浙江大学

本科课程论文

课程名称：项目实训

姓　　名：张溢弛

学　　院：计算机科学与技术学院

专　　业：软件工程 1801

学　　号：3180103772

指导教师：邵健

2021 年 7 月 21 日

目录

基于 SpringBoot 的层级化搜索与知识图谱开发

摘要：垂直搜索引擎是近年来的搜索引擎发展的突破口，本次暑期项目实训中，我们小组开发了一款足球领域的垂直搜索引擎，使用网络爬虫获取真实数据，使用 React 框架开发项目前端，并使用了 SpringBoot+ElasticSearch 的服务端架构开发了除了层级化的搜索服务模式和词云、知识图谱等一系列额外功能。本论文将重点介绍我们小组开发的搜索引擎的服务端架构设计和具体实现，并在此基础上总结垂直搜索引擎开发的一些世界观和方法论。

关键词： Spring Boot，垂直搜索引擎，服务端开发，层级化搜索

一　背景介绍

垂直搜索引擎是针对某一个行业的专业搜索引擎，是搜索引擎的细分和延伸，是对网页库中的某类专门的信息进行一次整合，定向分字段抽取出需要的数据进行处理后再以某种形式返回给用户。垂直搜索是相对通用搜索引擎的信息量大、查询不准确、深度不够等提出来的新的搜索引擎服务模式，通过针对某一特定领域、某一特定人群或某一特定需求提供的有一定价值的信息和相关服务。

本次项目实训中，我们小组成员完成了一个足球领域的垂直搜索引擎的开发工作，我负责了其中的服务端开发，小组成员使用了 SpringBoot+Mybatis+MySQL+ElasticSearch 的技术栈完成了服务端 API 的开发。

我们小组成员在完成了基于 ElasticSearch 的搜索功能的基础上，开发出了一系列附加功能，包括相关球员推荐，球员词云生成，球员知识图谱和球队知识图谱的生成等功能，在开发的过程中也逐渐了解了 SpringBoot 的架构和其背后的设计模式中所蕴含的软件工程思想，并在具体的实战开发中有了一些自己的心得。

我们小组成员在服务端设计了一套层级化的数据查询 API，通过合理分配 ElasticSearch 和 MySQL 数据库的搜索压力，并实现了知识图谱等高级功能。本论文将围绕项目实训中如何构建层级化搜索架构和知识图谱等增益功能展开详细叙述，介绍和总结项目实训中获得的一些实战经验。

二　相关技术

2.1　SpringBoot 框架

SpringBoot 是在 Spring 框架的基础上开发的一款 Java 微服务开发框架，其设计目的是用来简化新的 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配

