(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 106951231 A
(43)申请公布日 2017.07.14

(21)申请号 201710099952.7

(22)申请日 2017.02.23

(71)申请人 微梦创科网络科技(中国)有限公司
地址 100193 北京市海淀区东北旺西路中关村软件园二期(西扩)N-1、N-2地块新浪总部科研楼2层

(72)发明人 罗亮 王崇 任文越

(74)专利代理机构 北京国昊天诚知识产权代理有限公司 11315
代理人 黄熊

(51)Int.Cl.
G06F 9/44(2006.01)

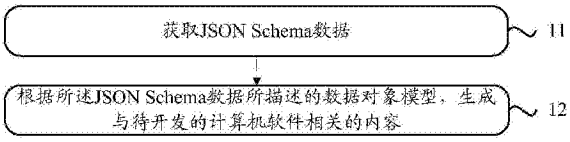
权利要求书3页 说明书14页 附图3页

(54)发明名称

一种计算机软件开发方法及装置

(57)摘要

本申请公开了一种计算机软件开发方法及装置，用以解决现有技术中采用瀑布式开发导致计算机软件开发过程整体效率较低，且容易导致人力资源浪费的问题。该方法包括：获取待开发的计算机软件对应的JSON Schema数据；所述待开发的计算机软件对应的JSON Schema数据，用于描述所述待开发的计算机软件的数据对象模型；根据所述JSON Schema数据所描述的数据对象模型，生成与所述待开发的计算机软件相关的内容；所述内容，包括下述至少一项：基础项目框架；接口文档；测试用例。



CN 106951231 A

1.一种计算机软件开发方法,其特征在于,包括:

获取待开发的计算机软件对应的JSON Schema数据;所述待开发的计算机软件对应的JSON Schema数据,用于描述所述待开发的计算机软件的数据对象模型;

根据所述JSON Schema数据所描述的数据对象模型,生成与所述待开发的计算机软件相关的内容;

所述内容,包括下述至少一项:

基础项目框架;接口文档;测试用例。

2.如权利要求1所述的方法,其特征在于,所述内容包括所述基础项目框架;所述基础项目框架包括第一数据表;

根据所述JSON Schema数据所描述的数据对象模型,生成与所述待开发的计算机软件相关的内容,具体包括:

按照预定转换规则,将所述JSON Schema数据所描述的数据对象模型转换为相应的数据定义语言DDL;

根据转换得到的DDL,生成所述第一数据表;

其中,所述预定转换规则,包括下述规则项中的至少一种:

将获取到的JSON Schema数据中包含的数据对象模型的标题,转换为用于定义第一数据表的表名的DDL;

将获取到的JSON Schema数据中包含的数据对象模型的描述,转换为用于定义第一数据表的备注的DDL;

将获取到的JSON Schema数据中包含的数据对象模型的属性名称,转换为用于定义第一数据表中字段的名称的DDL。

3.如权利要求2所述的方法,其特征在于,所述基础项目框架还包括模型层、服务层和控制层;

根据JSON Schema数据所描述的数据对象模型,生成与所述待开发的计算机软件相关的内容,还包括:

根据所述JSON Schema数据所描述的数据对象模型,按照模型层、服务层和控制层的生成规则,生成模型层、服务层和控制层;其中,模型层、服务层和控制层的生成规则,具体包括:根据所述JSON Schema数据所描述的数据对象模型的标题,分别生成模型层、服务层和控制层的类的名称;根据所述JSON Schema数据所描述的数据对象模型的属性,分别生成模型层、服务层和控制层的类的类属性和类属性的访问方法;

所述模型层,用于提供对所述第一数据表中的数据进行操作的操作函数;

所述服务层,包括业务逻辑代码框架;所述业务逻辑代码框架,用于软件开发人员在所述业务逻辑代码框架下编写用于实现业务逻辑的代码;

所述控制层,包括用于调用所述服务层的业务逻辑的接口的代码。

4.如权利要求3所述的方法,其特征在于,所述对所述第一数据表中的数据进行操作的操作函数,包括对所述第一数据表中的数据进行查询的查询函数;在所述业务逻辑代码框架中设置有业务逻辑的情况下,所述方法还包括:

通过调用所述控制层中包含的用于调用所述服务层的业务逻辑的接口,使得所述服务层被调用的业务逻辑调用所述模型层中的所述查询函数;

