Software Engineering and Applications 软件工程与应用, 2017, 6(2), 28-39
Published Online April 2017 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2017.62004

A Study of Key Technologies of REST-Style Web API Generation Tool

Sheng Gao, Yurong Deng, Junnian Fu, Xiaolan Xu

Guangdong Ocean University, Zhanjiang Guangdong
Email: minix@139.com, dengyurongrong6330@163.com, nqdmx2012@163.com, xxlanczj@163.com

Received: Mar. 28th, 2017; accepted: Apr. 11th, 2017; published: Apr. 17th, 2017

Abstract

To meet the requirement of rapid development of REST-Style Web API, the code generation tool is constructed, which combines with open source MVC and ORM framework. By using the concept of modular development, the architecture and function module are designed to construct REST-Style Web API generation tool with function of modular Java code and configuration file generation tool and extendibility. The technologies of code generation based on class information and data information analysis are discussed.

Keywords

Code Generation, Web API, Java, REST, HATEOAS

REST风格Web API生成工具关键技术的研究

高　升，邓煜荣，付军年，徐晓岚

广东海洋大学，广东　湛江
Email: minix@139.com, dengyurongrong6330@163.com, nqdmx2012@163.com, xxlanczj@163.com

收稿日期：2017年3月28日；录用日期：2017年4月11日；发布日期：2017年4月17日

摘　要

为了满足快速开发REST风格的Web API的需求，将现有的开源MVC和ORM框架技术相结合，实现代码生成工具，详细设计系统架构和功能模块，利用模块化的方式实现一个具有扩展性的模块化代码和配置文件生成功能的REST风格Web API生成工具，重点阐述以类信息为基础的代码生成、数据信息分析等关键技术。

文章引用: 高升, 邓煜荣, 付军年, 徐晓岚. REST 风格 Web API 生成工具关键技术的研究[J]. 软件工程与应用, 2017, 6(2): 28-39. https://doi.org/10.12677/sea.2017.62004

关键词

代码生成，Web API，Java，REST，HATEOAS

1. 引言

SOA 思想的提出和 Web Services 的出现改变了传统的软件开发思路。与传统的企业开发不一样，SOA 倡导将企业当中分散的功能组织成为共用的服务，系统开发过程中，通过对这些共用服务的组合和重用满足业务需求[1]。Web Services 是 SOA 思想的一种实现，其实现基本原理是通过 SOAP 协议描述数据，通过 HTTP 协议进行数据的传输，从而实现远程对象和过程调用。经过多年发展，Web Services 变得越来越完善的同时，也变得越来越臃肿，实现起来也变得越来越复杂。

移动互联网的发展、Ajax 技术大量被使用等因素将原本用于企业分布式开发的 Web Services 被广泛应用，臃肿和复杂实现带来的弊端逐渐显现，这些弊端在资源受限的设备上体现尤为突出。REST 架构风格提出以来，因为其简洁和优异的交互性能，不少的服务供应商都开始提供 REST 风格的 Web Services。同时提供基于 SOAP 与 REST 风格 Web 服务的 Amazon 宣称 REST 风格的 Web 服务比基于 SOAP 的 Web 服务快 6 倍[2]。相比起 Web Services，REST 风格的 Web API 在保留前者功能的前提下进一步简化。REST 风格 Web API 的兴起，出现了不少用于快速构建 REST 风格 Web Services 的开源和商用框架。这些框架当中不少拥有陡峭的学习曲线，而且难以避免出现大量重复代码。同时，开发人员对这些框架的掌握程度的参差不齐给软件质量带来潜在的风险；代码生成工具能够较为直接地解决上面问题，本文阐述的代码生成工具生成的是具有层次结构的 REST 风格 Web API 的 Java 实现。

2. REST 架构风格

REST 是由 Roy Thomas Fielding 在其博士学位论文中提出的一种软件架构风格。REST 架构风格中定义了客户端-服务器、无状态、缓存、统一接口、系统分层五个必备的约束和按需代码一个可选的约束[3]，满足五个必备约束的软件架构风格则可以称之为 REST 软件架构风格。

在实践当中，设计者把系统当中所有的信息、服务等抽象成为资源，这些资源可以是服务器上面的一份文档、一系列的文档，也可以是服务器上用于提供数据的服务。这些资源会分配一个唯一的标识符 URI，通过这个 URI，客户端可以通过 HTTP 方法作为统一的接口操作访问和服务器上的资源，服务等。对于一些静态资源，服务器在给出响应时可以建议客户端将这个数据进行缓存，减少网络访问的次数，提高系统的执行效率。可以认为，REST 是将 Web Services 从 RPC 的动词为中心的事务脚本转变为以资源等名字为中心的领域模型[4]。

随着 REST 架构风格研究的不断深入，Leonard Richardson 提出了一个成熟度模型。这个模型把系统按照对 REST 约束的实现程度分成从等级 0~等级 3 共四个等级[5]。仅仅使用 HTTP 作为传输协议的系统在这个模型中会被划分成等级 0，基于 SOAP 的 Web Services 就属于这个等级，等级 1 是在等级 0 的基础上引入资源概念和使用 URI 标识资源，等级 2 是在等级 1 的基础上使用 HTTP 方法作为统一的资源操作接口，并且使用 HTTP 状态码表示。等级 3 则是目前 REST 设计最理想的状态，这个等级使用 HATEOAS

