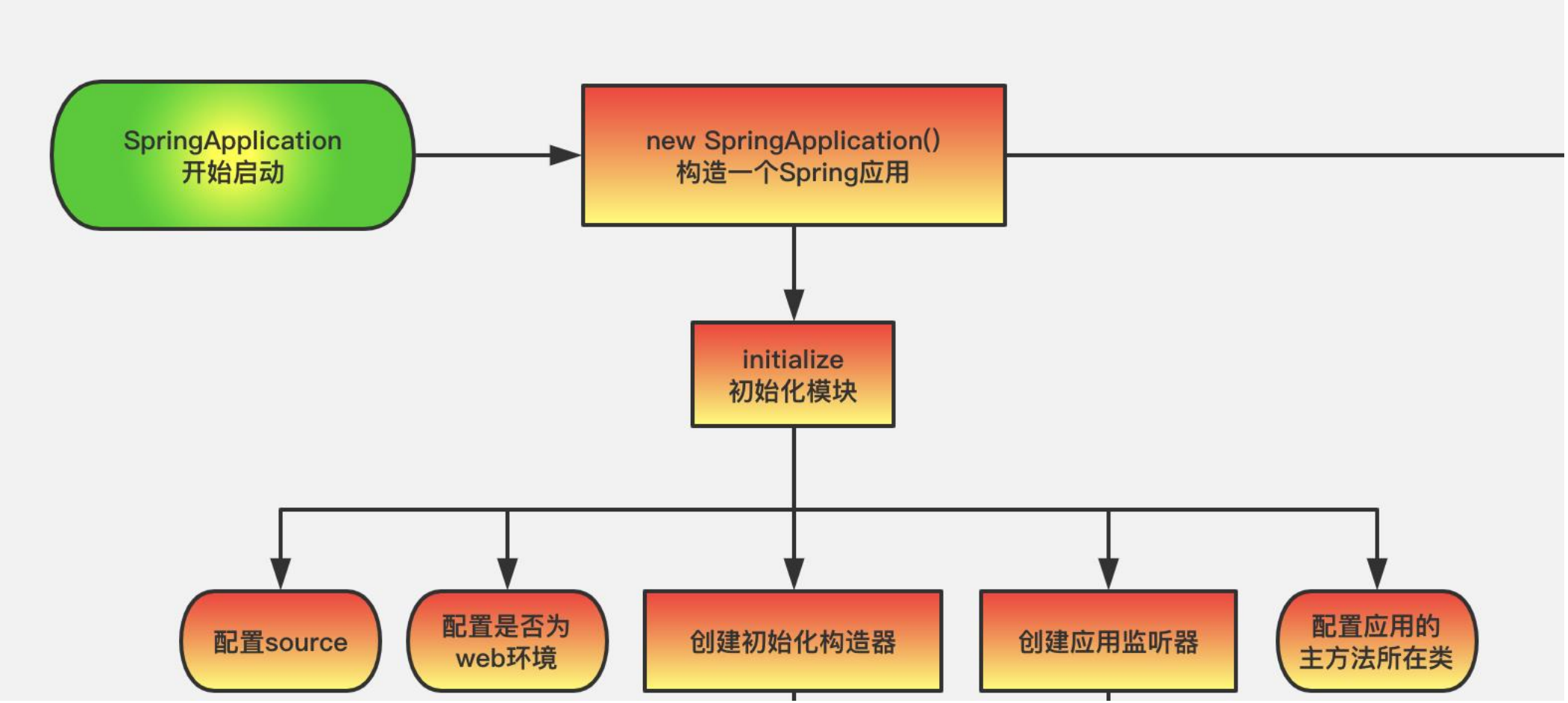


可以看出,大体框架已经拉出来了



spring boot 自动装配解析

Spring boot 启动流程解析



Spring boot 启动流程解析

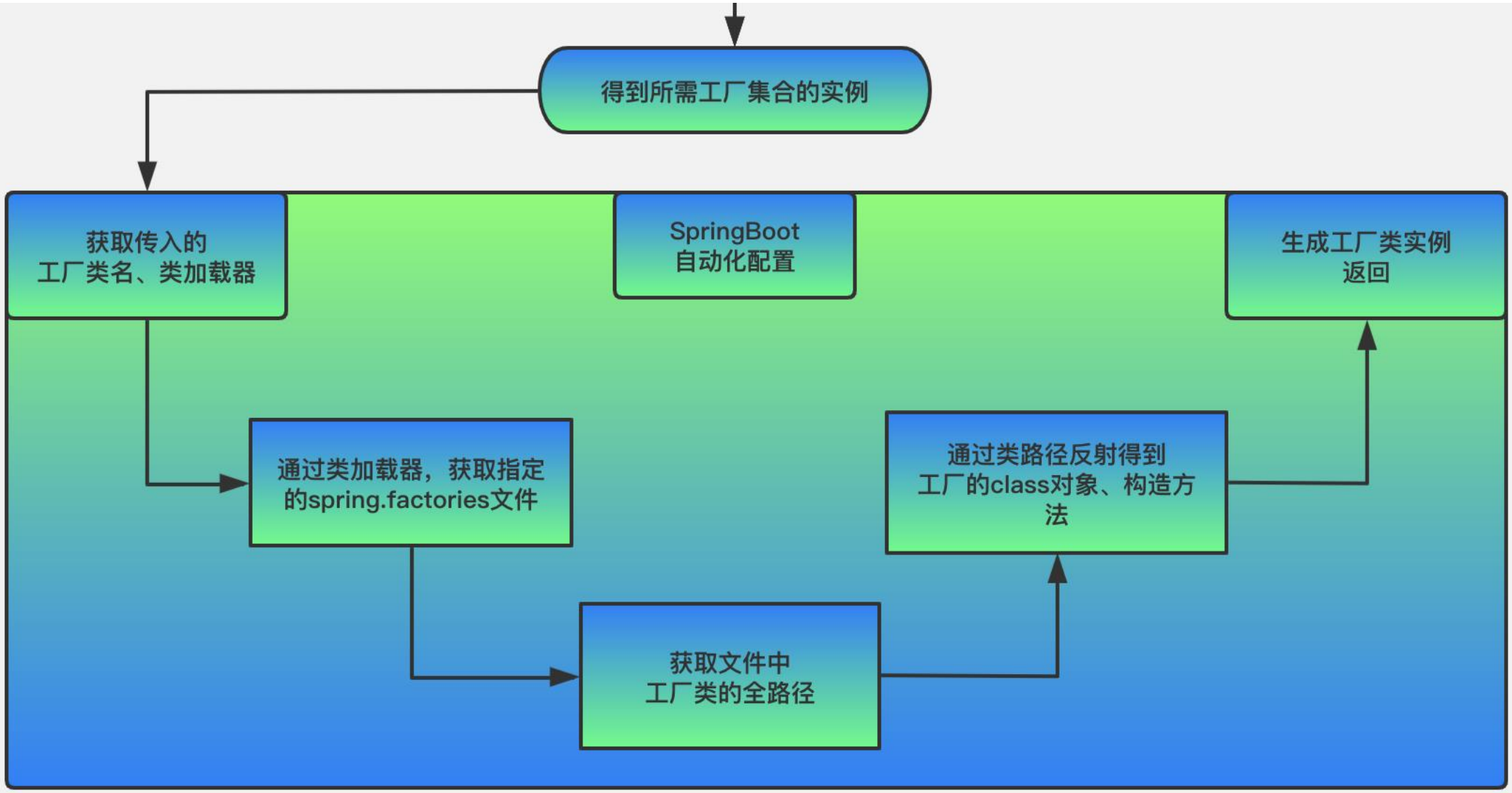
九章算法
www.jiuzhang.com

- 这里有两个重点需要掌握

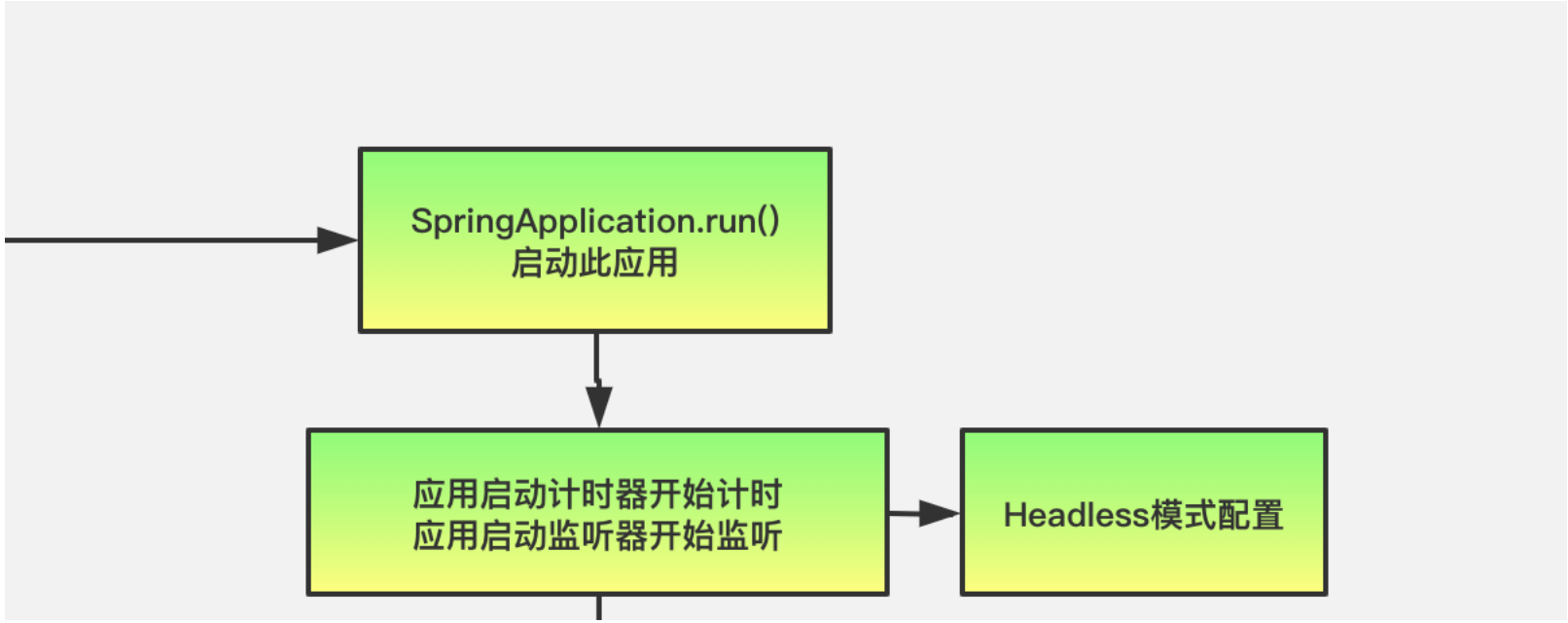
创建初始化构造器

创建应用监听器

获取构造器(工厂)对象


得到所需工厂集合的实例
获取传入的
工厂类名、类加载器
SpringBoot
自动化配置
生成工厂类实例
返回
通过类加载器，获取指定
的spring.factories文件
通过类路径反射得到
工厂的class对象、构造方
法
获取文件中
工厂类的全路径