获得所述接口返回的、由所述查询函数从所述第一数据表中查询到的目标数据；

将所述目标数据，转换为用于描述与所述目标数据的标识对应的数据对象模型的JSON Schema数据；

根据转换得到的JSON Schema数据，判断所述业务逻辑是否发生变化；

若是，则重新生成与待开发的计算机软件相关的内容；

重新生成的内容包括接口文档和/或测试用例。

5.如权利要求4所述的方法，其特征在于，根据转换得到的JSON Schema数据，判断所述业务逻辑是否发生变化，具体包括：

从存储有JSON Schema数据的第二数据表中，获取目标JSON Schema数据；其中，所述目标JSON Schema数据为：用于描述与所述目标数据的标识相对应的数据对象模型的JSON Schema数据中，最近一次更新的JSON Schema数据；

比较所述目标JSON Schema数据与转换得到的JSON Schema数据是否一致；

若一致，则判定所述业务逻辑没有发生变化；若不一致，则判定所述业务逻辑发生变化；

所述重新生成与待开发的计算机软件相关的内容，包括：

根据所述目标JSON Schema数据所描述的数据对象模型，重新生成与待开发的计算机软件相关的内容。

6.如权利要求4所述的方法，其特征在于，调用所述控制层中包含的用于调用所述服务层的业务逻辑的接口，具体包括：

通过扫描所述控制层，获取所述控制层包含的所述接口的地址；根据所述地址，调用所述接口；或

接收针对所述接口的指令；当确定出所述指令中存在特定参数值时，调用所述接口；其中，所述指令中包含所述特定参数值，表示所述指令用于触发调用所述接口。

7.一种计算机软件开发装置，其特征在于，包括：

JSON Schema数据获取模块，用于获取待开发的计算机软件对应的JSON Schema数据；所述待开发的计算机软件对应的JSON Schema数据，用于描述所述待开发的计算机软件的数据对象模型；

第一生成模块，用于根据所述JSON Schema数据所描述的数据对象模型，生成与所述待开发的计算机软件相关的内容；

所述内容，包括下述至少一项：

基础项目框架；接口文档；测试用例。

8.如权利要求7所述的装置，其特征在于，所述内容包括所述基础项目框架；所述基础项目框架包括第一数据表；

所述第一生成模块包括：

转换单元，用于按照预定转换规则，将所述JSON Schema数据所描述的数据对象模型转换为相应的数据定义语言DDL；

第一生成单元，用于根据转换得到的DDL，生成所述第一数据表；

其中，所述预定转换规则，包括下述规则项中的至少一种：

将获取到的JSON Schema数据中包含的数据对象模型的标题，转换为用于定义第一数

据表的表名的DDL；

将获取到的JSON Schema数据中包含的数据对象模型的描述，转换为用于定义第一数据表的备注的DDL；

将获取到的JSON Schema数据中包含的数据对象模型的属性名称，转换为用于定义第一数据表中字段的名称的DDL。

9.如权利要求8所述的装置，其特征在于，所述基础项目框架还包括模型层、服务层和控制层；

所述第一生成模块还包括：

第二生成单元，用于根据所述JSON Schema数据所描述的数据对象模型，按照模型层、服务层和控制层的生成规则，生成模型层、服务层和控制层；其中，模型层、服务层和控制层的生成规则，具体包括：根据所述JSON Schema数据所描述的数据对象模型的标题，分别生成模型层、服务层和控制层的类的名称；根据所述JSON Schema数据所描述的数据对象模型的属性，分别生成模型层、服务层和控制层的类的类属性和类属性的访问方法；

其中，所述模型层，用于提供对所述第一数据表中的数据进行操作的操作函数；

所述服务层，包括业务逻辑代码框架；所述业务逻辑代码框架，用于软件开发人员在所述业务逻辑代码框架下编写用于实现业务逻辑的代码；

所述控制层，包括用于调用所述服务层的业务逻辑的接口的代码。

10.如权利要求9所述的装置，其特征在于，所述对所述第一数据表中的数据进行操作的操作函数，包括对所述第一数据表中的数据进行查询的查询函数；所述装置还包括：

调用模块，用于在所述业务逻辑代码框架中设置有业务逻辑的情况下，通过调用所述控制层中包含的用于调用所述服务层的业务逻辑的接口，使得所述服务层被调用的业务逻辑调用所述模型层对象中的所述查询函数；