(Hypermedia as the Engine of Application State，超媒体作为应用状态引擎)的设计模式。在这个设计模式下，服务器仅仅需要暴露一个入口接口，通过资源或者业务上的关系将多个资源关联起来，响应客户请求的同时将这些关联资源的 URI 返回，客户端可以通过这些 URI 进行关联操作，实现客户端与服务器的完全解耦。

一个满足成熟度模型等级 3 的文献管理系统，要获得编号为 3 的学位论文。图 1 和图 2 则分别表示请求资源存在和不存在的两种情况。

如图 1 与图 2 所示，客户端请求资源时，采用 HTTP 方法当中的 GET 以及请求资源的 URI 标识资源和进行的操作。如果服务器上存在这个资源，则以 200 这个 HTTP 状态码表示找到这个资源，并返回资源本身的信息(文献信息)以及文献相关的信息(作者、参考文献信息)，如图 1 所示响应。如果这个资源不存在，则通过 404 这个 HTTP 状态码表示资源未找到，如图 2 所示响应。

3. 生成工具的设计

3.1. 工具的目标

这个代码生成工具的目的是生成基于现有成熟的开源框架的 REST 风格 Web API 实现。在《Code Generation in Action》一书中，作者 Jack Herrington 总结代码生成工具的十条准则[6]，在本代码生成工具设计时也尽量遵循下面的准则：

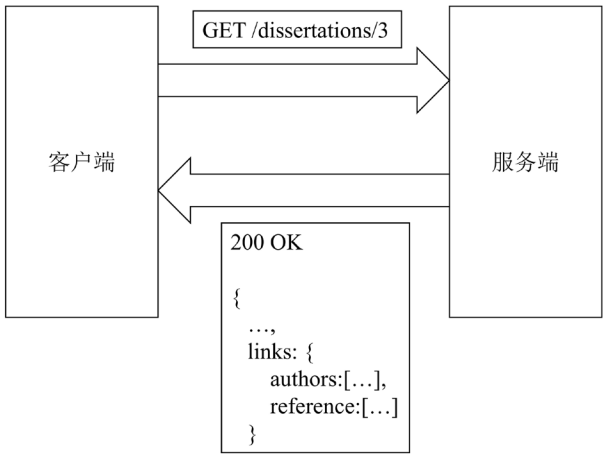

Figure 1. Response for an available resource
图 1. 请求资源存在时响应

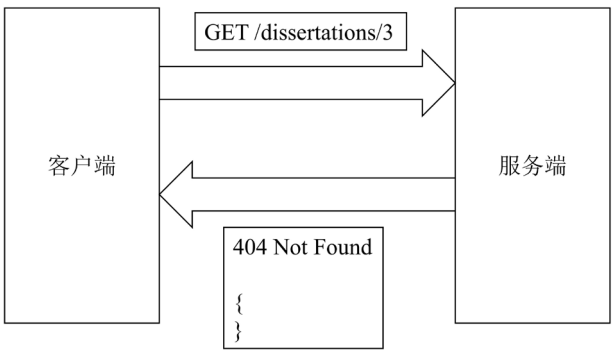

Figure 2. Response for an unavailable resource
图 2. 请求资源不存在时响应

1) 重视手工编码。代码生成工具解决的只是避免在重复代码的编写上面浪费过多的时间，在涉及业务细节的代码仍然需要手动进行编写。

2) 先进行手工编码。在生成代码前，可以手工编写目标系统基本的架构。这样可以对所生成的代码有所了解。便于在代码生成以后进行相应的修改。

3) 考虑生成工具使用的语言。根据需求，代码生成工具的目标系统的源码是 Java，利用 Java 的反射机制等技术，可以直接从框架当中获得相应信息，提高因为框架版本变化时生成工具的伸缩性，所以代码生成工具同样使用 Java 进行实现。

4) 整合到 IDE 当中。代码生成工具并不能做到一次就可以生成满足需求的系统，代码生成以后仍然需要由开发人员进行适当的修改。以插件形式将代码生成工具与 IDE 的结合，可以直接将生成代码以 IDE 工程的形式导入其中，与现有的开发流程结合。

5) 在生成代码当中以注释形式增加警告信息以及修改建议信息。在代码当中增加警告信息可以避免用户破坏代码，同时，对于代码生成工具无法完成的代码，增加引导性的注释，引导开发人员完成这部分代码的编写。

6) 包含文档。代码生成的文档有两种，一种是 Javadoc 文档，一种是生成的 REST API 文档。前者通过在源码当中添加满足一定规范的注释，使用工具自动生成。后者则采用模板的形式生成。

7) 不断地维护。新技术和框架的不断出现也促使代码生成工具需要不断去迎合这些新技术的要求。

3.2. 技术选择

Spring Boot 是一个用于 Spring 快速开发的框架，通过少量的配置，利用 Spring Boot 开发的程序即可使用 Spring Framework 当中核心的依赖注入，AOP 等技术。相比起传统的 Spring 应用程序开发，Spring Boot 摒弃了繁杂的 XML 和 Maven 依赖配置，提供一种快速开发 Spring 程序的途径，在一定程度上解决了一直被开发者诟病的复杂臃肿的配置引起复杂度增加的问题，近年来开始为人们所接受并开始应用至新开发的项目当中。

Spring HATEOAS 是 Spring 的子项目。Spring HATEOAS 提供一组快速实现 HATEOAS 的工具，与 Spring MVC 共同实现快速开发成熟度高的 REST Web Services 或者 Web API。