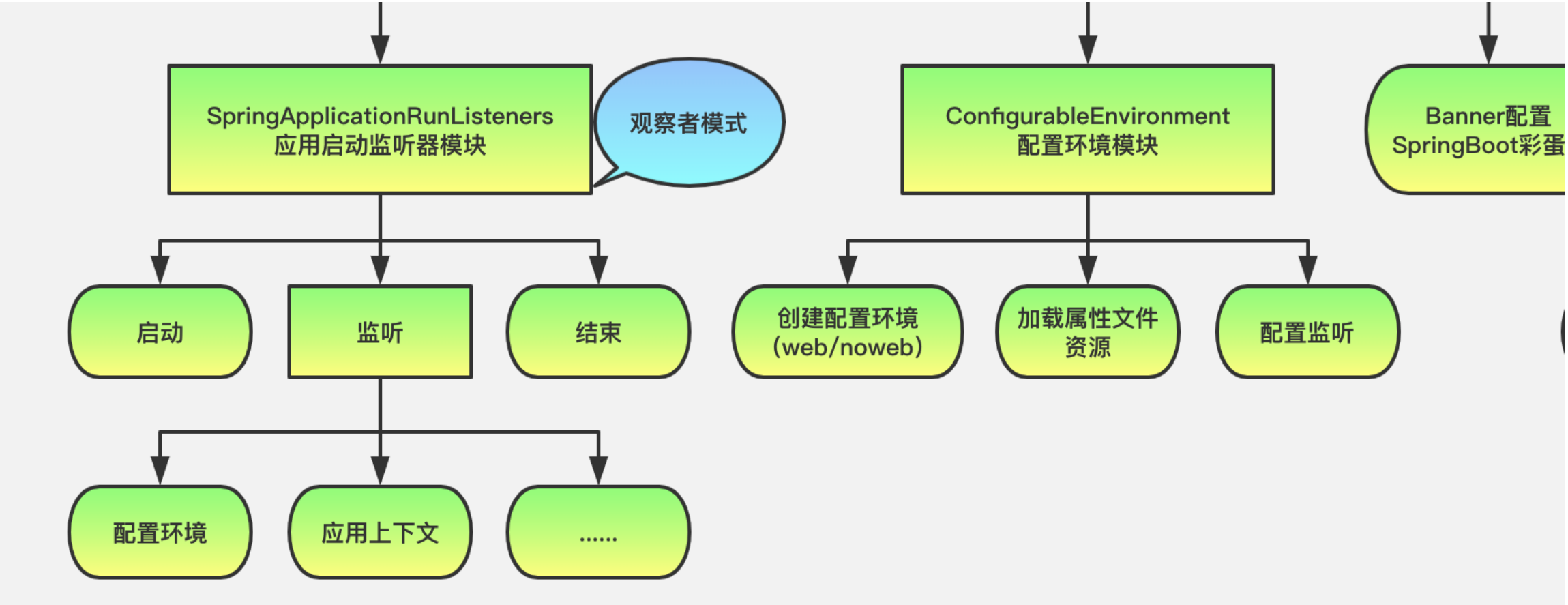
启动应用/启动监听器


SpringApplication.run()
启动此应用
应用启动计时器开始计时
应用启动监听器开始监听
Headless模式配置

什么是head less

Headless模式是系统的一种配置模式。在该模式下，系统缺少了显示设备、键盘或鼠标。

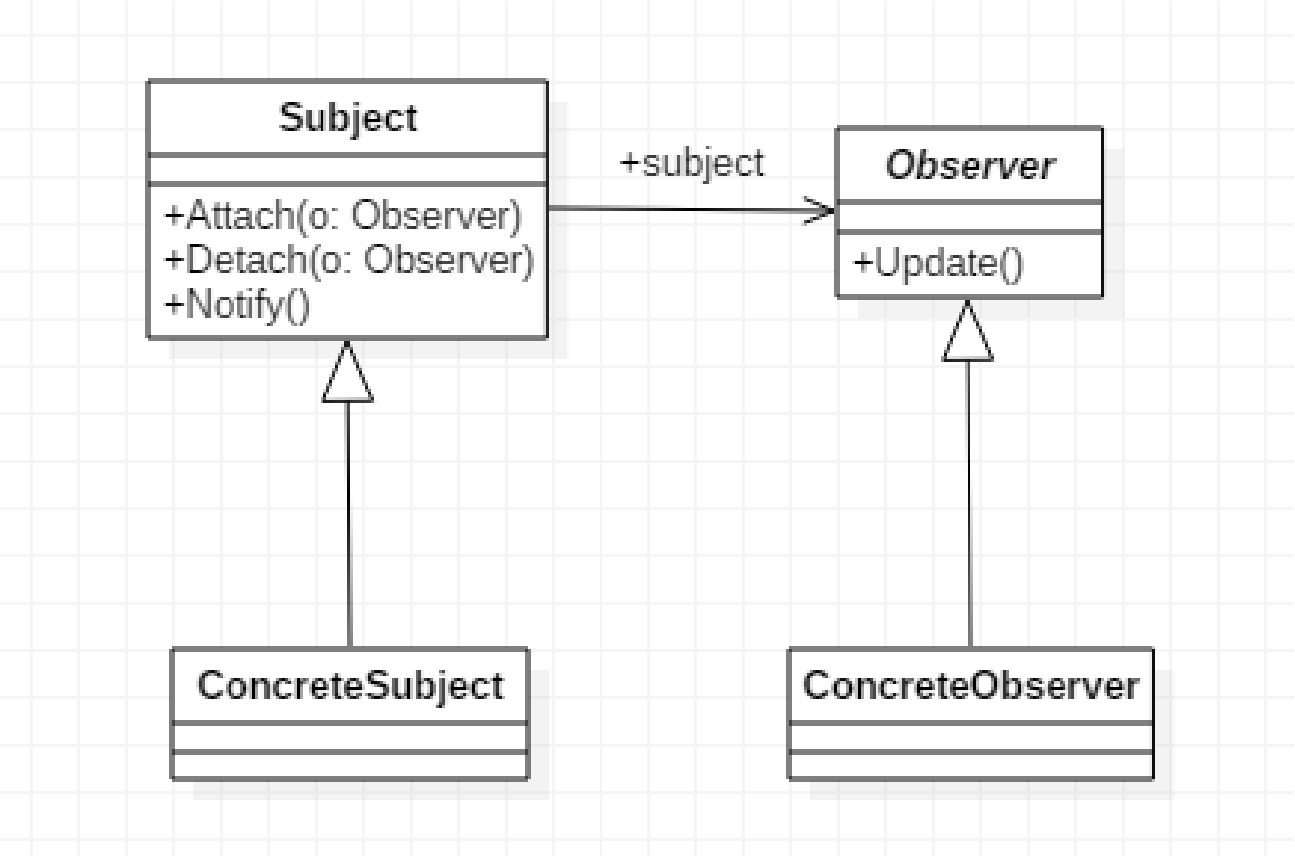
应用监听器的工作模式


SpringApplicationRunListeners
应用启动监听器模块
观察者模式
ConfigurableEnvironment
配置环境模块
Banner配置
SpringBoot彩蛋
启动
监听
结束
创建配置环境
(web/noweb)
加载属性文件
资源
配置监听
配置环境
应用上下文
.....

观察者模式（Observer）

又叫**发布-订阅模式（Publish/Subscribe）**，定义对象间一种一对多的依赖关系，使得每当一个对象改变状态，则所有依赖于它的对象都会得到通知并自动更新。

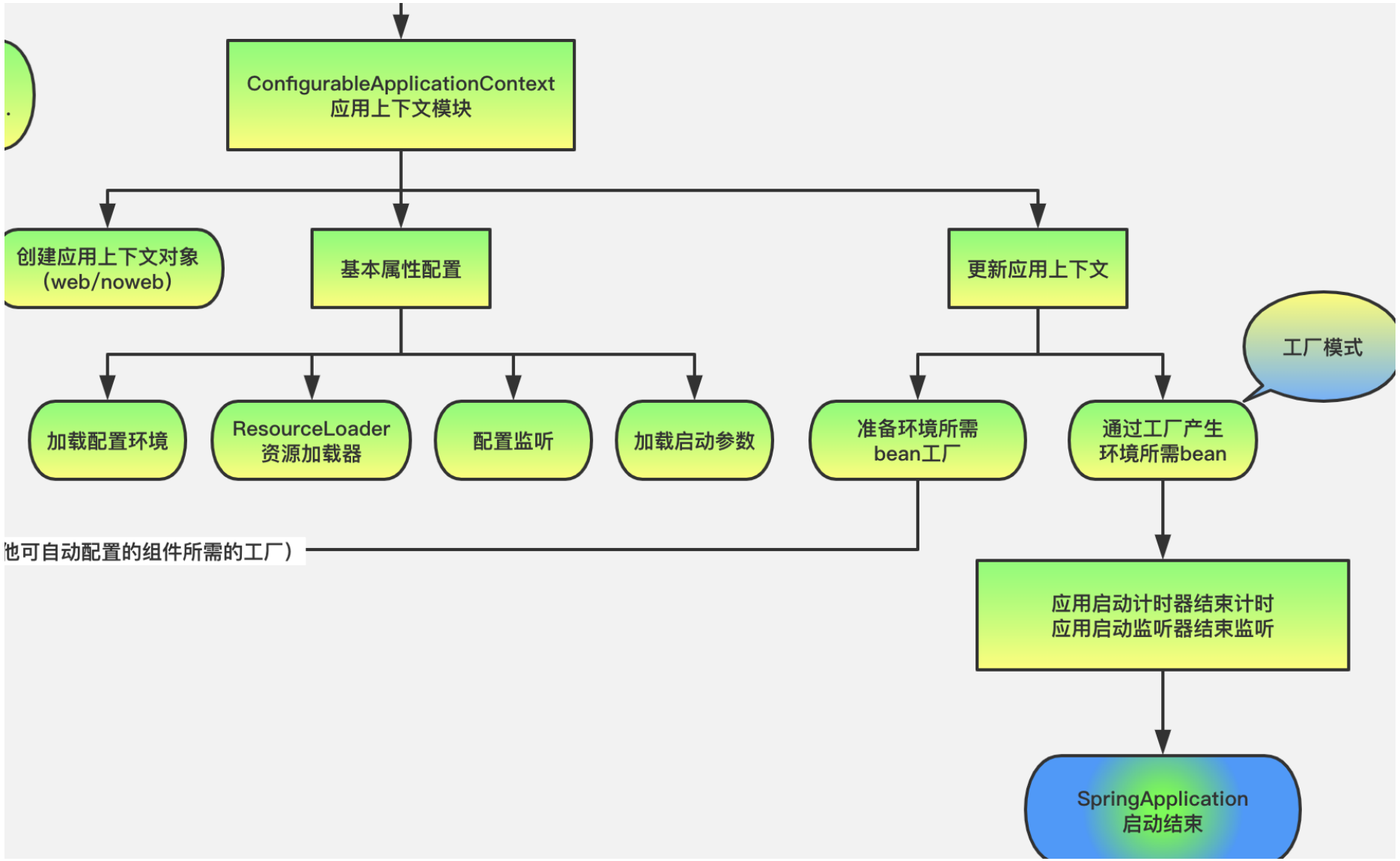
观察者模式


Subject
+Attach(o: Observer)
+Detach(o: Observer)
+Notify()
+subject
Observer
+Update()
ConcreteSubject
ConcreteObserver

观察者模式（Observer）

特点:一个主题对象
明确的观察者订阅主题
观察对象发布主题

真正的启动过程


ConfigurableApplicationContext
应用上下文模块
创建应用上下文对象
(web/noweb)
基本属性配置
更新应用上下文
加载配置环境
ResourceLoader
资源加载器
配置监听
加载启动参数
准备环境所需
bean工厂
通过工厂产生
环境所需bean
工厂模式
他可自动配置的组件所需的工厂)
应用启动计时器结束计时
应用启动监听器结束监听
SpringApplication
启动结束