置，从而使开发人员不再需要定义样板化的配置。其主要的特点有：

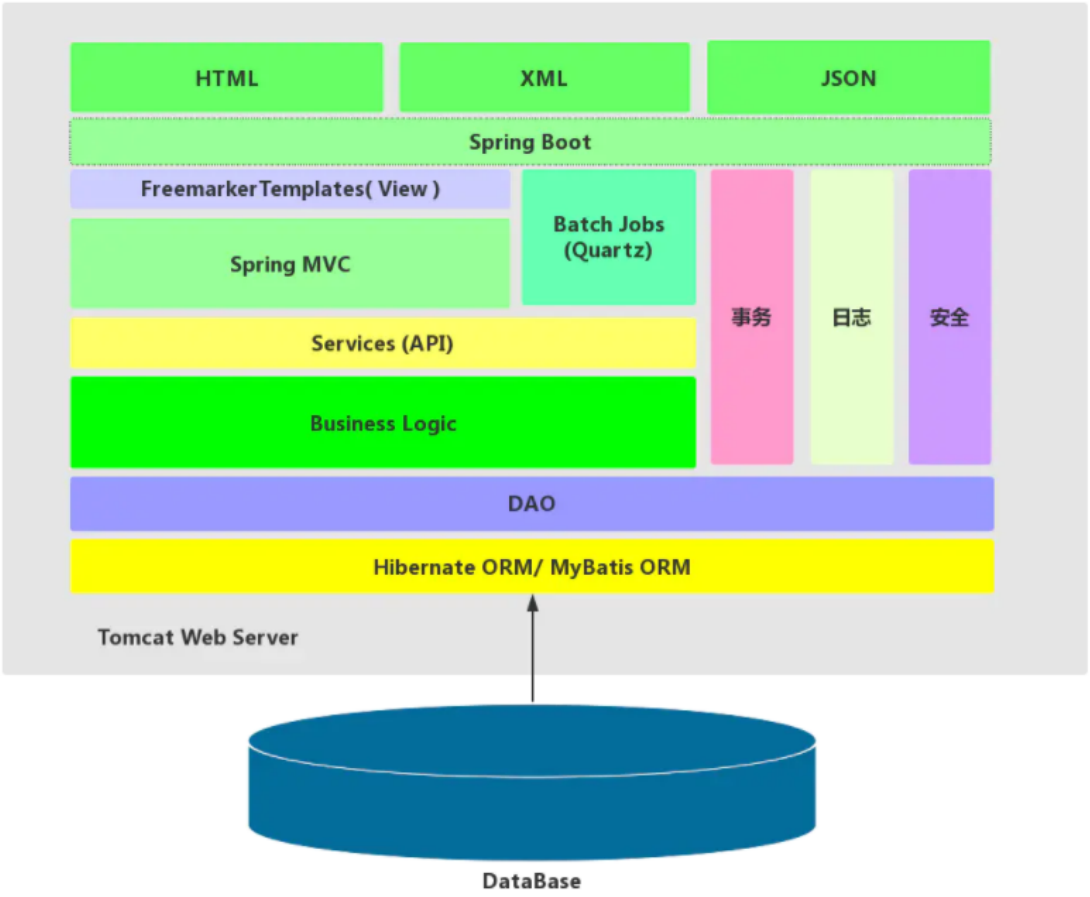
- 以创建独立的 Spring 应用程序，并且基于 Maven 等包管理工具，可以创建可执行的 JAR 文件
- 内嵌 Tomcat 或 Jetty 等 Servlet 容器
- 提供自动配置的“starter”项目对象模型（POMS）以简化 Maven 配置，避免了复杂而繁琐的 JAR 包引入工作
- 尽可能自动配置 Spring 容器
- 提供准备好的特性，如指标、健康检查和外部化配置
- 简化了 XML 的配置，不再需要代码生成

SpringBoot 框架中还有两个非常重要的策略：开箱即用和约定优于配置。开箱即用，Outofbox，是指在开发过程中，通过在 MAVEN 项目的 pom 文件中添加相关依赖包，然后使用对应注解来代替繁琐的 XML 配置文件以管理对象的生命周期。这个特点使得开发人员摆脱了复杂的配置工作以及依赖的管理工作，更加专注于业务逻辑。

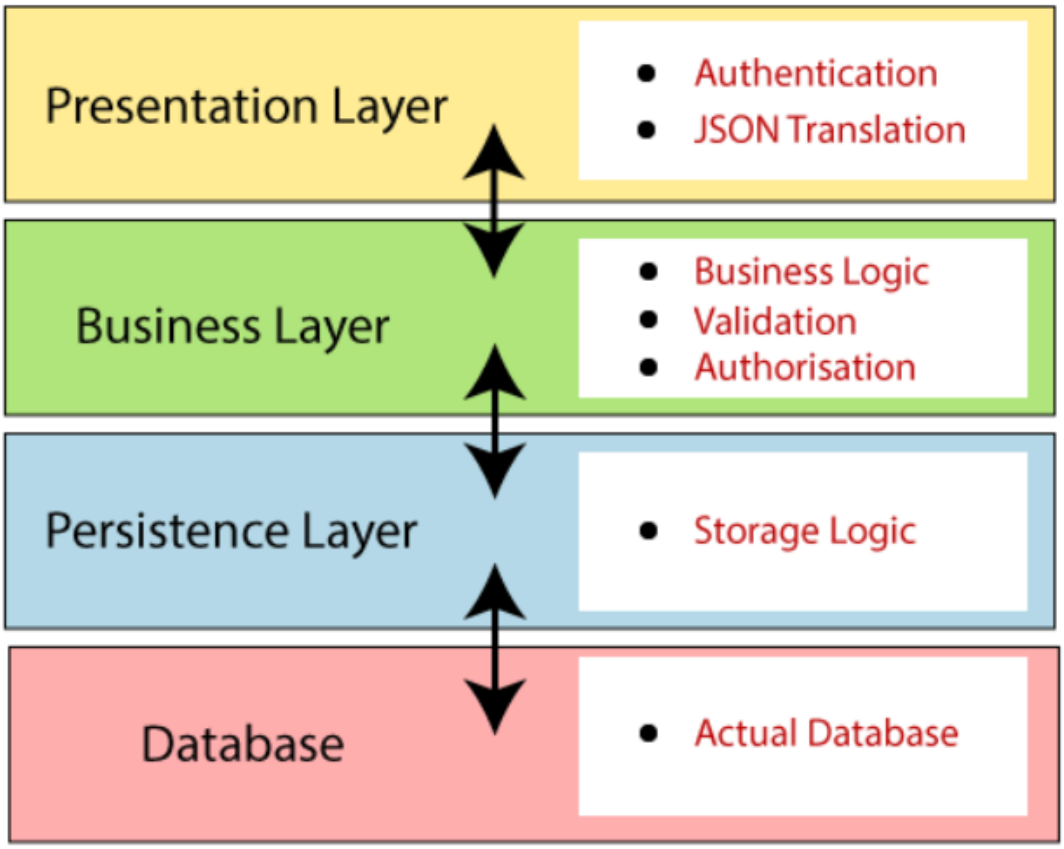
约定优于配置，Convention over configuration，是一种由 SpringBoot 本身来配置目标结构，由开发者在结构中添加信息的软件设计范式。这一特点虽降低了部分灵活性，增加了 BUG 定位的复杂性，但减少了开发人员需要做出决定的数量，同时减少了大量的 XML 配置，并且可以将代码编译、测试和打包等工作自动化。