JPA (Java Persistence API)是 Sun 提出在 Java SE 和 Java EE 下使用的 Java 持久化更规范[7]，作为 JSR-220 实现的一部分，可以脱离 EJB 3.0 而使用。JPA 本身不提供数据持久化的实现，仅仅用于描述 Java 类到关系型数据库记录之间的映射的规则，数据持久化的实现由第三方开源的 ORM 框架实现，比如 Hibernate。

Hibernate 是一个成熟的开源 ORM 框架，能够从配置文件，或者类当中 JPA 注解定义的映射配置实现自动的对象与关系型数据库记录之间的相互映射。Hibernate 将数据库常见的增删查改封装成一组 API，通过这组 API 即可以完成数据库的相应操作。另外，Hibernate 还提供了语法与 SQL 类似的 HQL，在持久化过程中，Hibernate 会自动将 HQL 翻译成 SQL 语句执行查询。Hibernate 框架把数据操作的细节进行了屏蔽，提供的透明的持久化过程，使得程序员在开发的过程不需要过于关心数据库的操作细节[8]，简化程序的开发。

3.3. 软件架构

图 3 为系统的体系架构，软件的主要逻辑与界面分离，可以通过 Swing，Web 调用，也可以作为插件的形式引入到开发环境当中。

在该架构当中，数据信息分析模块、代码生成模块和编译打包模块三个模块组成了生成工具。生成

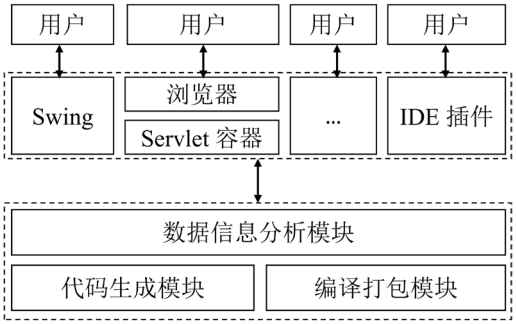

Figure 3. Architecture of system
图 3. 系统的体系架构

工具没有面向特定的用户界面，核心逻辑与界面逻辑相分离，降低了各自之间的耦合程度，从而降低系统整体的维护成本。该生成工具还可以作为第三方 jar 文件引入到其他的项目当中，便于未来需求发生变化时快速的重构。

3.4. 模块功能划分

1) 数据信息分析模块：接收用户的输入的信息，这些信息主要是用于描述实体类以及实体类之间的关联关系，数据信息分析模块会根据这些输入信息生成与之相对应的 Java 类信息。

2) 代码生成模块：该模块接收 Java 类的元信息，根据这些类信息生成相应的代码和配置系文件，整合成 Maven 项目。

3) 编译打包模块：将产生的 Maven 项目编译，打包成可以用于发布的 war 包。

这三个模块共同完成从用户需求的输入到生成项目的流程，图 4 描述了整个系统工作时的工作流程。

3.5. 生成工具数据的分析与抽象

该生成工具同时涉及到数据实体、实体之间关联关系信息、元信息等等，数据实体、实体关系是用于接收来自用户输入的数据。这些数据经过数据信息分析模块后会产生相应的类信息，这些类信息会被代码生成模块用于生成 Java 代码和相应的配置文件。无论是数据信息的数据，还是类信息数据，都采用面向对象的方式封装成各种各样的类。根据需求，把上述的数据抽象成两种类，第一种是用于描述数据信息的类：

数据实体(Entity)、实体之间的关系(EntityRelationship)、实体字段(EntityField)，三者关系如图 5 所示。

另外一种则是描述代码元信息的类：

类信息(ClassInfo)、字段信息(FieldInfo)、修饰符信息(ModifierInfo)、类型信息(TypeInfo)、注解信息(AnnotationInfo)、注解参数信息(AnnotationParameter)、方法信息(MethodInfo)，图 4 用于表示这 7 个类之间的关系。

图 6 展示了本系统从 Java 语法要素抽象得到类的类图。ClassInfo 表示一个 Java 的类信息，在 Java 语言当中，所有的字段、方法等都不能脱离类的存在而存在，抽象得到的类模型必然要以类为中心，这一点在图中则体现为 ClassInfo 类位于图的中心位置，表示方法和字段的 MethodInfo 类和 FieldInfo 类在 ClassInfo 中以字段的形式存在，表示这个类中定义的方法和字段。图中的 AnnotationInfo 表示注解信息。注解是 Java 在 JDK1.5 版本开始引入的一种在在代码当中的特殊标记。注解信息可以运用在类，字段和方法级别上，所以 MethodInfo、FieldInfo 和 ClassInfo 类当中都含有一个 AnnotationInfo 类型的集合类，用于储存这个方法、字段和类级别上的注解信息。Java 是一种强类型的面向对象编程语言，类型和访问