总的来说把握三个工厂模式的过程

九章算法
www.jiuzhang.com

- 工厂模式创建应用上下文

- 工厂模式创建资源加载器

- 工厂模式加载应用bean

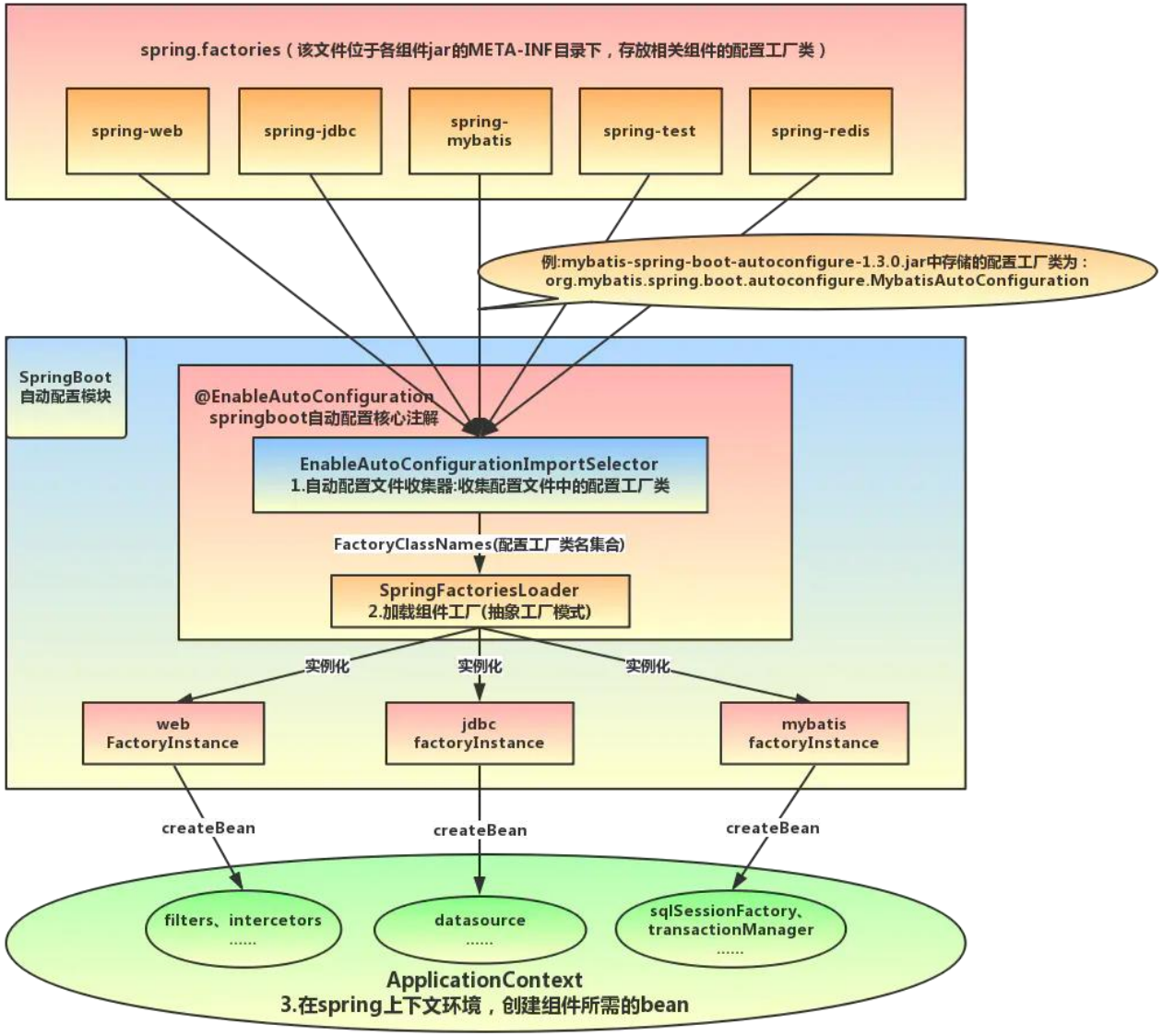
Spring boot自动配置装置过程是很常见的面试问题

频度:高

难度:中

通过率:低

Spring boot自动配置过程



配置文件的加载

九章算法
www.jiuzhang.com

- Starter组件的META-INF文件下均含有spring.factories文件

```
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter:2.1.13.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter/2.1.13.RELEASE/spring-boot-starter-2.1.13.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter/2.1.13.RELEASE/spring-boot-starter-2.1.13.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter/2.1.13.RELEASE/spring-boot-starter-2.1.13.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

配置文件的加载

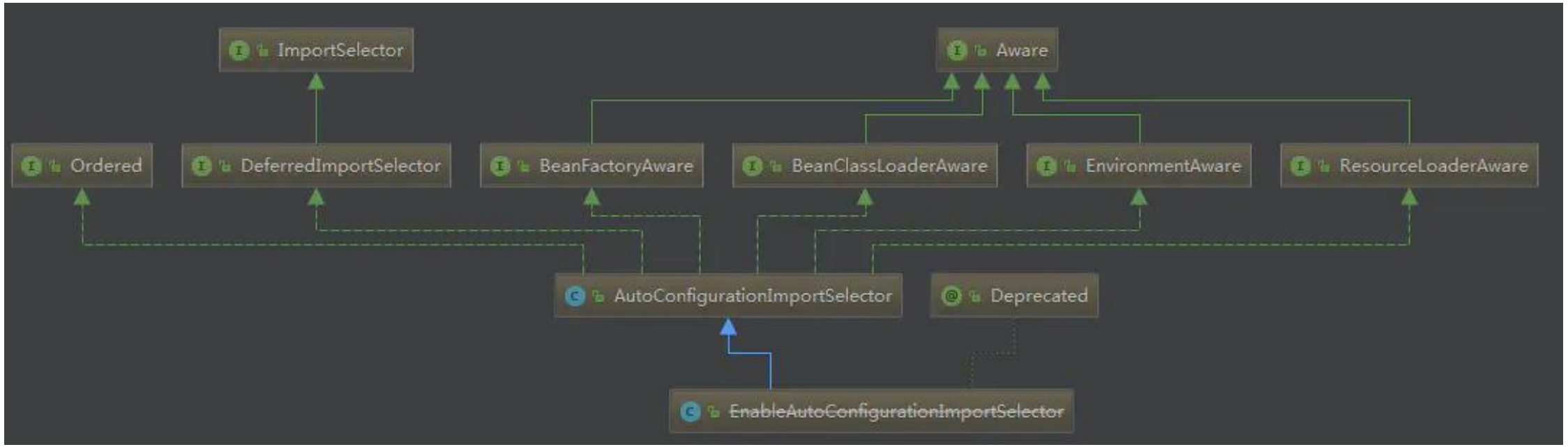
九章算法
www.jiuzhang.com

- Starter组件的META-INF文件下均含有spring.factories文件

```
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter:2.1.13.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter/2.1.13.RELEASE/spring-boot-starter-2.1.13.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter/2.1.13.RELEASE/spring-boot-starter-2.1.13.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter/2.1.13.RELEASE/spring-boot-starter-2.1.13.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

SpringFactoriesLoader收集到文件中的类全名并返回一个类全名的数组，返回的类全名通过反射被实例化，就形成了具体的工厂实例，工厂实例来生成组件具体需要的bean。

EnableAutoConfiguration注解



其最终实现了ImportSelector(选择器)和
BeanClassLoaderAware(bean类加载器中间件)

AutoConfigurationImportSelector的selectImports方法

九章算法
www.jiuzhang.com

```java
@Override
public String[] selectImports(AnnotationMetadata annotationMetadata) {
    if (!isEnabled(annotationMetadata)) {
        return NO_IMPORTS;
    }
    try {
        AutoConfigurationMetadata autoConfigurationMetadata = AutoConfigurationMetadataLoader
                .loadMetadata(this.beanClassLoader);
        AnnotationAttributes attributes = getAttributes(annotationMetadata);
        List<String> configurations = getCandidateConfigurations(annotationMetadata,
                attributes);
        configurations = removeDuplicates(configurations);
        configurations = sort(configurations, autoConfigurationMetadata);
        Set<String> exclusions = getExclusions(annotationMetadata, attributes);
        checkExcludedClasses(configurations, exclusions);
        configurations.removeAll(exclusions);
        configurations = filter(configurations, autoConfigurationMetadata);
        fireAutoConfigurationImportEvents(configurations, exclusions);
        return configurations.toArray(new String[configurations.size()]);
    }
    catch (IOException ex) {
        throw new IllegalStateException(ex);
    }
}
```

selectImports的意义

方法在springboot启动流程——bean实例化前被执行，返回要实例化的类信息列表。

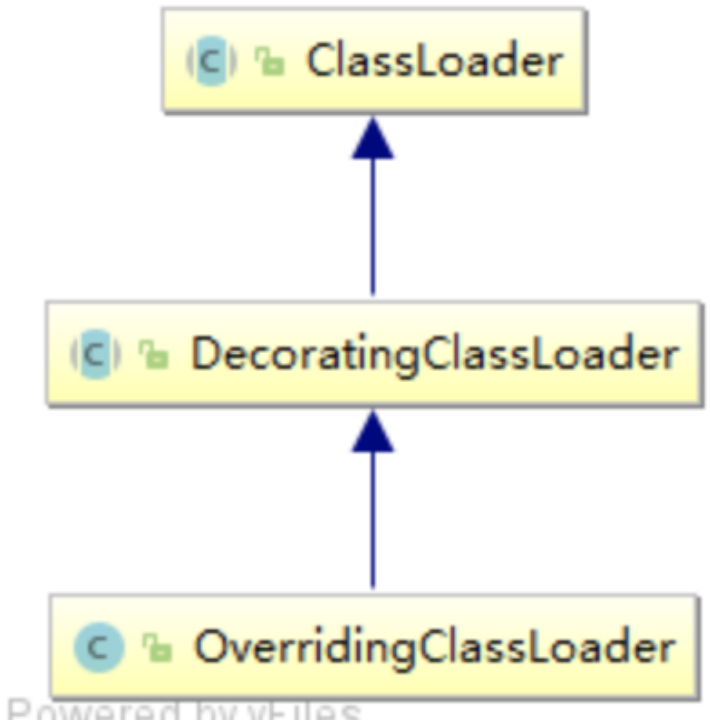
附加问题:spring中bean如何加载?

spring通过类加载器将类加载到jvm中

回答知识要点

Spring 类加载器

Spring 类加载器


ClassLoader
DecoratingClassLoader
OverridingClassLoader

OverridingClassLoader

Spring 自定义的类加载器，默认会先自己加载

OverridingClassLoader

loadClass

loadClassForOverriding

loadClass

九章算法
www.jiuzhang.com

```
@Override
public Class<?> loadClass(String name) throws ClassNotFoundException {
    if (this.overrideDelegate != null && isEligibleForOverriding(name)) {
        return this.overrideDelegate.loadClass(name);
    }
    return super.loadClass(name);
}
```

OverridingClassLoader

loadClassForOverriding

loadClassForOverriding

九章算法
www.jiuzhang.com

```
protected Class<?> loadClassForOverriding(String name) throws ClassNotFoundException {
    Class<?> result = findLoadedClass(name);
    if (result == null) {
        byte[] bytes = loadBytesForClass(name);
        if (bytes != null) {
            result = defineClass(name, bytes, 0, bytes.length);
        }
    }
    return result;
}
```

loadClassForOverriding

九章算法
www.jiuzhang.com

```
protected byte[] loadBytesForClass(String name) throws ClassNotFoundException {
    InputStream is = openStreamForClass(name);
    if (is == null) {
        return null;
    }
    try {
        byte[] bytes = FileCopyUtils.copyToByteArray(is);
        // transformIfNecessary 留给子类重写
        return transformIfNecessary(name, bytes);
    } catch (IOException ex) {
        throw new ClassNotFoundException("Cannot load resource for class [" + name + "]", ex);
    }
}
```

