|  |
| --- |
| **REST风格Web API自动生成系统的设计与实现** |

|  |
| --- |
| Design and Implementation of REST-Style Web API Generation Tool |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学生姓名 | 邓煜荣 | 学号 | | 201311701407 | |
| 所在学院 | 数学与计算机学院 | | 班级 | | 软件1134 |
| 所在专业 | 软件工程 | | | | |
| 申请学位 | 工学学士学位 | | | | |
| 指导教师 | 高升 | | 职称 | | 副教授 |
| 副指导教师 |  | | 职称 | |  |
| 答辩时间 | 年 月 日 | | | | |

**目 录**

[设计总说明 I](#_Toc482346226)

[**introduction** II](#_Toc482346227)

[1 绪论 1](#_Toc482346228)

[1.1 本设计的背景 1](#_Toc482346229)

[1.2 本设计的需求来源 2](#_Toc482346230)

[1.3 国内外的研究情况 2](#_Toc482346231)

[1.4 本设计要解决的问题及其意义 2](#_Toc482346232)

[1.5 本设计要达到的目标 3](#_Toc482346233)

[2 相关技术概况 3](#_Toc482346234)

[2.1 REST 架构风格 3](#_Toc482346235)

[2.2 Spring Boot 4](#_Toc482346236)

[2.3 Spring Data JPA 4](#_Toc482346237)

[2.4 Spring HATEOAS 4](#_Toc482346238)

[2.5 Hibernate 5](#_Toc482346239)

[2.6 Swagger UI 5](#_Toc482346240)

[3 需求分析 5](#_Toc482346241)

[3.1 功能分析 6](#_Toc482346242)

[3.1.1 项目基本信息管理功能 6](#_Toc482346243)

[3.1.2 实体信息管理功能 7](#_Toc482346244)

[3.1.3 关系信息管理功能 7](#_Toc482346245)

[3.1.4 代码生成功能 8](#_Toc482346246)

[3.2 用例说明 8](#_Toc482346247)

[3.2.1 项目信息修改用例 8](#_Toc482346248)

[3.2.2 增加实体信息用例 9](#_Toc482346249)

[3.2.3 修改实体信息用例 9](#_Toc482346250)

[3.2.4 删除实体信息用例 10](#_Toc482346251)

[3.2.5 增加关系信息用例 10](#_Toc482346252)

[3.2.6 修改关系信息用例 11](#_Toc482346253)

[3.2.7 删除关系信息用例 11](#_Toc482346254)

[3.2.8 代码生成用例 12](#_Toc482346255)

[3.2.9 代码生成配置用例 12](#_Toc482346256)

[3.3 非功能性需求 13](#_Toc482346257)

[3.3.1 代码生成工具扩展性 13](#_Toc482346258)

[3.3.2 生成代码可读性 13](#_Toc482346259)

[3.3.3 代码生成工具的易用性 13](#_Toc482346260)

[4 代码生成工具设计 14](#_Toc482346261)

[4.1 整体架构设计 14](#_Toc482346262)

[4.2 数据模型的建立 15](#_Toc482346263)

[4.2.1 实体信息数据模型 15](#_Toc482346264)

[4.2.2 元数据信息数据模型 16](#_Toc482346265)

[4.3 类元数据生成逻辑的设计 18](#_Toc482346266)

[4.3.1 模式固定的类元数据生成逻辑设计 18](#_Toc482346267)

[4.3.2 实体类元数据生成逻辑设计 19](#_Toc482346268)

[4.3.3 DAO接口元数据生成逻辑设计 24](#_Toc482346269)

[4.3.4 Service类元数据生成逻辑设计 25](#_Toc482346270)

[4.3.5 Controller类元数据生成逻辑设计 26](#_Toc482346271)

[4.3.6 HATEOAS资源类元数据生成逻辑设计 26](#_Toc482346272)

[4.3.7 HATEOAS资源装配类元数据生成逻辑设计 26](#_Toc482346273)

[4.3.8 解决Controller和HATEOAS资源装配类之间相互依赖逻辑设计 26](#_Toc482346274)

[4.4 代码生成逻辑的设计 26](#_Toc482346275)

[4.4.1 代码生成的策略 26](#_Toc482346276)

[4.5 用户界面的设计 27](#_Toc482346277)

[4.5.1 主界面布局的设计 27](#_Toc482346278)

[4.5.2 基本信息工作页面设计 27](#_Toc482346279)

[4.5.3 实体信息工作页面设计 28](#_Toc482346280)

[4.5.4 关系信息工作页面设计 28](#_Toc482346281)

[4.5.5 代码生成工作页面设计 29](#_Toc482346282)

[5 实现效果 29](#_Toc482346283)

[5.1 目标系统的描述 30](#_Toc482346284)

[鸣 谢 31](#_Toc482346285)

[参考文献 32](#_Toc482346286)

# 设计总说明

REST风格的Web API近年来得到广泛的应用。在REST风格Web API的开发实践当中，大量重复代码的编写调试工作对项目开发的进度造成极大的影响。重复代码的自动生成开始成为软件公司降低成本和缩短开发周期的解决方案，本设计是代码生成方案的具体实现。本设计根据用户所输入的实体信息，自动生成基于Spring Boot、Spring Data JPA、Spring HATEOAS的REST风格Web API中基本的实体增删查改和级联操作的实现代码。开发人员只需要将时间和精力投入到核心业务流程的开发当中，很大程度缩短了开发周期和降低开发和维护成本。

当前很多的代码生成工具均采用基于模板方式的代码生成，本设计尝试采用基于类信息进行代码生成。基于类信息代码生成方式会导致代码生成工具复杂程度的上升，基于类信息的代码生成工具本身的复杂度较高，本代码生成工具也不例外。本代码生成工具当前是采用代码对类信息的生成逻辑进行控制，可以改进为使用XML或其他的技术控制实体到类信息的转换过程，降低代码的复杂程度和耦合程度。

关键词：代码生成；Web API；REST风格

# **introduction**

REST-Style Web API has been widely used in recent years. A large number of duplicated code was written and debugged during the development of REST-Style, which has a negative effect on the development project. Generating these duplicated code is a solution to shorten the time and lower the cost of development, and this code generator is an implementation of this solution. A Spring Boot, Spring Data JPA, Spring HATEOAS based project will be generated by the code generator according to the entity information from user. The project generated includes the code for CURD of entities and cascade operations. The crew can put most their time and energy into the development of core business, which will play a positive effect to shorten the time and lower the cost of development and maintenance.

Nowadays, template based code generating is widely used in many code generator. This design makes an attempt to generate code based on class information. Class information based generating enables the generator tool handles use requirement more flexibly. However, class information based generating brings the complex of implementation of generator tool, this code generator is not the exception. The process of converting from entity information to class information is manipulated by code for now, and the improvement of this generator tool is to use XML or other similar technology to manipulate logic of converting, which will reduce the complex and coupling of code.

**Keywords: code generation; Web api; REST-Style**

**REST风格Web API自动生成系统的设计与实现**

软件工程，201311701407，邓煜荣

指导教师：高升

毕业设计说明书

# 绪论

## 本设计的背景

随着互联网的发展，Web服务在近年来得到了广泛的应用。Web服务在正确地部署到服务器以后，可以通过互联网进行访问。在这个过程当中，采用标准化的XML格式等多种平台无关的技术，大幅度提高了软件的重用率和互操作性。SOA（Service-Oriented Architecture，面向服务架构）让Web服务得到了很大的发展，相关的标准和技术不断地出现，对Web服务的每个细节都做出了相应的规定，使其更加标准化，进一步降低了异构平台的访问成本，其中对Web服务影响较大的协议有SOAP（Simple Object Access Protocol，简单对象访问协议）和XML-RPC（XML Remote Procedure Call，XML远程过程调用），但在标准化的过程中带来臃肿和复杂逐渐被使用者和开发人员诟病。

2000年，Roy Fielding在其论文《Architectural Styles and the Design of Network-based Software Architectures》中提出了REST（Representational State Transfer，表述性状态转移）软件架构风格，提倡无状态、使用HTTP（Hypertext Transfer Protocol，超文本传输协议）请求方法等简易的措施代替复杂的SOAP访问。

受到REST架构风格的启发，Web API（Web Application Programming Interface，Web应用程序编程接口）开始被广泛使用。Web API是Web服务的一种轻量级实现，Web API通常采用JSON（JavaScript Object Notation，JavaScript对象标记语言）作为数据交换格式，通过URI携带参数等等的更为简洁的方式实现Web服务的访问。相比起传统的Web服务，Web API的实现难度低，性能开销小，而且JSON格式的数据浏览器可以以较低的性能开销处理，这些的优势都让Web API成为当今开发者欢迎的技术。

近年来移动互联网的发展，移动设备的处理速度和储存容量使其难以离开云端的服务器完成计算任务。在资源受限设备上，Web API能够更好地完成移动设备与云端之间的数据交互。另外，在Web应用功能越来越多，架构越来越复杂时，Web API是Web应用的前后端分离可行的解决方案。无论是互联网行业还是传统的软件都有着较大的Web API开发需求。

## 本设计的需求来源

Web API的需求随着互联网发展的不断发展而日渐增加。采用Web API实现前后端分离的Web应用逐渐增多，根据现有的代码分析，大部分的Web API仅仅是实现简单的数据查询、封装等简单操作，这类简单操作的实现代码极为类似，这类的代码采用人工编写的方式会浪费大量时间。在总结代码库当中所有项目的数据访问、数据封装等功能的重复代码以后，得出一个Web API实现的模板，并以此为蓝本生成Web API的项目。

这个代码生成工具是一家创业公司用于快速开发外包项目的解决方案，可以生成用于评估实体设计是否合理的原型以及在此之上进行开发的工程框架。

## 国内外的研究情况

代码生成工具是根据用户输入的领域模型以及其他信息生成相应的代码。

国外的软件公司有着自己的代码生成解决方案。一种较为常见的解决方案是把代码生成功能整合在建模工具里面，例如蓝色巨人国际商业机器公司收购的Rational公司的产品Rose在代码生成方面比较成熟。用户在Rose软件上完成建模以后，可以生成相应的模型代码。与Rose软件相似的PowerDesigner，同样可以根据用户在其之上建立的模型生成相应的模型代码。另外一种较为常见解决方案是是将代码生成的功能整合到IDE（Integrated Development Environment，集成开发环境）当中，这种解决方案做得比较好的产品有微软公司旗下的开发工具Visual Studio。Visual Studio通过一系列的向导收集用户的需求信息，向导完成以后IDE会自动根据信息生成项目的基本框架，开发人员便可在此之上进行高层业务逻辑的开发。

国内在代码生成方面的研究相对较少。国内一些中间件厂商会有相关的一些代码生成工具，但是其作用仅仅是限于该中间件相关的代码，适用范围窄，应用场合单一等等的缺点。

## 本设计要解决的问题及其意义

REST风格的Web API概念自提出以来，产生了不少相关的开发框架和支撑技术。比如Spring MVC、RESTLet等，开发人员利用这些框架可以快速地构建和实现REST风格的Web API。但在实践当中，使用这些框架和工具的过程中难以避免涉及到大量数据库操作和文档撰写等机械而重复的工作。REST风格的Web API的操作已经被限定使用HTTP协议的GET、POST、DELETE、PUT等，对于不同对象的相同操作，无论是文档还是实现代码都有很高的相似程度，但是开发人员却不得不在这些重复的机械劳动上花费时间和精力，最终影响到了整个项目的开发周期和开发成本。

本设计就是为了解决REST风格Web API实现过程当中重复劳动的问题。开发人员只需要在开发过程中定义好系统的实体、关联关系等信息，生成工具会根据这些信息生成数据库操作、接口文档等文档。开发人员只需要编写整个系统当中核心的业务逻辑，这样可以把开发人员的精力集中于核心的业务逻辑开发当中。

## 本设计要达到的目标

本设计要达到的目的是一个能够生成基于Spring Boot、Hibernate、Swagger技术的带文档说明的层次化的Web API实现。生成的Web API项目按层次可以分为了数据访问层、业务逻辑层和数据表示层三层的结构。

数据访问层封装了数据库访问逻辑，往上层暴露相应的接口，在数据访问层之上的业务逻辑层无需了解数据访问的细节和具体实现。在生成的目标系统当中，数据访问层是通过Spring Data JPA和Hibernate两者共同实现的。

