

阿里云开发者社区
ALIBABA CLOUD DEVELOPER COMMUNITY

Spring Boot 2.6.0

电商网站开发实战

阿里云开发者学堂推荐配套教材



作者：侠客



扫码观看课程



阿里云开发者“藏经阁”
海量电子手册免费下载

卷首语

基于最新 Spring Boot 2.6。课程全面覆盖 Spring Boot 2.6 新特性、自动化配置原理、电商网站需求分析、架构设计、高并发电商网站架构、微服务架构、UML 绘图。实战 MySQL 8.0 数据库、Redis 高并发缓存、MongoDB 数据库、安全机制、性能监控、高级面试题等热门知识点。Java 工程师必备，书籍结合视频讲解，学习 Java Spring Cloud 微服务架构的必经之路。

目录

Spring Boot2.6 电商网站实战之需求分析和架构设计	5
一、电商网站需求分析	6
二、架构设计	8
三、电商系统模块架构图	8
四、三层架构（前后端分离）	10
五、创建第一个 Spring Boot 项目	11
Spring Boot2.6 电商网站实战之搭建架构连接 MySQL	17
一、三层架构	17
二、使用 Spring Data 简化 MySQL 数据访问	18
三、Spring Data（2.6）实战 MySQL	20
四、Spring Data JPA 框架	21
五、实操代码展示	21
Spring Boot2.6 电商网站实战之 MongoDB5.0 搜索附近的人车物	28
一、移动 App 搜索附近的人的原理	28
二、LBS 搜索附近的 X-解决方案应用技术	28
三、Spring Boot 2.6 实战 MongoDB 5.0 搜索附近的人车物	31
四、NoSQL 排名第一 MongoDB	32
五、安装 MongoDB 5.0 数据库	34
六、Spring Data 实战 MongoDB 数据库	35
Spring Boot 2.6 电商网站实战之 Redis 高并发缓存 6.0	37
一、背景介绍	37
二、Spring Boot 2.6 实战分布式缓存 Redis 6.0	38
三、实战演练 Linux 安装 Redis6.2	42
Spring Boot2.6 电商网站实战之安全机制	51
一、安全机制介绍	51
二、安全漏洞介绍	52
三、Java Spring Security 介绍	53
四、Java 安全框架 Shiro 介绍	54
五、Spring Security Demo 介绍	54
六、WebSecurityConfig 介绍	55
七、Web 全站安全验证配置	55
八、实战代码展示	57

Spring Boot2.6 电商网站实战之 需求分析和架构设计

——侠客
资深架构师

视频链接: <https://developer.aliyun.com/learning/course/903>

目录:

- 一、电商网站需求分析
- 二、架构设计
- 三、电商系统模块架构图
- 四、三层架构（前后端分离）与多层架构设计
- 五、创建第一个 Spring Boot2.6 项目

Spring Boot 是目前应用比较广泛的一种 Java 框架，java 语言亦也是比较普遍的一种编程语言。

目前，企业的应用开发模式主要为前后端分离，并且在开发过程中分成了很多个团队，系统庞大，需求复杂，迭代速度也不同。因此，大多数企业都采用了项目独立的微服务架构。其目的首先是拆解业务系统，独立敏捷开发，适应快速变化的需求迭代；其次是将系统解耦，各个团队独立开发，专注特定业务领域，实现业务拆分，以方便后期的 API 复用；再次，可以分摊风险，保证项目的正常运行，不会因为某一个成员的流失而导致整个项目停摆。

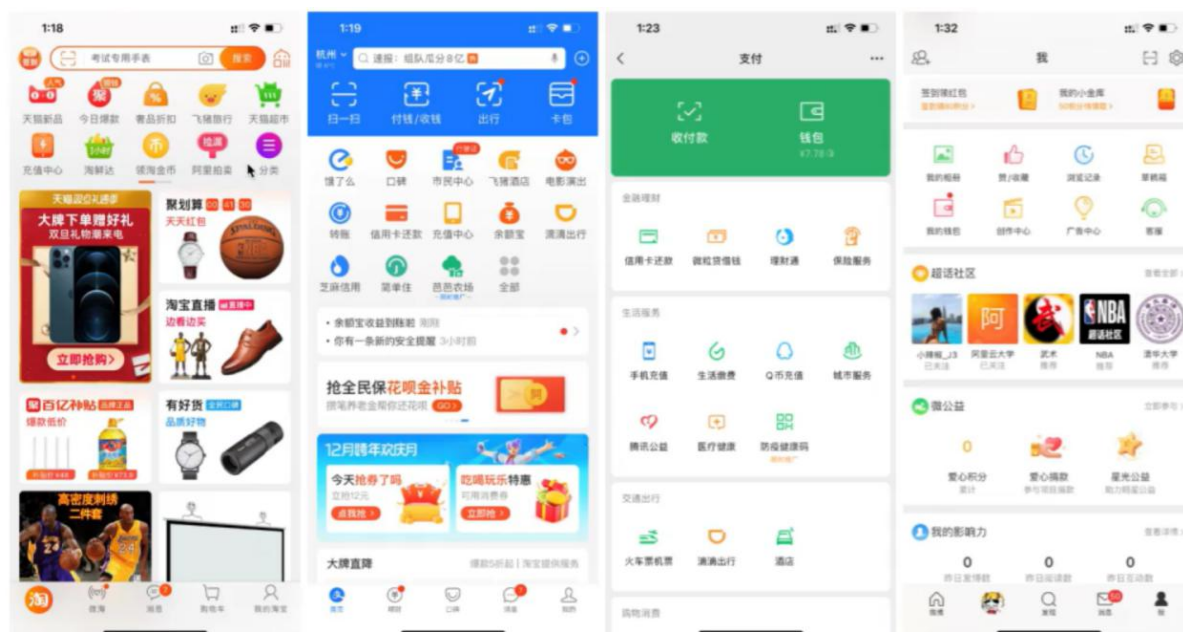
一般而言，创业公司/小公司更倾向于全栈型开发工程师，一个开发人员就能熟练使用前后端全部技术，能够节约成本，后期再慢慢扩充团队。与之相反的大公司的开发模式，主要倾向于复杂的多团队独立开发模式，比如现在流行的微服务架构。大公司系统庞大复杂，分工明确，讲究术业有专攻，只需要开发人员熟练地掌握某一方面的技术即可。

两种方式各有利弊，但是作为学习者，更需要完整地了解架构的整体知识，再根据实际情况去选择适合自己的开发模式。

目前，国内一些比较著名的电商平台都采用了前后端分离的模式，如淘宝、京东、拼多多等。

一、电商网站需求分析

① 电商产品原型：淘宝+支付宝+微信+微博



② 电商网站平台架构



电商网站的需求分析主要包括以下几个部分：

- 用户需求文档
- 产品原型，UI 界面
- 用例 Use Case
- 流程图



整个开发流程中存在以下几种角色：

① **后端开发工程师**，其中 Java 程序员的主要负责后端 API 开发、数据库的设计，此外还有使用 Spring Boot 进行开发定义的后端开发工程师，还可以细分为 DBA 数据库工程师、大数据架构师。架构师在了解完整个项目的需求之后，需要考虑使用什么架构。

② **前端开发工程师**，主要包括 IOS 开发、安卓开发，与后端数据库以及数据库接口一起实现全部的平台功能，其中用户登录、注册、下单都是电商中常见的场景。

③ **项目经理 PM**，主要负责协调整个项目，保证按时交付。

二、架构设计

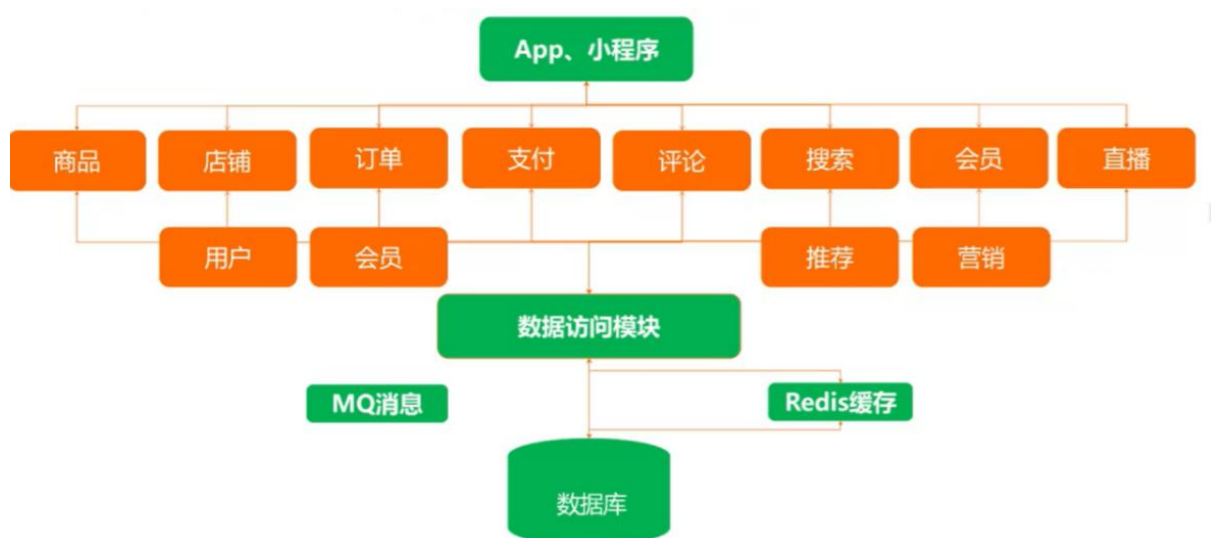
架构设计包括以下几个部分：技术选型、架构设计、三层架构、微服务架构、系统模块图、物理部署图。

曾经的开发模式是由产品经理（老板）将需求告知给技术人员，再由技术人员负责完成功能开发。然而此种模式已经几乎被淘汰，作为技术开发人员，必须掌握扎实的架构基础知识。

微服务架构主要应用于移动互联网公司，需求变化频繁，产品迭代迅速，对用户需求非常敏感。目前最流行最成熟的微服务架构是 Spring Cloud 微服务框架方案。此外，阿里开源的 Dubbo 也是很好的选择，开源技术领域也积攒了足够多的优秀经验和案例。淘宝双 11 正是微服务基于 Java 和 MySQL 数据库开源技术实现的。

阿里为其他互联网公司提供了很好的经验和案例。学习 Java Spring Cloud 微服务架构开发，可以在线观看课程《Spring Cloud 微服务架构设计与开发实战》。官方在线课程地址如下：
<https://developer.aliyun.com/learning/course/60>。