DecoratingClassLoader

内部维护了两个集合，如果你不想你的类被自定义的类加载器管理，
可以把它添加到这两个集合中

```java
private final Set<String> excludedPackages = Collections.newSetFromMap(new ConcurrentHashMap<>(8));
private final Set<String> excludedClasses = Collections.newSetFromMap(new ConcurrentHashMap<>(8));

// isExcluded 返回 true 时仍使用 JDK 的默认类加载机制，返回 false 时自定义的类加载器生效
protected boolean isExcluded(String className) {
    if (this.excludedClasses.contains(className)) {
        return true;
    }
    for (String packageName : this.excludedPackages) {
        if (className.startsWith(packageName)) {
            return true;
        }
    }
    return false;
}
```

下面将我们的项目部署一下

下面将我们的项目部署一下

环境准备

环境准备

九章算法
www.jiuzhang.com

- 根据我们之前的知识储备,可以识别到要在服务器上实际部署,需要做以下工作

一、服务器准备

- linux服务器(云服务器)

- 二、java运行环境准备

- 三、数据库准备

Linux服务器上安装jdk环境

九章算法
www.jiuzhang.com

- 安装jdk
- 命令:
- yum search java | grep -i jdk

列出可安装版本

九章算法
www.jiuzhang.com

```
java-1.8.0-openjdk.x86_64 : OpenJDK Runtime Environment
java-1.8.0-openjdk-accessibility.i686 : OpenJDK accessibility connector
java-1.8.0-openjdk-accessibility.x86_64 : OpenJDK accessibility connector
java-1.8.0-openjdk-accessibility-debug.i686 : OpenJDK accessibility connector
java-1.8.0-openjdk-accessibility-debug.x86_64 : OpenJDK accessibility connector
java-1.8.0-openjdk-debug.i686 : OpenJDK Runtime Environment with full debug on
java-1.8.0-openjdk-debug.x86_64 : OpenJDK Runtime Environment with full debug on
java-1.8.0-openjdk-demo.i686 : OpenJDK Demos
java-1.8.0-openjdk-demo.x86_64 : OpenJDK Demos
java-1.8.0-openjdk-demo-debug.i686 : OpenJDK Demos with full debug on
java-1.8.0-openjdk-demo-debug.x86_64 : OpenJDK Demos with full debug on
java-1.8.0-openjdk-devel.i686 : OpenJDK Development Environment
java-1.8.0-openjdk-devel.x86_64 : OpenJDK Development Environment
java-1.8.0-openjdk-devel-debug.i686 : OpenJDK Development Environment with full
java-1.8.0-openjdk-devel-debug.x86_64 : OpenJDK Development Environment with
java-1.8.0-openjdk-headless.i686 : OpenJDK Runtime Environment
java-1.8.0-openjdk-headless.x86_64 : OpenJDK Runtime Environment
java-1.8.0-openjdk-headless-debug.i686 : OpenJDK Runtime Environment with full
java-1.8.0-openjdk-headless-debug.x86_64 : OpenJDK Runtime Environment with full
java-1.8.0-openjdk-javadoc.noarch : OpenJDK API Documentation
java-1.8.0-openjdk-javadoc-debug.noarch : OpenJDK API Documentation for packages
java-1.8.0-openjdk-javadoc-zip.noarch : OpenJDK API Documentation compressed in
java-1.8.0-openjdk-javadoc-zip-debug.noarch : OpenJDK API Documentation
java-1.8.0-openjdk-src.i686 : OpenJDK Source Bundle
java-1.8.0-openjdk-src.x86_64 : OpenJDK Source Bundle
java-1.8.0-openjdk-src-debug.i686 : OpenJDK Source Bundle for packages with
java-1.8.0-openjdk-src-debug.x86_64 : OpenJDK Source Bundle for packages with
ldapjdk.noarch : The Mozilla LDAP Java SDK
```

选择版本进行安装

九章算法
www.jiuzhang.com

- 安装命令:
- yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel
- 等待命令执行成功

- 检查是否安装成功:

- java –version

```
[root@localhost html]# java -version
openjdk version "1.8.0_181"
OpenJDK Runtime Environment (build 1.8.0_181-b13)
OpenJDK 64-Bit Server VM (build 25.181-b13, mixed mode)
```

配置环境变量

九章算法
www.jiuzhang.com

- 在/etc/profile文件添加如下命令

- JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_64
- PATH=$PATH:$JAVA_HOME/bin
- CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
- export JAVA_HOME CLASSPATH PATH

Mysql数据库安装

九章算法
www.jiuzhang.com

- 安装命令:

- yum install mysql-community-server –y

- 这里安装时间较长,需要耐心等待一下

Mysql数据库安装

九章算法
www.jiuzhang.com

- 设置为开机启动:
- 命令:
- # chkconfig --list | grep mysqld

  # chkconfig mysqld on

Mysql数据库安装

九章算法
www.jiuzhang.com

- 启动mysql 数据库

- 命令:
- service mysqld start

- 设置root 密码:
- mysql_secure_installation

Mysql数据库安装

九章算法
www.jiuzhang.com

- 登录root账号

- 命令:
- mysql -uroot –p

- 注意:远程服务器还要建立远程root用户以便维护

- 命令:
- mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '你设置的密码' WITH GRANT OPTION;
- mysql> flush privileges;

初始化数据库

九章算法
www.jiuzhang.com

- Spring boot 本身有初始化数据表结构的能力,参看下面的配置

- 打开启动初始化数据表结构的功能
- spring.jpa.generate-ddl=true

- 启动时初始化策略设置为“update”
  spring.jpa.hibernate.ddl-auto=update

NOTE

应用第一次拉起后必须将配置改为关闭,否则有删库跑路的风险

Spring-data-jpa封装了数据库访问的能力

可以追溯要源码进行初步分析

Spring-data-jpa的底层封装

九章算法
www.jiuzhang.com

```
public List<T> findAllById(Iterable<ID> ids) {
    Assert.notNull(ids, message: "Ids must not be null!");
    if (!ids.iterator().hasNext()) {
        return Collections.emptyList();
    } else if (!this.entityInformation.hasCompositeId()) {
        Collection<ID> idCollection = toCollection(ids);
        SimpleJpaRepository.ByIdsSpecification<T> specification = new SimpleJpaRepository.ByIdsSpe
        TypedQuery<T> query = this.getQuery(specification, (Sort)Sort.unsorted());
        return query.setParameter(specification.parameter, idCollection).getResultList();
    } else {
        List<T> results = new ArrayList();
        Iterator var3 = ids.iterator();

        while(var3.hasNext()) {
            ID id = var3.next();
            this.findById(id).ifPresent(results::add);
        }

        return results;
    }
}
```

这里可以看到jpa的基本操作

JpaRepository

一个接口规约对业务层开放

JpaRepository

动态代理在运行时调用真正实现了Jpa规范的框架底层完成数据库访问.

POJO bean的实现

九章算法
www.jiuzhang.com

```java
@Entity
@Table(name = "user")
public class User implements Serializable {

    private static final long serialVersionUID = -4537601996396295771L;

    @Id
    @Column(name = "id", nullable = false, length = 48)
    private String id;

    @Column(name = "username", nullable = false, length = 16)
    private String username;

    @Column(name = "password", nullable = false, length = 64)
    private String password;

    @Column(name = "is_verified")
    private String isVerified;

    @Column(name = "email")
    private String email;

    @Column(name = "phone_number")
    private String phoneNumber;

    @Column(name = "area_number")
    private String areaNumber;


    public String getId() { return id; }
```

spring 数据开发中的常用注解

@Entity

表明该类 (UserEntity) 为一个实体类，它默认对应数据库中的表名是 user_entity。

@Entity name属性

@Entity(name = "xwj_user")

@Table

当实体类与其映射的数据库表名不同名时需要使用

@Column

@Column(name = “id”, nullable = false, length = 48)
列名;不可空;长度

@Id注释指定表的主键，它可以有多种生成方式：

九章算法
www.jiuzhang.com

- 1）TABLE：容器指定用底层的数据表确保唯一；
- 2）SEQUENCE：使用数据库的SEQUENCE列来保证唯一（Oracle数据库通过序列来生成唯一ID）；
-
- 3）IDENTITY：使用数据库的IDENTITY列来保证唯一；
-
- 4）AUTO：由容器挑选一个合适的方式来保证唯一；
- 5）NONE：容器不负责主键的生成，由程序来完成。

打包应用

命令:

mvn package

```
admindeMacBook-Pro-29:user-center senyang$ mvn package
[INFO] Scanning for projects...
Downloading: http://mvn.test.alipay.net:8080/artifactory/repo/org/springframework/boot/spring-boot-starter-parent/2.1.13.RELEASE/spring-boot-starter-parent-2.1.13.RELEASE.pom
[WARNING] Unable to get resource 'org.springframework.boot:spring-boot-starter-parent:pom:2.1.13.RELEASE' from repository central_prod (http://mvn.test.alipay.net:8080/artifactory/repo): Specif
ied destination directory cannot be created: /.m2/repository/org/springframework/boot/spring-boot-starter-parent/2.1.13.RELEASE
Downloading: http://mvn.dev.alipay.net:8080/artifactory/repo/org/springframework/boot/spring-boot-starter-parent/2.1.13.RELEASE/spring-boot-starter-parent-2.1.13.RELEASE.pom
[WARNING] Unable to get resource 'org.springframework.boot:spring-boot-starter-parent:pom:2.1.13.RELEASE' from repository central (http://mvn.dev.alipay.net:8080/artifactory/repo): Specified de
stination directory cannot be created: /.m2/repository/org/springframework/boot/spring-boot-starter-parent/2.1.13.RELEASE
Downloading: http://mvn.dev.alipay.net:8080/artifactory/repo/org/springframework/boot/spring-boot-starter-parent/2.1.13.RELEASE/spring-boot-starter-parent-2.1.13.RELEASE.pom
[WARNING] Unable to get resource 'org.springframework.boot:spring-boot-starter-parent:pom:2.1.13.RELEASE' from repository snapshots (http://mvn.dev.alipay.net:8080/artifactory/repo): Specified
destination directory cannot be created: /.m2/repository/org/springframework/boot/spring-boot-starter-parent/2.1.13.RELEASE
Downloading: http://repo1.maven.org/maven2/org/springframework/boot/spring-boot-starter-parent/2.1.13.RELEASE/spring-boot-starter-parent-2.1.13.RELEASE.pom
```

拉起应用

九章算法
www.jiuzhang.com

- 命令:

- java -jar lean_1-0.0.1-SNAPSHOT.jar

拉起应用

九章算法
www.jiuzhang.com

```
admindeMacBook-Pro-29:target senyang$ java -jar lean_1-0.0.1-SNAPSHOT.jar

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.13.RELEASE)

2020-04-26 00:23:41.680  INFO 9252 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2020-04-26 00:23:41.780  INFO 9252 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 91ms. Found 8 JPA repository interfaces.
2020-04-26 00:23:42.437  INFO 9252 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8099 (http)
2020-04-26 00:23:42.469  INFO 9252 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-04-26 00:23:42.469  INFO 9252 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.31]
2020-04-26 00:23:42.558  INFO 9252 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/jzsf]   : Initializing Spring embedded WebApplicationContext
2020-04-26 00:23:42.558  INFO 9252 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1540 ms
2020-04-26 00:23:42.820  INFO 9252 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [
        name: default
        ...]
2020-04-26 00:23:42.882  INFO 9252 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate Core {5.3.15.Final}
2020-04-26 00:23:42.883  INFO 9252 --- [           main] org.hibernate.cfg.Environment            : HHH000206: hibernate.properties not found
2020-04-26 00:23:43.015  INFO 9252 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.0.4.Final}
2020-04-26 00:23:43.201  INFO 9252 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2020-04-26 00:23:43.621  INFO 9252 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2020-04-26 00:23:43.636  INFO 9252 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
2020-04-26 00:23:44.303  INFO 9252 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2020-04-26 00:23:44.846  INFO 9252 --- [           main] o.h.h.i.QueryTranslatorFactoryInitiator  : HHH000397: Using ASTQueryTranslatorFactory
2020-04-26 00:23:45.190  INFO 9252 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2020-04-26 00:23:45.250  WARN 9252 --- [           main] aWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed
during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2020-04-26 00:23:45.467  INFO 9252 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8099 (http) with context path '/jzsf'
```