业务逻辑层是整个系统的核心部分，不同的目标系统处理的业务逻辑不一样，所以业务逻辑是留空由开发人员进行实现的。生成的系统当中，业务逻辑层包含有最基本的级联保存和删除的代码。

数据表示层主要由各种的Controller类组成，Controller则是响应用户的Web API调用的组件。这些组件通过Spring MVC的数据绑定机制获得用户调用Web API传递过来的参数，经过简单的预处理逻辑以后传递到业务逻辑层，业务逻辑层执行业务逻辑以后，将返回的数据转换层Spring HATEOAS的Resource类，传递到Spring MVC并自动将其序列化成JSON响应并输出返回给调用者，完成整个API的调用流程。

# 相关技术概况

本设计生成的Web API采用REST架构风格。在实现上，采用了Spring Boot、Spring HATEOAS、Spring Data JPA以及Hibernate技术，Web API的文档和相应的展示页面采用Swagger UI技术、Web API工具的用户界面采用Java Swing实现。

## REST 架构风格

REST架构风格是Roy Thomas Fielding所提出的一种软件架构风格。这种的架构风格提倡将超媒体的分布式系统做到从无到有、利用客户端-服务器方式、无状态会话、缓存、统一接口、分层以及按需编码等等原则，通过资源、资源标识符、表示、表示元数据、资源元数据以及控制数据等概念对软件系统当中涉及的数据进行抽象。

上面这些原则在实践当中，将由系统分析得出的实体信息、元数据等抽象为资源，由服务器为这些资源分配一个唯一的标识符，具体实现为URI（Uniform Resource Identifier，统一资源标识符）。把业务的逻辑抽象成为对服务器上不同资源的操作，操作采用HTTP协议中定义的方法来制定对资源进行什么样的操作，采用HTTP的状态码标识操作的执行结果。另外，对于一些变化较少的资源信息，可以同时给出缓存相应的数据，由客户端对这些资源进行缓存，降低了网络的访问次数，继而提高服务的效率。不难看出，REST风格的软件架构并不是什么新的概念，只是对HTTP协议当中的概念进行了回归。

随着REST架构风格的广泛应用，Leonard Richardson提出了一个用于衡量Web API以及Web服务的成熟度的模型。这个成熟度模型根据特定的标准将成熟等级分成等级0到等级3共四个等级，采用HTTP协议作为传输协议即满足等级0的标准；等级1在等级0的基础上采用了分治法把复杂的系统分解成资源，所有的数据和服务都抽象成为资源，采用资源的唯一标识符进行访问；等级2在等级1的基础之上引入了采用HTTP动词的约束，对所有资源的操作进行了统一，避免了不必要的复杂性；等级3引入了HATEOAS（Hypermedia As The Engine Of Application State，超媒体作为应用状态引擎）的约束，满足该约束的Web服务仅仅需要暴露一个入口的接口，该接口的返回值当中含有其他的资源的URI，通过这种方式，为客户端提供了自发现和自我描述的能力，客户端可以如同人类访问网站自如使用Web服务。

## Spring Boot

Spring Boot是由著名的Java开发框架Spring Framework的管理者Pivotal软件公司提供的一种用于快速开发Spring Framework程序的解决方案。Spring Framework强大的功能受到不少Java开发人员的热爱。但随着其发展已经功能的不断增加，Spring Framework的配置难度和繁琐程度也随之增大，其组件化的设计在依赖管理方面增加了相应的成本。Spring Boot的出现就是为了这些问题。Spring Boot大量采用约定优于配置的原则，内置大量预置的参数，开发人员只需要进行极少量的配置即可以完成Spring Framework的IoC（Inversion of Control，控制反转）和AOP（Aspect Oriented Programming，面向切面编程）功能以及Spring MVC的配置，接近“开箱即用”的效果。另外，在依赖管理方面，Spring Boot将复杂的Maven依赖封装成了一个又一个的Starter，开发人员只需要根据所需的功能引入相应的Starter即可，在很大程度上降低了配置的难度和复杂程度。

## Spring Data JPA

Spring Data同样是由Pivotal软件公司为了简化Spring Framework数据访问的技术，提供了包括传统的关系型数据库和非关系型数据库的访问接口。Spring Data JPA（Java Persistence API，Java持久化应用程序编程接口）是Spring Data项目下的子项目，通过JPA元数据定义数据操作过程持久化的行为，常常被用于访问关系型数据库。使用Spring Data，开发人员要进行数据访问，所需的操作只是按照一定的规则对DAO（Data Access Object，数据访问对象）的方法进行命名，Spring Data会解析方法的名称并转换成相应的数据访问操作，免去了数据访问代码的编写，进一步提高了软件系统的开发效率。Spring Data项目本身仅仅提供的是一种编码模式，不提供数据访问的实现，数据访问的实现需要由第三方的ORM（Object/Relationship Mapping，对象关系映射）框架提供。

## Spring HATEOAS

Spring HATEOAS是由Pivotal软件公司提供，用于Spring Framework快速开发成熟度较高的Web API和Web 服务的一组工具，Spring HATEOAS提供一组快速构建具备HATEOAS功能的响应对象，是一种快速实现HATEOAS风格的Web API的解决方案。

## Hibernate

Hibernate是一个成熟的开源全自动ORM框架。使用Hibernate，开发人员可以无需对数据访问的底层细节进行关注，无需编写SQL（Structured Query Language，结构化查询语言）以及数据记录与对象之间转换的代码即可完成数据库的访问。在使用Spring Data JPA以后，无需进行任何配置即可与Spring Data JPA进行整合。

Hibernate高度的自动化，难以避免会带来一定的性能开销，而且Hibernate生成的SQL语句晦涩难懂。所以Hibernate优化技能学习曲线极其陡峭，这在一定程度上制约了Hibernate的实践当中的应用。本设计在实现过程中会预留ORM框架更换的条件，可供更换使用其他的ORM框架。

## Swagger UI

Swagger UI是一个用于生成Web API可视化和交互界面的一个工具。Swagger工具的工作原理是通过在控制器方法以及模型对象至上增加Swagger的注解信息，描述这个方法返回值、参数等信息，Swagger UI会自动根据这些注解信息生成相应的说明和交互界面，开发人员不需要为此编写一句的代码和一个的HTML页面。

# 需求分析

本设计的核心需求是不同用途的REST风格Web API的快速开发以及快速构建原型，主要面向的用户是软件的设计人员和开发人员。这个代码生成工具的使用是在需求和设计人员完成对软件系统的需求分析的设计以后快速生成相应的代码框架。这个框架可以用于设计合理性的评估，也可以作为开发人员的开发基础，在生成的代码之上进行编码工作。

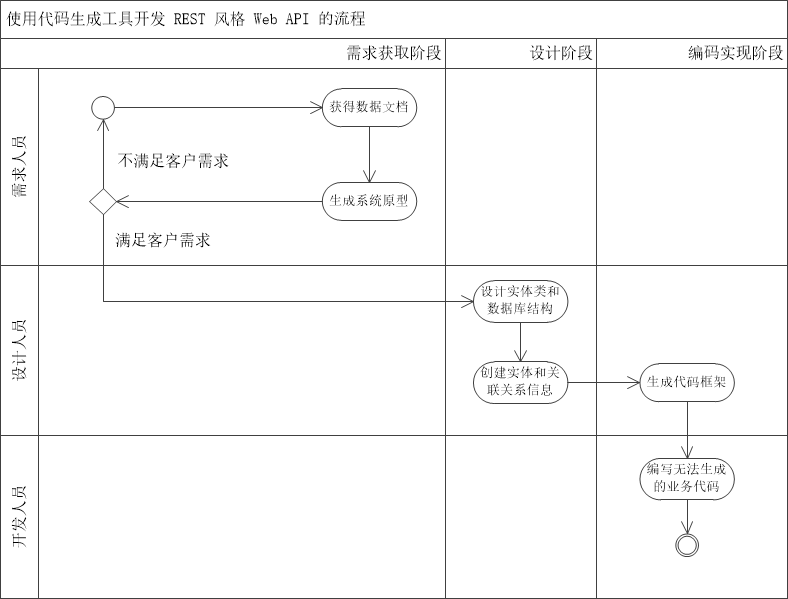


图3-1 使用代码生成工具开发 REST 风格 Web API 的流程图

## 功能分析

本代码生成工具在执行代码生成以前，需要能够接受项目的基本信息、项目的实体信息、项目的关系信息的输入，同时需要工程的基本信息、实体信息、关系信息等信息进行管理。

### 项目基本信息管理功能

项目信息是描述所要生成的目标项目的一组数据。这组数据对目标系统的业务流程不会产生较大的影响，但是会影响到最终生成系统的配置信息。项目的基本信息包括以下的内容：

(1) 项目名称

(2) 项目版本

(3) 项目的开发者

(4) 项目的包名

(5) 数据库的账号密码

(6) 数据库的服务器地址

这些信息最终会体现在项目的配置文件和工程结构当中，属于项目的必填项目。项目基本信息的修改允许用户通过表单的形式对项目基本信息的每一项进行填写、查看和保存。

### 实体信息管理功能

实体信息是用于描述由软件设计人员根据软件需求设计所设计的实体的信息。每个实体信息包含有下面的内容：

(1) 实体的名称

(2) 实体的描述

(3) 实体当中的属性信息

每个属性信息又包含下面的内容：

(1) 属性的名称

(2) 属性的描述信息

(3) 属性的数据类型

(4) 这个属性是否会被用作实体的唯一标识符

(5) 这个属性是否会被用作数据库的查询条件

本代码生成工具需要完成实体信息的管理功能，即实体信息的增加、删除、修改功能。实体信息具备一定的约束条件，在实体管理功能当中必须遵循这些约束。实体信息管理的约束有：

(1) 在一个项目中，每个实体的名称都必须是唯一的。项目范围内不能出现同名实体信息

(2) 每个实体都必须具有一个可以用于唯一标识实体的标识属性。

### 关系信息管理功能

关系信息用于描述由软件设计人员根据软件需求设计所得到的实体间关系的信息。每个关系信息包含下面的内容：

(1) 关系名称

(2) 关系的描述

(3) 关系双方的实体信息

(4) 关系的维护方

(5) 关系的类型

(6) 关系相关的字段信息

本代码生成工具需要完成关系的管理功能，即关系信息的增加、删除、修改功能。与实体信息管理类似，关系信息在管理过程中同样需要遵循相关的约束。关系信息管理功能需要遵循的约束有：

(1) 在一个项目中，每个关系的名称都必须是唯一的。项目范围内不能出现同名的关系信息

(2) 关系双方的实体信息必须是存在并且有效的实体

(3) 关系的维护方实体必须是关系双方其中之一

(4) 关系的类型必须是一对一、一对多、多对一和多对多四种之中的其中一种

(5) 关系相关的字段名称不能与这个实体属性相关的字段名称以及这个实体其他关系相关字段同名

### 代码生成功能

代码生成功能是在软件设计人员或者软件需求人员在完成实体信息和关系信息的录入以后，由软件设计人员执行的产生最终代码的功能。代码生成功能中的逻辑是整个系统中的核心逻辑。代码生成功能所需的数据由实体信息管理和关系信息管理进行输入，代码生成功能在执行时无需用户输入额外的信息，只需要指定用于存放所生成的代码文件的路径即可。