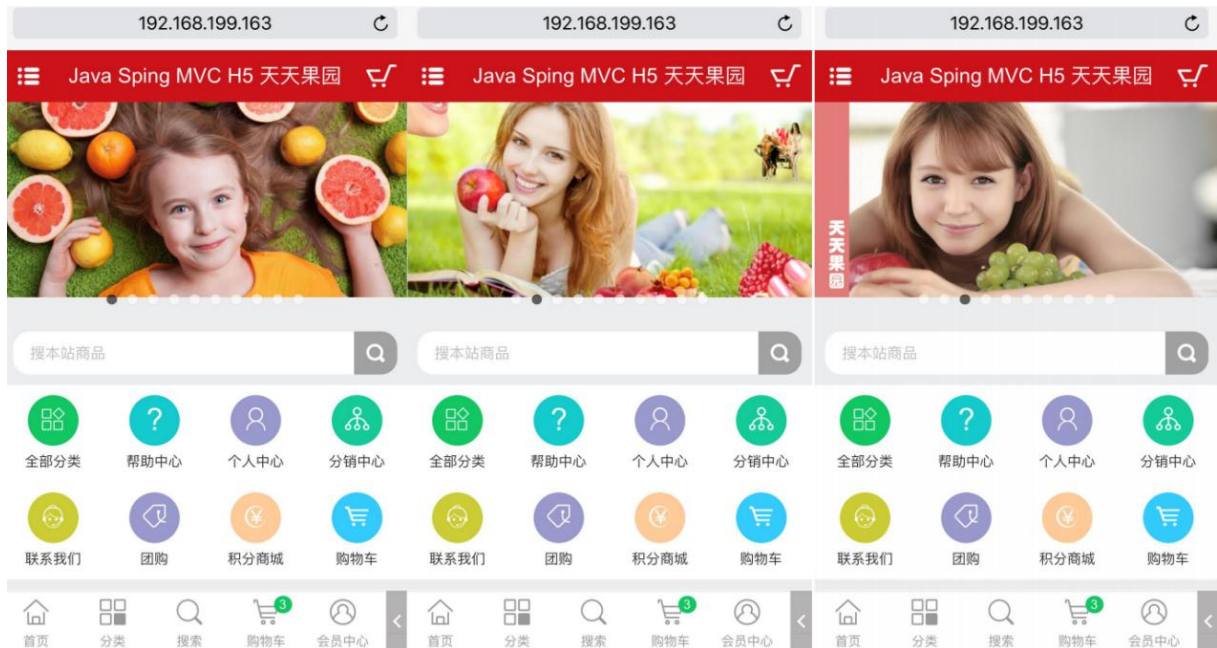
三、电商系统模块架构图



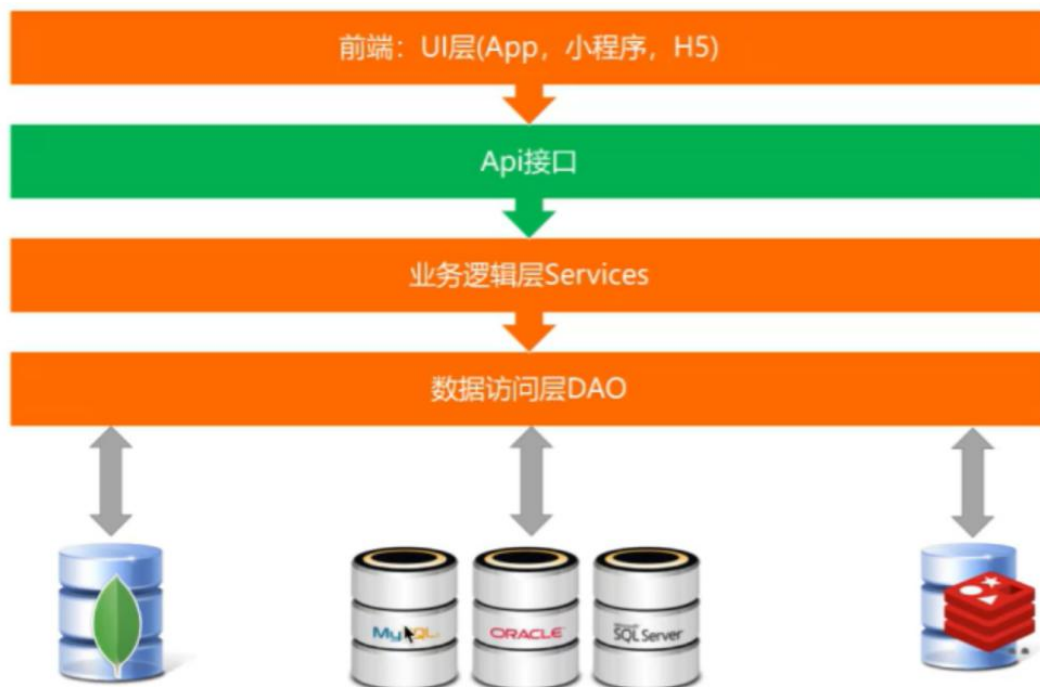
如今很多大学的计算机开发课程比较陈旧，与当下互联网公司的实际应用脱节，没有跟上企业最新的技术需求。从某种角度来看，一些大型互联网公司的技术课程质量不一定比大学差，如京东、今日头条、阿里等，甚至可能更胜一筹，因为他们的技术知识都经过了实践的检验、总结和沉淀。

Spring Boot 可以用来开发电商后台微服务 API，但是它还提供了更好的选择——SpringCloud 微服务。目前，很多 App 的商推荐功能，如淘宝的商品推荐、抖音的视频推荐等，主要依靠推荐算法和大数据的分析功能。上图大致展示了基本的开发流程，但每个电商平台的推荐算法都有所不同。

此外，当今基于大数据的营销推荐可以做到非常精准。可以通过模拟的淘宝电商平台，小公司一般只需要使用三层架构。后续随着业务的发展可以扩展至五层、六层等更复杂的架构。

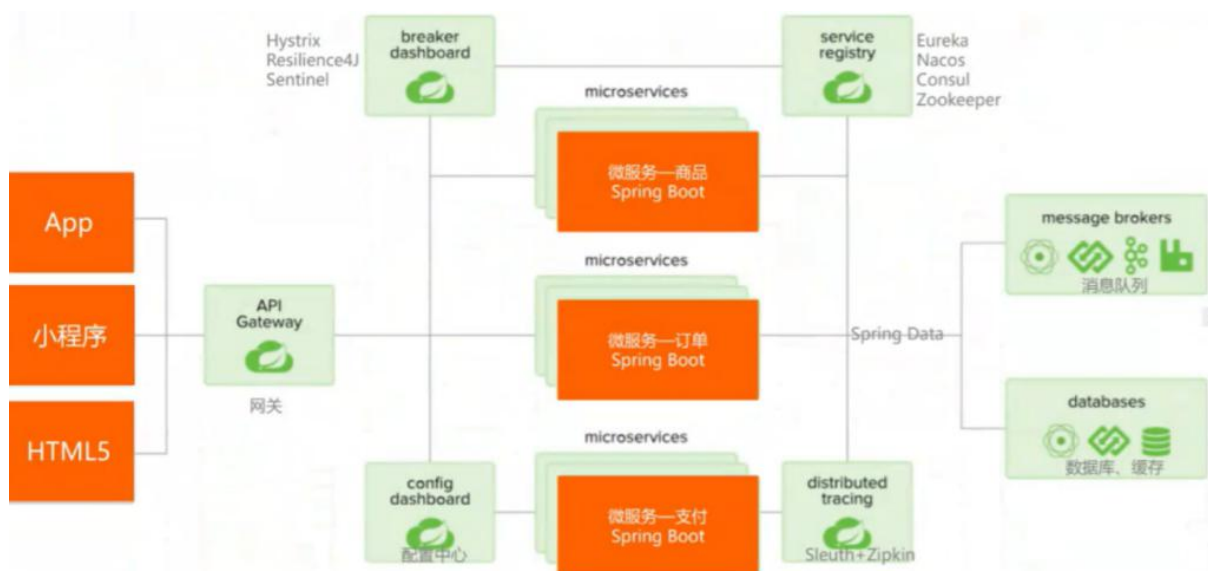


四、三层架构（前后端分离）



可以使用 MongoDB 等数据库来实现地理位置定位以及搜索附近的人。用户使用手机的时候会上传个人的经纬度位置，然后计算出两个人的距离。其中，基于 GEO 地理位置的社交是比较有吸引力的。

中大型网站 Spring Cloud 微服务架构图如下：



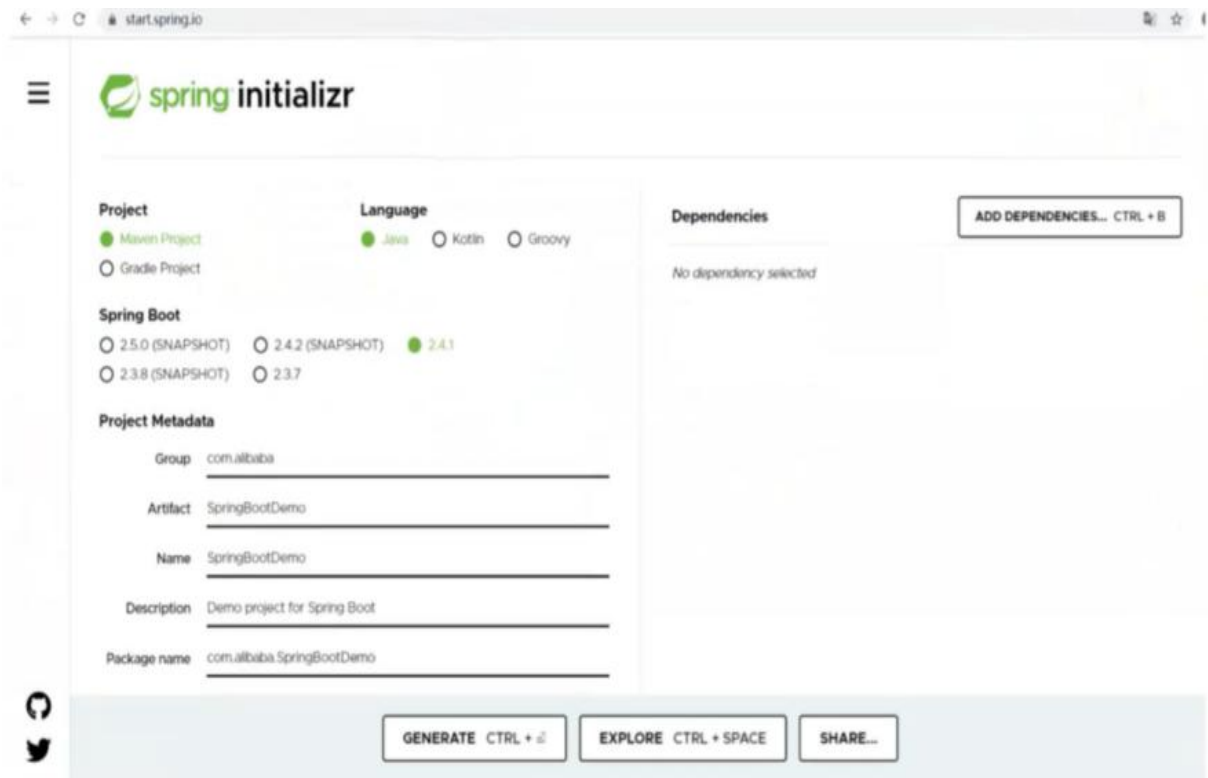
五、创建第一个 Spring Boot 项目

下图为 Spring Boot 开发环境准备：

1. Open JDK 1.8
2. Eclipse 4.6+开发工具
3. 或者IDEA开发工具



1、Spring Boot2.6 实战 Demo



第一步：下载解压缩

下载解压缩

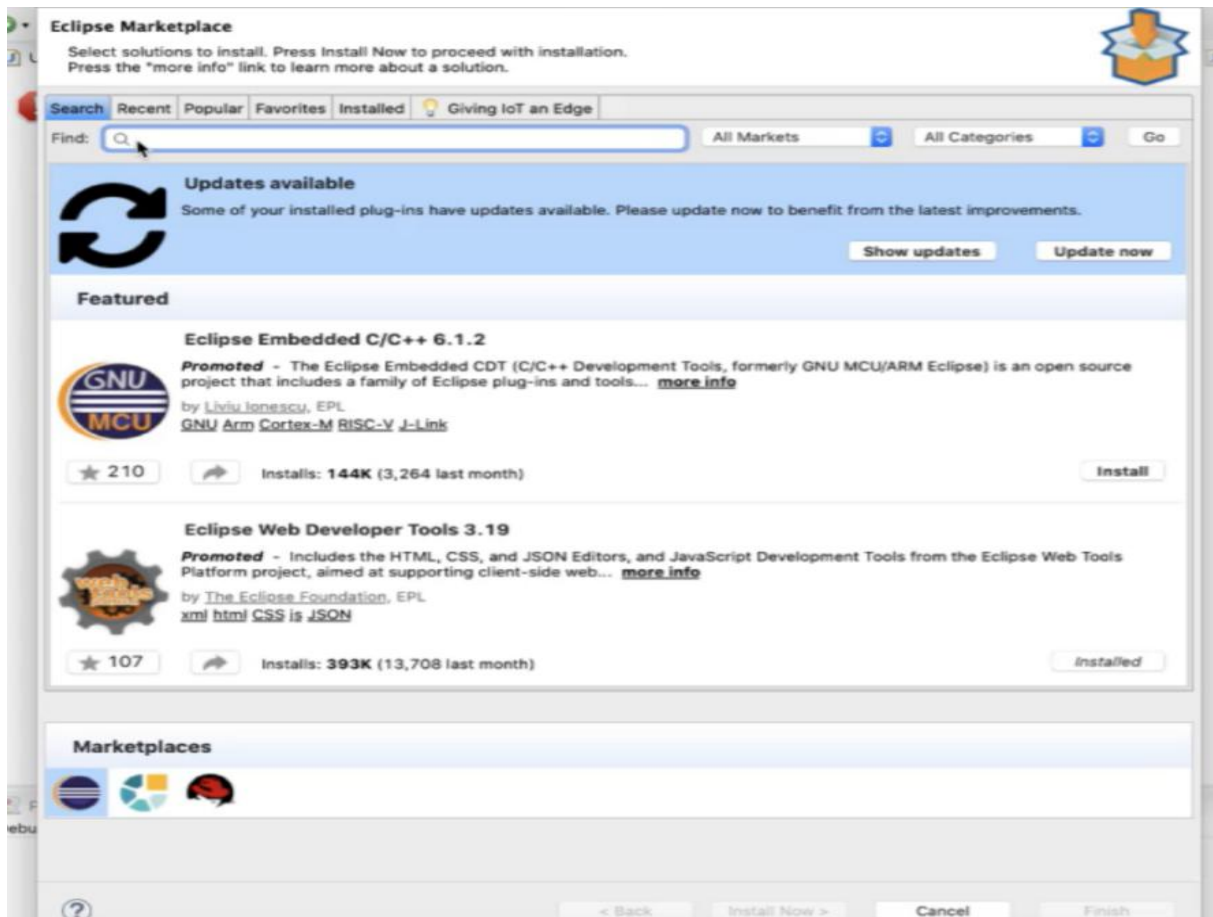
名称	修改日期	类型	大小
.mvn	2019/1/26 5:05	文件夹	
.settings	2019/1/26 13:07	文件夹	
src	2019/1/26 5:05	文件夹	
target	2019/1/26 13:07	文件夹	
.classpath	2019/1/26 13:07	CLASSPATH 文件	2 KB
.gitignore	2019/1/26 5:05	文本文档	1 KB
.project	2019/1/26 13:07	PROJECT 文件	1 KB
mvnw	2019/1/26 5:05	文件	9 KB
mvnw.cmd	2019/1/26 5:05	Windows 命令脚本	6 KB
pom.xml	2019/1/26 5:05	XML 文档	2 KB