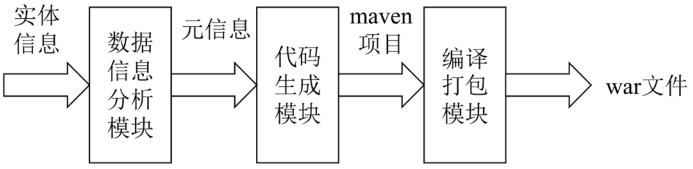

Figure 4. The workflow of system
图 4. 系统工作流程

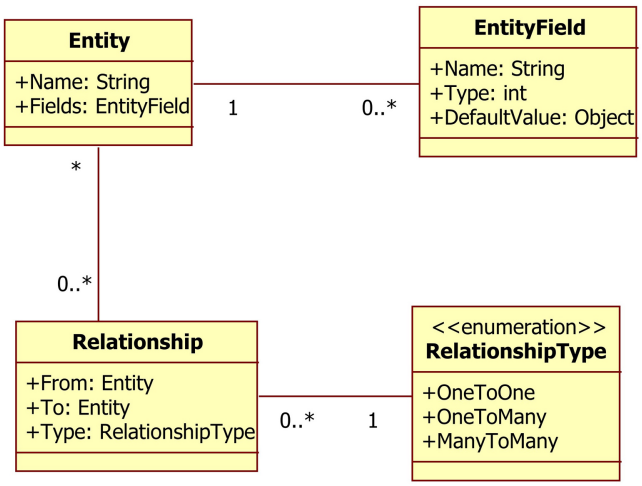

Figure 5. Class diagram of data information
图 5. 数据信息类的类图

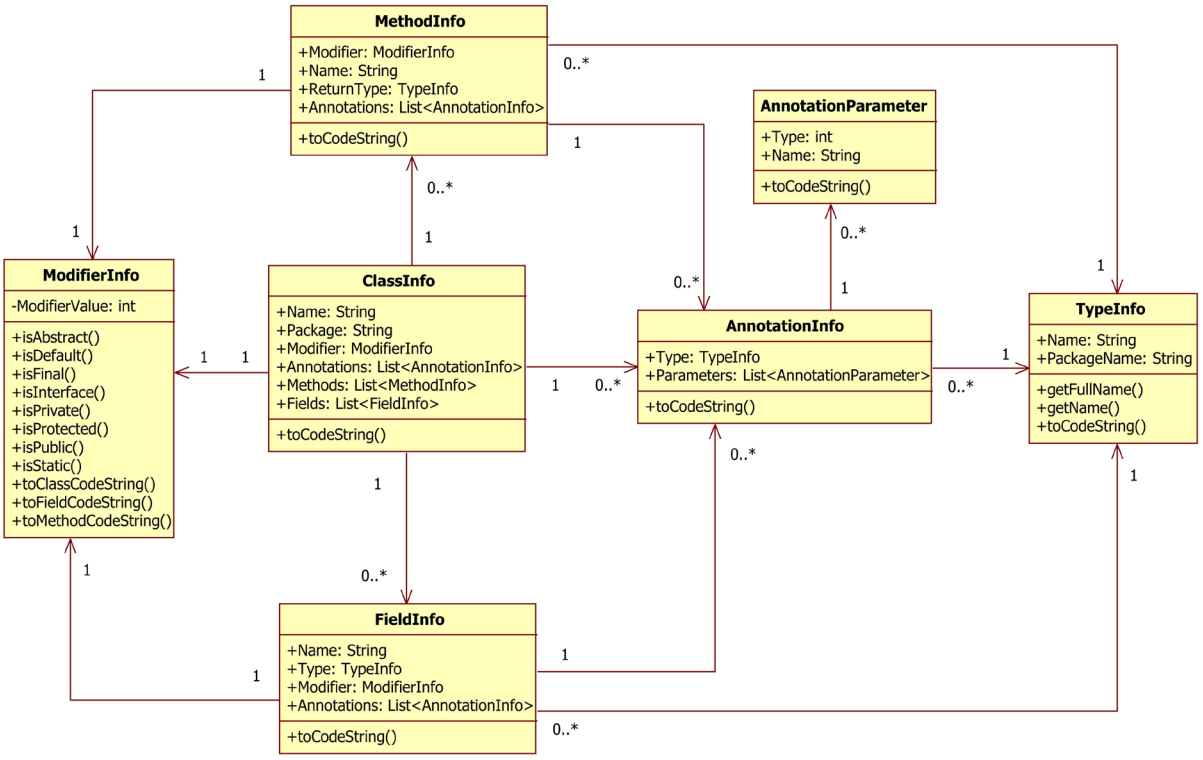

Figure 6. Class diagram of mate information
图 6. 元信息类的类图

控制机制是 Java 当中十分重要的内容。图 6 中 TypeInfo 类和 ModifierInfo 类分别用于表示类型和访问控制的访问修饰符，这两项信息在 Java 当中是极其重要的信息。其中，类、方法和字段都涉及到访问修饰符的信息，所以在这三者当中都含有一个 ModifierInfo 对象实例。方法、字段和注解都涉及类型信息，这三者对象当中都含有一个 TypeInfo 对象实例。

这些用于存放类信息的类可以根据 Java 语法规则的层次关系组织成树状的结构。树的根节点是表示类信息的 ClassInfo 对象，这个类下面的字段，方法都是这个根节点的子节点。字段和方法结点下面有这个字段的类型、修饰符信息等，这些信息对应的对象均以被修饰对象结点的子结点的形式存在。图 7 为一个带有 JPA 注解的 Java 类对应的类信息结构。

通过构建这样的一个树状的结构，可以层次化地将一个类的字段，方法，注解等信息进行完整描述。按树状的类信息组织，在代码生成阶段，把整个类的代码生成工作拆分成很多的注解代码生成，字段声明语句生成等等大量的工作，避免了整个系统的复杂度集中在少数的逻辑代码当中引起的系统复杂度的增加。