在用户执行代码生成的指令以后，系统会完成下面的几个子操作：

(1) 检查工程信息是否有误

(2) 检查实体和关系信息是否违反了相应的约束条件

(3) 执行代码的生成逻辑

如果上述的三个步骤均顺利完成，代码生成工具会将生成的代码文件存放到用户所指令的目录；否则会弹出相应的错误信息，以供用户对信息进行检查和修改。

## 用例说明

根据系统所需的功能，可以得出这个系统的九个用例。本代码生成工具的所有用例如图3-2所示。

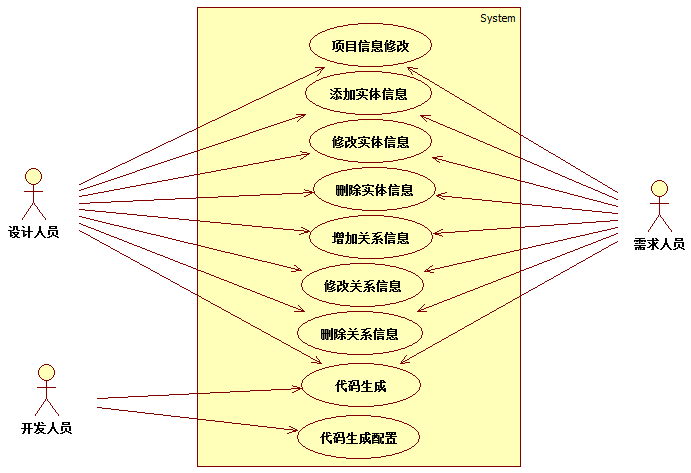


图3-2 系统用例图

### 项目信息修改用例

表3-1 项目信息修改用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 项目信息修改 |
| 用例ID | CGS-XMGL-1 |
| 角色 | 设计人员、需求人员 |
| 用例说明 | 设计人员、需求人员对项目的信息进行修改 |
| 前置条件 | 无 |
| 基本事件流 | 1. 用户打开项目信息编辑界面  2. 系统检索当前项目的项目信息，显示到界面上  3. 用户通过界面修改项目信息  4. 用户点击“保存”，系统将当前界面上的项目数据保存 |
| 可选事件流 | 无 |
| 异常事件流 | 1. 事件流中每一步发生的异常都会弹出相应的提示供用户处理   4a. 当系统没法正确保存修改后的项目信息，弹出相应的错误消息 |
| 后置条件 | 修改用户界面中显示项目名称的位置 |

### 增加实体信息用例

表3-2 增加实体信息修改用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 增加实体信息 |
| 用例 | CGS-STGL-1 |
| 角色 | 设计人员、需求人员 |
| 用例说明 | 设计人员、需求人员增加一个实体信息到系统中 |
| 前置条件 | 系统已经打开一个有效工程，处于可以添加实体的状态 |
| 基本事件流 | 1. 用户单击工具栏或者菜单栏的添加实体按钮  2. 系统创建一个实体编辑的页面并打开  3. 用户在实体编辑页面中输入实体的信息  4. 系统对输入的信息进行校验  5. 系统将用户实体的实体信息保存到项目当中 |
| 可选事件流 | 4a. 如果用户要保存的实体名称与工程中已有的实体名称相同，则弹出相应的提示信息，通知用户进行修改。  4b. 如果这个实体中不存在用作唯一标识符的属性字段，则弹出相应的提示信息，通知用户进行修改。 |
| 异常事件流 | 1. 事件流中每一个步骤发生异常均会提示用户  2a. 如果系统无法创建新的实体编辑页面，弹出提示消息并给出一些解决措施  5a. 如果用户在保存实体过程发生错误，弹出提示消息并给出一些解决过措施 |
| 后置条件 | 更新界面中实体信息的列表 |

### 修改实体信息用例

表3-3 修改实体信息修改用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 修改实体信息 |
| 用例 | CGS-STGL-2 |
| 角色 | 设计人员、需求人员 |
| 用例说明 | 设计人员、需求人员修改实体信息 |
| 前置条件 | 系统已经打开一个工程，而且要进行修改的实体存在有效 |
| 基本事件流 | 1. 用户通过主界面左侧的工程结构树中双击要进行修改的实体信息  2. 系统在工程数据中找出用户双击的实体信息  3. 如果这个实体对应的页面已经打开，则将这个页面放到前台；否则创建一个实体编辑页面，将工程当中的数据填充到工作页面当中。  4. 用户在实体编辑页面修改实体信息。  5. 用户点击保存，校验数据，通过以后将实体数据保存到工程数据当中。 |
| 可选事件流 | 3a. 如果要进行编辑的实体信息不存在，则弹出相应的提示信息。  5a. 如果用户输入的实体信息检验不通过，将检验错误和修正建议显示，供用户检查改正 |
| 异常事件流 | 1. 事件流当中每个步骤抛出的异常会以提示的方式提醒用户  3a. 如果用查询实体信息或者往界面填充实体信息失败时产生相应提示 |
| 后置条件 | 更新界面中的实体列表 |

### 删除实体信息用例

表3-4 删除实体信息用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 删除实体信息 |
| 用例 | CGS-STGL-3 |
| 角色 | 设计人员、需求人员 |
| 用例说明 | 设计人员、需求人员删除实体信息 |
| 前置条件 | 系统已经打开一个工程，而且要执行删除的实体存在 |
| 基本事件流 | 1. 用户在实体列表中选择要删除的实体，通过右键菜单或者工具栏点击删除实体 2. 弹出一个询问框，询问用户是否确认删除实体 3. 扫描工程中的关系数据，判断这个实体是否存在相关的关系，并且要删除的实体不是关系的维护方。如果存在，则弹出相应的提示信息，告知用户当前实体不能被删除 4. 扫描工程中的关系数据，将这个实体维护的关系信息删除 5. 删除选中的实体信息 6. 刷新界面的实体列表 |
| 可选事件流 | 2a. 如果用户没有确认删除实体，也不会执行任何的操作 |
| 异常事件流 | 1. 事件流中每个步骤抛出的异常都会以提示的方式提醒用户 |
| 后置条件 | 更新界面中的实体列表 |

### 增加关系信息用例

表3-5 增加关系用例的用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 添加关系信息 |
| 用例 | CGS-GXGL-1 |
| 角色 | 设计人员、需求人员 |
| 用例说明 | 设计人员、需求人员增加关系信息 |
| 前置条件 | 系统已经打开一个工程，而且工程当中已经存在有效的实体信息 |
| 基本事件流 | 1. 如果当前工程当中不存在有效的实体信息，则弹出相应的提示信息，供用户检查修改 2. 系统创建一个关系信息编辑页面，并将这个页面显示 3. 用户输入关系的数据 4. 用户点击保存关系。系统会对用户输入的关系信息进行检验。 5. 系统保存用户输入的关系信息 |
| 可选事件流 | 1a. 如果当前的工程数据中不存在任何有效的实体信息，也弹出相应的提示信息，告知用户进行检查 |
|  | 4a. 如果用户输入或者选择的关系两端的实体不存在，或者关系的维护方实体不属于关系两端的实体，则显示相应的提示信息，由用户检查并处理  4b. 如果工程当中存在同名的关系，则显示相应的提示信息，由用户检查并处理 |
| 异常事件流 | 1. 事件流中每个步骤发生的异常都会弹出相应的提示信息 |
| 后置条件 | 更新主界面中的关系列表 |

### 修改关系信息用例

表3-6 修改关系信息用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 修改实体信息 |
| 用例 | CGS-GXGL-2 |
| 角色 | 设计人员、需求人员 |
| 用例说明 | 设计人员、需求人员修改关系信息 |
| 前置条件 | 系统已经打开一个工程，而且要进行修改的关系存在有效 |
| 基本事件流 | 1. 用户通过主界面左侧的工程结构树中双击要进行修改的实体信息  2. 系统在工程数据中找出用户双击的关系信息  3. 如果这个关系对应的页面已经打开，则将这个页面放到前台；否则创建一个关系编辑页面，将工程当中的数据填充到工作页面当中。  4. 用户在关系编辑页面修改关系信息。  5. 用户点击保存，校验数据，通过以后将关系数据保存到工程数据当中。 |
| 可选事件流 | 3a. 如果要进行编辑的关系信息不存在，则弹出相应的提示信息。  5a. 如果用户输入的关系信息检验不通过，将检验错误和修正建议显示，供用户检查改正 |
| 异常事件流 | 1. 事件流中每个步骤抛出的异常都会弹出相应的提示信息  3a. 如果用查询实体信息或者往界面填充实体信息失败时产生相应提示 |
| 后置条件 | 更新界面中的实关系列表 |

### 删除关系信息用例

表3-7 删除关系信息用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 删除关系信息 |
| 用例 | CGS-GXGL-3 |
| 角色 | 设计人员、需求人员 |
| 用例说明 | 设计人员、需求人员删除关系信息 |
| 前置条件 | 系统已经打开一个工程，而且要进行修改的关系信息存在且有效 |
| 基本事件流 | 1. 用户通过主界面左侧的关系列表选择要删除的关系信息  2. 弹出一个询问框，向用户确认是否要删除这个关系  3. 从工程数据中删除这个关系信息 |
| 可选事件流 | 2a. 如果用户不确认要对关系进行删除，则不会对工程数据进行变更 |
| 异常事件流 | 3a. 如果在删除关系数据时发生错误，则以提示的方式提醒用户 |
| 后置条件 | 更新用户界面的关系列表 |

### 代码生成用例

表3-8 代码生成用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 代码生成用例 |
| 用例 | CGS-DMSC-1 |
| 角色 | 设计人员、需求人员、开发人员 |
| 用例说明 | 设计人员、需求人员、开发人员根据已经输入的实体关系数据生成目标系统 |
| 前置条件 | 系统已经打开一个工程 |
| 基本事件流 | 1. 用户通过主界面的代码生成的导航链接进入代码生成页面  2. 如果代码生成页面已经打开，就打开这个页面。否则创建生成代码的页面并打开。  3. 用户在代码生成页面中选择输出路径  4. 用户点击代码生成按钮 |
| 可选事件流 | 无 |
| 异常事件流 | 1. 事件流中每个步骤发生错误都会以消息提示的方式提醒用户  4a. 如果用户指定的输出路径无效，则提出相应的提示信息  4b. 如果代码生成过程中发生文件操作异常，则提示相应的提示信息 |
| 后置条件 | 无 |

### 代码生成配置用例

表3-9 代码生成配置用例规约

|  |  |
| --- | --- |
| 项目名称 | 内容 |
| 用例名称 | 代码生成配置用例 |
| 用例 | CGS-DMSC-2 |
| 角色 | 开发人员 |
| 用例说明 | 开发人员代码生成配置 |
| 前置条件 | 系统已经打开一个工程 |
| 基本事件流 | 1. 用户通过主界面的导航打开代码生成配置的界面  2. 用户可以在代码生成配置的界面对代码生成的风格进行配置  3. 用户在代码生成配置页面点击保存 |
| 可选事件流 | 无 |
| 异常事件流 | 1. 事件流中所有步骤抛出异常都会弹出相应的提示信息 |
| 后置条件 | 无 |

## 非功能性需求

### 代码生成工具扩展性

软件开发技术发展速度较快，软件开发需要不断跟随技术的发展而发展。当前代码生成工具所生成的系统是基于Spring Boot、Hibernate等开源技术。这些开源的技术本身在不断地发展，所以代码生成工具也需要不断进行扩展以跟随技术发展的脚步，这一点在Jack Herrington在《Code Generation in Action》一书中也提及过，所有的代码生成工具都需要不断地维护去迎合新技术[1]。大量新技术出现就意味着代码生成工具需要很好的扩展性和可维护性，良好的扩展性和维护性能够使得代码生成工具持续发挥其作用，降低代码生成工具本身的维护成本。

### 生成代码可读性

软件的生命周期包括编码和维护过程，软件代码的质量和可读性是维护成本一个较大的影响因素。本代码生成工具的定位是生成软件的基本骨架，开发人员在所生成的代码之上进行开发， 所以生成代码的可读性存在一定的要求。本代码生成工具在生成代码的过程中会增加相应的注释，以供开发人员对所生成代码的理解。

### 代码生成工具的易用性

代码生成工具的目标人群是软件设计和软件开发人员，但在用户体验方面同样需要考虑。现有的开发工具和开发框架当中都存在这学习曲线过于陡峭的问题。甚至在一些的技术栈，学习曲线已经影响到了开发团队协作和开发效率。本代码生成工具的目的就是为了提高开发效率，如果代码生成工具具备一定的难度，不但不能起到提高开发效率的作用，反而会因为增加学习成本而导致开发效率降低。

易用性在本代码生成工具会通过下面的准则予以体现：

(1) 采用层次明显的界面展示所有的功能，用户无需通过菜单或者工具栏等导航工具寻找所需的功能

(2) 尽可能采用下拉选框代替文本输入。可以降低用户错误输入引起程序错误的行为，降低用户排错的时间和精力

(3) 内置默认属性。在表单上自动填写默认的参数，用户只需要在默认参数不符合需求的情况下才进行输入和调整，减少参数的填写上花费的时间。

(4) 内置常用的实体信息和属性。不少的软件系统都含有类似的实体信息，比如用户实体。这些实体可以一键创建。除了实体，一些常用的属性会在多个不同的实体中出现。比如实体的唯一标识符。这些属性同样可以通过一键创建，减少用户鼠标和键盘的点击次数。