可以看见,应用已经拉起来了

面向场景开始我们的设计

前序回顾

之前的课程主要以技能课为主,贯穿面试问题的讲解

教学目标

通过简单的项目实践,训练真正的工程思维

面试对标

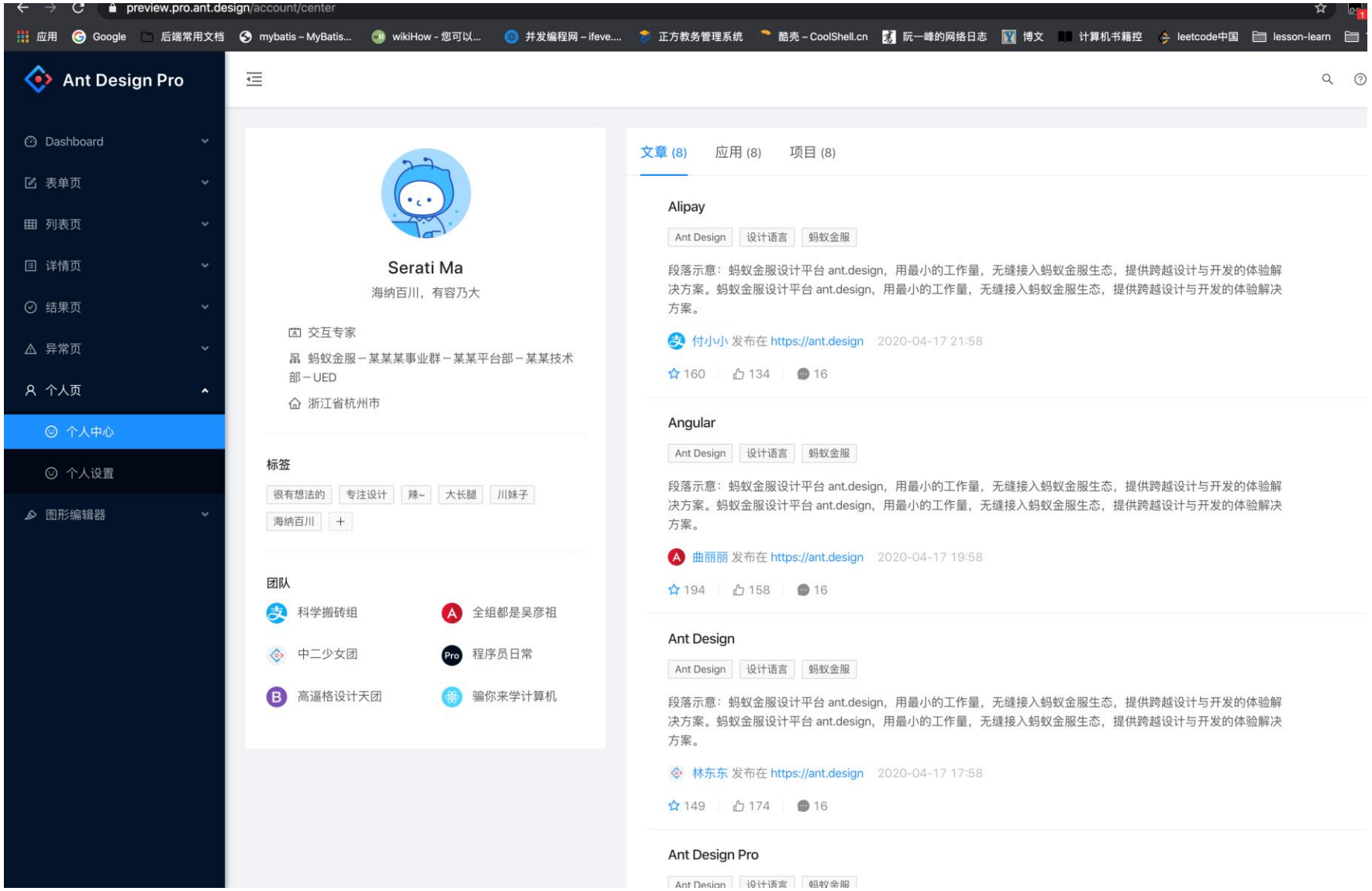
三面即项目面试的内容

九章算法
www.jiuzhang.com

系统分析

项目实战准备

实现目标


preview.pro.ant.design/account/center
Ant Design Pro
Dashboard
表单页
列表页
详情页
结果页
异常页
个人页
个人中心
个人设置
图形编辑器
Serati Ma
海纳百川，有容乃大
交互专家
蚂蚁金服－某某某事业群－某某平台部－某某技术部－UED
浙江省杭州市
标签
很有想法的
专注设计
辣~
大长腿
川妹子
海纳百川
团队
科学搬砖组
全组都是吴彦祖
中二少女团
程序员日常
高逼格设计天团
骗你来学计算机
文章 (8)
应用 (8)
项目 (8)
Alipay
Ant Design
设计语言
蚂蚁金服
段落示意：蚂蚁金服设计平台 ant.design，用最小的工作量，无缝接入蚂蚁金服生态，提供跨越设计与开发的体验解决方案。蚂蚁金服设计平台 ant.design，用最小的工作量，无缝接入蚂蚁金服生态，提供跨越设计与开发的体验解决方案。
付小小 发布在 https://ant.design 2020-04-17 21:58
160
134
16
Angular
Ant Design
设计语言
蚂蚁金服
段落示意：蚂蚁金服设计平台 ant.design，用最小的工作量，无缝接入蚂蚁金服生态，提供跨越设计与开发的体验解决方案。蚂蚁金服设计平台 ant.design，用最小的工作量，无缝接入蚂蚁金服生态，提供跨越设计与开发的体验解决方案。
曲丽丽 发布在 https://ant.design 2020-04-17 19:58
194
158
16
Ant Design
Ant Design
设计语言
蚂蚁金服
段落示意：蚂蚁金服设计平台 ant.design，用最小的工作量，无缝接入蚂蚁金服生态，提供跨越设计与开发的体验解决方案。蚂蚁金服设计平台 ant.design，用最小的工作量，无缝接入蚂蚁金服生态，提供跨越设计与开发的体验解决方案。
林东东 发布在 https://ant.design 2020-04-17 17:58
149
174
16
Ant Design Pro

从一个场景需求入手

九章算法
www.jiuzhang.com

- 实现一个互联网场景中的用户中心功能

用户登录

用户注册

用户偏好设置+个人信息设置

文章模块:发表+查看

系统分析

“快反”——快速反应,是互联网项目中的系统分析区别于传统的系统分析主要特征

系统分析

业务需求分析+功能需求分析+项目架构及技术选型+系统实现路径
= 互联网项目的系统分析

系统分析

业务需求分析和功能分析主要由脑图来实现

面试题

请谈谈你研发XXX项目的过程:

频度:高

难度:中

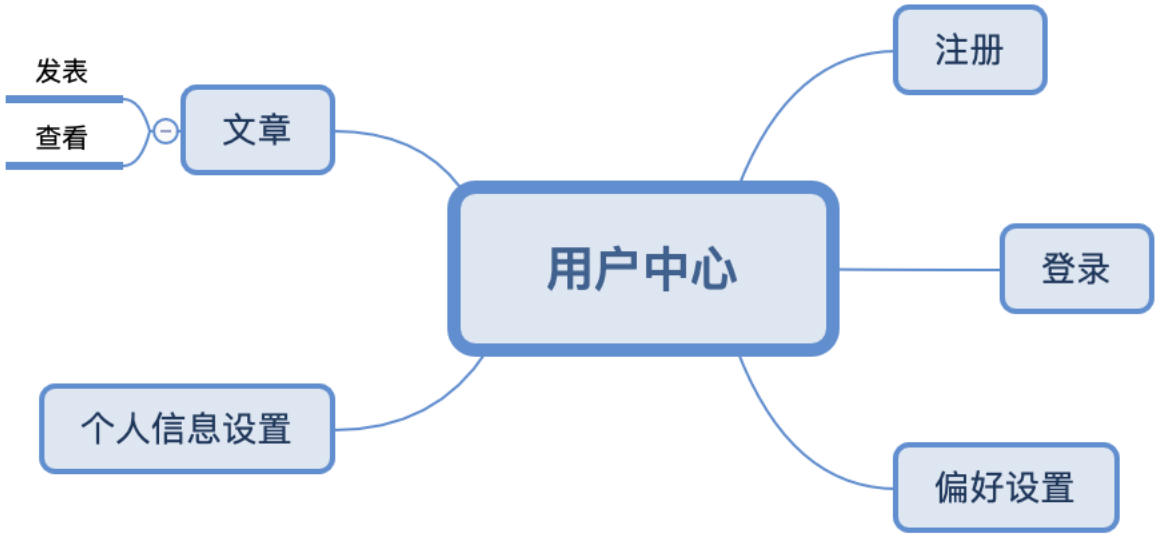
通过率:低

答案(框架)