4. 关键技术的研究

4.1. 实体信息到元信息的转换

生成工具的目标系统结构采用的是目前应用比较广泛的分层结构，其项目层次如图 8 所示。

数据访问层封装数据库操作的逻辑，往上层暴露一系列操作数据库的接口。上层需要操作数据库时只需要调用数据访问层提供的接口，无需了解数据库操作的细节。数据访问层对具体的数据库访问操作对上层进行了屏蔽。在生成的目标系统当中，数据访问层是通过 Spring Data JPA 与 Hibernate 共同实现。

业务逻辑层是整个系统的核心，负责整个系统的主要业务流程。因为不同的系统的业务逻辑不尽相同，所以在默认生成的代码当中，业务实现类仅是调用数据访问层的接口，实现数据的传递功能，具体的业务逻辑代码需要使用者手动根据具体的业务逻辑进行编写。

表示层会根据用户的请求的内容格式将业务逻辑层返回的数据进行呈现。如果用户请求数据通过 XML 的格式进行呈现时，表示层会将业务逻辑层返回的数据序列化成为 XML 返回至客户端，其他格式的操作类似。

在元信息转换的过程中，一个核心过程是将用户输入的实体信息转换成含有用于 ORM 框架进行持久化的映射元数据的类信息。

本系统在实现实体信息转换成元信息过程中，采用“多次转换，逐步完善”的策略，通过多次对用户输入的数据信息进行分析，不断向元信息当中添加细节信息，最终完成类信息数据的构建。每次的转换都有其明确的目标：

1) 第一次转换是根据一定的规则，由实体名称生成类名，由实体字段的名称生成相应的字段名称。当类名和字段名与 Java 关键字重合时，根据相应的规则对类和字段名称进行重命名，为类和字段添加相应的注解信息。如果输入模型当中，实体类不存在一个能够唯一标识实体的主键，则自动创建一个名为 _id 用于标识实体的标识符，在持久化过程作为表的主键。

2) 第二次转换会对实体与实体之间关系进行处理，找出为类添加关联关系以后有可能出现的重名的字段，根据一定的策略对字段的名称进行修改。

3) 第三次转换是处理实体之间的关联关系，根据关联关系的信息，例如关系类型(一对多、一对一、多对多)、关系的方向等等为类信息添加相应的集合类或字段，并增加相应的注解。为各字段创建 get 和 set 方法。