# 代码生成工具设计

## 整体架构设计

代码生成工具采用模块化的设计，用户界面和主要的生成逻辑均为单独的模块。通过与不同的用户界面整合，可以把代码生成工具实现为普通的桌面应用程序、Web应用程序甚至是IDE插件形式。

本代码生成工具采用的不是以模板引擎为基础的代码生成策略，而是基于代码结构进行代码生成。所以，代码生成工具必须同时具备构造目标系统代码结构和根据目标系统代码结构生成的能力。根据这两个目标，把代码生成的逻辑拆分成为了生成目标系统代码结构的逻辑与根据目标系统代码结构生成代码两个主要的逻辑。根据这两个主要功能，把整个代码生成生成的模块继续细分为完成构造目标系统代码结构的数据信息分析模块以及将系统代码结构生成相应代码的模块的代码生成模块。另外，在完成代码的生成以后，可以根据用户的选择，决定是否对生成的代码进行编译打包，编译打包的工作由编译打包模块完成。

三个模块在整个逻辑当中的顺序如图4-1所示。

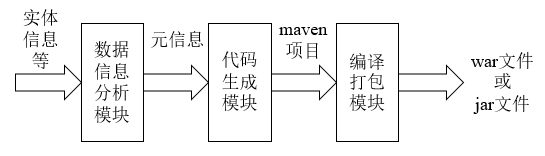


图4-1 系统工作流程

数据信息分析模块直接接收来自于用户界面的数据输入，这些数据包含有工程基本信息、实体类以及实体类之间关联关系的数据，这些数据在数据信息分析模块会被转换成相应的元信息，这些信息包含有Java类信息和工程的配置信息。

代码生成模块接收来自数据信息分析模块的元数据，根据特定的规则基于元信息生成相应的配置文件和代码文件。

编译打包模块则将会把代码生成模块生成的工程通过maven工具将其打包成为一个内嵌有Servlet容器的jar文件或者是用于部署到Servlet容器的war包文件。

系统整体的架构如图4-2所示。

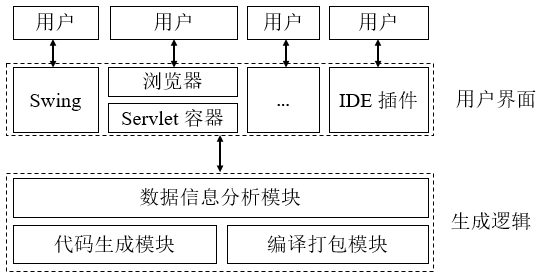


图4-2 系统整体架构示意图

## 数据模型的建立

无论是数据信息分析模块还是代码生成模块，其输入和输出的数据都需要按照一定的结构进行组织。在面向对象的编程语言当中，这些数据的组织通常采用类的形式进行组织，本设计采用Java语言实现，所以本设计同样也以类为单位对数据进行组织。

### 实体信息数据模型

实体信息数据模型主要包含实体信息、实体属性信息以及实体关系等信息。该模型如图4-3所示。

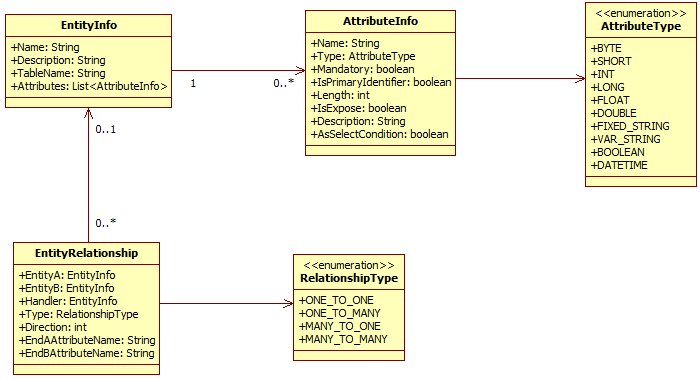


图4-3 实体信息数据模型

实体信息模型主要是根据实体设计过程进行抽象得到的结果。该模型通过实体信息（对应图中EntityInfo）、实体属性（AttributeInfo）和实体关联（EntityRelationship）三个实体以及由属性数据类型（AttributeType）和关系类型（RelationshipType）两个枚举组成。

属性数据类型的枚举主要用于表示实体当中属性的数据类型，关系类型则是主要用于描述关联关系的重数关系。关系类型的描述具有方向性，描述关联类型时关联A端描述在前，关联B端的关系描述在后。属性数据类型枚举成员对应的Java数据类型和MySQL字段类型如表4-1所示，四种关系类型的语义如图4-4所示。

表4-1 属性数据类型枚举成员与Java和MySQL数据类型对应关系

|  |  |  |
| --- | --- | --- |
| 枚举成员 | 对应的Java数据类型 | 对应的MySQL字段类型 |
| BYTE | byte | TINYINT |
| SHORT | short | SMALLINT |
| INT | int | INTEGER |
| LONG | long | BIGINT |
| FLOAT | float | FLOAT |
| DOUBLE | double | DOUBLE |
| FIXED\_STRING | String | CHAR |
| VAR\_STRING | String | VARCHAR |
| DATETIME | Date | DATETIME |

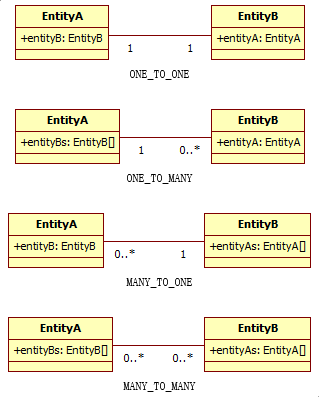


图4-4 关联关系枚举成员对应的语义示意图

实体信息类代表系统当中的一个实体信息，实体信息中含有实体名称、属性等信息。实体关联当中含有三个实体信息类实例。图中EntityRelationship类中的EntityA和EntityB字段分别表示关联关系相关的两个实体类，可以称之为关联关系A端和关联关系B端。Handler字段则表示关联关系的维护实体。关联关系类当中含有一个关系类型的枚举字段，这个字段的作用是指定关联关系的类型。

### 元数据信息数据模型

元数据信息模型主要用于组织类的元数据。这些元数据主要是用于描述类、类当中字段、方法、注解等等Java元素的数据。元数据信息模型是根据Java语言的语法要素抽象得到的，元数据信息模型当中有10个与Java语法相关实体信息，这10个的实体分别是类信息（ClassInfo）、访问修饰符信息（ModifierInfo）、字段信息（FieldInfo）、方法信息（MethodInfo）、注解信息（AnnotationInfo）、类型信息（TypeInfo）、注解参数信息（AnnotationParameter）、参数信息（Parameter）指令信息（Instruction）以及值表达式信息（ValueExpression）。这10个实体类的关系如图4-5所示。

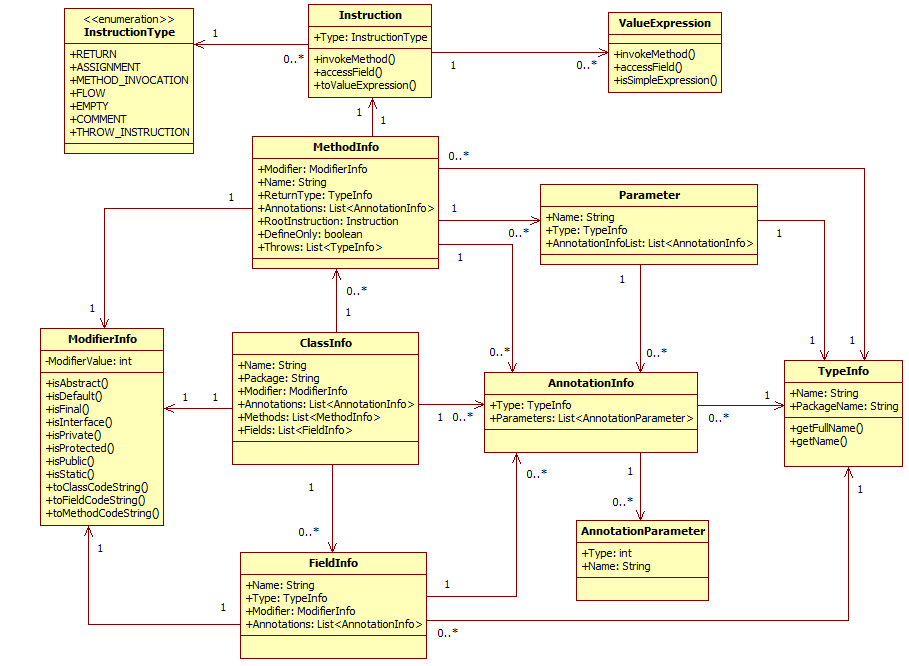


图4-5 元数据信息模型

在Java语言当中，所有的方法、字段等信息不能脱离类而存在，所以在抽象得到的元数据模型中，ClassInfo类位于模型的中间位置，方法信息和字段信息类实例在类信息当中以字段的形式存在，并以此反映了方法、类字段和类之间的关系。位于左右两边外部的访问修饰符信息和类型信息会被多个不同的实体所使用。具体体现在Java当中类、字段和方法均有相关的访问修饰符。

注解是JDK1.5版本以后引入的用于在代码当中添加额外信息的特殊标记。注解信息使用在类、字段、方法以及方法的参数之中，所以模型当中，注解信息的位置同样是相对中心的位置。注解同样有类型和参数信息，所以注解信息对注解参数以及类型信息有着一定的关联关系。

在模型顶端的Instruction类和ValueExpression类分别表示一条Java指令以及一个就有表达值功能的表达式。在本工程当中，指令有着较为广泛的含义，一方面可以指一条具体的操作指令，也可以指通过某种流程控制结构组织起来的多条程序指令。指令具体的实现则通过对Instruction类的继承，正因为如此，在方法的指令属性上面不需要采用Instruction的集合类来存放多个的指令实例。ValueExpression与Instruction类似，其作用仅仅用于表示一个值，这个值可以应用在很多地方，比如在函数调用时用作传递的实参，用作返回值等等。正如Java支持链式的方法调用，值表达式对象同样支持在表达式的基础上调用方法以及访问字段等操作，操作的返回值仍然是一个值对象。

在实际的代码生成过程当中，上面模型当中所有的实体会根据Java的语法规则组织成树状的结构，接近Java程序当中的语法元素的组织。

## 类元数据生成逻辑的设计

实体信息到类元数据转换过程，输入数据是工程信息（包含有工程的开发者信息、工程名称、版本、项目包名等信息）以及这个工程当中的实体信息和实体关系信息，输出的是相关的类元数据信息。数据信息分析模块的主要任务就是完成实体信息到元数据信息转换的逻辑，所产生的元数据会被用于代码生成模块生成相应的实现代码。

从实体信息到类元数据转换的逻辑会根据类的类型进行拆分。本代码生成工具所生成的类根据其作用可以分成十类，按照生成步骤的分别是Spring Boot的主类、Swagger UI的配置类、自定义异常类、所有组件都会依赖的公用类、实体类、DAO接口、Service类、Controller类、HATEOAS资源类、HATEOAS资源装配器。但是，Controller类与HATEOAS资源装配类之间存在相互依赖的关系，所以在生成相应的Controller和HATEOAS资源装配类时没有创建相关的字段，需要两个类均生成完毕以后添加彼此之间的依赖关系。所以从实体信息到整个系统的类信息转换一共被拆分成了十一个步骤，这十一个步骤当中所使用的逻辑有八种。

### 模式固定的类元数据生成逻辑设计

主流的开源框架当中，搭建开发环境所必须的配置和代码是高度接近甚至是一样的。在本代码生成工具所生成的目标系统当中同样存在着类似的大量重复代码，Spring Boot的主类、Swagger UI的配置类、自定义异常类、所有组件依赖的公用类均属于这类代码。这一类的代码的生成逻辑较为简单而且固定，这一类代码的生成逻辑流程图如4-6所示。