第二步：简化配置

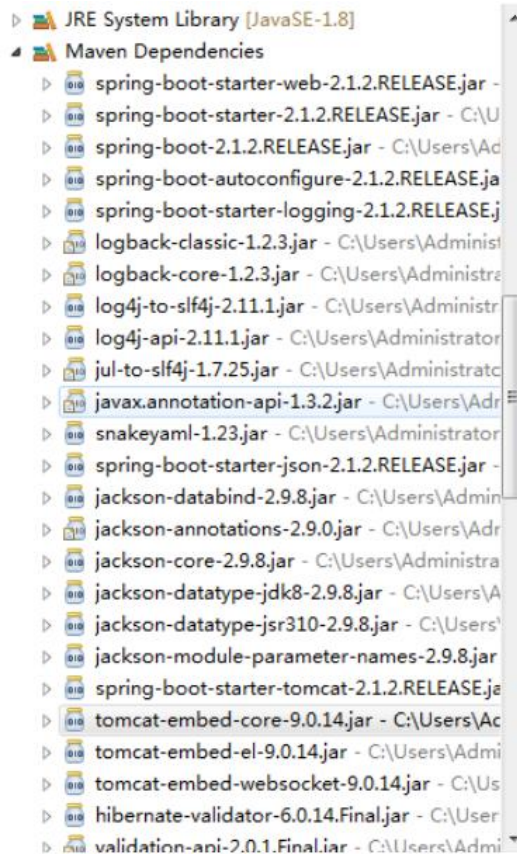
简化配置

```
16
17< properties>
18    <java.version>1.8</java.version>
19</ properties>
20
21< dependencies>
22    <dependency>
23        <groupId>org.springframework.boot</groupId>
24        <artifactId>spring-boot-starter-web</artifactId>
25    </dependency>
26
27    <dependency>
28        <groupId>org.springframework.boot</groupId>
29        <artifactId>spring-boot-starter-test</artifactId>
30        <scope>test</scope>
31    </dependency>
32</ dependencies>
33
34< build>
35    <plugins>
```

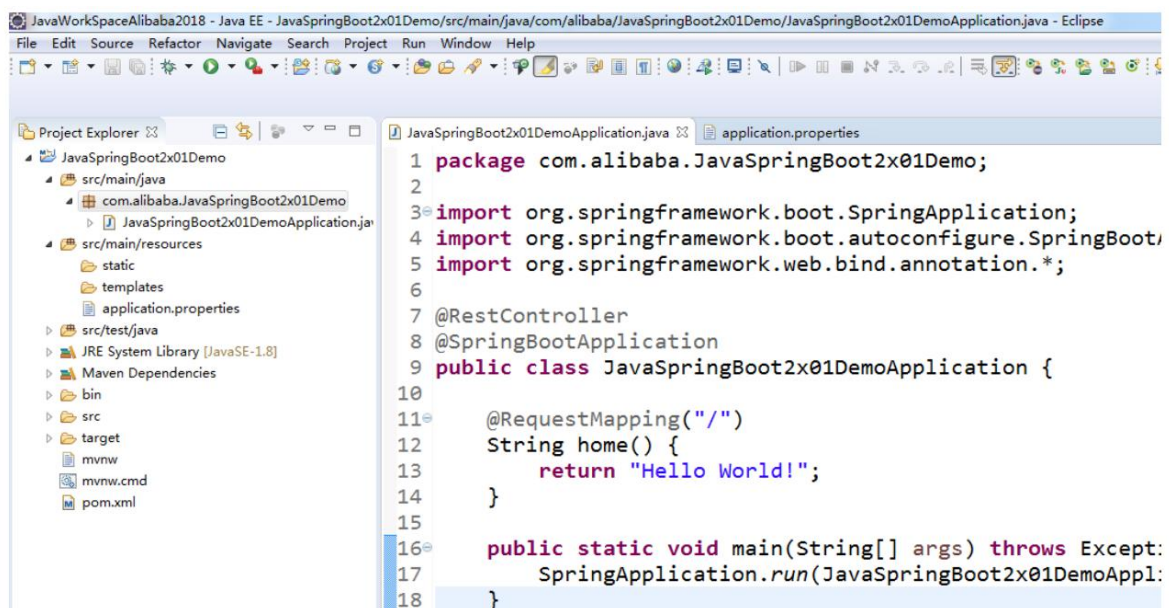
在 Eclipse 工具里搜索 spring tools 安装即可



第三步：内置 Tomcat9.0



2、Spring Boot 2.6 快速开发 REST API



第一步：浏览器测试 API



第二步：选择对应的功能。

第三步：等待容器拉包。

第四步：启动器展示。

第五步：接口层的模拟接口。

具体代码展示如下：

```
@Api(tags = "User 用户相关接口", description = "提供用户相关的 Rest API")
@RestController
@RequestMapping("/user")
public class UsersController {
    @ApiOperation("新增用户接口")
    @PostMapping("/add")
    public boolean addUser(@RequestBody User user){
        return false;
    }
    @ApiOperation("查询用户接口")
```



```
@GetMapping(" /getAll")
public List<User> getAll(){
    List<User> list = new ArrayList<User>();
    for (int i = 0; i < 100; i++){
        User user= new User( );
        user.setId(i + 1);
        user.setName("Java" + i);
        user.setPassword("1234qwer" );
    }
}
```

Spring Boot2.6 电商网站实战之 搭建架构连接 MySQL

——侠客
资深架构师

视频链接: <https://developer.aliyun.com/learning/course/903>

一、三层架构

多数互联网大公司采用移动端+后台 API 服务的架构实现公司平台, 这种前后端分离的架构主要是出于对业务需求、高并发问题的考量。目前数据库以 MySQL 为主, 后期可以根据需要在 NoSQL 数据库 (如 MongoDB、Redis) 中进行扩展, 这是一个循序渐进的过程。将早期的基础架构搭好后, 能够更容易地进行微服务架构升级的改造。

三层架构指的是显示层、业务逻辑层和数据访问层。顾名思义, 每一层都有自己的职责, 这与计算机编码规范相关。其中, 业务逻辑层是由项目代码统一封装业务逻辑, 但由于翻译问题存在一定的概念混淆, 如 API 服务接口早期叫 Web Service, 现在是 Rest Service。

首先需要提前安装 MySQL 以及 MongoDB、Redis。对于 JAVA 开发, 早期可以使用原生 JDBC 或者 Mybatis 等其他框架, 目前 Spring 提供了 Spring Data 开源框架以简化数据库访问, 可以直接在底层使用。在 Spring Data 框架的支持下, 我们可以配置 Template 或者 Repository 来简化数据接口的编写工作, 这样在使用 Java 进行增删改查时, 和数据库的交互会相对简化很多。

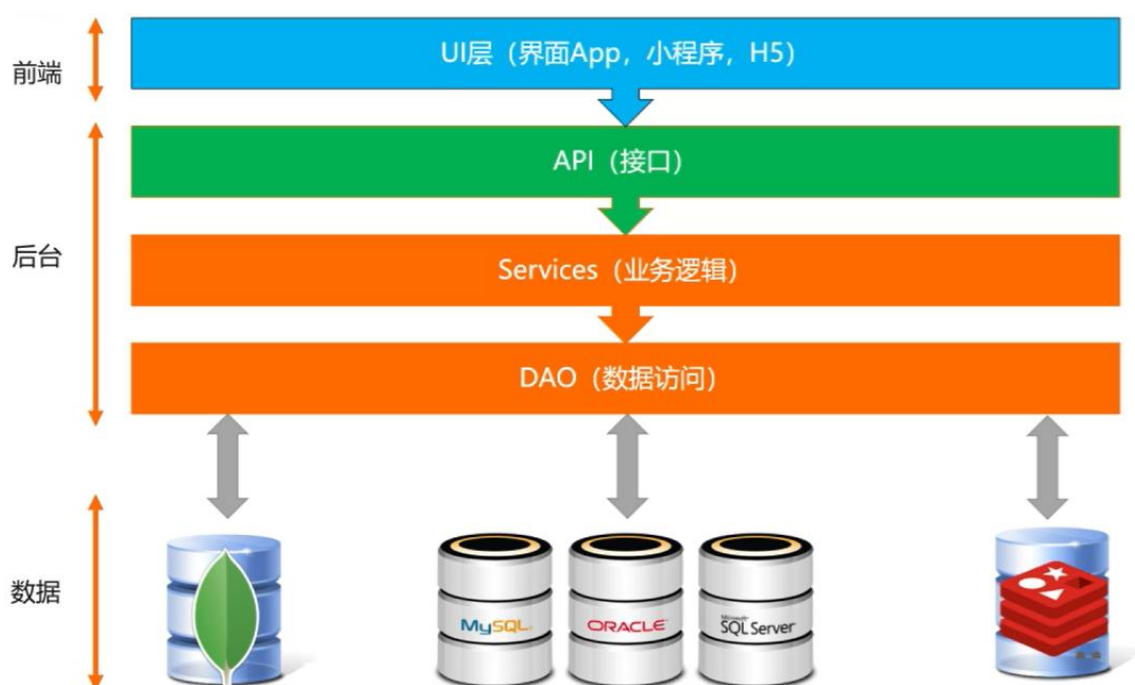
在做微服务时常常会看到一个词——Domain (领域), 它在软件工程领域指的是业务领域, 如金融业务、支付业务、商品业务等, 这些都是单独的业务领域, 每个领域都有自己的规则。

实际项目中, 难以实现覆盖每个数据库, 因此只需要覆盖使用频率比较高的那些数据库,

其他较为生僻的数据库可以在产生需求时再自行学习相关知识，像惯性数据库、非惯性数据库等都有提供相关的学习支持。

首先，以 MySQL 作为例子进行数据库的设计，在创建数据库后设计一个表，比如商品表 Products、用户表 Users 等，每个表中包含一些字段。在设计数据库时一般需要遵循第三范式，但特殊情况下也并不一定要严格执行。

前后端分离主要考虑的是业务规模，包括复杂度、高并发等关键因素。



上图展示的是前后端分离的三层架构——接口层、业务逻辑层、数据访问层，其中每一层都有自己的职责，这与计算机编码规范相关。

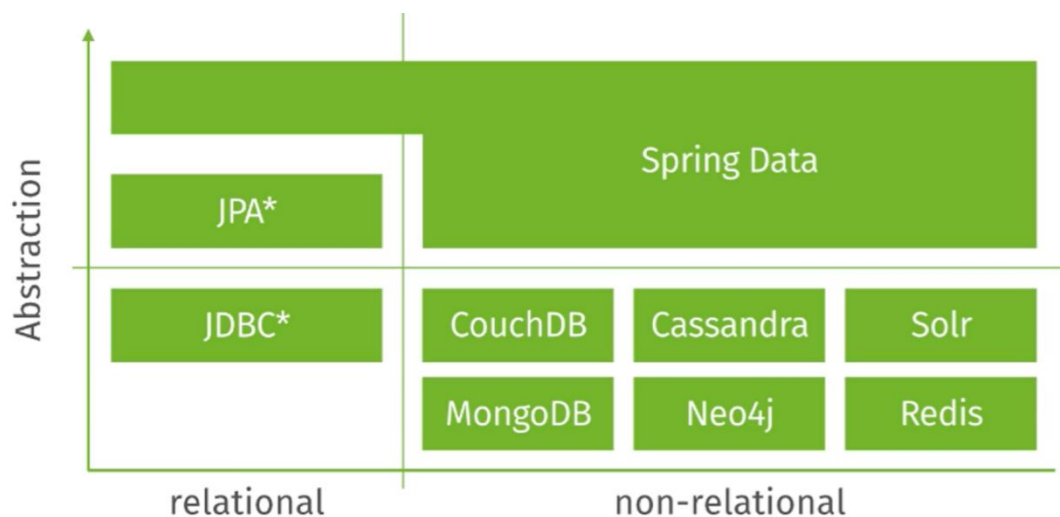
学员们可以提前装好 MySQL、MongoDB 等软件，JAVA 开发可以使用 Spring Data 来优化开发流程，简化和数据库的数据交互。

二、使用 Spring Data 简化 MySQL 数据访问

Spring Data 新特性：

1. 快速数据访问框架，提供统一的编程模型。
2. 强大的 repository 仓储和自定义对象映射 ORM 抽象。
3. 从 repository 方法名称派生动态查询接口。
4. 实现 Domain 域基类提供基本属性。
5. 支持透明审计日志（创建，最后更改）。
6. 可以自定义 repository 代码。
7. 通过 JavaConfig 和自定义 XML 命名空间轻松实现 Spring 集成。
8. 与 Spring MVC 控制器的高级集成。
9. 跨库持久性的实验支持。