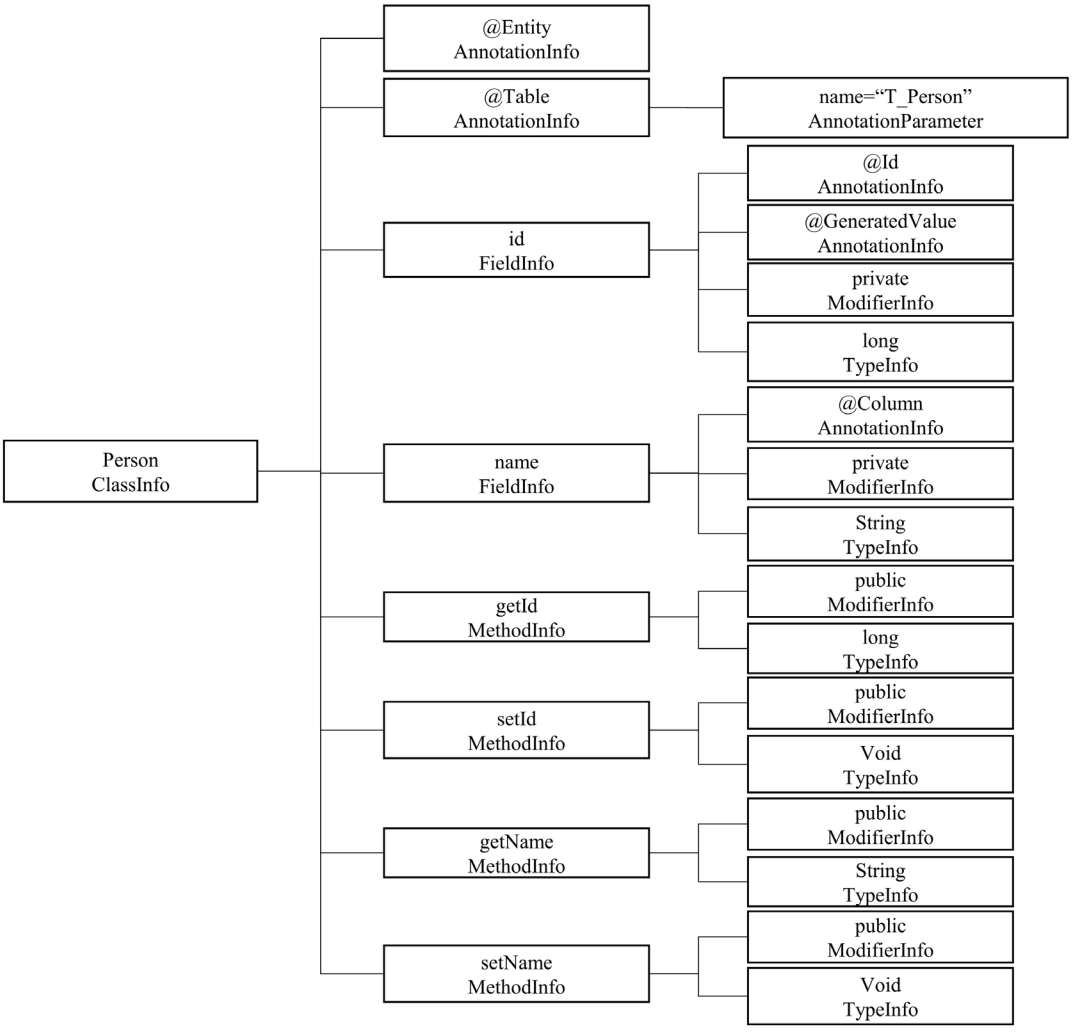

Figure 7. The structure of Person class information
图 7. Person 类对应的类信息结构

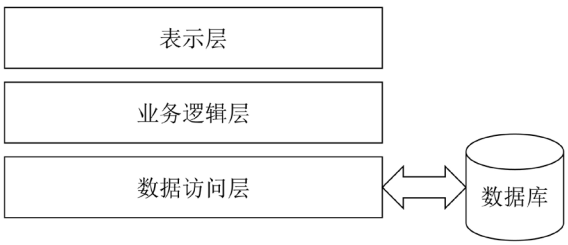

Figure 8. The hierarchy diagram of generated project
图 8. 生成项目层次图

4) 第四次转换是处理双向的实体关联关系中，在实体之间关联相关的字段，利用 Hibernate 的懒加载机制，增加相应的配置信息，解决从数据库取出数据时关联查询引起的死循环。

经过这四次的转换，用户输入的数据信息便可以被转换为相应的类信息。图 9 是一个用于描述学生和班级的模型，作为输入模型。

图 10 和图 11 是根据输入模型生成的类信息。

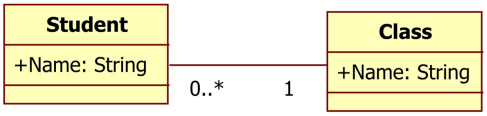

Figure 9. Input model
图 9. 输入模型

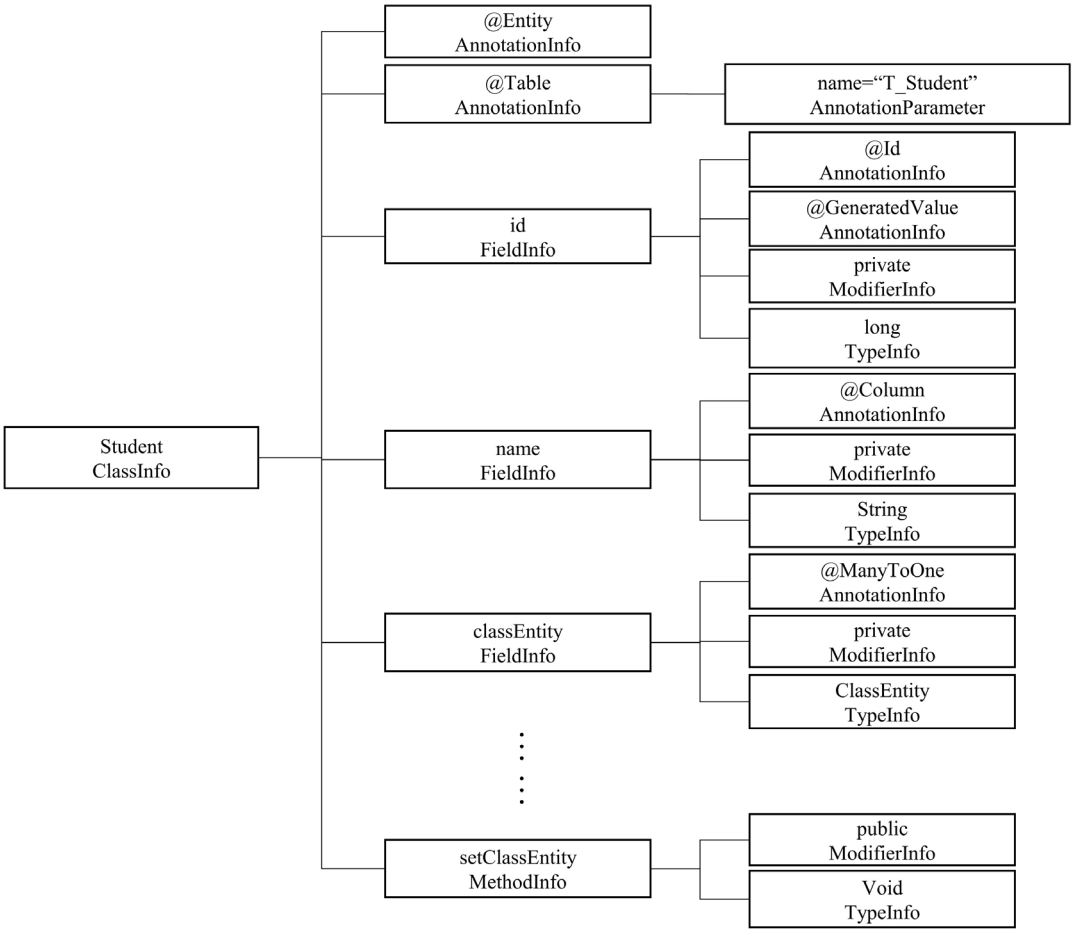

Figure 10. Information of Student class (get, set methods are omitted)
图 10. 生成的 Student 类的类信息(get, set 方法已省略)

4.2. 元信息生成代码的过程

在大部分的编程语言当中，语言当中的各个元素都可以通过树状的结构进行组织。储存类数据被设计成可以根据 Java 语法组织成树状结构的类，从而在代码生成的过程中可以充分利用树状结构的特点，从最小的元素开始生成，通过不断的组合过程最终生成类的完整代码。下面以一个 DAO 类为例，阐述代码生成的整个流程。DAO 类在项目当中被用于程序与数据库的交互，把所有的数据库操作封装为增删查改四个方法，对应的类信息的结构如图 12 所示。