图4-6 模式固定的类信息的生成逻辑

### 实体类元数据生成逻辑设计

实体类是本系统当中用于储存和组织数据的类。实体类会被系统中的所有层次的组件使用，所以实体类的设计可以认为是基础性的类。不能随便进行更改，而且这个类当中含有用于ORM的配置信息，所以实体类的生成在整个系统代码的生成过程中属于基础性逻辑，生成逻辑必须确保能够恰当处理。

相比内容较为固定的类元数据的转换，实体类元数据的转换较为复杂。用户输入的实体信息没有固定格式和规范，只要用户所输入的信息不违反强制性的约束条件，实体类转换逻辑都必须对实体信息做出最合适的处理。

实体信息到实体类元数据的转换有三个要点。第一是实体信息当中的属性名称和属性类型恰当地转换成相应的字段名称和字段类型，对于一些与编程语言保留字冲突的属性名称，转换逻辑通过添加一定的前缀后缀避免与语言的关键字冲突；第二是正确处理实体之间关联关系相应字段。实体之间的关联关系粗略可以分为一对一、一对多、多对一和多对多四种。转换逻辑需要正确创建关联关系对应字段的字段类型，创建的是对方实体类的实例或者是对方实体类的实例的集合；第三是恰当地根据实体信息以及实体关联信息创建用于ORM框架的注解信息。这些信息直接控制ORM框架执行数据库操作时的实际行为，是一项十分重要的数据。这些数据的处理难点是处理关联关系维护方的配置，这项信息配置不当会引起数据级联操作数据出错，是一项十分重要的数据。

根据代码转换过程流程，可以将实体信息到类元数据转换的逻辑拆分为三个步骤。第一步骤是生成实体类基本信息以及实体本身的属性对应的字段信息，忽略实体关系相关的字段。第二个步骤是在第一个步骤所生成的实体元数据基础上，通过解析关系信息，在实体类元数据当中添加相应的字段信息。第三个步骤是在前两个步骤生成的字段基础之上，生成相应的get/set方法，使得生成的实体类能满足JavaBean的规范。

第一个步骤的逻辑可以分成两部分，第一部分是根据实体的信息产生类元数据的基本信息，这些信息包括类名、包名的数据；第二部分的逻辑是根据实体的属性信息创建出相应的字段并添加到对应的实体类当中。数据信息分析模块会迭代工程当中所有的实体信息，将每个实体信息都作为逻辑的输入输出，进而产生与之相对应的实体类元数据。实体类的元数据生成整体逻辑如图4-7所示，非关联关系字段元数据的生成逻辑如图4-8所示。

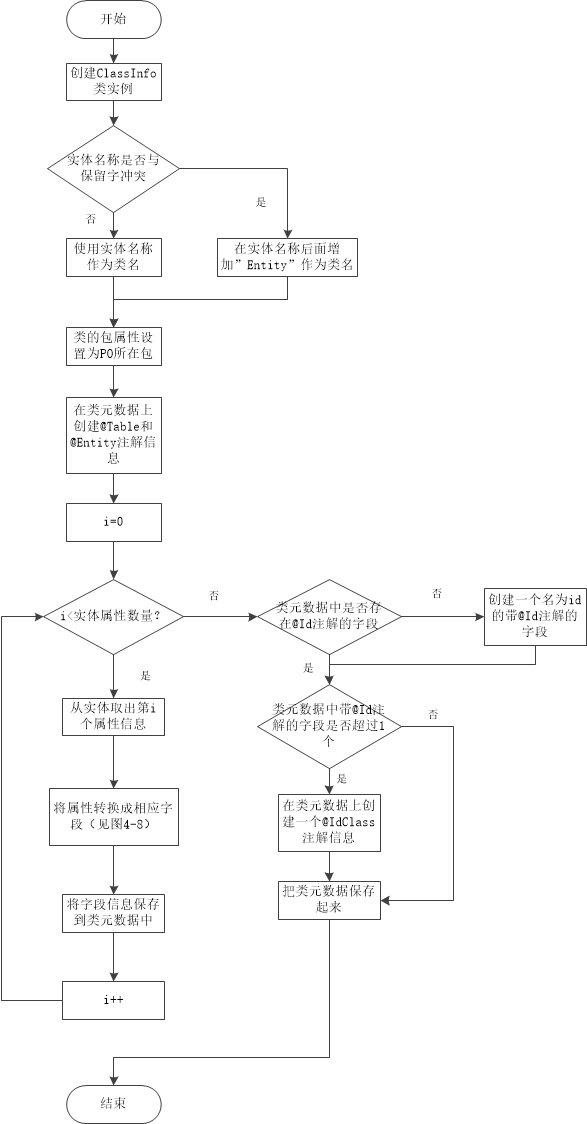


图4-7 实体类元数据生成的整体逻辑

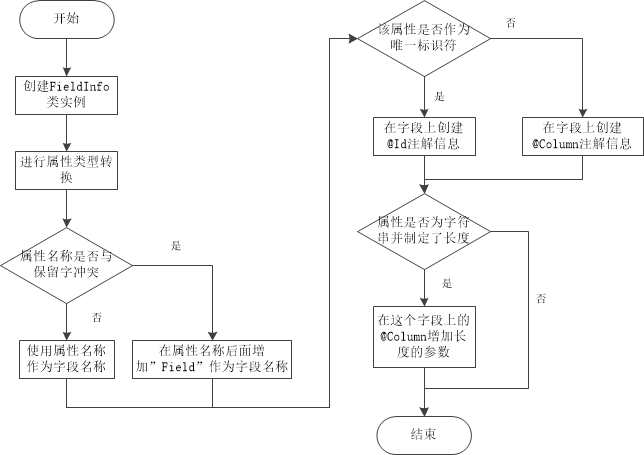


图4-8 非关联属性字段的生成逻辑

第一个步骤完成以后，数据转换模块开始执行第二个步骤。与第一个步骤不同的是，第二个步骤是使用实体关联关系作为输入数据，模块工作时会迭代工程当中所有的关联关系数据，将每一个关联关系都作为输入数据，相关的逻辑会根据输入数据产生相应的字段信息，并且会把这些字段添加到相应的实体类当中。与实体属性不同，一个实体间的关联关系有在关联关系相关的两个类实体中产生两个不同的字段，所以处理同一个关联关系信息时，需要根据所产生字段最终被添加到的类执行两种不同的业务逻辑。所以第二个步骤当中共涉及了三个不同的逻辑。

处理关联字段转换的整体逻辑如4-9所示，产生用于关系A端实体类元数据的字段逻辑与产生用于关系B端实体类元数据的字段逻辑存在不同的地方，图4-10和图4-11分别描述产生用于关系A端实体类元数据的字段逻辑和B端实体类元数据的字段逻辑。

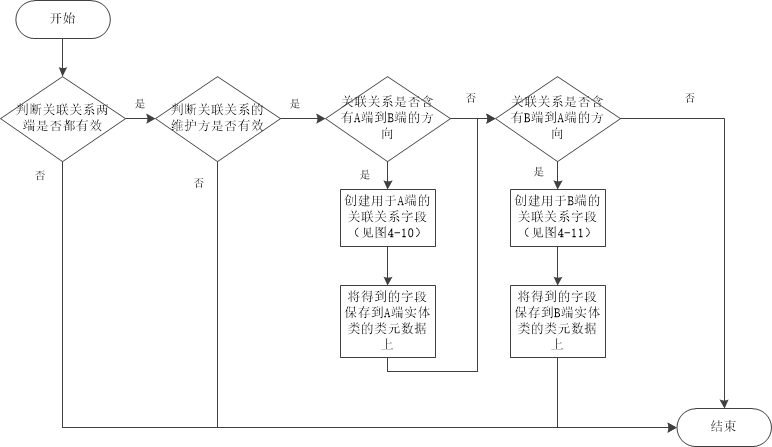


图4-9 关联关系字段转换的整体逻辑

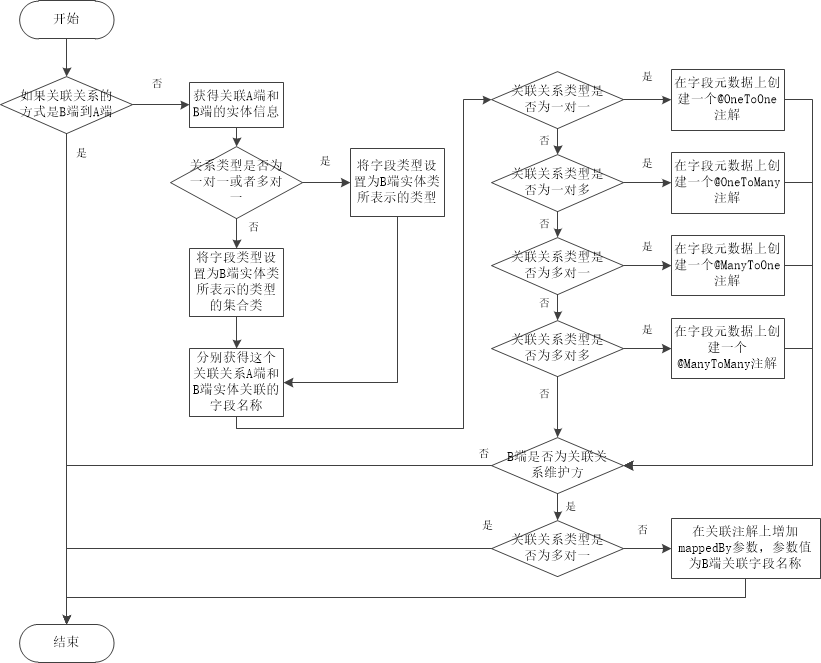


图4-10 用于A端的关联字段的生成逻辑

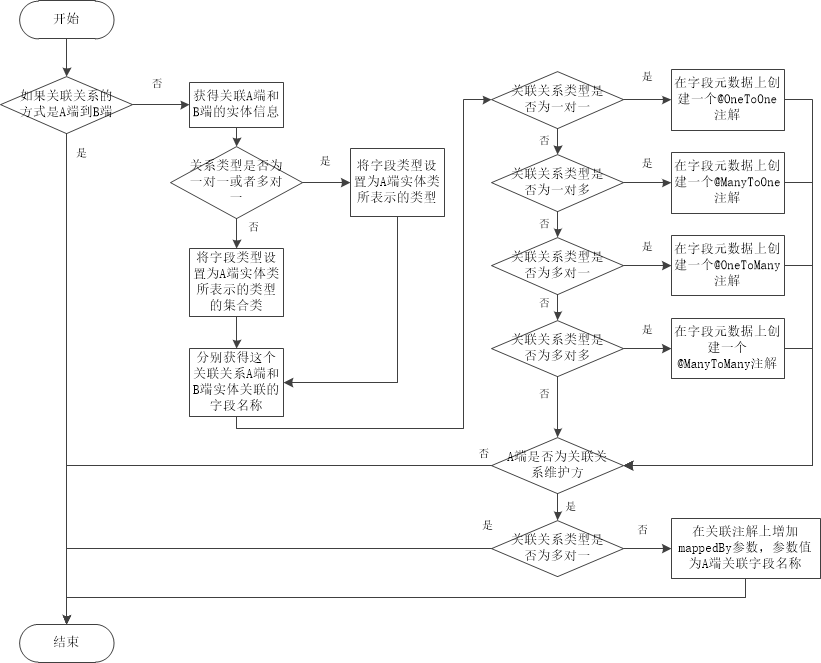


图4-11 用于B端的关联字段的生成逻辑

在完成第二个步骤以后，工程当中所有已经定义的关联关系都会被转换成相应的关联字段。第三个步骤的内容就是在前两个步骤生成字段的基础上产生相应的get/set方法。产生get/set方法的逻辑如图4-12所示。