九章算法
www.jiuzhang.com

- 1、从需求分析脑图入手
- 2、合并需求脑图到功能脑图
- 3、输出实体对象和领域对象
- 4、技术框架(技术栈)选型
- 5、稳定性和异常挽回

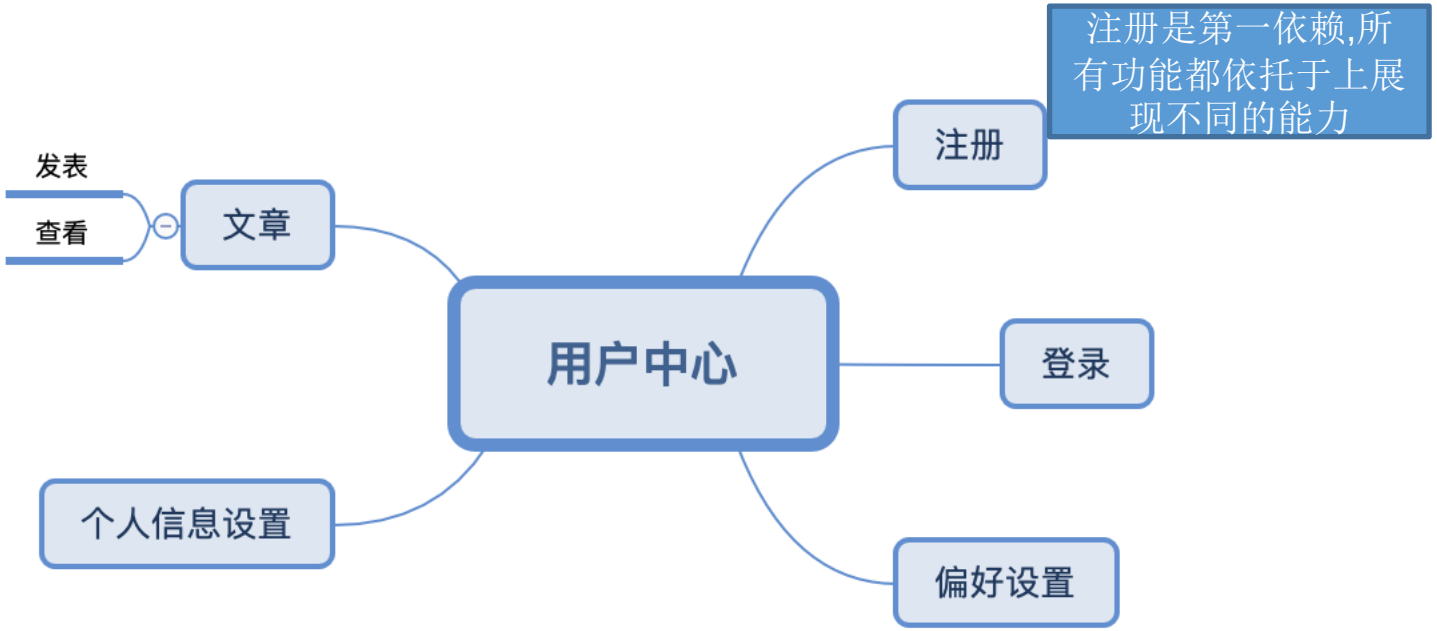
需求分析脑图



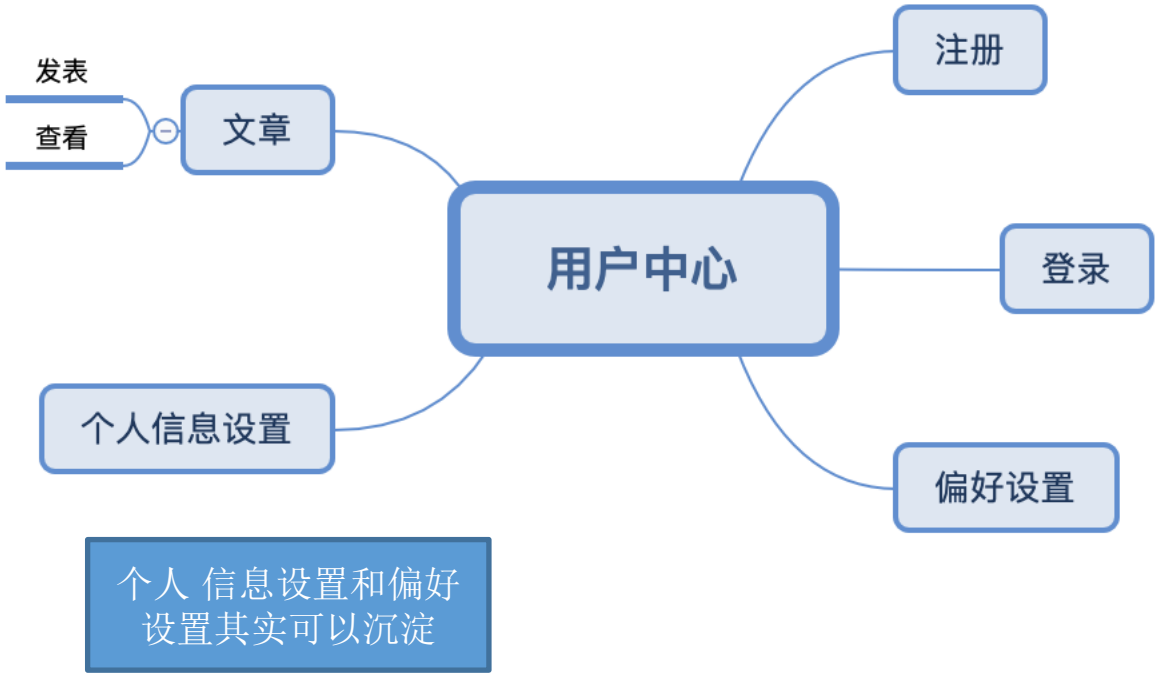
合并需求脑图到功能脑图

业务视角到系统视角,合并的方略是注入逻辑性(依赖性)和沉淀共性

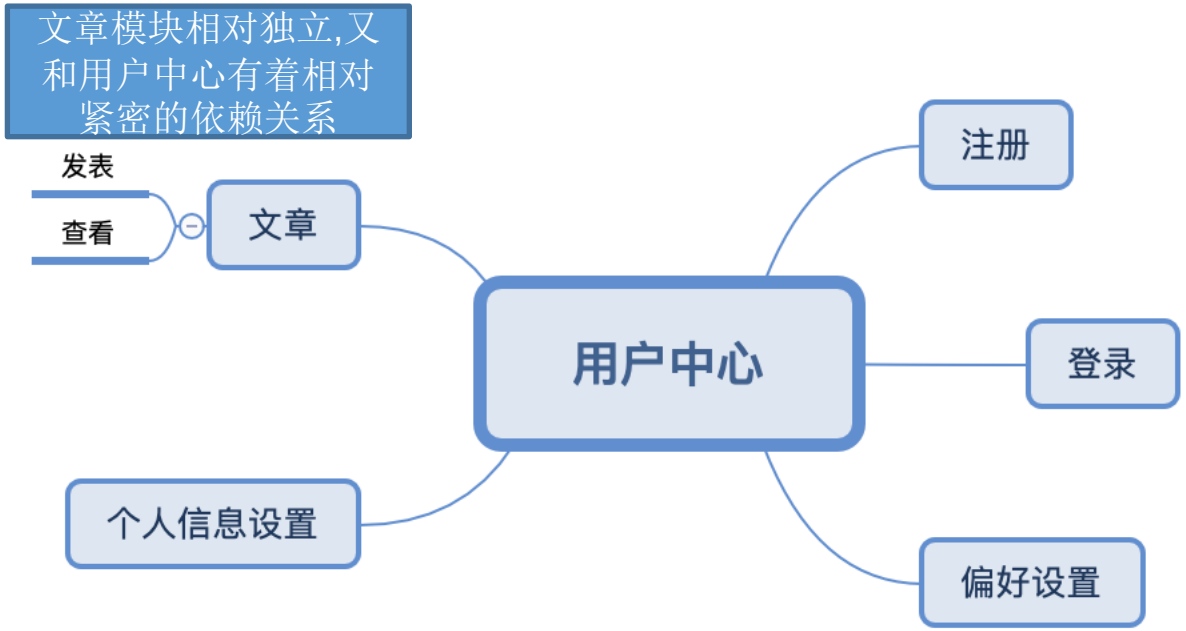
需求分析脑图



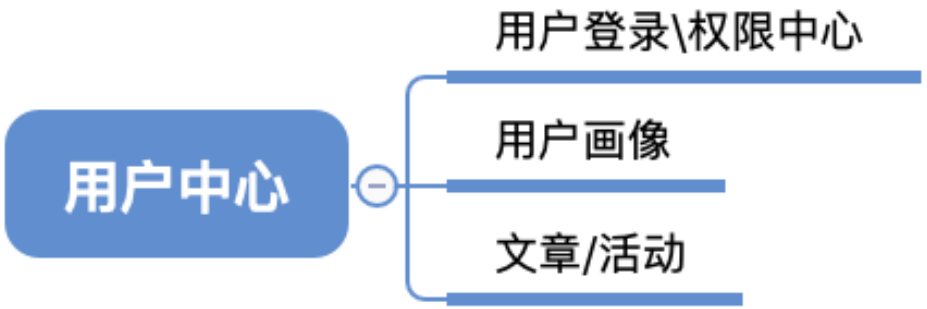
需求分析脑图



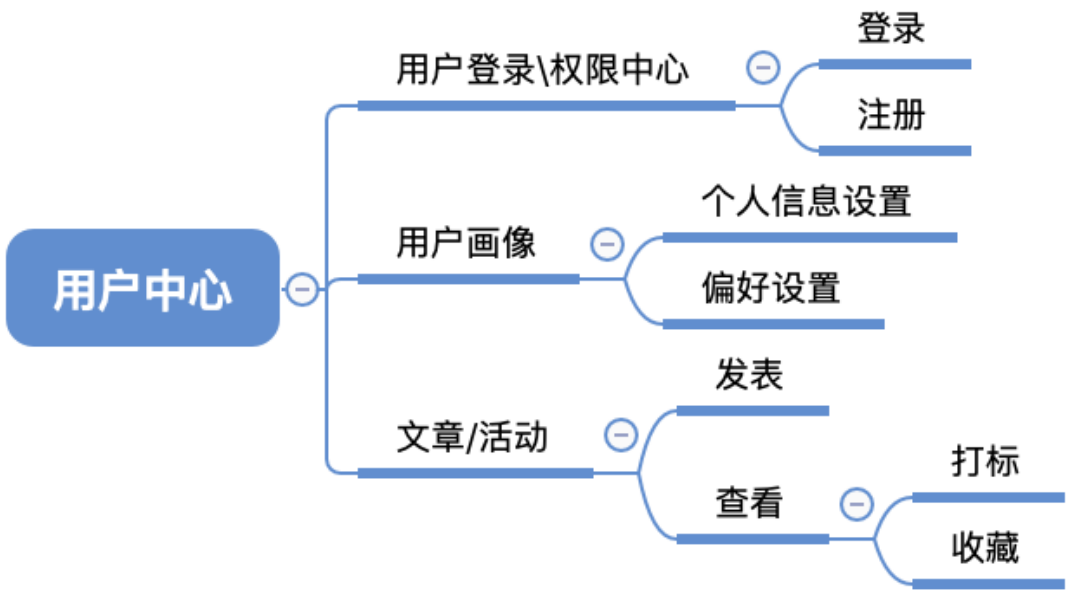
需求分析脑图



功能脑图——step 1



功能脑图——step 2



功能模块结构通过这样的合并就能完整展示

需求脑图到功能脑图是一个比较重要的项目面试考察环节,候选人描述的清晰度和方法论可以给面试官全面的信息量去考察简历的真实性

功能模块结构通过这样的合并就能完整展示

换言之,简历中提到的项目,自己都应该去做这样的功课,把这个环节补齐

输出实体对象和领域对象

领域对象是一个相对专业的课题,对于工程人员来说,简单认为,领域对象就是系统间交互的业务对象

领域对象

在我们这个场景里,简单的说,前端请求上来,后端系统用一个对象去“接住”这个请求,这里就需要定义一系列的领域对象

如用户登录请求,我们用这样一个领域对象来接住它

九章算法
www.jiuzhang.com

```java
public class LoginReq implements Serializable {

    private static final long serialVersionUID = -7559061990780206659L;

    private String username;
    private String password;
    //private String role;

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }


}
```

同样的,用户的画像信息,用这样一个领域对象来接住

九章算法
www.jiuzhang.com

```java
public class UserProfileReq implements Serializable {

    private static final long serialVersionUID = -8119755907886576088L;

    private String username;
    private String email;
    private String personalProfile;
    private String country;
    private String province;
    private String city;
    private String streetAddress;
    private String areaNumber;
    private String phoneNumber;
```

实体对象

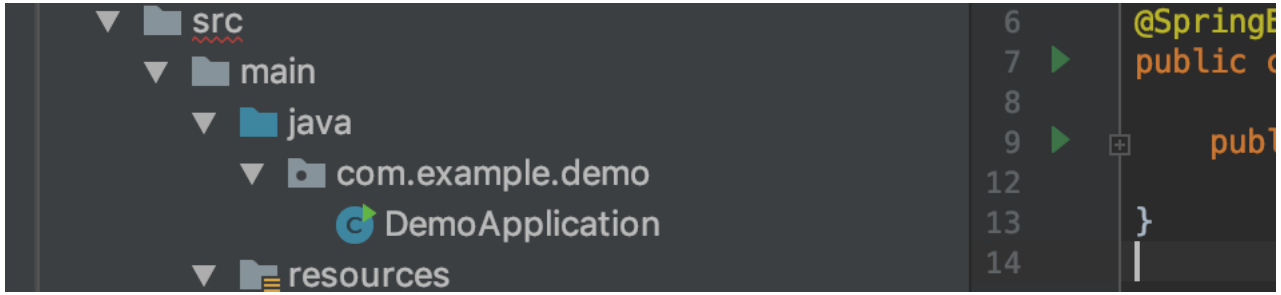
实体对象由领域对象推导而来,体现的是数据库中的存储关系

用上述两个领域对象,合并出用户信息这个实体对象

九章算法
www.jiuzhang.com

```java
@Entity
@Table(name = "user")
public class User implements Serializable {

    private static final long serialVersionUID = -4537601996396295771L;

    @Id
    @Column(name = "id", nullable = false, length = 48)
    private String id;

    @Column(name = "username", nullable = false, length = 16)
    private String username;

    @Column(name = "password", nullable = false, length = 64)
    private String password;

    @Column(name = "is_verified")
    private String isVerified;

    @Column(name = "email")
    private String email;

    @Column(name = "phone_number")
    private String phoneNumber;

    @Column(name = "area_number")
    private String areaNumber;
```