在这个树状结构的每一个结点的对象均有一个 toCodeString 方法，这个方法的作用是将结点自身转化为符合 Java 语法的表达式。在这棵树的非叶子结点调用 toCodeString 方法时，会根据 Java 的语法按照一定的顺序调用子结点的 toCodeString 方法，将返回的所有的子结点表达式按照 Java 语法规则组合成一个合法的表达式。

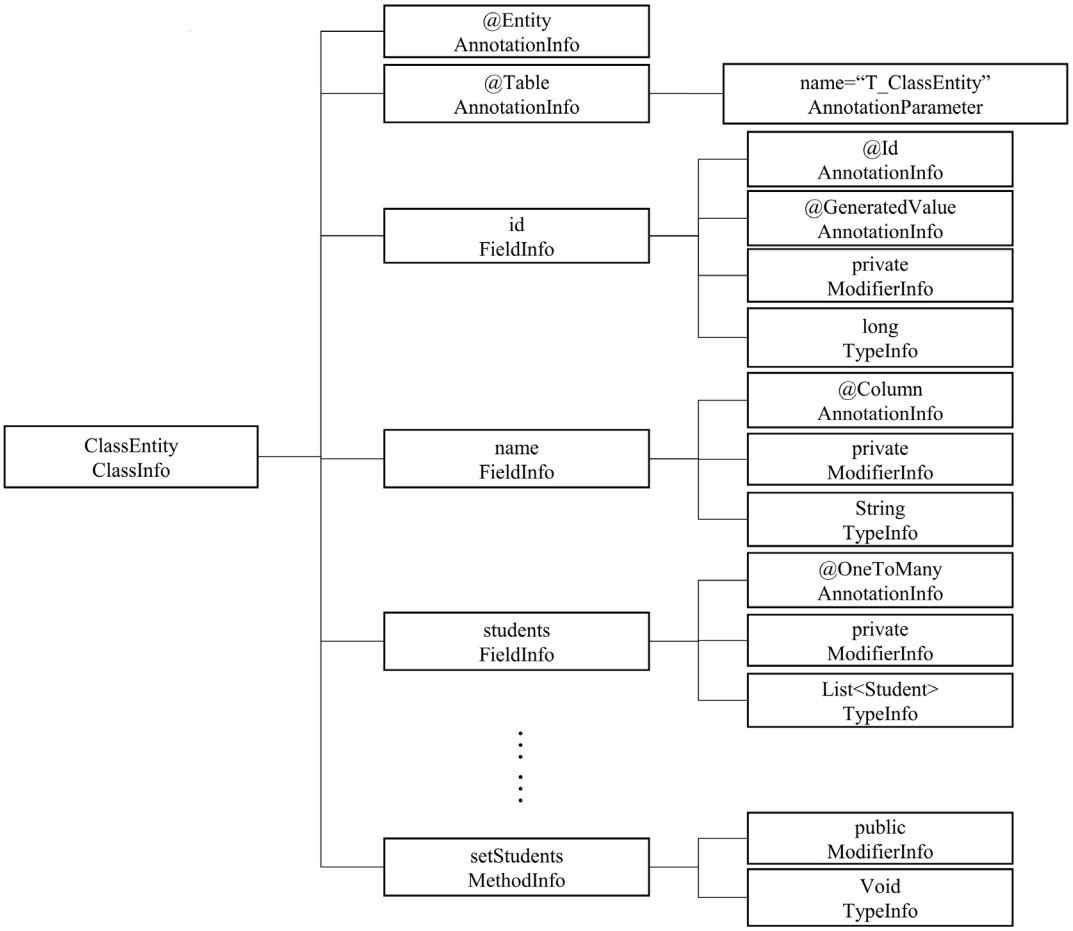

Figure 11. Information of ClassEntity class (get, set methods are omitted)
图 11. 生成的 ClassEntity 类信息(get, set 方法已省略)

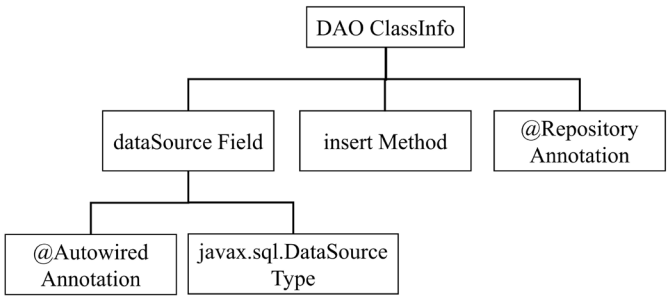

Figure 12. Class information of a simplified DAO class
图 12. 一个 DAO 类的类信息(简化后)

Java 的 import 语句可以在引用其他类时避免输入类所在包的包名，减少代码量。在本系统的代码生成模块当中，存在一个用于在当前上下文中管理 import 语句的组件，其主要目的是为了充分发挥 Java 当中 import 语句而达到减少代码数量和提高代码可读性的作用。生成上述 DAO 当中 dataSource 字段代码的示意图如图 13 所示。

import 管理组件当中保存着在当前的上下文当中所有的 import 语句，import 语句会在树状形式组织的根结点元素代码生成完成以后进行一次性的生成。示意图如图 14 所示。