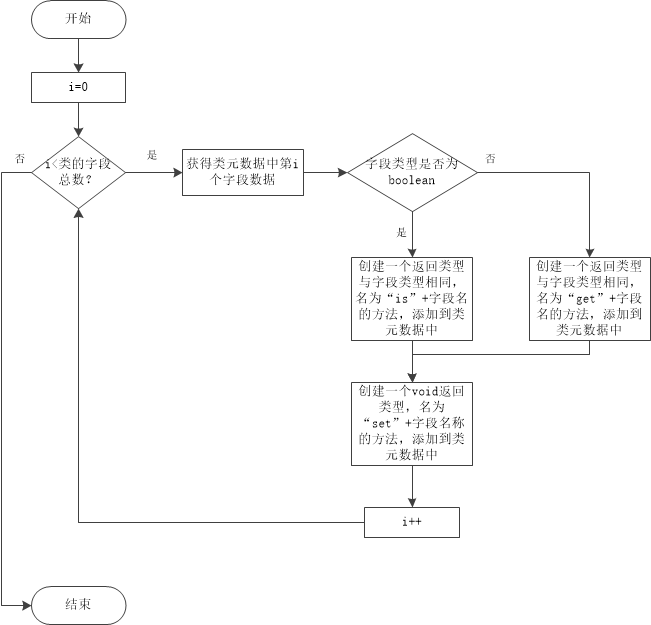


图4-12 实体类get/set方法的生成逻辑

三个步骤完成以后，该工程项目的实体类生成完成，可以继续执行后续的生成逻辑。

### DAO接口元数据生成逻辑设计

DAO在目标系统中的职责是进行数据库的访问操作。本代码生成工具生成的系统是基于Spring Data JPA，无需生成操作数据库的基本实现，所需要的仅仅是生成一个用于定义操作的接口即可。Spring Data JPA已经将常用的增加、删除、修改和查询进行了封装，DAO接口生成逻辑需要生成的仅仅是自定义查询条件以及关联关系查询两种的查询操作。

DAO接口元数据的生成可以进一步分解为三个步骤，分别是生成DAO接口的基本信息（基本信息包括接口名称和基类信息）、生成自定义查询条件和关联关系查询三个步骤。三个步骤的逻辑分别如图4-13、图4-14和图4-15所示。

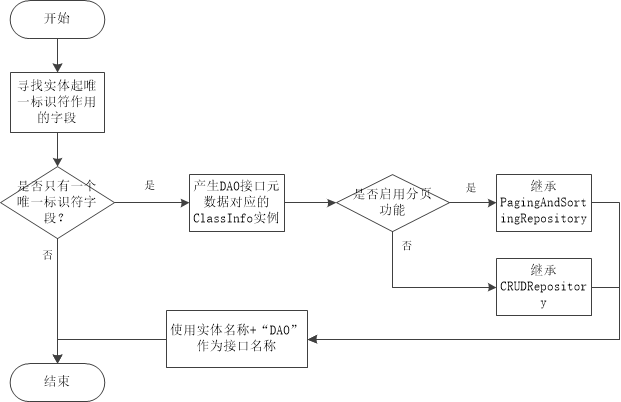


图4-13 DAO接口基本信息生成逻辑

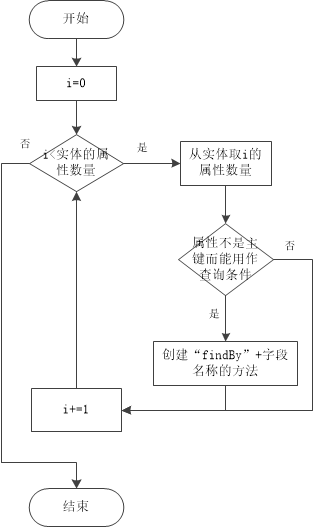


图4-14 自定义查询条件方法生成逻辑

### Service类元数据生成逻辑设计

### Controller类元数据生成逻辑设计

### HATEOAS资源类元数据生成逻辑设计

### HATEOAS资源装配类元数据生成逻辑设计

### 解决Controller和HATEOAS资源装配类之间相互依赖逻辑设计

## 代码生成逻辑的设计

### 代码生成的策略

本代码生成的策略采用问题拆解的策略。在最终的代码生成的逻辑执行之前，类元数据相关的信息的根据Java的语法规则组织成一个树状的结构（下称元数据树）。图展示用于表示常见的实体类结构的元数据树。

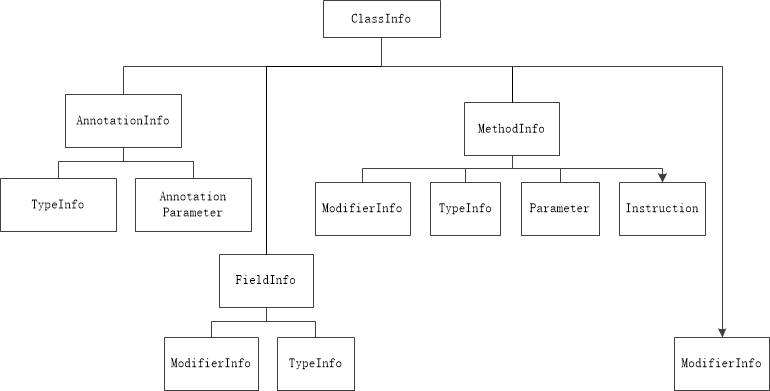


图 元数据树的结构

Java是一种以类为基础的语言，所以整个项目代码都是以类为基础的。

本代码生成工具会将一整个项目的代码生成拆解成多个类代码文件的生成；一个类的代码生成拆解成注解生成、修饰符生成、字段生成和方法生成四种不同语法元素的生成。所有的语法元素生成以后，将生成得到的代码段按照一定的规则整合成符合语法规则的代码。在代码实现过程中，元数据树上的每一个结点类型都通过复写toString方法以实现将自身表示的语法元素转换成相应的语句。

## 用户界面的设计

本代码生成工具的定位是软件开发的辅助工具，采用传统的桌面应用程序较Web应用程序更为合适。为了降低用户界面的复杂程度，用户界面采用Java的Swing进行用户界面开发。

### 主界面布局的设计

本代码生成工具通过菜单栏和工具栏对功能进行组织。与大部分的桌面软件类似，主界面当中除去菜单栏、工具栏和状态栏的工作区会被划分成左右两部分。左边窗格放置一个以树状结构展示当前工程信息结构的控件，用户能够直观地了解到当前工程当中的信息结构，右边窗格是用户的主要工作区域，工作区域则会根据用户打开的信息类型而不一致。工作区域是程序接收用户输入的主要位置。主界面的布局如图所示



图 主界面布局原型设计图

### 基本信息工作页面设计

基本信息工作页面主要用于修改工程基本信息。工程的基本信息包含程序的开发者名称、项目名称、版本、包名、数据库信息等基本信息。除了数据库类型需要通过下拉选单进行选择以外，其他的信息都需要通过文本框进行输入。工程基本信息的工作页面设计如图所示



图 工程基本信息工作页面原型设计图

### 实体信息工作页面设计

实体信息编辑界面包括实体名称、实体描述和属性等信息的编辑。其中，实体名称和实体描述两项的信息采用文本框的方式接受用户的输入，属性采用的表格的形式进行呈现。属性名称、属性描述两项的内容采用文本框的形式接受用户的输入，而属性的类型则采用下拉选框的方式供用户选择，其他布尔类型的属性数据则采用复选框的方式呈现和接受用户数据。实体信息编辑页面如图所示。



图 实体基本信息工作页面原型设计图

### 关系信息工作页面设计

关联关系编辑界面能够选择关系两端的实体，同时含有关系名称、关系类型以及关系相关字段的配置。关系名称和字段名称采用文本框的接受用户输入。用于确定关联关系两端实体采用下拉选框进行选择，这样可以确保用户不会因为输出错误而导致的配置错误。四种关联关系的选择采用单选框的形式。关联关系信息编辑界面如图所示。

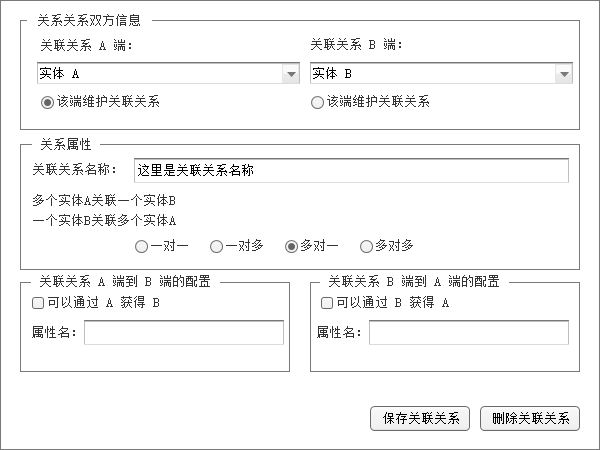


图 关联关系信息工作页面界面原型设计图

### 代码生成工作页面设计

代码生成的页面仅仅需要用户选择好代码的输出目录，用户即可点击生成按钮。代码生成工作页面的设计相比起其他的工作页面简单。如图所示

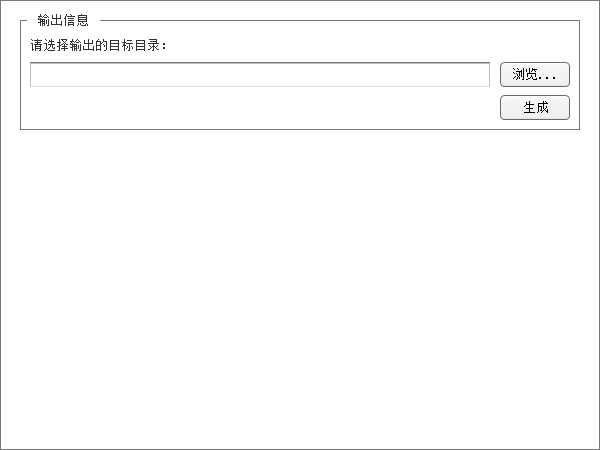


图 代码生成工作页面界面原型设计图

# 实现效果

为了演示本代码生成工具在软件开发中的使用流程，下面以一个手机通讯录备份软件的后台系统作为例子进行本代码生成工具的实现效果展示。

## 目标系统的需求描述

通讯录助手软件是一个用于手机通讯录备份和转移的手机软件。该软件的主要作用是为了避免手机在意外情况下的数据丢失以及更换手机过程当中通讯录数据的迁移。这个软件整个系统可以分为手机APP，服务器程序和数据库三部分。其中，手机APP和数据库两者与代码生成工具的效果演示没有关系，所以不在本文的讨论范围以内。本文只探讨服务器程序的实现。

服务器程序实现了数据保存、组织功能；通过Web API以Web Service的形式对外暴露一系列的接口，供手机APP完成通讯录备份和转移的功能。

## 目标系统的功能描述

手机通讯录的服务器程序通过用户、备份、联系人记录和联系人数据记录四个概念对手机通讯录数据进行组织和处理。服务器程序的功能需要根据整个系统的功能进行设计，整个系统的用例模型如图5-1所示。

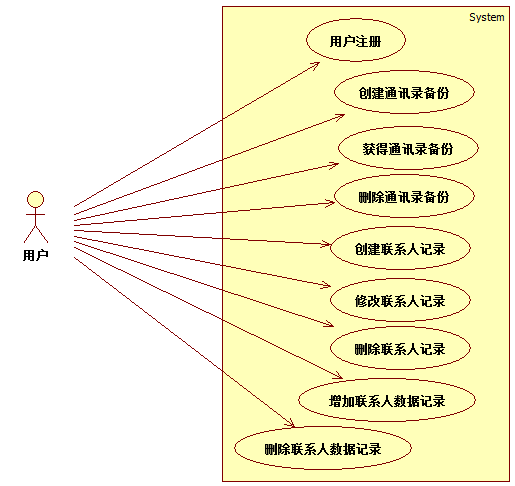


图5-1 目标系统的用例模型

## 目标系统的设计

根据系统的用例模型，可以完成系统数据模型的建立和用作接口的URI的设计。目标系统的数据模型如图5-2所示。

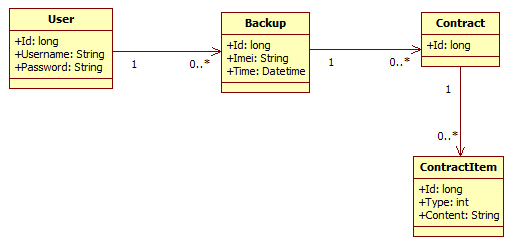


图5-2 目标系统的数据模型

数据模型中四个类分别表示用户（User）、备份（Backup）、联系人记录（Contract）以及联系人数据记录（ContractItem）。因为目标系统是采用REST架构风格的Web API实现，所以URI的设计极为重要，根据图5-2中的数据模型设计的URI的表5-1所示