目标数据获取模块，用于获得所述接口返回的、由所述查询函数从所述第一数据表中查询到的目标数据；

转换模块，用于将所述目标数据转换为用于描述与所述目标数据的标识对应的数据对象模型的JSON Schema数据；

判断模块，用于根据转换得到的JSON Schema数据，判断所述业务逻辑是否发生变化；

第二生成模块，用于若判断所述业务逻辑发生变化，则重新生成与待开发的计算机软件相关的内容；

重新生成的内容包括接口文档和/或测试用例。

一种计算机软件开发方法及装置

技术领域

[0001] 本申请涉及计算机技术领域，尤其涉及一种计算机软件开发方法及装置。

背景技术

[0002] 在计算机软件开发中，瀑布式开发，是最典型的预见性的开发方法。瀑布式开发大体分为这几个阶段：需求分析、设计、编码和测试。开发过程严格按这几个阶段依次进行，每个阶段都强依赖于前一个阶段。针对编码阶段而言，在该阶段中，软件开发人员会采用编写代码的方式，产出基础项目框架、接口文档和测试用例。

[0003] 其中，基础项目框架，是指开发出的计算机软件最为基础的框架，用于实现所述计算机软件最为基础的功能，如实现最基本的诸如增删改查(即CURD，是Create、Update、Read和Delete的简称)的数据操作中的至少一种操作。在一些具体的应用场景中，所述的基础项目框架，可能包括基础代码和用于保存数据的数据表。

[0004] 接口文档，是用于记载开发出的计算机软件中包含的接口的相关信息的文档。该接口文档由软件开发人员编写，记载的内容往往包括接口的名称、描述、请求地址、请求参数、返回数据、请求统一资源定位符(Uniform Resource Locator，URL)实例和返回数据实例中的多项或所有项。

[0005] 测试用例，即Test Case，是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果。

[0006] 就瀑布式开发而言，虽然它多年来一直是大型的复杂项目开发的标准，但它有两个明显的缺陷：

[0007] 1.针对待开发的计算机软件的开发需求往往会随着时间而变化，在新的开发需求下，软件开发人员先前已经产出的一些内容(如基础项目框架、接口文档或测试用例)可能会被作废，需要软件开发人员重新产出该些内容，影响计算机软件开发过程整体效率，且导致大量人力资源的浪费。

[0008] 2.在瀑布式开发中，测试是最后一个阶段，该阶段要依赖于之前各个阶段的工作均已完成。然而，等待测试之前的各个阶段的工作完成，需要一段较长的时间，等待这段时间导致计算机软件开发过程整体效率较低。

发明内容

[0009] 本申请实施例提供一种计算机软件开发方法，用以解决现有技术中采用瀑布式开发导致计算机软件开发过程整体效率较低，且容易导致人力资源浪费的问题。

[0010] 本申请实施例还提供一种计算机软件开发装置，用以解决现有技术中采用瀑布式开发导致计算机软件开发过程整体效率较低，且容易导致人力资源浪费的问题。

[0011] 本申请实施例采用下述技术方案：

[0012] 一种计算机软件开发方法，该方法包括：

[0013] 获取待开发的计算机软件对应的JSON Schema数据；所述待开发的计算机软件对

应的JSON Schema数据，用于描述所述待开发的计算机软件的数据对象模型；

[0014] 根据所述JSON Schema数据所描述的数据对象模型，生成与所述待开发的计算机软件相关的内容；

[0015] 所述内容，包括下述至少一项：

[0016] 基础项目框架；接口文档；测试用例。

[0017] 一种计算机软件开发装置，该装置包括：

[0018] JSON Schema数据获取模块，用于获取待开发的计算机软件对应的JSON Schema数据；所述待开发的计算机软件对应的JSON Schema数据，用于描述所述待开发的计算机软件的数据对象模型；

[0019] 第一生成模块，用于根据所述JSON Schema数据所描述的数据对象模型，生成与所述待开发的计算机软件相关的内容；

[0020] 所述内容，包括下述至少一项：

[0021] 基础项目框架；接口文档；测试用例。

[0022] 本申请实施例采用的上述至少一个技术方案能够达到以下有益效果：

[0023] 由于可以根据用于描述待开发的计算机软件的数据对象模型的JSON Schema数据所描述的数据对象模型，生成与待开发的计算机软件相关的内容(具体而言是基础项目框架、接口文档和测试用例中的至少一项)，因此，无需依赖软件开发人员采用编写代码的方式逐一产出与待开发的计算机软件相关的内容，从而可以提升计算机软件开发过程的整体效率。

附图说明

[0024] 此处所说明的附图用来提供对本申请的进一步理解，构成本申请的一部分，本申请的示意性实施例及其说明用于解释本申请，并不构成对本申请的不当限定。在附图中：