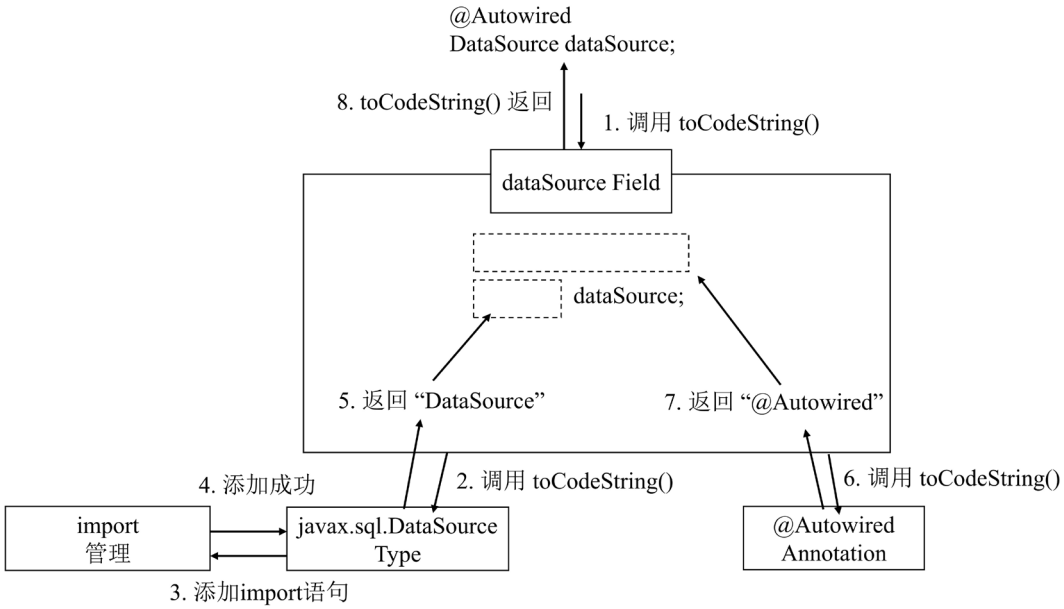

Figure 13. The workflow of field statement generating
图 13. 类字段生成的示意图

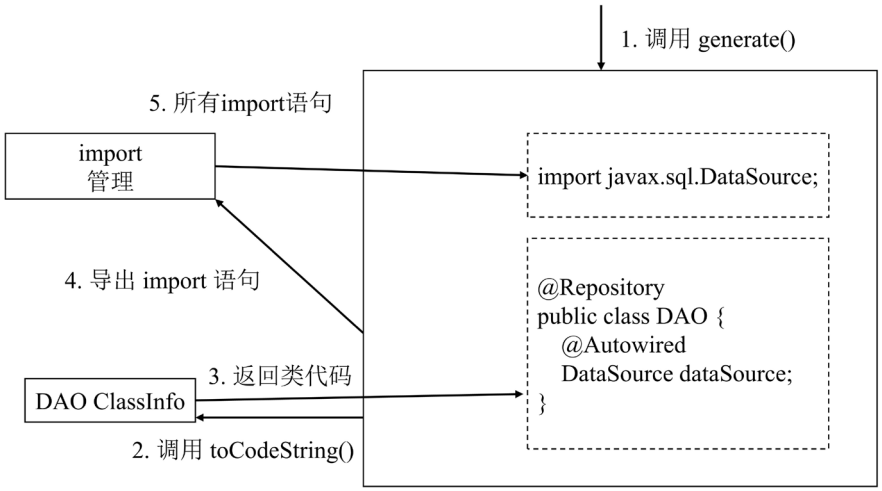

Figure 14. The workflow of import statements generating
图 14. Import 语句生成示意图

5. 结语

该 REST Web API 生成工具改变了现有代码生成工具以模板为基础实现代码生成的思路，采用了基于元信息代码生成的方式，可以在今后不断维护的过程中轻易地增加对新的开源框架或者技术的支持，也能够用于其他场合下代码生成的利用。另外，这个代码生成器当前仅仅能够生成较为简单的逻辑，对于复杂的业务逻辑，仍然需要在生成的代码当中人为地编写代码。在实践中代码重复较多的部分已经能够通过工具生成，开发人员可以从中解放出来，将主要的精力集中于软件主要逻辑代码的编写当中。

基金项目

本文得到广东海洋大学大学生创新创业训练计划项目(编号：CXXL2016133)资助。

参考文献 (References)

[1] 王建伟. 基于 Web Services 的 SOA 架构设计方法的研究[D]: [硕士学位论文]. 大连: 大连海事大学, 2006.

[2] Trachtenberg, A. (2003) PHP Web Services without SOAP. http://www.onlamp.com/pub/a/php/2003/10/30/amazon_rest.html

[3] Fielding, R.T. (2000) Architectural Styles and the Design of Network-Based Software Architectures. University of California, Oakland, 88.

[4] Fowler, M. (2002) Patterns of Enterprise Application Architecture. Addison-Wesley Professional, Boston.

[5] Fowler, M. (2010) Richardson Maturity Model. https://martinfowler.com/articles/richardsonMaturityModel.html

[6] Herrington, J. (2003) Code Generation in Action. Manning Publications Co., Greenwich.

[7] JSR 220, Enterprise JavaBeans, Version 3.0 Java Persistence API.

[8] 林寒超, 张南平. Hibernate 技术的研究[J]. 计算机技术与发展, 2006, 16(11): 112-113.