表5-1 目标系统的URI设计

|  |  |  |
| --- | --- | --- |
| URI | HTTP方法 | 对应的业务逻辑 |
| /users | POST | 用户注册 |
| /users | GET | 用户登录 |
| /users/{id} | GET | 查询特定编号的用户信息 |
| /users/{id}/backups | GET | 获得这个用户的所有通讯录备份信息 |
| /users/{id}/backups | POST | 为这个用户创建一个通讯录备份信息 |
| /backups/{id} | DELETE | 删除给定的通讯录备份信息 |
| /backups/{id} | GET | 获得通讯录备份的详细信息 |
| /backups/{id}/contracts | POST | 在备份当中创建一个联系人记录 |
| /backups/{id}/contracts | GET | 获得这个备份当中的联系人记录列表 |
| /contracts/{id} | DELETE | 删除这个联系人记录 |
| /contracts/{id} | PUT | 修改这个联系人记录 |
| /contracts/{id}/items | POST | 在联系人记录当中创建一条联系人数据记录 |
| /contracts/{id}/items | GET | 获得这个联系人记录当中所有的联系人数据记录 |
| /items/{id} | GET | 获得这个联系人数据记录的详情 |
| /items/{id} | DELETE | 删除这个联系人数据记录 |

在这个系统的URI设计中，将系统中所有的业务操作抽象成对用户、备份、联系人记录和联系人数据记录四种资源的操作。通过上表的接口可以完成系统中所有的操作。

## 目标系统的生成

根据目标数据模型，得到了四个实体信息和三个关系信息。四个实体信息分别用户实体（User）、备份实体（Backup）、联系人实体（Contract）和联系人数据实体（ContractItem）；三个关联关系分别为用户与备份之间的一对多关系（rel\_user\_back）、备份与联系人记录之间的一对多关系（rel\_back\_cont）、联系人记录与联系人数据记录之间的一对多关系（rel\_cont\_item）。四个实体属性信息分别如表5-2，表5-3，表5-4，表5-5所示；三个关系的信息如表5-6所示。

表5-2 用户实体的属性信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 属性说明 | 类型 | 实体唯一标识符 | 必填 | 用作查询条件 |
| id | 用户编号 | long | 是 | 是 | 是 |
| username | 用户名 | varchar | 否 | 是 | 是 |
| password | 密码 | varchar | 否 | 是 | 是 |

表5-3 备份实体的属性信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 属性说明 | 类型 | 实体唯一标识符 | 必填 | 用作查询条件 |
| id | 备份编号 | long | 是 | 是 | 是 |
| imei | 移动设备标识码 | varchar | 否 | 是 | 是 |
| time | 备份时间 | datetime | 否 | 否 | 否 |

表5-4 联系人实体的属性信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 属性说明 | 类型 | 实体唯一标识符 | 必填 | 用作查询条件 |
| id | 联系人记录编号 | long | 是 | 是 | 是 |

表5-5 联系人数据实体的属性信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名称 | 属性说明 | 类型 | 实体唯一标识符 | 必填 | 用作查询条件 |
| id | 联系人数据记录编号 | long | 是 | 是 | 是 |
| type | 联系人数据记录类型 | int | 否 | 是 | 否 |
| content | 联系人数据记录内容 | varchar | 否 | 是 | 否 |

表5-6 目标系统中的关系数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 关系名称 | A端实体 | B端实体 | 维护方 | 类型 | 方向 |
| rel\_user\_back | 用户实体 | 备份实体 | 备份实体 | 一对多 | 双向 |
| rel\_back\_cont | 备份实体 | 联系人实体 | 备份实体 | 一对多 | 一对多单向 |
| rel\_cont\_item | 联系人实体 | 联系人数据实体 | 联系人数据实体 | 一对多 | 一对多单向 |

得出目标系统的实体和关系数据以后，将这些数据通过代码生成工具的界面添加到代码生成工具当中。这些实体信息通过实体编辑页面输入到代码生成系统当中，四个实体的输入界面分别如图5-3，图5-4，图5-5，图5-6所示



图5-3 用户实体的实体信息编辑界面



图5-4 备份实体的实体信息编辑界面



图5-5 联系人实体的实体信息编辑界面



图5-6 联系人数据实体的实体信息编辑界面

除了实体信息以外，关系数据也是项目当中十分重要的数据。本工程的三个关系的编辑页面如图5-7，图5-8，图5-9所示。

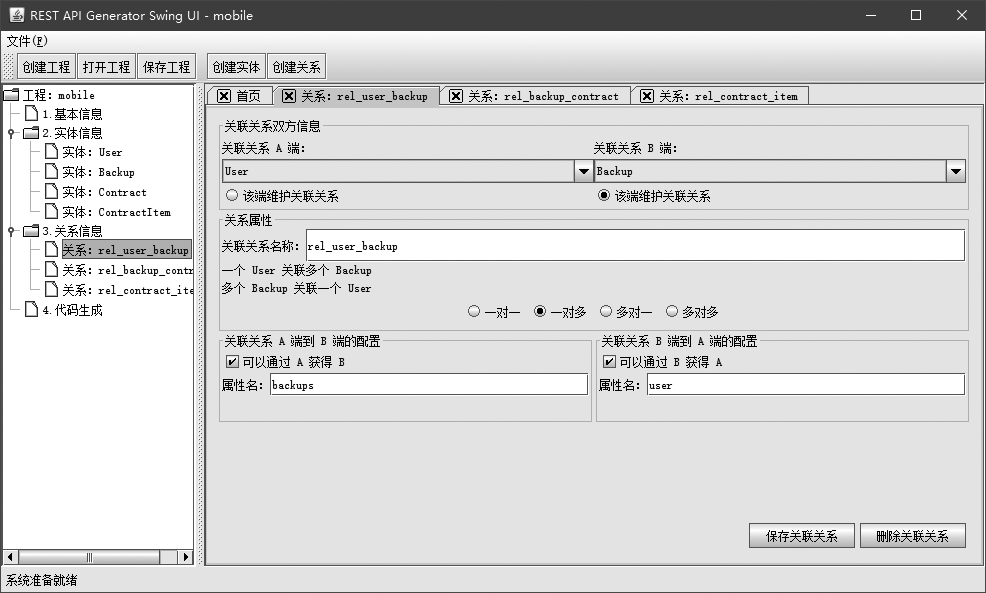


图5-7 用户-备份关系信息编辑界面

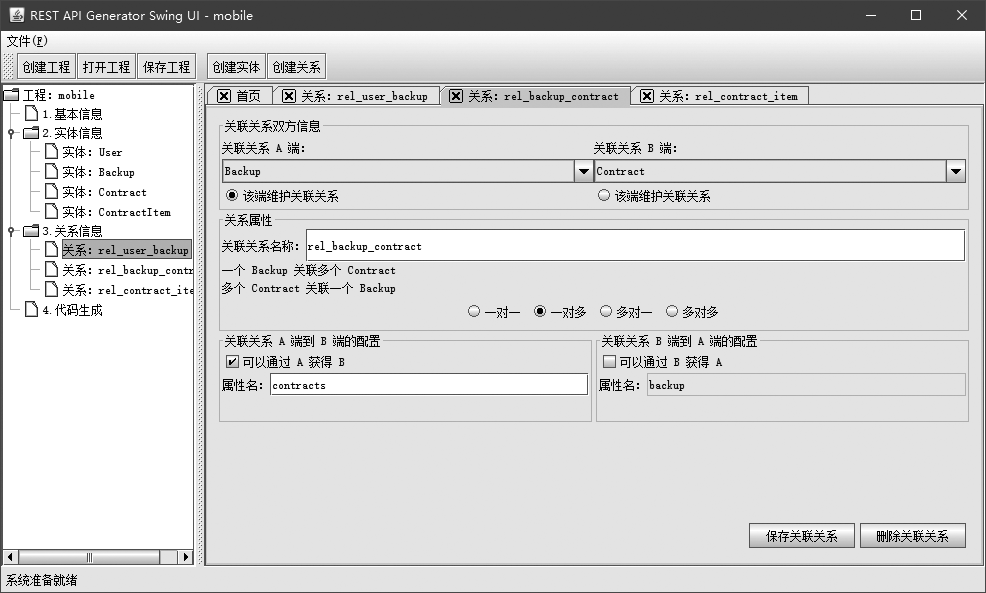


图5-8 备份-联系人关系信息编辑界面



图5-9 联系人-联系人数据关系信息编辑界面

当用户完成所有数据的输入以后，可以在代码生成页面中选择代码的输出路径，点击生成即可完成代码的生成。代码生成完成以后会弹出相应的提示消息，如图5-10所示。



图5-10 代码生成成功的提示界面

## 目标系统的生成结果

用户单击代码生成按钮以后，会在指定的输出目录中生成一个Maven项目，这个项目实现了由用户定义的实体和关系基本的增加、删除、查找和修改操作。图5-11是由代码生成工具所生成的项目的目录结构。

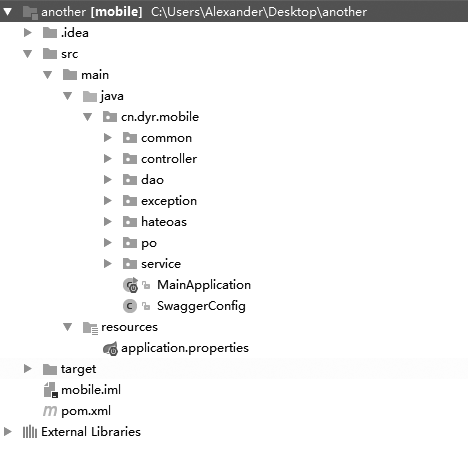


图5-11 代码生成工具生成的工程的目录结构

将代码生成工具所生成的工程导入到IDE中，执行编译并运行。运行以后通过浏览器访问项目Swagger UI的页面。可以看到这个工程当中所有的API接口的列表，如图5-12所示。