Spring Data 架构示意图：

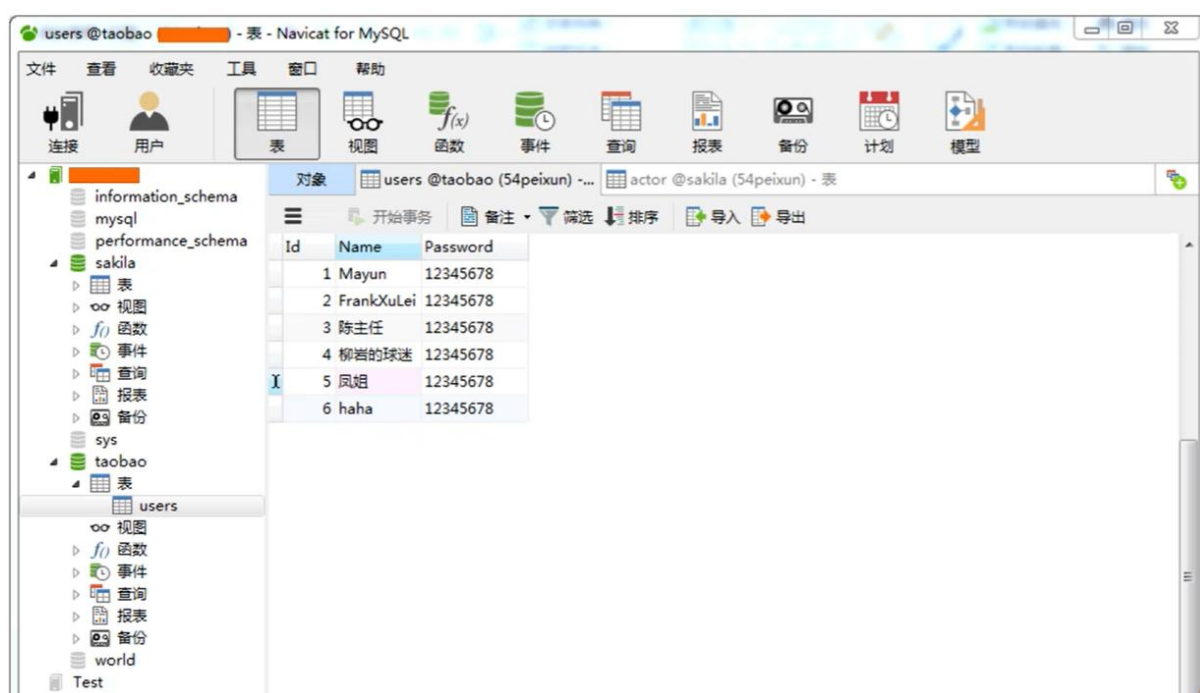


Spring Data 核心模块：

1. Spring Data Commons -支持每个 Spring Data 模块的 Core Spring 概念。
2. Spring Data JDBC -对 JDBC 的 Spring Data 存储库支持。
3. Spring Data JDBC Ext-支持标准 JDBC 的数据库特定扩展，包括对 Oracle RAC 快速连接故障转移的支持，AQJMS 支持以及对使用高级数据类型的支持。
4. Spring Data JPA - JPA 的 Spring Data 存储库支持。
5. Spring Data KeyValue -基于映射的存储库和 SPI，可轻松构建用于键值存储的 Spring Data 模块。

6. Spring Data LDAP -对 Spring LDAP 的 Spring Data 存储库支持。
7. Spring Data MongoDB-基于 Spring 的对象文档支持和 MongoDB 的存储库。
8. Spring Data Redis - 从 Spring 应用程序轻松配置和访问 Redis。
9. Spring Data REST-将 Spring Data 存储库导出为超媒体驱动的 RESTful 资源。
10. Spring Data Apache Cassandra-轻松配置和访问 Apache Cassandra 或大规模，高可用性，面向数据的 Spring 应用程序。
11. Spring Data Apache Geode -轻松配置和访问 Apache Geode, 实现高度一致，低延迟，面向数据的 Spring 应用程序。
12. Spring Data Apache Solr-为面向搜索的 Spring 应用程序轻松配置和访问 Apache Solr。
13. Spring Data Pivotal GemFire-轻松配置和访问 Pivotal GemFire。

MySQL 数据库设计表：



三、Spring Data (2.6) 实战 MySQL

1. Spring JDBC and JdbcTemplate。
2. Spring Data JPA and Hibernate framework。

3. Spring Data 简化连接不同的数据库。
4. 使用 Spring Data JPA 框架连接 MySQL。
5. 当然也可以使用原始的 JDBC。
6. 默认底层使用 Hibernate 框架。
7. 支持 Repository 仓储模式。
8. 引入最重要的 2 个包。
9. spring-boot-starter-data-jpa。
10. MySQL-connector-java。

注意：第十个依赖（MySQL-connector-java）非常重要。

四、Spring Data JPA 框架

Spring Data JPA 框架具有以下特性：

- Spring Data JPA 简化数据访问层的开发工作。
- 基于 Spring 和 JPA 构建存储库的完美支持。
- 支持 Querydsl 谓词，从而支持类型安全的 JPA 查询.Domain 类的透明审核。
- 分页支持，动态查询执行，集成自定义数据访问代码的能力。
- 在引导时验证@Query 带注释的查询。
- 支持基于 XML 的实体映射。
- 引入@EnableJpaRepositories，基于 JavaConfig 的存储库配置。

五、实操代码展示

1) 快速创建 Spring Boot 项目的两种方式：

第一种，在浏览器中输入网址 start.spring.io 进入官网，勾选 Maven Project ，Language 栏选择 Java，最后选择好对应的版本，加入对应的依赖即可，点击 GENERATE CTRL 将项目打包好，导入对应的包即可使用。

第二种，点击 File，选择 New，勾选 Spring Starter Project，在弹窗内修改项目名称，在 Available 中搜索 Web，MySQL Driver 并加入对应的依赖，同时，勾选 SpringBootDevTools，

点击 Finish 即可完成项目的创建。

2) 项目代码展示：

- 启动器：

```
package com.alibaba;

import org.springframework.boot. CommandLineRunner;□

@SpringBootApplication
public class Application implements CommandLineRunnerr
public static void main(String[] args) {

    SpringApplication.run(Application.class, args);
}

@Override
public void run(String.. . argo) throws Exception {

    // TODO Auto-generated method stub
    system.out.println("阿里巴巴 Java Spring Boot 2.5 实战开发课程");
}
}
```

- 在 API 接口处创建一个 Usercontroller
- 在 DAO 层下创建一个 UserDAO 和 UserRepository 类
- 在 Services 层下创建一个 UserService

- 创建一个 Entity 包，在其包下创建一 User 的实体类
- 进入数据库查看对应的字段
- 实体类 User 代码展示如下：

```
//两种方法生成 get set 方法
//使用快捷键一键生成
//加入 Lombok Config 依赖，再添加@Data 注解
package com.alibaba.entity;

//@Data 使用该注解可以不用生成 get set 方法
public class User {
    private Integer id;
    private String name;
    private String password;
}

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName(){
        return name;
    }

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

    public String getPwd() {
        return pwd;
    }
}
```

```
    }

    public void setPwd(String pwd) {
        this.pwd = pwd;
    }
}
```

- 在 **Application.properties** 下添加如下配置：

```
spring.application.name=SpringBoot260Demoserver.port=8088
spring.datasource.url=jdbc:mysql://localhost/alibaba?useSSL=false&serverTimezo
spring.datasource.username=root
spring.datasource.password=1234qwer
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect
```

- **UserRepository** 类代码展示：

```
package com.alibaba.repository;

import java.util.List;

public interface UsersRepository extends CrudRepository<Users, Integer> {

    @Query( "select u from Users u where u.name =?1")

    public List<Users> getUserByName(@Param( "name" )String name);

    @Query("select u from Users u where u.name=?1 and u.password=?2")
    public Users getUserByNameAndPassword(String name, String password);

    @Query ( "select u from Users u where u.id =?1")
```

```
public Users getUserById(int id);  
}
```

- 在 pom.xml 中添加以下依赖：

```
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>
```

- 刷新 Maven 更新 jar 包
- UserController 代码展示如下：

```
package com.alibaba.api;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import com.alibaba.entity.Users;  
  
import com.alibaba.repository.UsersRepository;  
  
@RequestMapping("/getAll")  
@RestController  
public class UsersController {  
    @Autowired UsersRepository userRepository;  
  
    @RequestMapping("/getAll") public List<Users> getAll(){  
        return userRepository.getAll();  
    }  
}
```

```
}  
}
```

- 启动项目查看是否报错
- 进入浏览器输入以下指令：

Localhost: 8088/users/getAll

- UserDAO 类的代码展示如下：

```
package com.alibaba.dao;  
import java.util.List;  
import com.alibaba.entity.Users;  
import com.alibaba.repository.UserRepository;  
  
public class UsersDAO {  
    @Autowired UserRepository userRepository;  
    public List<Users> getAll(String name) {  
        return userRepository.getUserByName(name);  
    }  
}
```

- 进入 UserService 层更新代码：

```
Import org.springframework.beans.factory.annotation.Autowired;  
  
import com.alibaba.dao.UsersDAO;  
import com.alibaba.entity.Users;  
  
public class Userservice {  
    @Autowired UsersDAO userDAO;
```

```
public List<Users> getAll(String name) {  
    return userDao.getUserByName( name) ;  
}  
}
```

- **进入 UserController 层更新代码:**

```
@RequestMapping(" /users")@RestController  
public class UsersControllererr  
  
    @Autowired UserService;  
    @RequestMapping( "/getAll/{name}")  
    public List<Users> getAll(@Param( "name" ) String name) {  
        [ / String name ="java";  
        return userService.getAll( name) ;  
    }  
}
```

Spring Boot2.6 电商网站实战之 MongoDB5.0 搜索附近的人车物

——侠客
资深架构师

视频链接: <https://developer.aliyun.com/learning/course/903>

一、移动 App 搜索附近的人的原理

移动互联网时代下，出现了许多基于地理位置 LBS 的 App。比如高德地图 App 可以定位当前位置，也可以搜索附近的加油站；美团、饿了么 APP 可以搜索外卖店铺，并且可以按照距离排序；许多交友类 App 也具备“附近的人”搜索功能；以及出行类的滴滴打车、货拉拉、哈罗单车等。

此类平台的后台都有着存储目标任务或店铺坐标数据的功能。以 MongoDB 为例，它使用了经典的 GEO 哈希算法来搜索附近的人或车辆，支持地理位置 GEO 搜索，支持平面索引和三维球面体系数据的搜索。且可以直接使用它来存储当前用户的经纬度。

对于同城或 100 公里以内范围的搜索，GEO 哈希搜索算法的性能能够基本满足需求。但是在范围比如 500 公里以上的、涉及到地球表面弧度问题的搜索，结果就可能出现偏差，因此一般建议使用球体距离计算算法来获取更准确的距离，但不是所有的数据库都支持三维球面距离索引。

二、LBS 搜索附近的 X-解决方案应用技术

基于位置服务（Location Based Services, LBS）主要的算法包含以下几种：

- MongoDB

- Redis
- ElasticSearch
- MySQL:SPATIALINDEX
- PostgresqlPostGis 索引

 阿里云 开发者训练营

LBS搜索附近的X—GEO算法

1. Geohash,地图分割不同区域, 二维坐标生成一个字符串

geohash码长度	宽度	高度
1	5,009.4km	
2	1,252.3km	624.1km
3	156.5km	156km
4	39.1km	19.5km
5	4.9km	4.9km
6	1.2km	609.4m
7	152.9m	152.4m
8	38.2m	19m
9	4.8m	4.8m
10	1.2m	59.5cm
11	14.9cm	14.9cm
12	3.7cm	1.9cm

MongoDB 不仅提供了二维和三维的球面体系, 还可以指定多边形的搜索策略。

 阿里云 开发者训练营

LBS搜索附近的X—Redis支持

1. Redis 3.2提供了基于GeoHash和数据结构Zset地理位置功能
2. GEOADD: 新增位置 (纬度、经度、名字) 添加到指定的key;
3. GEOPOS: 从key里面返回对象的位置 (经度和纬度);
4. GEODIST: 返回两个给定位置之间的距离;
5. GEOHASH: 返回一个或多个位置对象的Geohash表示;
6. GEORADIUS: 以给定经纬度为中心, 半径内所有位置对象;
7. GEORADIUSBYMEMBER: 以给定的对象为中心, 半径内所有位置对象。
8. Redis 6.2 版本为 Geo 新增了 GEOSEARCH 和 GEOSEARCHSTORE 指令, 阿里云 矩形区域查询

(-) 阿里云 开发者训练营

LBS搜索附近的人—MongoDB支持GEO索引

1. GEO地理数据相关的索引有两种 2dsphere 和 2d
2. 2d 支持平面搜索，2dsphere主要支持球面
3. 地理位置存储为GeoJSON Point类型
4. \$geoNear用于聚合查询
5. 普通查询
6. db.createCollection("users")
7. db.users.save({ _id: "jack", position: [0.1121, -0.1211]})
8. db.users.save({ _id: "musk", position: [1.1231, 1.12312]})
9. db.users.ensureIndex({position: "2d"})
10. db.users.find({position: { \$near: [0,0], \$maxDistance: 10 } })