2.1.1 SpringBoot 基本架构

SpringBoot 继承了 Spring 框架的架构和设计模式，其核心的架构图如下图所示：



同时 SpringBoot 框架在具体开发中被分成了若干层级结构，不同的层级分别对应着不同的功能，具体的划分如下图所示：



ORM 层即数据库实体层，也被称为 entity 层，一般数据库一张表对应一个实体类，类属性同表字段一一对应。dto 全称为：Data Transfer Object，即数据传输对象，一般用于向数据层外围提供仅需的数据。

dao 层即数据持久层，也被称为 mapper 层。dao 层的作用为访问数据库，向数据库发送 sql 语句，完成数据的增删改查任务。而在具体的开发过程中，我们使用了 Mybatis 作为持久层框架，通过编写基于 XML 的 SQL 文件的形式实现了持久层的功能。

service 层即业务逻辑层。service 层的作用为完成功能设计。service 层调用 dao 层接口，接收 dao 层返回的数据，完成项目的基本功能设计。

controller 层即控制层。controller 层的功能为请求和响应控制。controller 层负责前后端交互，接受前端请求，调用 service 层，接收 service 层返回的数据，最后返回具体的页面和数据到客户端。

同时使用 SpringBoot 框架进行开发的过程中，需要使用注解来标明所写的类和放大对应的类型，常用的注解有：

- @Controller：用于标注控制器层组件
- @Service：用于标注业务层组件
- @Repository：用于标注数据访问组件，即 DAO 组件
- @Bean：方法级别的注解，主要用在 @Configuration 和 @Component 注解的类里，@Bean 注解的方法会产生一个 Bean 对象，该对象由 Spring 管理并放到 IoC 容器中。

引用名称是方法名，也可以用 @Bean(name = ”beanID”) 指定组件名

- @Autowired：默认按类型进行自动装配。在容器查找匹配的 Bean，当有且仅有一个匹配的 Bean 时，Spring 将其注入 @Autowired 标注的变量中。

2.1.2 IoC 和 AOP

SpringBoot 框架继承了 Spring 两个最要的特性，就是控制反转 (IoC) 和面向切片编程 AOP，Spring 核心容器的主要组件是 Bean 工厂（BeanFactory），Bean 工厂使用控制反转（IoC）模式来降低程序代码之间的耦合度，并提供了面向切面编程（AOP）的实现。

控制反转实际上就是将初始化一个对象的权力被交给了 Spring 框架本身，而不需要我们手动 new 出一些关键对象。

面向切面编程（AOP）就是纵向的编程。比如业务 A 和业务 B 现在需要一个相同的操作，传统方法我们可能需要在 A、B 中都加入相关操作代码，而应用 AOP 就可以只写一遍代码，A、B 共用这段代码。并且，当 A、B 需要增加新的操作时，可以在不改动原代码的情况下，灵活添加新的业务逻辑实现。