[0025] 图1a为本申请实施例提供的一种计算机软件开发方法的流程图；

[0026] 图1b为本申请实施例中业务逻辑代码框架的目录结构示意图；

[0027] 图2为本申请实施例提供的计算机软件开发方法在实际中的应用流程图；

[0028] 图3为本申请实施例提供的一种计算机软件开发装置的结构示意图。

具体实施方式

[0029] 为使本申请的目的、技术方案和优点更加清楚，下面将结合本申请具体实施例及相应的附图对本申请技术方案进行清楚、完整地描述。显然，所描述的实施例仅是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

[0030] 以下结合附图，详细说明本申请各实施例提供的技术方案。

[0031] 实施例1

[0032] 为解决现有技术存在的采用瀑布式开发导致计算机软件开发过程整体效率较低，且容易导致人力资源浪费的问题，本申请实施例提供一种计算机软件开发方法。

[0033] 该方法的执行主体，可以但不限于是软件开发人员使用的电脑或者其他计算设备。

[0034] 为便于描述，本申请实施例中以该方法的执行主体为电脑为例，对该方法进行详细介绍。在实际应用中，所述电脑可以通过运行设置在所述电脑中的、用于实现该方法的程序(如脚本)，来实现该方法。

[0035] 具体地，该方法的实现流程如图1a所示，包括下述步骤：

[0036] 步骤11，软件开发人员使用的电脑获取待开发的计算机软件对应的JSON Schema数据。

[0037] 其中，JSON Schema可以用于描述存在的数据格式(describes your existing data format)，是一种清晰的、用户和机器均可读的文件(clear,human-and machine-readable documentation)。该文件中的用于描述数据格式的内容，即为本申请实施例中所述的JSON Schema数据。

[0038] 在本申请实施例中，在计算机软件开发场景中，所述电脑获取的JSON Schema数据，用于描述待开发的计算机软件的数据对象模型。

[0039] 本申请实施例中，数据对象(Data Object)，是性质相同的数据元素的集合，是数据的一个子集。数据对象比如可以是外部实体(例如，产生或使用信息的任何事物)、事物(例如，报表)、行为(例如，打电话)、事件(例如，响警报)、角色(例如，教师、学生)、单位(例如，会计科)、地点(例如，仓库)或结构(例如，文件)等。总之，可以由一组属性来定义的实体都可以被认为是数据对象。数据对象模型，是指利用适当的规则对数据对象的主要特征进行描述而得到的模式。

[0040] 针对数据对象模型而言，请参考下表1，表1第一列中包含的各项，为待开发的计算机软件的数据对象模型的各种信息(也即主要特征)，第二列中包含的各项，则是第一列中各项的中文释义。

[0041] 表1：

[0042]

title	标题
description	数据对象模型描述
schemaCode	code码，数据对象模型标识
type	数据类型约束(array,string,number,boolean,null)
properties	各种键与其对应的值类型
required	必要属性列表
code	键对应的code
minimum	接受的最小值(integer与number类型数据)
maximum	接受的最大值(integer与number类型数据)
minLength	最小长度(string类型数据)
maxLength	最大长度(string类型数据)
pattern	正则表达式匹配模式(string类型数据)
default	缺省值

[0043] 本申请实施例中，以所述数据对象模型的信息如表1所示为例，在一个具体的实例中，采用JSON Schema数据描述一种数据对象模型时，具体的JSON Schema数据如下所示：

[0044]
```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "user",
    "description": "user description",
    "type": "object",
    "schemaCode": 1000,
    "properties": {
        "id": {
            "description": "id",
            "type": "integer",
            "code": 1001
        },
        "name": {
            "description": "name",
            "type": "string",
            "minLength": 6,
            "maxLength": 32,
            "code": 1002
        },
        "age": {
            "description": "age",
            "type": "integer",
            "minimum": 18,
            "maximum": 60,
            "default": 0,
            "code": 1003
        }
    },
    "required": [
        "id",
        "name",
        "age"
    ]
}
```

[0046] 本申请实施例中，所述电脑获取的JSON Schema数据，可以是由软件开发人员根据待开发的计算机软件的数据对象模型的各种信息，进行JSON Schema数据编译而得到的。编译得到的JSON Schema数据，可以保存在所述电脑中，以便所述电脑根据需要进行获取。