找到项目入口 项目名+Application.java



可以看到该项目可以直接运行

九章算法
www.jiuzhang.com

```
idea
mvn
src
main
java
com.example.demo
DemoApplication
resources
application.properties
test
java
com.example.demo
arget
gitignore
emo.iml
```

```
import ...

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) { SpringApplication.run(DemoApplication.class, args); }

}
```

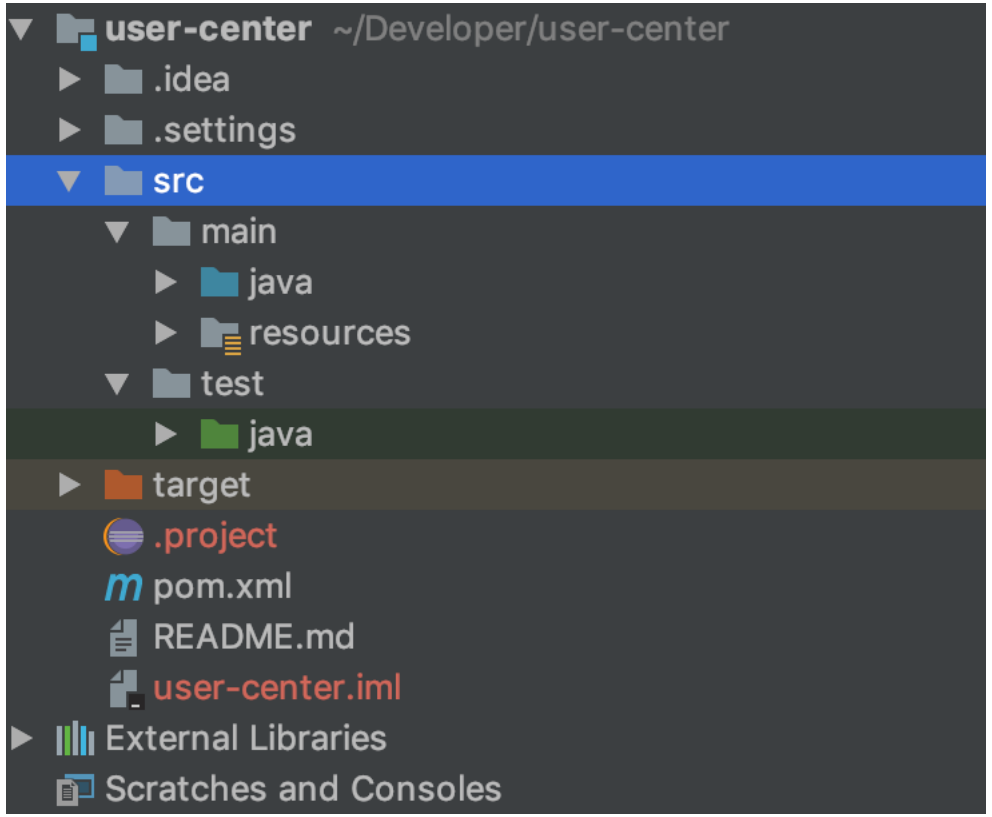
DemoApplication

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.13.RELEASE)

2020-04-18 16:08:40.201  INFO 2479 --- [           main] com.example.demo.DemoApplication         : Starting DemoApplication on admindeMacBook-Pro-29.local with PID 2479 (/Users/senyang/Dow
2020-04-18 16:08:40.203  INFO 2479 --- [           main] com.example.demo.DemoApplication         : No active profile set, falling back to default profiles: default
2020-04-18 16:08:40.663  INFO 2479 --- [           main] com.example.demo.DemoApplication         : Started DemoApplication in 0.877 seconds (JVM running for 1.242)