(-) 阿里云 开发者训练营

LBS搜索附近的人—MongoDB支持

1. MongoDB主要两种地理空间索引 2dsphere 和 2d。
2. 两种索引的底层依然是基于Geohash来进行构建的。
3. 但与国际通用的Geohash还有一些不同，具体参考官方文档。
4. 2dsphere 索引仅支持球形表面的几何形状查询。
5. 2d 索引支持平面几何形状和一些球形查询。
6. 2dsphere索引和相应的操作符来寻找附近的人
7. geoNear命令搜索附近的点
8. GeoJSON对象类型存储位置信息经纬度信息，支持类型：
9. Point点
10. LineString线
11. Polygon多边形
12. MultiPoint多点
13. MultiLineString多线
14. MultiPolygon多面体
15. GeometryCollection地理集合

(-) 阿里云 开发者训练营

搜索附近的X—SpringBoot+MongoDB解决方案

- <dependency>
- <groupId>org.springframework.boot</groupId>
- <artifactId>spring-boot-starter-data-mongodb</artifactId>
- </dependency>

```

/**
 * 2D圆形查询,geoNear查询点附近的所有车辆，指定半径
 * @param point 中心点
 * @param radius 半径，米
 * @param limit 限制数量
 * @return
 */
public List<CarInfo> geoNearQuery(Point point,double radius,int limit) {
    List<CarInfo> result =
        mongoTemplate.find(new Query(
            Criteria.where("location")
                .near(point)
                .maxDistance(radius))
                .limit(limit)
                ,CarInfo.class);
    return result;
}

```

安装好 MongoDB 后，首先要建立一个基于经纬度的数据记录作为索引，供客户端调用。客户端 APP 调用索引后，即可很快查询到所需数据。

但这里还需要指定半径 r 。在二维平面上，搜索圆圈内的点才有意义。基于半径对搜索数据提前进行过滤，能够优化搜索性能。

比如共享单车一般搜索附近 200 米或 500 米以内的车辆，距离太远用户也不会考虑使用。在规定半径内搜索不到车辆时，则会增加半径。搜索范围越广，搜索速度越慢。除了限制搜索半径，还需要限制返回的数据。比如限制返回最多一百辆或最多十辆的数据。

此外，比如打车平台的后台有人工或辅助程序的订单调度系统。司机可以主动刷新订单列表，搜索附近用户的订单，也可以通过 APP 平台派单给附近的司机。派单策略比较复杂，可以基于司机等级、信用、好评、当天累计单量、车辆是否空载等因素来综合决策。

三、Spring Boot 2.6 实战 MongoDB 5.0 搜索附近的人车物

阿里云 开发者训练营

《阿里巴巴MongoDB4.0高级实战》

1. MongoDB数据库入门：MongoDB概览、4.0新特性、下载安装、Shell连接及基本操作等
2. MongoDB数据库数据查询与分析：MongoDB查询命令、分析、聚合等
3. MongoDB数据库核心知识：MongoDB数据库操作、集合、存储引擎、数据模型等
4. MongoDB数据库管理：MongoDB数据相关操作、数据备份与恢复等
5. MongoDB数据库性能分析与调优：MongoDB索引、算法、查询计划等
6. MongoDB与Java开发实战：基于Java Spring Boot和云数据库MongoDB开发HTML5博客应用
7. MongoDB数据库排错日志分析：MongoDB日志收集、经典问题解析、常用问题排查工具等
8. MongoDB数据库安全机制：MongoDB身份验证、加密、典型机制等
9. MongoDB数据库HA高可用集群架构：主从复制、读写分离、自动化故障转移、HA集群，阿里云数据库集群等
10. MongoDB数据库运维：数据库维护、升级，云数据库MongoDB版运维、监控工具，数据库容灾方案等
11. MongoDB优化实战案例：讲解MongoDB的索引原理，以及常见的优化手段，并分析一些具体的优化案例
12. MongoDB Sharding集群原理与架构优化：讲解Sharding集群原理、架构设计方法，以及常见的架构优化手段
13. 官方网站：<https://edu.aliyun.com/workshop/3/course/1044>



四、NoSQL 排名第一 MongoDB

MongoDB 属于 NoSQL 数据库（文档型数据库），存储单位是 Document（文档）。SpringData for MongoDB 可以直接集成到项目，它提供了 MongoDB 仓储的接口，简化了 CRUD 操作接口，直接生成访问对象，简化数据库开发工作。

MongoDB简介

阿里云 开发者训练营

1. NoSQL排名第一，BAT互联网公司必备
2. 分布式数据库，
3. 由C++语言编写，特点是高性能、易部署、易使用、存储数据非常方便，
4. 旨在为Web应用提供可扩展的高性能数据存储解决方案
5. MongoDB由10gen团队所开发，于2009年2月首度推出
6. MongoDB开源、跨平台，
7. 支持 Windows、Linux、OS X和Solaris系统
8. MongoDB最新版本为4.0，支持跨文档事务



MongoDB优点

阿里云 开发者训练营



Spring Data for MongoDB 整个数据访问层的开发工作已经愈发简单，可以称得上是“傻瓜式开发”。

MongoDB版本特性

版本	关键特性	建议
2.X	index、writeConcern、readPreference	强烈建议升级
3.0	Pluggable Storage Engine、Wiredtiger、improved mmapv1	建议升级
3.2	Raft 协议、文档校验、部分索引、inMemory、\$lookup	建议升级
3.4	并行复制、sharding迁移改进、collation、\$facet、\$graphLookup	强烈建议使用
3.6	安全、并行性能、\$lookup、Online 维护（在线 oplog维护、在线添加认证）	已经发布
4.0	分布式事务Transaction	已经发布

首先，启动 MongoDB 数据库服务。仓储模式需要配置文件和数据库地址。Mongo shell 打开一个命令行，对接 MongoDB 的服务链接，发送给客户端，随后直接启动即可。启动以后，MongoDB 会自动创建一个日志文件和数据文件。此外，它还提供了可视化的管理界面。相比于 MySQL 等关系型数据库，MongoDB 并不需要提前创建数据库，使用上更加灵活。

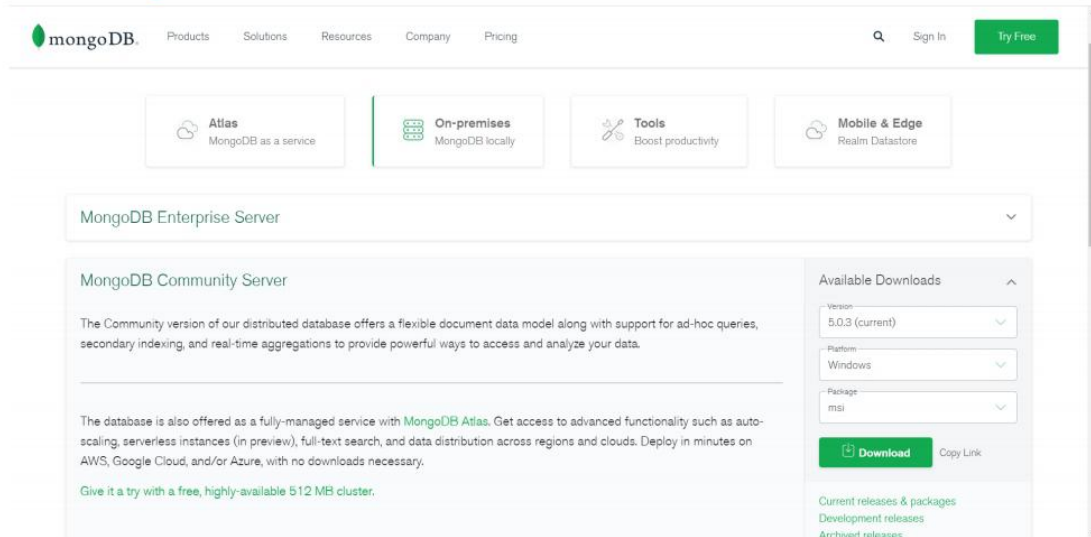
那么，仓储层如何与数据库进行交互？首先，保存一个自定义对象。比如搜索附近的车辆的属性有车的名字、经纬度，也可以加一些编号比如车牌号、车的颜色、司机的名字、司机的手机号等。

接下来是与 MongoDB 进行对接。定义一个 user 或者汽车 car 的类型，搜索的时候会涉及到对二维数据的过滤问题。简单的数据可以直接用仓储进行，稍微复杂一点数据可以通过自定义参数查询接口过滤后再进行封装

五、安装 MongoDB 5.0 数据库

下载MongoDB5.0

阿里云 开发者训练营



安装MongoDB5.0

阿里云 开发者训练营



六、Spring Data 实战 MongoDB 数据库

(-) 阿里云 开发者训练营

Spring Data 2.6 MongoDB新特性

1. 简化Java的MongoDB数据库开发API
2. 提供一致的基于Spring的编程模型
3. Spring configuration 支持@Configuration classes or an XML namespace
4. 方便编写Repository仓储模式的DAO层代码
5. 自动实现Repository interface的CRUD常用操作
6. 自动进行POJO和MongoDB文档数据的映射转换, Spring' s Conversion Service
7. 可以自定义扩展方法
8. MongoTemplate helper class 提升MongoDB开发效率
9. Low-level mapping using MongoReader/MongoWriter abstractions
10. Java based Query, Criteria, and Update DSLs
11. Log4j log appender
12. GeoSpatial集成
13. Map-Reduce 集成
14. JMX administration and monitoring
15. CDI support for repositories
16. GridFS 支持

(-) 阿里云 开发者训练营

Repository仓储层代码

```
public interface BlogRepository extends MongoRepository<Blog, ObjectId> {  
    public Blog findById(ObjectId id);  
    public void delete(ObjectId id);  
    public List<Blog> findAll();  
}
```

代码实操演示

首先加入 MongoDB 的依赖:

```
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-data-mongodb</artifactId>  
</dependency>
```

从 [MongoDB 官网](#) 下载 MongoDB=》选择社区版的 MongoDB (MongoDBCommunityServer)。下载完后在 Windows 上启动使用, 进入 bin 目录找到 mongo.exe (客户端命令)、mongod.exe (MongoDB 的数据库服务命令)、mongos.exe (集群路由器)

启动 MongoDB 流程如下：

- ① 打开命令行界面，进入 MongoDB 的 bin 目录 (cd\MongoDB\bin) 后，输入如下指令：

```
mongod.exe--port27017--dbpath="..\data"--logpath="..\log/mongo.log"
```

注：linux 和苹果操作系统与 Windows 的操作同理。

- ② 在命令行界面输入一下指令可查看当前持有的数据库：

```
Show dbs
```

- ③ 在 MavenDependencies 中可添加对应的 MongoDB 修改包：

```
spring.application.name=SpringBoot260Demo
server.port=8088
spring.datasource.url=jdbc:mysql://localhost/alibaba?useSSL=false&serverTimez
spring.datasource.username=root
spring.datasource.password=1234qwer
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.databaseplatform=org.hibernate.dialect.MySQL5InnoDBDialect
#mongodb
spring.data.mongodb.database=alibaba
spring.data.mongodb.host=localhost
spring.data.mongodb.port=270172
#spring.data.mongodb.username=root
#spring.data.mongodb.password=1234qwer
```

- ④ 在 entity 包中创建多个实体类进行测试。

Spring Boot 2.6 电商网站实战之 Redis 高并发缓存 6.0

——侠客

资深架构师

视频链接：[【1024 创造营】基于 Mysql 和 MongoDB 的 Java Spring Boot 2.6.0 电商网站开发实战-学习中心-阿里云开发者社区 \(aliyun.com\)](#)

一、背景介绍

当前的互联网公司比如淘宝、各直播平台、支付宝、微信、微博等都在大量使用 Redis 缓存来解决电商系统的高并发问题。以最典型的淘宝的双 11 为例，并发峰值订单能够达到 58 笔每秒。这个并发值在全世界的电商平台里都是第一位。此前传统的高并发方案大部分是基于 IOE 实现的，而完全使用开源技术解决极大规模的电商高并发问题，阿里是第一家。

Redis 在高并发系统中属于非常重要的开源技术，经常被用来解决热点数据缓存类的场景，强大到几乎没有竞争对手。

有一种说法是“Redis 是单线程，使用单线程实现高并发”，从严格意义上说，它是不准确的，Redis 整体上是多线程模式。Redis 本质上是内存读写操作，因为它是基于内存的来进行存取数据的。所以读取速度比基于磁盘 IO 要快很多，当然这也与数据结构算法有关。

理论上，如果涉及到分布式架构，还需要考虑另外一个问题：请求消息的数量 QPS 和请求数据的大小。比如需要经过网络传输的时候，网络本身也要消耗资源。如果请求来自于不同的客户端，与服务器建立连接的时候会对服务器造成比较大的压力。而同一个节点、同一个客户端，在很短的时间内发送重复的请求，性能则会高很多。

所以压力测试的客户端和网络分布也会影响测试的结果。一般的测试都是来自于同一个客户端模拟压力测试的场景。但双十一或其他某些高并发的场景，实际用户可能来自于全国各地。

淘宝使用的缓存是 Tair，并且阿里对此进行了一部分优化改进。

Redis 提供了完善的客户端驱动和强大类型的系统，能够灵活扩展。支持丰富的数据类型，方便用户处理不同的数据，主要支持以下几种数据类型：

1. 字符串 Strings：二进制存储、图片、对象。
2. 字符串列表 Lists。
3. 无序不重复的字符串集合 Sets。
4. 有序不重复的字符串集合 Sorted Sets。
5. 键、值都为字符串的哈希表 Hashes。
6. 其它复杂的数据结构。
7. 辅助索引 Index 和 GEO 地理位置索引。
8. Stream 流数据类型。
9. ReJSON（特殊的 JSON 数据类型）。