[0047] 步骤12，所述电脑根据通过执行步骤11获取到的JSON Schema数据所描述的数据对象模型，生成与待开发的计算机软件相关的内容。

[0048] 其中，这里所述的内容，可以但不限于包括下述至少一项：

[0049] 基础项目框架；

[0050] 接口文档；

[0051] 测试用例。

[0052] 以下，分别说明所述电脑如何根据获取到的JSON Schema数据，生成与待开发的计算机软件相关的各内容。

[0053] 首先，介绍所述电脑如何根据获取到的JSON Schema数据，生成基础项目框架。

[0054] 在实际应用中，当期望生成的基础项目框架包括数据表(为与后文提到的其他数据表相区分，该数据表可称为第一数据表)时，针对步骤12而言，所述电脑根据获取到的JSON Schema数据所描述的数据对象模型生成第一数据表的方式，具体可以包括：

[0055] 按照预定转换规则，将获取到的JSON Schema数据所描述的数据对象模型转换为相应的数据定义语言(Data Definition Language，DDL)；根据转换得到的DDL，生成第一数据表。

[0056] 其中，所述预定转换规则，可以是预先设置的规则。该规则比如可以包括但不限于下述规则项中的至少一种：

[0057] 将获取到的JSON Schema数据中包含的数据对象模型的标题，转换为用于定义第一数据表的表名的DDL；

[0058] 将获取到的JSON Schema数据中包含的数据对象模型的描述，转换为用于定义第一数据表的备注的DDL；

[0059] 将获取到的JSON Schema数据中包含的数据对象模型的属性名称，转换为用于定义第一数据表中字段的名称的DDL。

[0060] 在将获取到的JSON Schema数据所描述的数据对象模型转换为相应的DDL后，就可以根据转换得到的DDL生成第一数据表。

[0061] 由于如何将JSON Schema数据转换为DDL，以及如何根据DDL生成数据表，已经是比较成熟的相关技术，本申请实施例对此不再赘述。

[0062] 本申请实施例中，生成的第一数据表，后续可以用于保存数据。具体而言，所述数据可以是通过执行后文所述的服务层中编写好的业务逻辑而创建的。

[0063] 在实际应用中，当期望生成的基础项目框架包括基础代码时，针对步骤12而言，所述电脑根据获取到的JSON Schema数据所描述的数据对象模型生成基础代码的方式具体可以包括：根据获取到的JSON Schema数据所描述的数据对象模型，按照模型层、服务层和控制层的生成规则，生成模型层、服务层和控制层。

[0064] 所述的模型层、服务层和控制层，共同构成所述基础代码。

[0065] 以获取到的JSON Schema数据所描述的数据对象模型为“user”这个数据对象模型为例，照模型层、服务层和控制层的生成规则生成模型层、服务层和控制层的具体实现方式，可以包括：

[0066] 所述电脑将获取到的用于描述“user”这个数据对象模型的JSON Schema数据，转换为超文本预处理器语言(Hypertext Preprocessor，简称php)的文件(文件的后缀一般为.php)，从而得到userController.php、userModel.php和userService.php这几个php文件。

[0067] 得到的这三个文件，依次分别为控制层、模型层和服务层。得到的各个文件中，都会包含一个与文件名同名的类，类中都包括create\update\delete\read函数(也称类方法)，即用于实现CURD的函数。

[0068] 其中，模型层，包括模型层目录下的类。在所述电脑将获取到的JSON Schema数据所描述的数据对象模型转换为控制层的过程中，具体地，所述电脑可以根据JSON Schema数据中的数据对象模型的title(表1中第一行信息)生成模型层的类名——若title为User，则类名一般为：UserModel。此外，根据JSON Schema数据中的数据对象模型的properties(表1中第五行信息)生成类属性及类属性的访问方法。具体而言，假设JSON Schema数据中的properties有id(标识)、name(姓名)和age(年纪)这几个属性，那么，所述电脑可以根据该些属性，通过运行转换脚本转换得到UserModel.php文件。这个文件中，会包含一个与文

件名同名的类，类中都包括create\update\delete\read函数(也称类方法)，这几个方法是缺省方法，用于实现CURD的函数。以create创建用户(User)为例，方法“create”可以接收的参数就包含name、age(id做为特殊属性，使用数据库的id自增字段，这相当于一个统一约束)。create接收参数后，调用对象关系映射(Object-Relational Mapping，ORM)类，将数据写入所述第一数据表。

[0069] 模型层目录下的类用于提供对第一数据表中的数据进行操作的操作函数。其中，第一数据表的具体生成方式请见上文的介绍，此处不再赘述。这里所说的对第一数据表中的数据进行操作的操作函数，具体而言可以包括对第一数据表中的数据进行CURD的数据操作中的至少一种操作，或者，可以包括根据第一数据表中的数据输出数据清单的数据操作(该操作可以称为list)。