Process finished with exit code 0
```

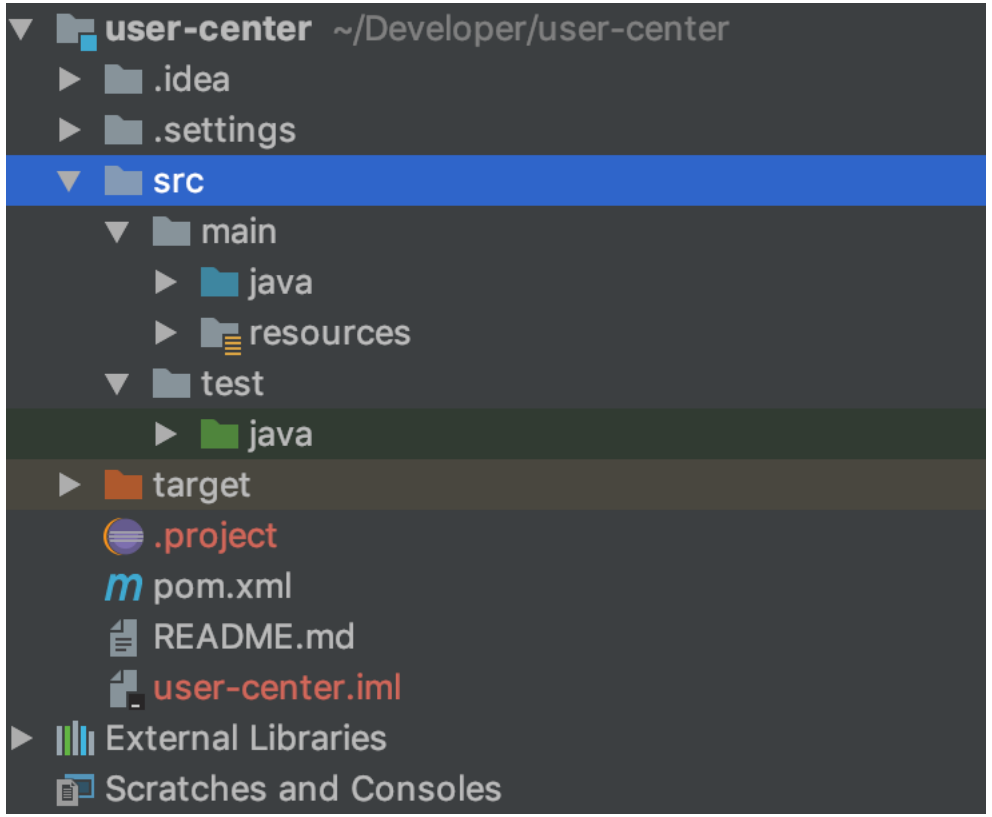
下面我们总体上的架构拉起来

利用互联网常用的领域模型知识逐步落地为系统架构

先观察我们现在得到的项目结构



先观察我们现在得到的项目结构

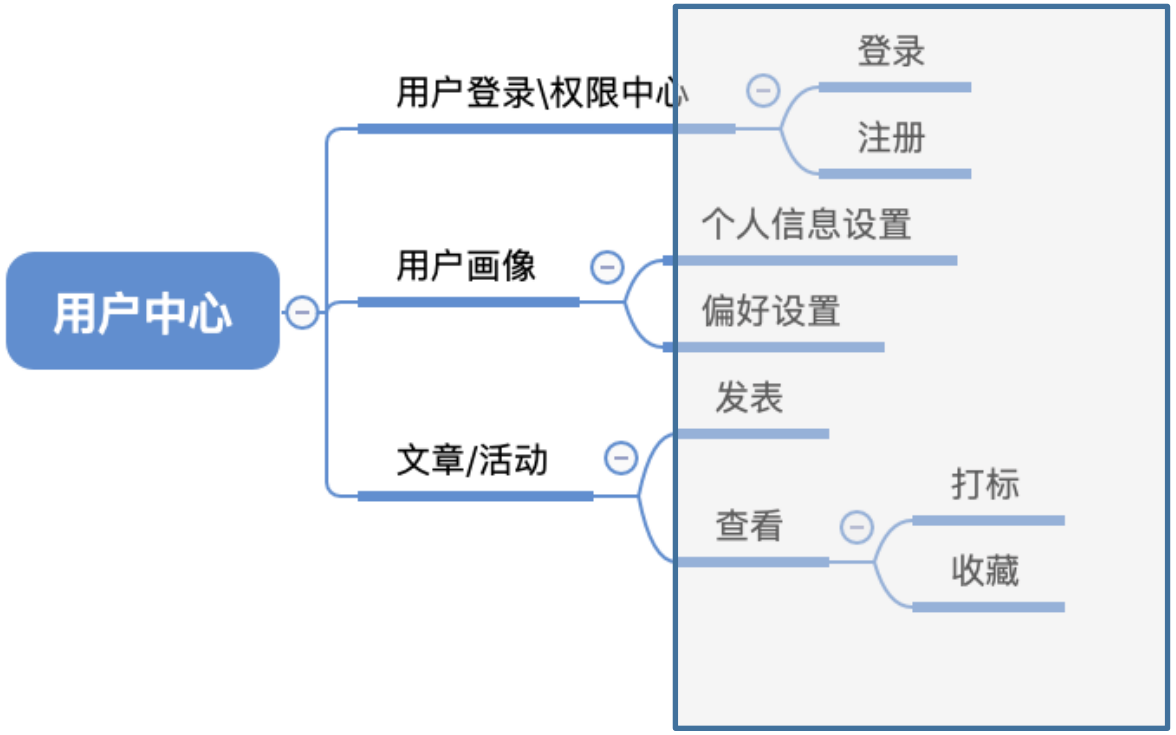


一个业务bundle

一个业务test bundle

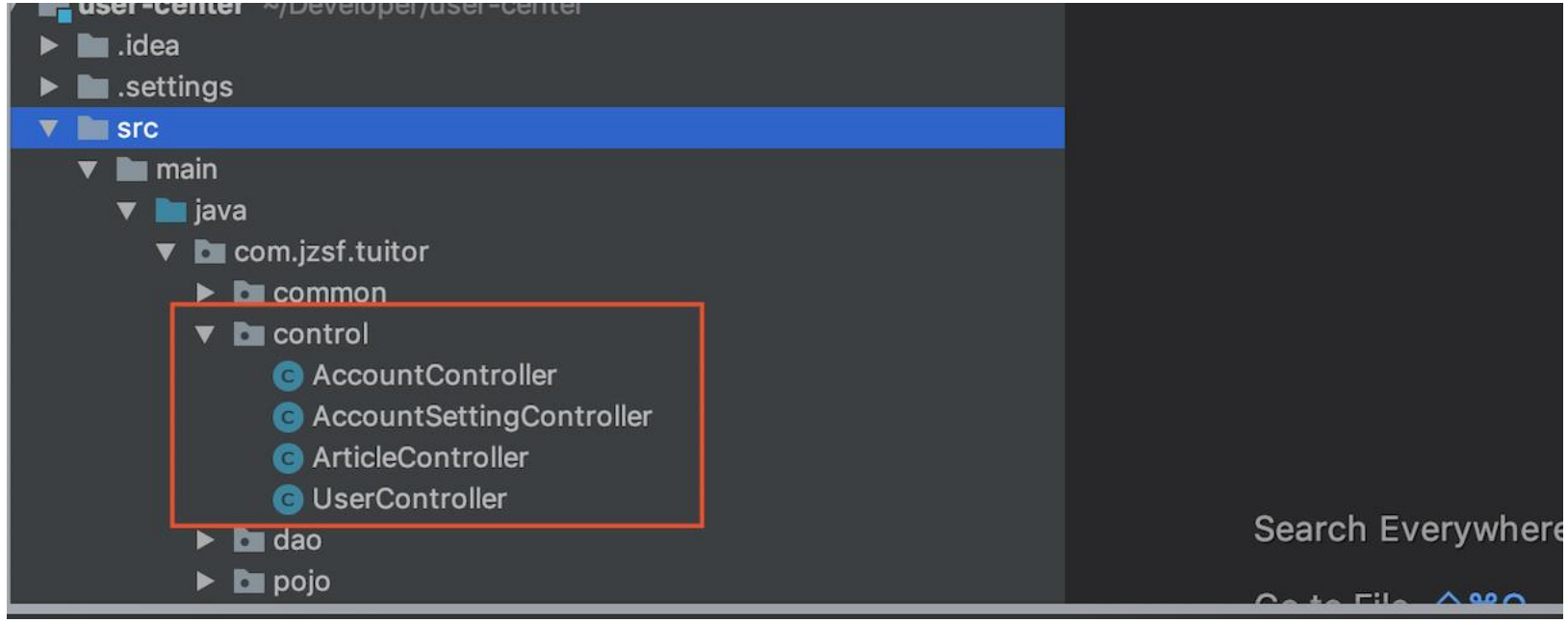
通过我们之前得到的脑图划分系统结构层次

作为后端服务,我们的第一个层次:rpc服务接口层,在mvc框架中,对应我们的contorller,微服务架构中,对应façade层



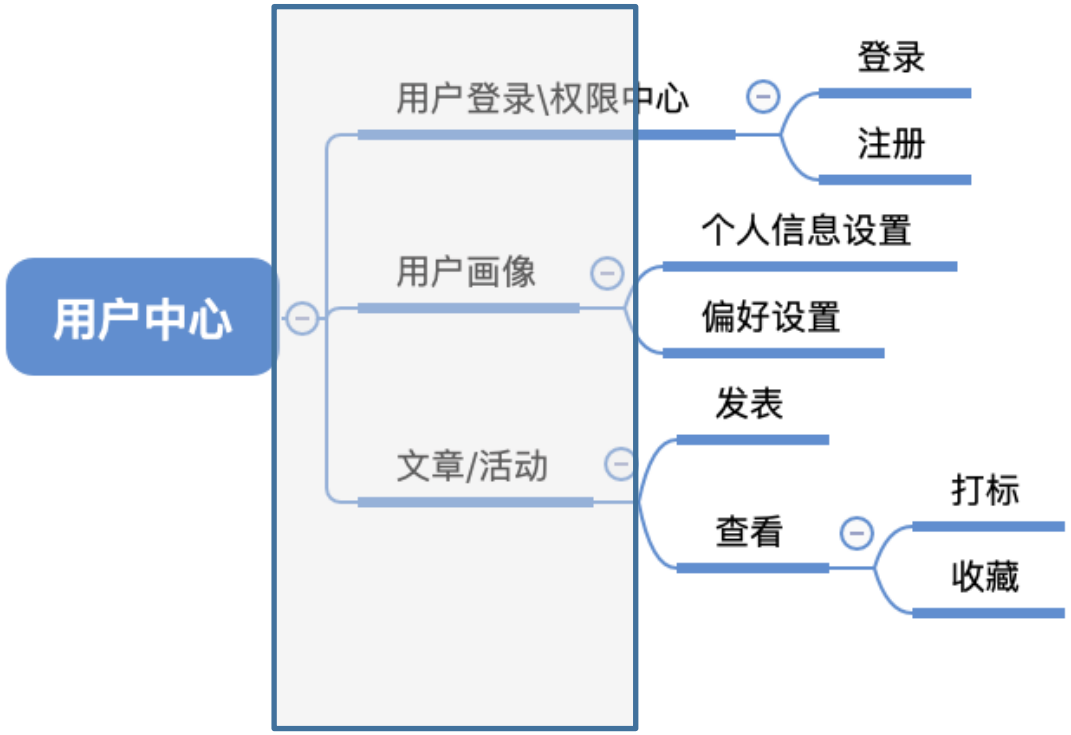
Rpc接口层

- 在mvc框架下,所谓的接口层就是contorller层



通过我们之前得到的脑图划分系统结构层次

Model层是后端中比较“重”的一个系统层次,也是我们架构分解的重点

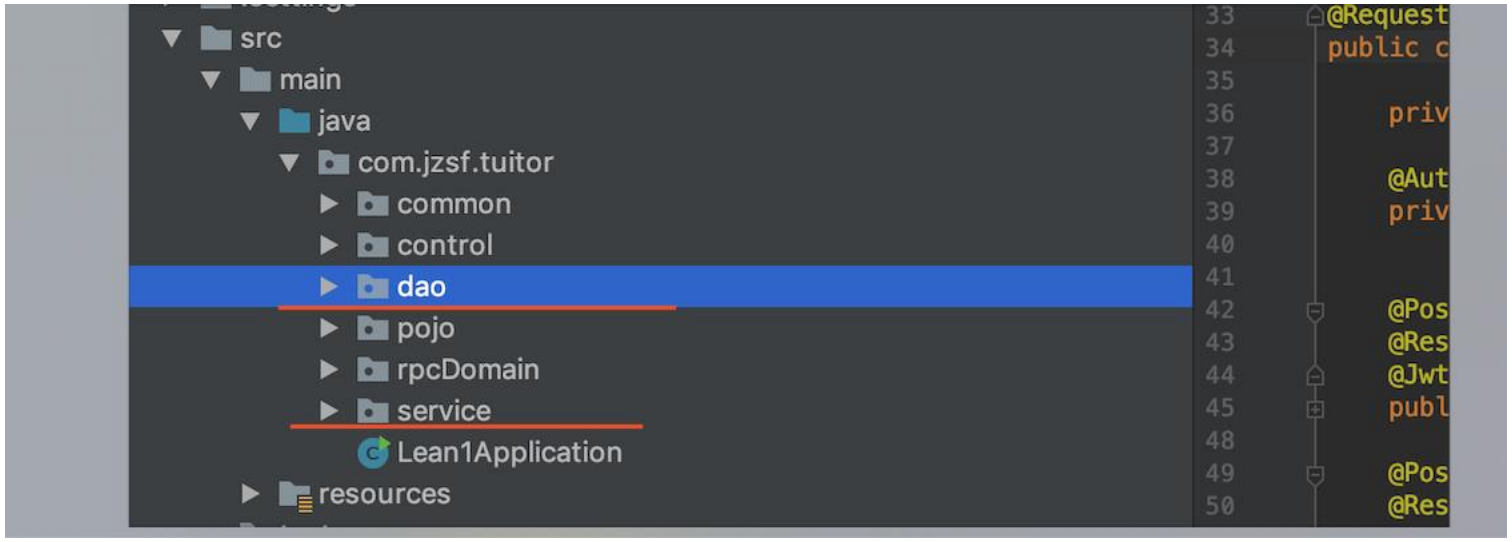


modle

Web应用中用于处理数据逻辑的部分，包括Service层和Dao层；
Service层用于和数据库联动，放置业务逻辑代码，处理数据库的增删改查，
Dao层用于放各种接口，以备调用

modle

直接拉开service和dao层

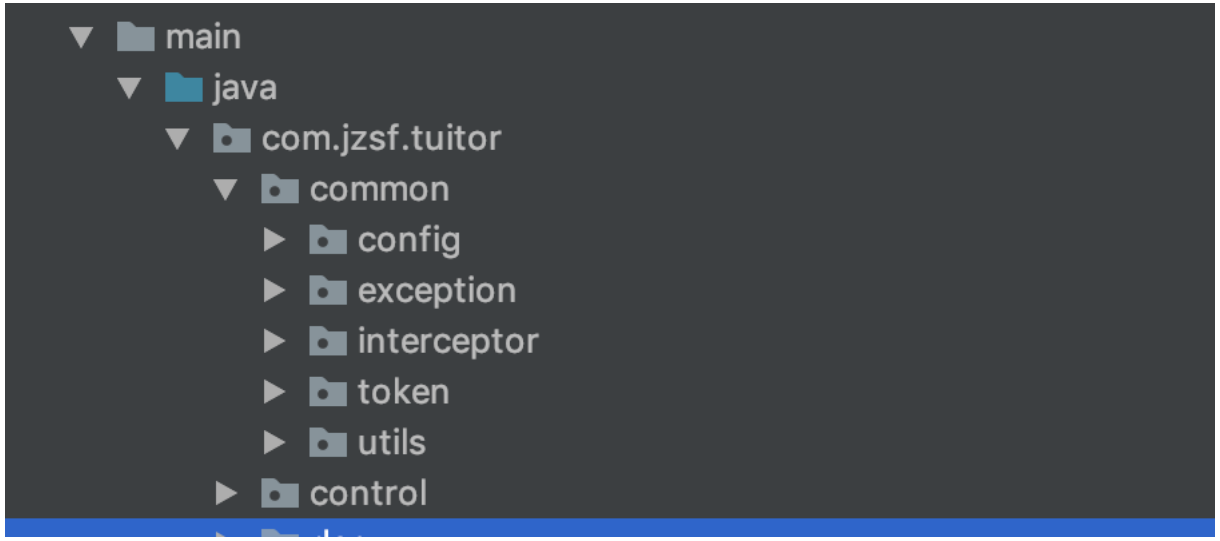

src
main
java
com.jzsf.tuitor
common
control
dao
pojo
rpcDomain
service
Lean1Application
resources
33 @Request
34 public c
35
36 priv
37
38 @Aut
39 priv
40
41
42 @Pos
43 @Res
44 @Jwt
45 publ
48
49 @Pos
50 @Res

Dao层既然拉开了,就必然存在实体Bean这一层:POJO

九章算法
www.jiuzhang.com

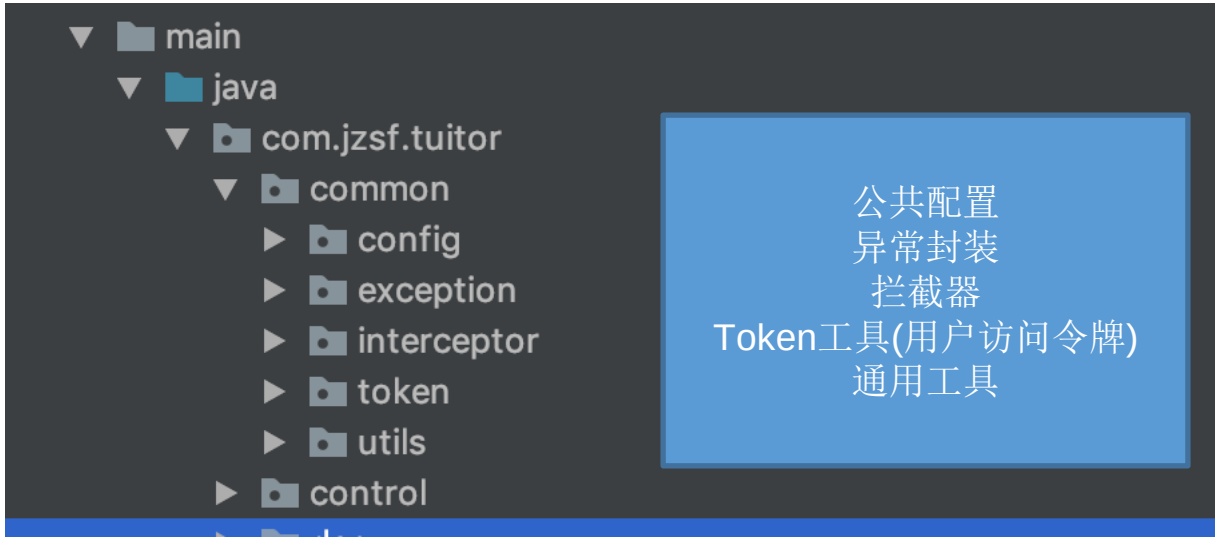
既然是一个应用,就有一些公共的能力要往底层沉淀-common层

九章算法
www.jiuzhang.com



既然是一个应用,就有一些公共的能力要往底层沉淀-common层

九章算法
www.jiuzhang.com



系统架构总结

九章算法
www.jiuzhang.com

前端

rpc

controller

service

common

dao