此外，Redis 6.0 新引入了 SSL、RESP3 协议、ACL、客户端缓存、无磁盘复制、I/O threads 等功能。其中，RESP 全称 REdis Serialization Protocol，是 Redis 服务端与客户端之间通信的协议。

二、Spring Boot 2.6 实战分布式缓存 Redis 6.0

Java Spring Data 2.x for Redis新特性

阿里云 开发者训练营

1. 支持多种Redis驱动程序/连接器的低级抽象（Jedis和Lettuce。JRedis和SRP过期）
2. Spring Data Access exception和Redis driver exceptions转换
3. RedisTemplate高级抽象封装Redis操作，异常转换和序列化工作
4. Pubsub发布订阅模式支持（例如消息驱动POJO的MessageListenerContainer）
5. 支持Redis Sentinel和Redis Cluster集群模式
6. JDK, String, JSON和Spring Object / XML映射序列化器
7. 基于Redis的JDK Collection实现
8. Atomic counter原子计数器
9. Sorting and Pipelining功能
10. 专门API支持SORT, SORT / GET模式和返回批量值数据
11. Redis实现了Spring 3.1缓存抽象
12. 自动实现Repository接口，@EnableRedisRepositories支持自定义查找方法
13. 支持存储库的CDI

Spring Data for Redis 是 Redis 缓存的快速开发框架，封装了简单友好的 API，并且基于 2.0 版本做了许多改进：

1. 支持多种 Redis 驱动程序/连接器的低级抽象（Jedis 和 Lettuce。JRedis 和 SRP 过期）。
2. 支持 Spring Data Access exception 和 Redis driver exceptions 转换。
3. 支持 RedisTemplate 高级抽象封装 Redis 操作，异常转换和序列化工作。
4. 提供了 Pubsub 发布订阅模式支持（例如消息驱动 POJO 的 MessageListenerContainer）。
5. 支持 Redis Sentinel 和 Redis Cluster 集群模式。
6. 支持 JDK，String，JSON 和 Spring Object / XML 映射序列化器。
7. 基于 Redis 实现了 JDK Collection。
8. 支持 Atomic counter 原子计数器。
9. Sorting and Pipelining 功能。
10. 专门 API 支持 SORT、SORT / GET 模式和返回批量值数据。
11. 实现了 Spring 3.1 缓存抽象。
12. 自动实现 Repository 接口，@EnableRedisRepositories 支持自定义查找方法。
13. 支持存储库的 CDI。

RedisConnection 解析

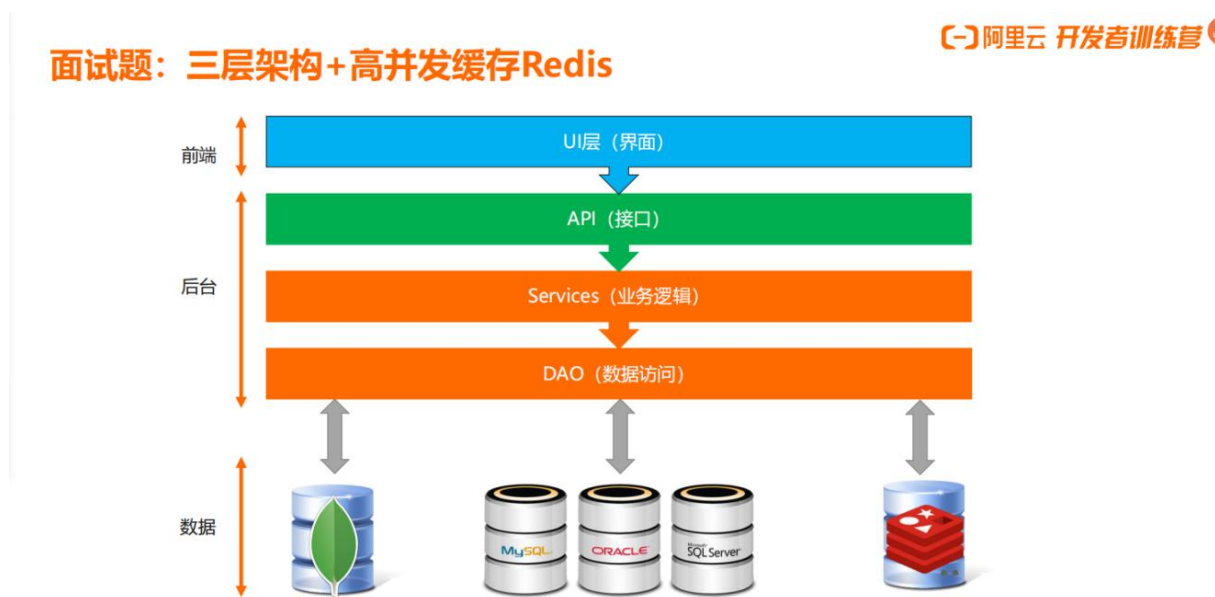
阿里云 开发者训练营

1. RedisConnection 为 Redis 通信提供核心组件
2. 处理与 Redis 服务器后端的通信。
3. 自动将底层连接异常转换为 Spring DAO 异常
4. 可以在不更改任何代码的情况下切换连接器，
5. 操作语义保持不变。
6. 统一接口
7. 工厂模式
8. 仓储模式

RedisConnection 对象提供了对 Redis 网络连接的封装，并且简化了统一的接口功能定义，主要包括以下功能：

1. RedisConnection 为 Redis 通信提供了核心组件。
2. 处理与 Redis 服务器后端的通信。

3. 自动将底层连接异常转换为 Spring DAO 异常。
4. 可以在不更改任何代码的情况下切换连接器。
5. 操作语义保持不变。
6. 统一的接口。
7. 工厂模式。
8. 仓储模式。



安装 Redis 6.0 服务，不支持 windows，推荐使用 linux 环境。直接在官网下载并启动即可。

Spring Data 2.x for Redis 提供了友好的接口方便用户操作 Redis 缓存服务器，可以直接在 Spring Boot 项目中集成，使用方式与配置 MongoDB 的项目依赖类似。

Spring Data 2.x for Redis

阿里云 开发者训练营

1. Upgrade to Java 8.
2. Upgrade to Lettuce 5.0.
3. Removed support for SRP and JRedis drivers.
4. Reactive connection support using Lettuce.
5. Introduce Redis feature-specific interfaces for RedisConnection.
6. Improved RedisConnectionFactory configuration with JedisClientConfiguration and LettuceClientConfiguration.
7. Revised RedisCache implementation.
8. Add SPOP with count command for Redis 3.2.

阿里云 开发者训练营

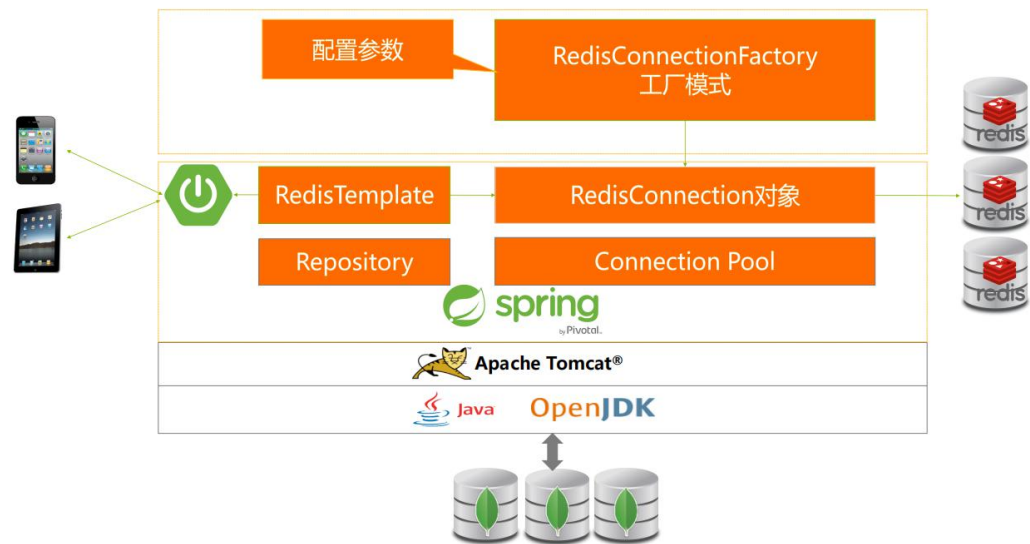
```
org.springframework.data.redis.c  
ore
```

Redis API

阿里云 开发者训练营

- org.springframework.data.redis.connection包
- RedisConnection
- RedisConnectionFactory Interface

Java Spring Data for Redis架构



RedisTemplate

Interface	Description
GeoOperations	Redis geospatial操作, 例如GEOADD, GEORADIUS,...
HashOperations	Redis hash操作
HyperLogLogOperations	Redis HyperLogLog操作,例如PFADD, PFCOUNT,...
ListOperations	Redis list操作
SetOperations	Redis set操作
ValueOperations	Redis string (or value)操作
ZSetOperations	Redis zset (or sorted set)操作

RedisTemplate 提供了不同数据类型的操作方法，用户能够快速上手，轻松使用。

三、实战演练 Linux 安装 Redis6.2

Linux安装Redis 6.2

- 1.下载安装
 - \$ wget https://download.redis.io/releases/redis-6.2.1.tar.gz
 - \$ tar xzf redis-6.2.1.tar.gz
 - \$ cd redis-6.2.1
 - \$ make
- 2.启动服务器
 - \$ src/redis-server
- 3.启动命令客户端
 - \$ src/redis-cli
- 4.新增查询Key value
 - redis> set 1 java
 - OK
 - redis> get 1
 - "java"

1、下载安装命令如下：

```
$ wget https://download.redis.io/releases/redis-6.2.1.tar.gz
$ tar xzf redis-6.2.1.tar.gz
$ cd redis-6.2.1
$ make
```

2、启动服务器：

```
$ src/redis-server
```

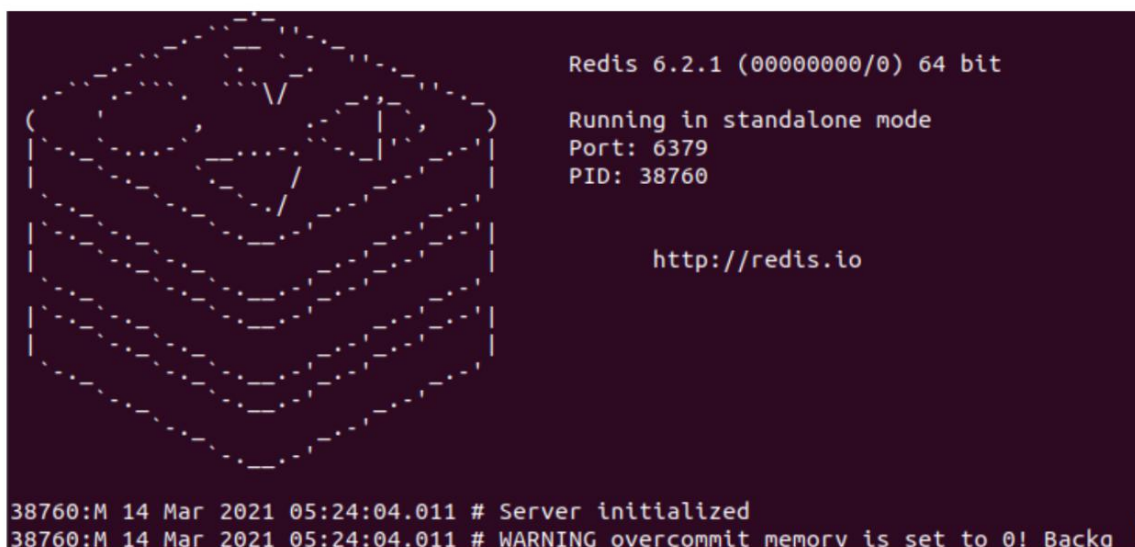
3、启动命令客户端：

```
$ src/redis-cli
```

4、新增查询 Key value：

```
redis> set 1 java
OK
redis> get 1
"java"
```

Linux启动Redis Server



Redis配置文件修改允许远程链接

- 配置文件redis.conf,
- 绑定本机地址: bind 127.0.0.1
- 注释掉# bind 127.0.0.1
- 或者修改为
- bind 0.0.0.0