[0070] 服务层，包括服务层目录下的类。在所述电脑将获取到的JSON Schema数据转换为服务层的过程中，具体地，所述电脑可以根据JSON Schema数据中的title(表1中第一行信息)生成控制层的类名——若title为User，则类名一般为：UserService。此外，所述电脑还生成类方法，包括create/update/read/delete/lists类方法。其中，电脑生成create/update/read/delete/lists类方法采用比较成熟的相关技术即可实现，此处不再赘述。

[0071] 服务层目录下的类，即为业务逻辑代码。所述电脑将获取到的JSON Schema数据所描述的数据对象模型，转换为服务层后，初始得到的是服务层目录下的空的业务逻辑代码框架。该业务逻辑代码框架，用于软件开发人员在业务逻辑代码框架下编写用于实现业务逻辑的代码。在实际应用中，所述电脑可以按照如图1b的形式，对业务逻辑代码框架进行展示。若采用这样的展示形式，则文件夹图标的层级关系用来表示业务逻辑代码框架的目录结构，文件夹名称则用来表示业务逻辑代码框架下需要编译的内容。软件开发人员通过点击文件夹图标，可以打开文件夹，以便在打开的文件夹下进行代码的编写。

[0072] 软件开发人员在业务逻辑代码框架下编写的所述代码，用于实现一定的业务逻辑。视实际需求的不同，该业务逻辑比如可能是创建某个用户的数据并保存在第一数据表中，或者从第一数据表中读取数据，等等。

[0073] 控制层，包括控制层目录下的类。在所述电脑将获取到的JSON Schema数据转换为控制层的过程中，具体地，所述电脑可以根据JSON Schema数据中的title(表1中第一行信息)生成控制层的类名——若title为User，则类名一般为：UserController。此外，所述电脑还生成类方法，包括create/update/read/delete/lists类方法。其中，电脑生成create/update/read/delete/lists类方法采用比较成熟的相关技术即可实现，此处不再赘述。

[0074] 控制层目录下的类，即为用于调用服务层的业务逻辑的接口的代码。其中，用于调用服务层的业务逻辑的接口，可以开放给期望调用所述业务逻辑的调用方。该调用方，比如可以是所述电脑上运行的一些应用程序，或者是待开发的计算机软件的一些软件开发人员所使用的电脑，等等。

[0075] 以上，介绍的是作为所述执行主体的所述电脑如何根据获取到的JSON Schema数据生成基础项目框架。

[0076] 以下，对所述电脑如何根据获取到的JSON Schema数据生成接口文档和测试用例进行介绍。

[0077] 其中，这里所说的接口文档，是用于记载开发出的计算机软件中包含的接口的相

关信息的文档。

[0078] 针对接口文档的生成方式而言，本申请实施例中，所述电脑可以根据获取到的JSON Schema数据所描述的数据对象模型的特定信息(如表1中所示的“description”、“type”、“minimum”、“maximum”、“minLength”和“maxLength”)所对应的参数值，生成接口文档。该些参数值，可以是控制层的类方法与方法接收的参数值。如上述UserController的create方法，接收的参数值举例来说可以有上述参数的参数值，根据这些参数值可以生成一个user_create.html文件，生成的该文件中描述了暴露给外部调用的接口的调用地址、方式，接受参数与返回值(JSON格式)等信息。生成的该超文本标记语言(HyperText Markup Language，HTML)格式的文件，即为接口文档。该格式的接口文档可以方便软件开发人员采用浏览网页的方式来查看。由于如何将JSON Schema数据转换为相应的HTML，已经是比较成熟的相关技术，本申请实施例不再进行详细介绍。

[0079] 针对测试用例的生成方式而言，本申请实施例中，所述电脑比如可以使用开源的phpunit生成脚本，来实现根据控制层的类方法生成测试用例代码。如针对create类方法，生成对应的createTest方法(该方法即为测试用例的一个实例)，此createTest方法可以调用UserController这个类中的create类方法。

[0080] 以上，是对本申请实施例提供的计算机软件开发方法的具体实施方式进行的介绍。采用本申请实施例提供的该方法，由于可以根据用于描述待开发的计算机软件的数据对象模型的JSON Schema数据，生成与待开发的计算机软件相关的内容(具体而言是基础项目框架、接口文档和测试用例中的至少一项)，因此，无需依赖软件开发人员采用编写代码的方式逐一产出与待开发的计算机软件相关的内容，从而可以提升计算机软件开发过程的整体效率。