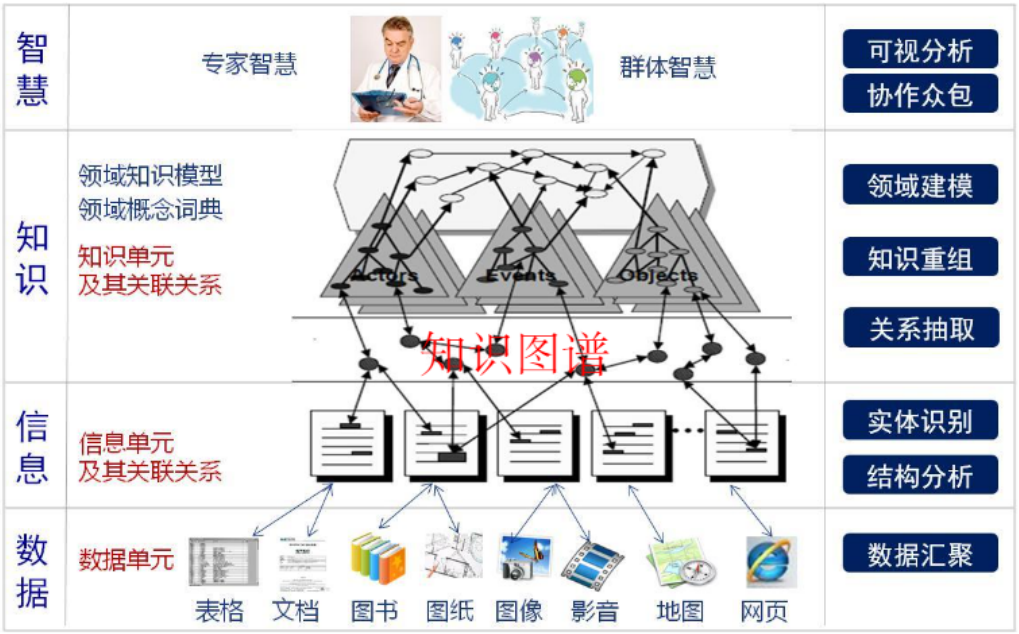
2.2 ElasticSearch

Elasticsearch 是一个分布式、RESTful 风格的搜索和数据分析引擎，能够解决不断涌现出的各种用例，支持非常方便的建立索引，导入数据和查询数据。

本项目中使用了 ElasticSearch 用来构建数据索引并进行多种模式的数据查询。同时 Elasticsearch 提供了多种编程语言的接口，本项目通过在 SpringBoot 的 pom 配置文件中引入 Elasticsearch 相关包使得项目可以调用 Elasticsearch 提供的 RESTful 接口。

2.3 知识图谱与关系抽取

知识图谱是一种使用三元组来表示某种关系的方式，三元组由 Entity-Relation-Entity 的形式构建，后端的主要工作是根据结构化和非结构化的数据生成一系列三元组，并按照一定的格式发送给前端，由前端调用可视化相关的 JavaScript 库进行知识图谱的可视化呈现。



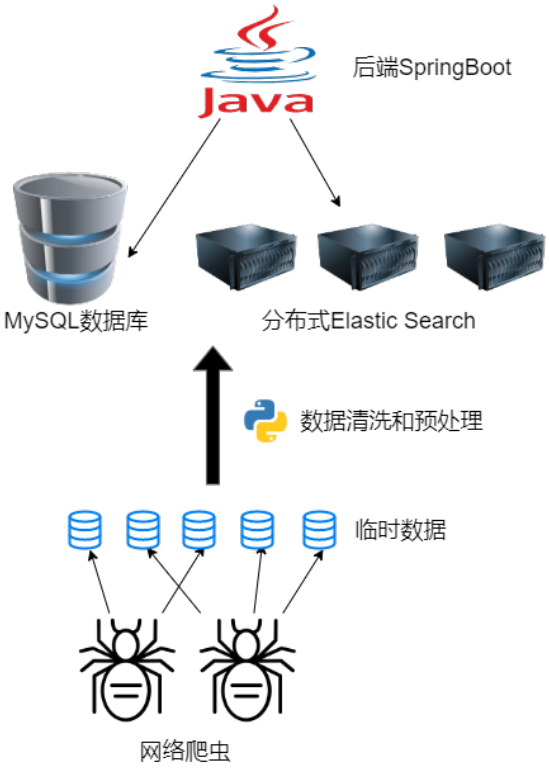
而关系抽取就是通过一定的算法和规则，在我们使用爬虫获取到的结构化和非结构化文本中抽取出一系列三元组。本项目中对爬虫获取的新闻等内容进行了关系抽取，形成了球员的词云和知识图谱三元组。