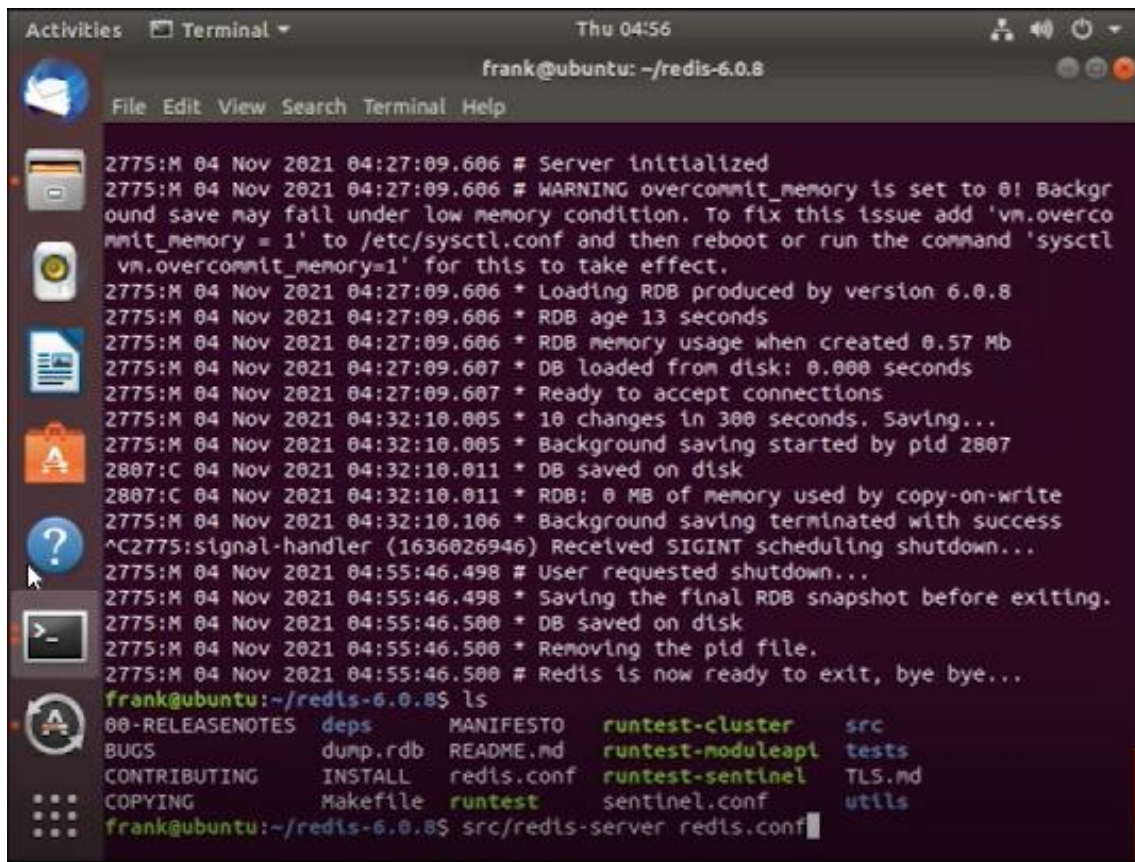
如果希望 Redis 能够被外部程序访问, 务必修改本地的 redis 配置文件 redis.conf。

如果要新增配置项, 需要先加入对配置项 resemblance 连接词的配置相关参数, 包括序列化等。

Redis 可以缓存用户名、密码、用户头像、手机号、地址、经纬度等信息。用户登录 App 时, 会优先查询 Cache 缓存, 减轻数据库压力, 大大提高性能。

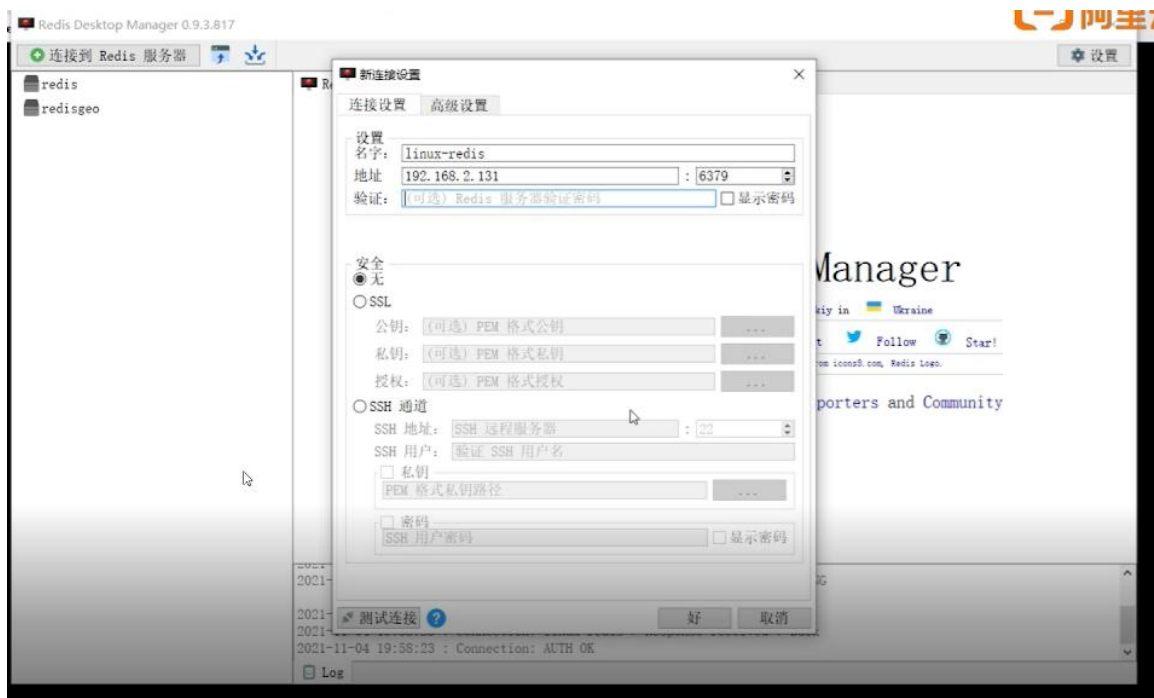
Redis 也会备份内存数据, 保存一份 RDB 文件。此外, 它还能够支持很多算法。

Redis、MongoDB 属于最优秀的 NoSQL 数据库, 非常推荐 java 用户学习以及研究其背后的算法和底层原理。



```
Activities Terminal Thu 04:56
frank@ubuntu: ~/redis-6.0.8
File Edit View Search Terminal Help

2775:M 04 Nov 2021 04:27:09.606 # Server initialized
2775:M 04 Nov 2021 04:27:09.606 # WARNING overcommit_memory is set to 0! Backgr
ound save may fail under low memory condition. To fix this issue add 'vm.overco
mmit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl
vm.overcommit_memory=1' for this to take effect.
2775:M 04 Nov 2021 04:27:09.606 * Loading RDB produced by version 6.0.8
2775:M 04 Nov 2021 04:27:09.606 * RDB age 13 seconds
2775:M 04 Nov 2021 04:27:09.606 * RDB memory usage when created 0.57 Mb
2775:M 04 Nov 2021 04:27:09.607 * DB loaded from disk: 0.000 seconds
2775:M 04 Nov 2021 04:27:09.607 * Ready to accept connections
2775:M 04 Nov 2021 04:32:10.005 * 10 changes in 300 seconds. Saving...
2775:M 04 Nov 2021 04:32:10.005 * Background saving started by pid 2807
2807:C 04 Nov 2021 04:32:10.011 * DB saved on disk
2807:C 04 Nov 2021 04:32:10.011 * RDB: 0 MB of memory used by copy-on-write
2775:M 04 Nov 2021 04:32:10.106 * Background saving terminated with success
^C2775:signal-handler (1636026946) Received SIGINT scheduling shutdown...
2775:M 04 Nov 2021 04:55:46.498 # User requested shutdown...
2775:M 04 Nov 2021 04:55:46.498 * Saving the final RDB snapshot before exiting.
2775:M 04 Nov 2021 04:55:46.500 * DB saved on disk
2775:M 04 Nov 2021 04:55:46.500 * Removing the pid file.
2775:M 04 Nov 2021 04:55:46.500 # Redis is now ready to exit, bye bye...
frank@ubuntu:~/redis-6.0.8$ ls
00-RELEASENOTES  deps      MANIFESTO  runtest-cluster  src
BUGS             dump.rdb  README.md  runtest-moduleapi tests
CONTRIBUTING    INSTALL  redis.conf runtest-sentinel  TLS.md
COPYING          Makefile  runtest    sentinel.conf    utils
frank@ubuntu:~/redis-6.0.8$ src/redis-server redis.conf
```



使用开源的 Redis Desktop Manager 可视化管理客户端，可以很方便地管理 redis。安装完毕后，只需新建连接，设置 redis 服务器参数即可使用。

Spring Boot 2.6 版本下，电商网站连接 Redis 服务器需要加入项目依赖，并且修改配置文件以及加入 Redis 服务器参数。

① 项目加入快速启动需要依赖 spring-boot-starter-data-redis，具体代码如下：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
    <exclusions>
        <exclusion>
            <groupId>io.lettuce</groupId>
            <artifactId>lettuce-core</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-pool2</artifactId>
</dependency>
<!-- https://mvnrepository.com/artifact/redis.clients/jedis -->
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
</dependency>
```

此处使用的是 jedis 驱动库，是一款相对比较成熟的驱动库，支持连接池机制，方便性能优化。

② 配置文件参数如下：

```
spring.application.name=SpringBoot251
server.port=8081
# REDIS (RedisProperties)
spring.redis.database=0
spring.redis.host=192.168.1.100
spring.redis.port=6379
```

③ 对于数据访问层，它也统一了仓储模式，封装了对于 Redis 缓存的 CRUD 操作。定义的接口如下：

```
/*
 * @Package: com.alibaba
 * @ClassName: RedisRepository 仓储模式
 * @Desc : 《阿里巴巴 Java Spring Boot 2.6 开发实战课程》参考代码
 */
public interface RedisRepository {
    Users find(Integer id);
    Map<Integer, Users> findAll();
    void save(Users user);
    void update(Users user);
    void delete(Integer id);
}
```

④ 实现如下自定义接口，以实现对于用户 User 的操作：

```
@Repository
public class RedisRepositoryImpl implements RedisRepository {
    private static final String KEY = "Users";

    private RedisTemplate<String, Object> redisTemplate;
    private HashOperations<String, Integer, Users> hashOperations;
```

```
@Autowired
public RedisRepositoryImpl(RedisTemplate<String, Object> redisTemplate){
    this.redisTemplate = redisTemplate;
}

@PostConstruct
private void init(){
    hashOperations = redisTemplate.opsForHash();
}

public Users find(final Integer id){
    return (Users) hashOperations.get(KEY, id);
}

public Map<Integer, Users> findAll(){
    return hashOperations.entries(KEY);
}

public void save(final Users user) {
    hashOperations.put(KEY, user.getId(), user);
}

public void update(final Users user) {
    hashOperations.put(KEY, user.getId(), user);
}

public void delete(final Integer id) {
    hashOperations.delete(KEY, id);
}
}
```

⑤ 配置 RedisTemplate 的连接池和序列化器参数：

```
@Configuration
@ComponentScan("com.alibaba")
public class RedisConfig {

    @Bean
    public JedisConnectionFactory redisConnectionFactory() {

        RedisStandaloneConfiguration config = new RedisStandaloneConfiguration
("192.168.1.100", 6379);
        return new JedisConnectionFactory(config);
    }

    @Bean
    public RedisTemplate<String, Object> redisTemplate() {
        final RedisTemplate<String, Object> template = new RedisTemplate<String,
Object>();
        template.setConnectionFactory(redisConnectionFactory());
        template.setValueSerializer(new GenericToStringSerializer<Object>(Object.class));

        return template;
    }

}
```

此处使用的序列化器是 GenericToStringSerializer。

⑥ 封装对于对象的操作逻辑。本示例通过 RedisRepository 对象查询所有的用户信息：

```
@RequestMapping("/Redis")
@Controller
public class RedisController {
```

```
@Autowired//Spring 创建对象，注入进来
private RedisRepository redisRepository;