[0081] 对本申请实施例提供的该方法进行改进，还可以进一步提升计算机软件开发过程的整体效率。

[0082] 具体而言，针对现有技术中采用的计算机软件开发方法，影响开发过程整体效率的另一原因，是在开发需求发生变化而导致业务逻辑变化的情况下，软件开发人员往往采用的是聚在一起开会的形式，口头通知参与软件开发的相关人员根据变化后的业务逻辑进行相应的内容调整。比如，根据变化后的业务逻辑，通过手动编辑的方式更新JSON Schema数据、接口文档和测试用例等。由于召集软件开发人员开会的方式往往费时费力，并且，即便开会完毕后，也是要通过软件开发人员手动编辑的方式调整接口文档和/或测试用例等，从而严重影响软件开发过程的整体效率。

[0083] 为解决上述问题，在假设前文所述的对所述第一数据表中的数据进行操作的操作函数包括“对所述第一数据表中的数据进行查询的查询函数”，且所述业务逻辑代码框架中设置有业务逻辑的情况下，本申请实施例提供的上述方法可以做如下改进。

[0084] 具体地，所述的改进在于，本申请实施例提供的上述方法可以进一步包括下述步骤一～步骤六：

[0085] 步骤一：所述电脑通过调用控制层中包含的接口(该接口用于调用服务层的业务逻辑)调用服务层的业务逻辑，使得服务层被调用的业务逻辑调用模型层中的所述查询函数；

[0086] 其中，控制层中包含的接口可以接收待查询的目标数据的标识，并根据目标数据

的标识调用服务层的业务逻辑。若业务逻辑的执行会需要调用模型层中的所述查询函数，则该业务逻辑可以根据目标数据的标识，调用所述查询函数，以便所述查询函数根据目标数据的标识，在第一数据表中查询具备该标识的数据，即查询目标数据。查询函数在查询到目标数据后，会将该目标数据返回至该业务逻辑，再由该业务逻辑将该目标数据返回至控制层。

[0087] 针对步骤一的具体实现方式而言，本申请实施例中提供两种具体实现方式。第一种是自动调用的方式，第二种是手动调用的方式。具体如下：

[0088] 自动调用的方式，即：所述电脑通过扫描所述控制层，获取所述控制层包含的用于调用服务层的业务逻辑的接口的地址；而后，根据获取到的所述地址，调用该接口。其中，所述电脑在根据获取到的所述地址调用该接口时，具体可以是根据目标数据的标识(该标识可以是由软件开发人员输入所述电脑的)，来调用所述地址对应的该接口。

[0089] 手动调用的方式，即：所述电脑接收针对控制层包含的用于调用服务层的业务逻辑的接口的指令；当确定出所述指令中存在特定参数值时，调用该接口。其中，该指令可以是软件开发人员向所述电脑发出的。软件开发人员可以设置包含所述特定参数值(特定参数比如可以是一个名为contrast的参数；特定参数值可以为1，即contrast的值为1，也即contrast＝1)的指令并发送给所述电脑。当所述电脑确定出该指令中存在这样的特定参数值时，确定该指令是用于触发调用上述“用于调用服务层的业务逻辑的接口”的指令，从而执行调用该接口的操作。

[0090] 需要说明的是，当待开发的计算机软件的开发需求发生变化时，在本申请实施例中，软件开发人员可以变更服务层的业务逻辑代码，以及按照业务逻辑代码发生的变更，适应性地变更控制层的用于调用服务层的业务逻辑的接口的代码。

[0091] 步骤二：所述电脑获得控制层中被调用的所述接口返回的、由所述查询函数从第一数据表中查询到的目标数据；

[0092] 需要说明的是，由于不同的业务逻辑往往需求不同属性的目标数据，因此，变更后的业务逻辑代码(后称新业务逻辑)调用模型层中的所述查询函数对第一数据表进行查询后返回的目标数据，往往与变更前的业务逻辑(后称原业务逻辑)调用所述查询函数对第一数据表进行查询后返回的目标数据在属性上有所不同。

[0093] 比如，原业务逻辑调用所述查询函数对所述第一数据表进行查询后返回的目标数据可能是array类型，而新业务逻辑调用所述查询函数对第一数据表进行查询后返回的目标数据则可能是string类型。

[0094] 此外需要说明的是，控制层中被调用的所述接口返回的、由所述查询函数从第一数据表中查询到的目标数据的格式为JavaScript对象表示JSON(JavaScript Object Notation))格式。