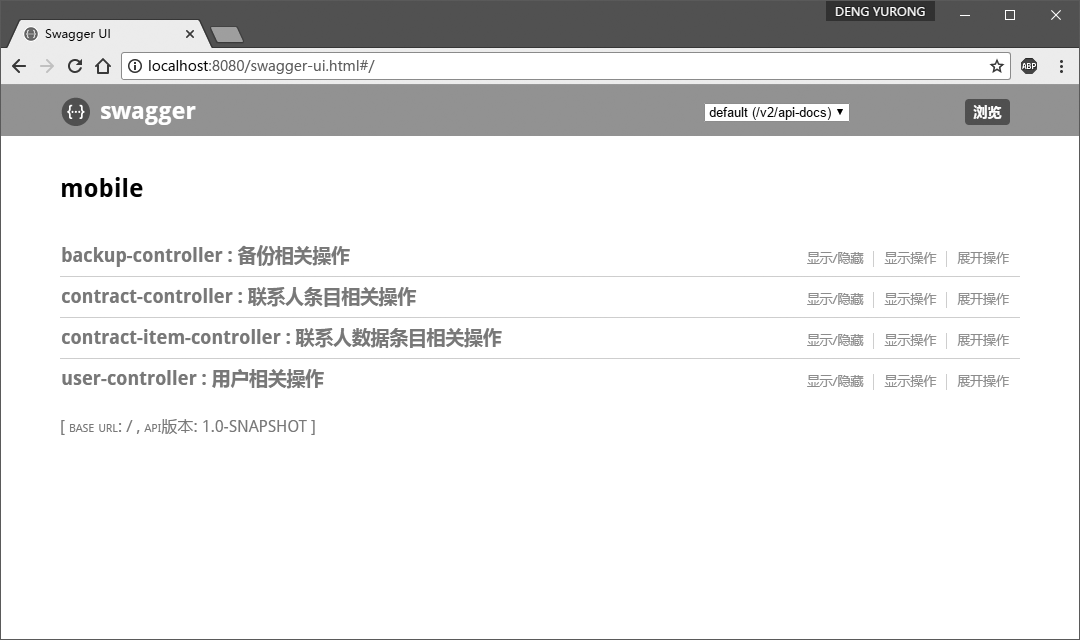


图5-12 生成的目标项目的 Swagger UI 页面

### 普通接口的调用过程

在Swagger UI页面中找到用户注册的接口，可以通过点击操作展开用户注册接口的详细信息。如图5-13所示。

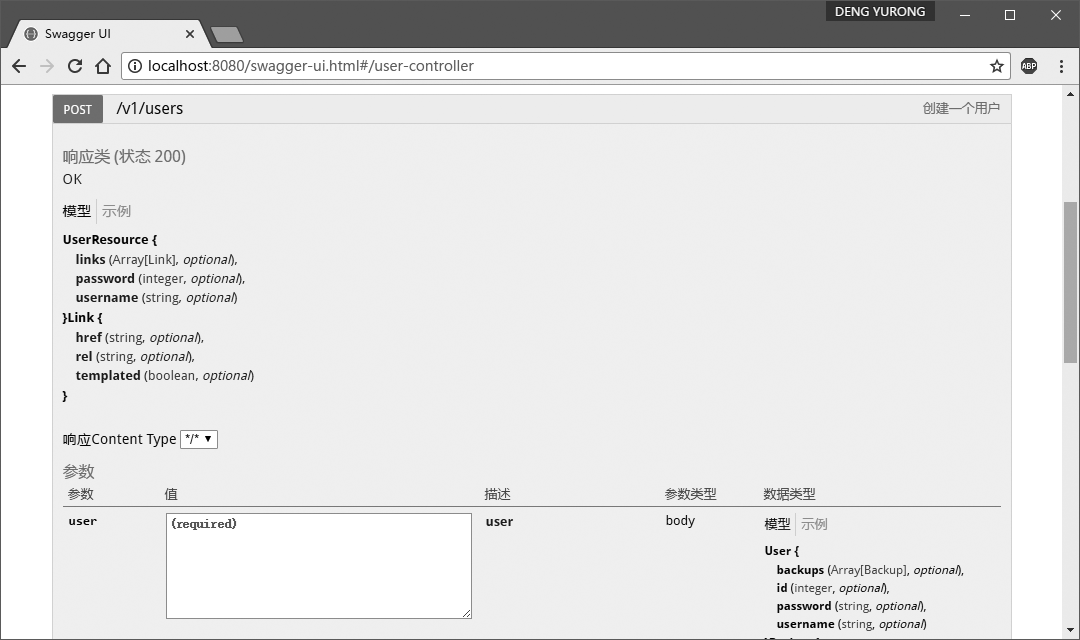


图5-13 用户注册的接口测试页面

用户在打开接口的测试页面后，可以在相应的位置当中输入参数的值，这些值输入完毕以后都会提交给所生成的接口。这里注册一个名为“root”，密码为“123456”的用户，参数以json格式进行组织，参数正确填写在相应的参数框当中，如图5-14所示。

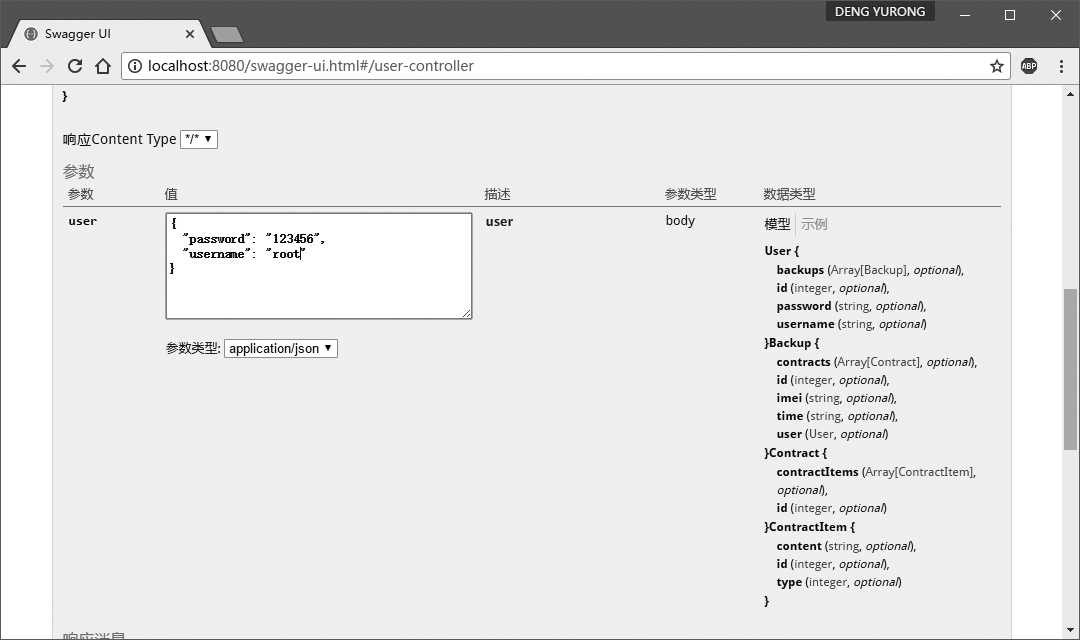


图5-14 输入用户注册接口的测试数据

用户正确输入接口的访问参数以后，点击下面的“试一下！”按钮，即可以特定的参数调用这个接口。在调用接口以后，Swagger UI会将服务器返回的结果以及HTTP响应头等信息呈现到屏幕上，如图5-15所示。

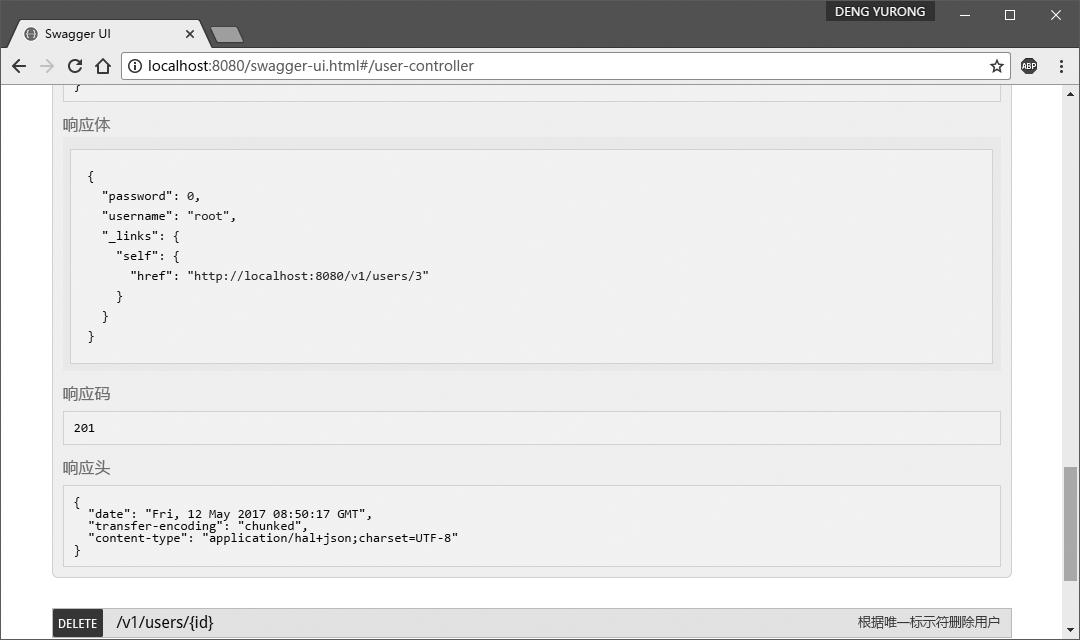


图5-15 Swagger UI显示接口调用返回的结果

这个接口正确调用以后，会根据在代码生成过程配置的属性对参数进行处理，数据经过Controller、Service、DAO，最终到达并写入到数据库，接口调用完成。图5-16是参数经过处理以后被写入到数据库方式

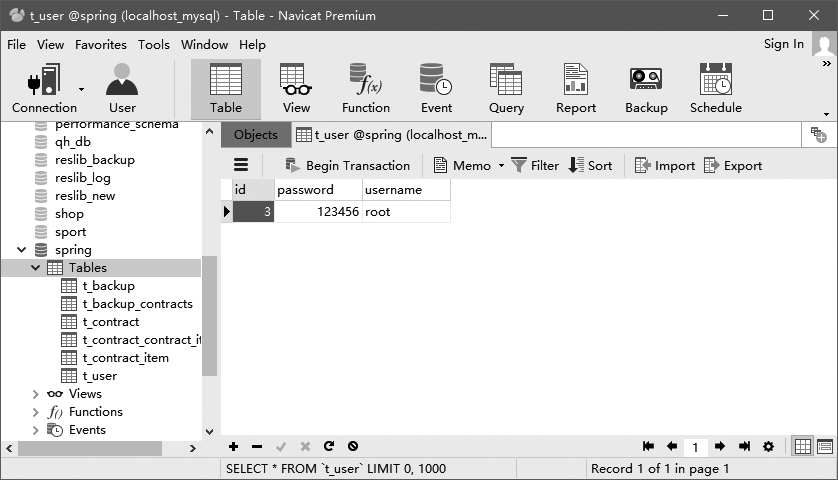


图5-16 用户注册接口调用以后对数据库的变更操作

### 外键约束删除接口调用失败过程

某些业务操作可能会涉及到对象的删除，删除操作有可能会违反外键约束。所以在涉及到实体删除操作的接口，会有与之对应的响应码表示这个错误。如图5-17当中400状态码。

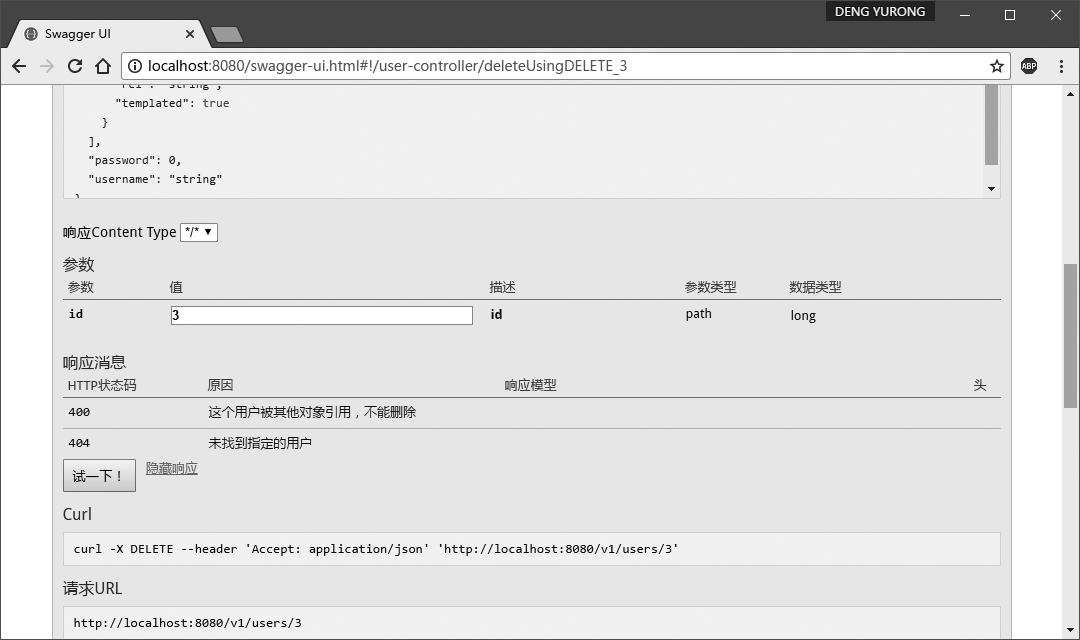


图5-17 有可能引起违反外键约束操作的状态码信息

如果接口在执行的过程中违反了数据库外键约束，则会以400状态码返回，并在响应的正文部分提供相应的错误信息，如图5-18所示。



图5-18 接口调用过程违反数据库外键约束的错误信息

# 鸣 谢

本毕业设计得以顺利完成，首先要感谢导师高升老师在代码生成工具的设计、实现以及毕业设计说明书的撰写过程中给予帮助。

# 参考文献

[1] Jack, Herrington. Code Generation in Action[M]. Greenwich, CT, USA:Manning Publications, 2003. 25-26

附 录

（宋体五号字，行距为固定值20磅）（要求：附录的有无根据说明书（设计）情况而定，内容一般包括正文内不便列出的冗长公式推导、符号说明（含缩写）、计算机程序等。附录中有程序源代码的因篇幅限制可酌情考虑内容的字号。）