@GetMapping
public String hello() {
    return "Hello Java Spring Data Redis";
}

@RequestMapping("/addUsers")
@ResponseBody
public int addUsers() {
    for (int i = 0; i < 100; i++) {
        Users entity = new Users();
        entity.setId(i);
        entity.setName("Java Spring Boot 2.6x 实战 Redis: " + i);
        entity.setPassword("password" + i);
        redisRepository.save(entity);
    }
    Map<Integer, Users> listUsers = redisRepository.findAll();
    System.out.println(listUsers.size());
    return listUsers.size();
}
}
```

Spring Boot2.0 之后，提供了 Redis 响应式接口，是基于 GEO 地理位置搜索的新特性支持，背后的原理是 GEOHash 算法。不同的数据库对于 GEO 搜索的支持存在差异，比如多边形搜索就会比较弱。而阿里优化后的 Redis 或 MongoDB 以及官方的 MongoDB 都能提供多边形搜索。

Spring Boot 2.6 可以与 Redis 最新版进行对接，阿里的优化实现了集群包括 IO 请求处理的多线程支持，以支持更高规模的并发。

Spring Boot2.6 电商网站实战之安全机制

——侠客
资深架构师

视频链接: <https://developer.aliyun.com/learning/course/903>

一、安全机制介绍

Spring Boot 2.6 安全机制具有以下特点:

1. 自定义实现安全验证
2. Apache Shiro 开源框架
3. Spring Security 开源框架
4. 大量使用 AOP
5. 依赖注入思想
6. 灵活扩展

Spring Boot 2.6 安全机制如下图所示:



Java Spring Boot 2.6 的安全机制实现了一些强化改进。开发一些电商、社交或者游戏等 app 的后端，开放 API 接口给客户端的 app 调用，要如何保证接口的安全？

大家日常使用的 app，例如淘宝、微信、抖音或者其他的一些电商平台，登录时需要输入用户密码，或者直接输入手机号验证，还有一些小程序直接集成微信、支付宝或者淘宝账号登录验证功能，当然也可以集成使用微博账号验证。在一些重要的行业（银行金融、政府部门）可能采用更严格的安全验证方案，目前一种非常流行的安全验证方式就是：人脸识别。

人脸识别的安全度已经颇高，虽然目前无法做到百分之百的准确，但已经可以保证 95% 以上的准确率。现在市场上还有指纹识别验证，这也是一种生物安全验证方式。

其中，安全度较高的是指纹识别、瞳孔识别、人脸识别。常见的账号密码验证过程中，首先会确定账号密码是否有效，然后再查询权限。国内目前绝大部分的 APP 都可以使用微信、淘宝、支付宝登录，这是一种比较常见的集成开放 OAuth 登录方式。

此外，还有基于令牌登录的，捆绑 APP 去识别后台 api 接口的验证请求。

二、安全漏洞介绍

软件框架没有绝对的安全，只有未发现的安全漏洞，Spring Boot 也不例外，不过它一直在持续更新完善，它主要的修复历程如下：

1. Spring Boot 2020 年 9 月份修复漏洞。
2. Spring Boot Actuator 未授权访问远程代码执行漏洞。
3. 紧急修复 Spring Framework 版本包含一个安全漏洞（CVE-2020-5421）的修复程序。此漏洞可以通过 sessionId 绕过 RFD（反射型文件下载）保护。
4. Spring Boot 2018 年修复了一些安全漏洞。
5. 建议使用最新的 Spring 5.0+ 版本。
6. Spring 框架升级 5.0.0 - 5.0.2。
7. Spring 框架升级 4.3.0 - 4.3.13。
8. Spring Boot 1.5.10。

这里面涉及到一个框架的漏洞问题，无论 cpp 还是 go，还是 java，都有程序的漏洞。像框架级的漏洞，一般公司都有专门的安全专家或者安全工程师应对。大部分互联网中的数据泄露或者网络攻击，都是属于此类普通的漏洞。比如 12306 网站出现过用户密码和用户账号泄露，里面的手机号都是真实的；还有国内在 2013 年左右，出现了如 csdn 以及一些社交网站的大规模账号泄露。很多网站的数据库建立的用户密码字段都是明文的，而这其实应该是尽量避免的。

三、Java Spring Security 介绍

Java Spring Security 有如下优点：

1. Spring Security 是功能强大且高度可自定义的 Java 开源安全框架。
2. 保护 Spring 应用系统的安全标准。
3. Spring Security 专注于身份验证和授权。
4. 容易扩展、自定义开发。
5. 前身是 Acegi Security。
6. 提供安全认证服务的框架。
7. Spring Security 为基于 J2EE 企业应用提供了全面安全机制。
8. Authentication 验证和 Authorization 授权。
9. 抵御会话攻击，点击劫持,CSRF 跨站请求伪造。

开源社区里面比较著名的两个框架，其中之一就是 spring 框架。最初 spring 只是个工具类，它来解决依赖注入和控制反转。后面又出现 spring boot、spring cloud、spring data、spring security 各种各样的产品，包括定制任务等。

在今天 spring 严格来说是一个体系，它给 java 应用开发面临的各种典型的、关注性的问题都提供了解决方案。Spring Security 是一种专门解决安全问题、应用程序安全问题的集成。它也可以和 Spring Mvc 集成，测试方式都是比较典型的。

四、Java 安全框架 Shiro 介绍

java 安全框架 Shiro 有如下优点：

1. Apache Shiro 简单易用的开源 Java 安全框架。
2. 轻松实现身份验证、授权、加密和会话管理。
3. 使用 Shiro 可以快速实现系统安全。
4. Shiro 其前身是 Jsecurity 项目。
5. Shiro 可以轻易实现 Java 网站安全验证。
6. 可应用于 Web 环境，非 Web 环境。
7. 支持多种数据源 MySQL 等。
8. 如 LDAP, JDBC, Kerberos,ActiveDirectory 等。

另一个框架是 Apache shiro，这是一个开源的安全框架。它和 Spring Security 框架是对等的，功能上很像，唯一的缺点可能就是它本身并不是出自于 spring 家族，与 Spring 的兼容性不完美，所以可能和 Spring Security 这个框架的版本不兼容。但它内部也提供了对于多种安全验证方式的支持。

五、Spring Security Demo 介绍

建项目时，加入 spring-boot-starter-security 依赖即可，具体代码如下：

```
<dependencies>
...
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
...
</dependencies>
```

注意：此处 starter 的本意是启动者或者入门者，快速构建 Java 项目。这里采用了最少依赖原则，现在很多框架都支持简化开发的模式，也称为傻瓜式开发。早期 Java 开发对程序员技能要求非常高，而现在将 spring-boot-starter-security 引用到项目中，即可完美解决安全验证问题。

六、WebSecurityConfig 介绍

WebSecurityConfig 具有以下特性：

1. Spring Security 的配置类
2. WebSecurityConfig
3. 可以配置安全规则
4. 默认启用 basic 验证
5. # Spring Security 可以在配置文件中关闭
6. security.basic.enabled = false

七、Web 全站安全验证配置

实现 web 全站安全验证配置，只需加一个 WebSecurity 配置类将请求进行拦截即可，具体代码实现如下：

```
@Configuration
@Order(SecurityProperties.BASIC_AUTH_ORDER - 10)
public class ApplicationConfigurerAdapter extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception{
        http.antMatcher("/admin/**")
        authorizeRequests()
        antMatchers("/admin/users").hasRole( "usersAdmin")
        antMatchers("/admin/orders").hasRole(
```

```
ordersAdmin")
anyRequest().isAuthenticated();}}
```

Spring Boot 底层用到了 Spring MVC 框架，MVC 框架底层基于 Servlet API。MVC 框架已经把底层拦截的类全部封装好，只需要进行 `antMatcher` 等参数的配置。这个匹配的参数可以是正则表达式或者字符串。上述代码中，如果请求地址里面包含 `/admin/` 类型的字符串，就认为会要访问一个 `admin` 根目录下的网页。MVC 对 `/admin/` 这类 url 访问的时候，Spring 判断它进入的是管理后台，并且启用了身份验证。前台显示没有问题，但系统必须进行验证登录。`.isAuthenticated()` 的意思就是对请求进行身份验证。这个可以定义多个 URL 地址，比如店铺地址等，最后通过地址进行拦截以及安全验证。

目前大部分的电商平台都有图片或者视频审核机制，很多网站都设置了管理员，其职责各不相同。比如前段时间比较火的鉴黄师，是负责对用户上传的资料、视频、图片审核，防止用户在平台上传非法的图片。

在网站平台打造的过程中，可以自己实现一个万能的请求拦截器，当用户访问登录模块时，就可以将请求拦截下来，从用户的请求消息头里提取用户密码进行验证，再生成统一的 Token 令牌。APP 端可以持有 Token，方便后续的使用。后续的访问操作会传回之前的 token，后台 API 实现请求验证。

现在的大型网站的安全问题包括很多方面，不仅仅是账号身份验证问题。App 在互联网上数据传输的过程其实也是存在安全隐患的，比如有一些嗅探工具、拦截工具，拦截数据流，再使用暴力破解的方法，导致敏感数据泄露或账号被盗。

当然，如果数据库的安全措施做得不彻底，也可能拦截明文的 SQL，通过 SQL 注入破解数据库，并还有可能进行撞库去暴力破解其他平台账号，造成更大的数据安全问题。经常出现平台攻击或者数据泄露的新闻，大部分与安全措施做得不规范有直接关系，可见使用一套成熟的 security 框架是很有必要的。

八、实战代码展示

用户登录接口代码：

```
@RequestMapping(value = "/login", method = RequestMethod.POST)
public ModelAndView login(String username, string password, HttpServletRequest
request,HttpServletResponse response) {
    system.out.println(username) ;
    system. out.println(password);
    ModelAndView mv = new ModelAndView("redirect:/Home/index" );try
    //Hash 密码加密，
    //SQL 注入，防止 SQL 注入
    Users user = userServiceImpl.getUserByNameAndPassword(username,password (user
!= null)
    request.getSession().setAttribute("UserName",username);}else {
    mv.setViewName( "redirect: ./ login");
    }
    catch (Exception e) {
    System.out.println(e.getMessage());
    }
    return mv;
```

注：Redis 可用于防止数据丢失。抖音及微信都存在大数据的相关技术，应用的比较多的就是行为大数据。

WebSecurityConfig 代码展示：

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders
import org.springframework.security.config.annotation.web.builders.HttpSecuri
```

```
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity

public class webSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/", "/Home").permitAll().antMatchers("/Users/**").denyAll()
            .access("hasRole('ROLE_ADMIN')").anyRequest().authenticated().and()
            .formLogin();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth
            .inMemoryAuthentication()
            .withUser("frankxu")
            .password("1234qwer")
            .roles("ADMIN");
    }

    @Bean
    public static PasswordEncoder passwordEncoder(){
        return NoOpPasswordEncoder.getInstance();
    }
}
```

AccountController 层代码展示:

```

@Controller
@RequestMapping( "/Account")
public class AccountController {
    @Autowired
    private UserserviceImpluserService;

    @RequestMapping(value = "/login", method = RequestMethod.GET)public string login() {
        system.out.println("打开登录页面");return "Account/ login";
    }

    @RequestMapping(value = "/login", method = RequestMethod.PoST)
    public ModelAndView login(String username, String password, HttpServletRequest request, HttpServletResponse response) {
        System.out.println(username );
        system.out.println(password);
        ModelAndView mV = new ModelAndView("redirect:/Home/index");try {
            //Hash 密码加密 ,
            / /SQL 注入, 防止 sQL 注入
            Users user = userServiceImpl.getUserByNameAndPassword(username;if (user != null) {
                request.getSession( ).setAttribute("UserName", username);} else {
                mv.setViewName( "redirect: ./ login");
            }
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return mv;
    }
}

```

```
@RequestMapping(value = "/logout", method = RequestMethod.POST)public int logout(String username) {  
    // redis 缓存删除状态信息,  
    // Redis 缓存更新一个 status 01, 下线 offline。 return 1;  
}  
  
@RequestMapping(value = "/login", method = RequestMethod.POST)  
public Integer login(string username, string password, HttpServletRequest  
request, HttpServletResponse response) {  
    //Redis 缓存更新一个 status 1, 在线 online。 @RequestMapping(value = "/logout", method = RequestMethod.POST)public int logout(String username) {  
    // redis 缓存删除状态信息,  
    // Redis 缓存更新一个 status 01, 下线 offline。 return 1;  
}  
  
@RequestMapping(value = "/login", method = RequestMethod.POST)  
public Integer login(string username, string password, HttpServletRequest  
request, HttpServletResponse response) {  
    //Redis 缓存更新一个 status 1, 在线 online。  
}  
  
@RequestMapping(value = "/login", method = RequestMethod.POST)  
public Integer login(String username, String password, HttpServletRequest  
request, HttpServletResponse response) {  
    // Redis 缓存更新一个 status 1, 在线 online。  
  
    @RequestMapping(value = "/logout" , method = RequestMethod.POST)public int logout(String username) {  
        //redis 缓存删除状态信息,  
        //Redis 缓存更新一个 status 01, 下线 offline。 return 1;  
    }  
  
    @RequestMapping(value = "/login",method = RequestMethod.POST)  
    public Integer login(string username, String password, HttpServletRequest  
    request, HttpServletResponse response) {  
        // Redis 缓存更新一个 status 1, 在线 online。
```



```

* try { //Hash 密码加密, //SQL 注入, 防止 SQL 注入//Users user:

@Controller
@RequestMapping( "/Account")
public class AccountController {
    @Autowired
    private UserserviceImpluserService;
    @RequestMapping(value = "/login", method = RequestMethod.GET)public string lo
gin() {
    system.out.println("打开登录页面");return "Account/ login";
    }
    @RequestMapping(value = "/login", method = RequestMethod.PoST)
    public ModelAndView login(String username, String password, HttpServletRe
request, HttpServletResponse response) {
    System.out.println(username );
    system.out.println(password);
    ModelAndViewmv=new ModelAndView("redirect:/Home/index");try i
//Hash 密码加密 ,
/ /SQL 注入, 防止 sQL 注入
Usersuser= userServiceImpl.getUserByNameAndPassword(username;if (user != nul1)
{
    request.getSession( ).setAttribute("UserName", username);} else {
    mv.setViewName( "redirect: ./ login");
    }
    }
    catch (Exception e) {
    System.out.println(e.getMessage());
    }
    return m1;
    }}

```

UserController 层代码展示:

```
UsersRepository;
@Autowired
UserService;
@RequestMapping( "/getAll/{name} ")
public List<Users> getAll(@PathVariable( "name") String name){
// / string name ="java" ;
return userService.getAll(name);
}
//查询所有的用户数据
@RequestMapping( "/getAll")public List<Users> getAll(){
List<Users> listusers = (List<Users>) usersRepository.findAll();return listUsers;
}
}
```