三　实战开发

3.1　服务端架构设计

我们发现 ElasticSearch 在面对文档级别的数据时查询效率要远高于 MySQL，而 Elastic Search 在面对结构化的关系型数据时，精确查询的语句比较繁琐而结果不够精确，MySQL 在关系型数据的查询中比 Elastic Search 的查询结果更为准确、迅速而方便。

因此在服务端的具体开发过程中，我们综合考虑了 SpringBoot 框架，Elastic Search 各自的特性以及需要实现的具体功能，采用了 Elastic Search+MySQL 混合检索的方式，即将关键的核心信息，如球员球队的基本信息和新闻的文本导入到 ElasticSearch 中建立索引，而将一些次要的，用于具体展示的相关数据以结构化的形式存放在 MySQL 中，并使用球员、球队专属的 ID 进行针对性的查询。总体的架构设计如下图所示：



同时我们也采用了多台服务器的集群，使得 ElasticSearch 可以提供分布式的查询服务，这样的服务端架构设计实现了关键数据的层级化检索，可以有效减轻 ElasticSearch 的服务压力，提高搜索的精度和准确率，并且使用 Python 编写的爬虫脚本和数据清洗预处理脚本对数据进行实时的爬取和更新。

3.2 层级化搜索的实现

在本项目中，我们在 ElasticSearch 中建立了三个索引，分别是球员的主要信息，球队的主要信息和足球新闻，主要信息包含 ID 和一些关键的元数据，同时在 MySQL 中存储了若干表来记录一些球员和球队相关的数据，比如球员的比赛数据、转会记录、伤病情况、词云和知识图谱，以及球队的成员、荣誉记录等等。

这些数据都可以使用球员和球队和 ID 进行针对性的查询，我们使用 SpingBoot+MyBatis 的技术栈开发了一系列查询 API，并封装成了一系列 Service，因此一次搜索的工作逻辑就可以用下面的流程图来表示：

3.3 具体功能开发

3.3.1 球员个人信息

我们的搜索引擎中，为每条球员搜索结果开发了球员个人信息详细展示界面，展示的内容包括了球员的基本信息，比赛数据和可视化呈现，个人能力值的可视化呈现，转会数据、伤病数据、相关热词 (词云) 和知识图谱等等功能。

这些具体的球员数据都在 MySQL 数据库中用数据表存储，并且可以用球员的 ID 查询到，而球员的 ID 等信息可以在 ElasticSearch 中使用输入的关键词获取，这就构成了一个层级化的搜索体系，在提高查询效率的同时，也提高了 ElasticSearch 搜索结果的准确度，避免了过多的冗余信息带来的精度下降问题。

3.3.2 球员词云

本项目使用爬虫爬取的数据中，有很多结构化的数据，比如球员的个人信息，球员的能力值数据，也有很多非结构化的数据，比如足球新闻和球员球队的热门新闻标题等等。

我们首先通过对这些结构化的数据进行基于规则的信息抽取，构造了若干球员关键词，比如对于射门命中率特别高的球员，就给他打上一个“神射手”的标签。

此外，我们对非结构化的文本进行了分词和正则表达式过滤等预处理，并去掉 stop word 和一些没有价值的单字之后，得到了若干和球员相关的词语，然后我们使用词频统计，选取了频率最高的 20 个词语和前面得到的球员关键词进行混合，并赋予一定的权重，作为词云的组成部分。前端在调用后端的 API 之后就可以调用可视化库将词云结果可视化。

3.3.3 球员知识图谱

我们采用结构化的数据提取出了一些和球员/球队的三元组，并按照一定的格式发送给前端，由前端进行渲染得到最终的结果。我们使用球员的球队，队友，伤病，比赛荣誉，转会记录等等，使用基于规则的方法构造出了一系列球员、球队的三元组。

四 总结

层级化的搜索架构在搜索引擎的开发过程中帮助我们实现了高效、高性能、高可用性的软件工程开发，在提高搜索质量和搜索效率的同时降低了后端各个模块的耦合度。而知识图谱和词云使得我们可以更好地处理结构化和非结构化的数据进行信息的提取和呈现，使得整个系统的功能和用户友好度得到提升。

使用 SpringBoot+ElasticSearch 开发搜索引擎服务端总的来说还是充满了挑战的，本论文中提出的一些方法只是对本次项目实训的一个简单总结，未来还需要更多的实际开发来实践上课所学的软件工程理论，技术框架和系统设计思想。