[0095] 步骤三：所述电脑将目标数据转换为用于描述与目标数据的标识对应的数据对象模型的JSON Schema数据；

[0096] 前文已说明，目标数据的格式为JSON格式。通过执行步骤三，可以将JSON格式的目标数据，转换为相应的JSON Schema数据。

[0097] 举例来说，沿用上例，若新业务逻辑调用所述查询函数对第一数据表进行查询后返回的目标数据是string类型，那么，将JSON格式的目标数据，转换为相应的JSON Schema

数据后，得到的JSON Schema数据所描述的数据对象模型的类型(如表1中所示的type)为string。

[0098] 步骤四：所述电脑根据转换得到的JSON Schema数据，判断所述业务逻辑是否发生变化，若是，执行步骤五；若否，执行步骤六；

[0099] 步骤五：所述电脑重新生成与待开发的计算机软件相关的内容，流程结束。其中，这里所说的重新生成的内容，包括接口文档和/或测试用例。

[0100] 步骤六：所述电脑不执行任何操作，或者显示“业务逻辑无变化”这样的文字，以便提示所述电脑的使用者业务逻辑没有发生变化。

[0101] 针对上述步骤四而言，在一种实施方式中，所述电脑根据转换得到的JSON Schema数据，判定所述业务逻辑是否发生变化，具体可以包括下述子步骤：

[0102] 子步骤41：所述电脑从存储有JSON Schema数据所描述的数据对象模型的第二数据表中，获取目标JSON Schema数据；

[0103] 其中，所述第二数据表，为所述电脑中的一个数据表，它用于对应存储描述数据对象模型的JSON Schema数据和相应的数据对象模型的标识。针对同一数据对象模型而言，该第二数据表中，保存有用于描述该数据对象模型的不同版本的JSON Schema数据。用于描述该数据对象模型的不同版本的JSON Schema数据，对应于在计算机软件开发过程中对于该数据对象模型的不同需求，也即对应于不同的业务逻辑。

[0104] 举个简单的例子来说，比如，第一版本的JSON Schema数据所描述的该数据对象模型，它的类型(type)可能是“array”；但当对于该数据对象模型的需求发生变化导致业务逻辑发生变更后，它的“type”可能就变成“string”，从而用于描述该数据对象模型的JSON Schema数据就会与第一版本的JSON Schema数据有所不同，相应地，版本也会从第一版本变化为第二版本。

[0105] 所述目标JSON Schema数据，为用于描述与目标数据的标识相对应的数据对象模型的JSON Schema数据中，最近一次更新的JSON Schema数据。

[0106] 当然，在业务逻辑由原业务逻辑变更为新业务逻辑后，最近一次更新的JSON Schema数据有可能还是对应于原业务逻辑的JSON Schema数据。也即，业务逻辑变更为新业务逻辑后，第二数据表中的JSON Schema数据还没有发生相应变化。那么，在这样的情况下，就会出现通过执行子步骤42获取的所述目标JSON Schema数据与通过执行步骤三转换得到的JSON Schema数据不一致的情况，详见下述子步骤42。

[0107] 子步骤42：所述电脑比较通过执行子步骤42获取的所述目标JSON Schema数据与通过执行步骤三转换得到的JSON Schema数据是否一致；若比较结果为一致，则执行子步骤43；若比较结果为不一致，则执行子步骤44。

[0108] 举例来说，沿用上例，若新业务逻辑调用所述查询函数对第一数据表进行查询后返回的目标数据是string类型，那么，将JSON格式的目标数据，转换为相应的JSON Schema数据后，得到的JSON Schema数据所描述的数据对象模型的类型(如表1中所示的type)为string。

[0109] 这样的情况下，如果目标JSON Schema数据所描述的该数据对象模型的类型为array，那么，所述电脑对目标JSON Schema数据和转换得到的JSON Schema数据进行比较，得到的比较结果为不一致；而如果目标JSON Schema数据所描述的该数据对象模型的类型

为string，那么，所述电脑对目标JSON Schema数据和转换得到的JSON Schema数据进行比较，得到的比较结果为一致。

[0110] 当然，上文只是举例说明所述比较结果什么情况下为一致，什么情况下为不一致。在实际应用中，软件开发人员可以视实际情况来设置比较规则，以使得所述电脑根据设置的比较规则得到比较结果。比如，所述比较规则一般可以为“目标JSON Schema数据和转换得到的JSON Schema数据分别描述的数据对象模型的数据属性完全一致”。那么，按照该比较规则，所述电脑在判断出目标JSON Schema数据和转换得到的JSON Schema数据分别描述的数据对象模型的数据属性完全一致时，得到的比较结果为一致，否则，得到的比较结果为不一致。

[0111] 子步骤43，所述电脑判定所述业务逻辑没有发生变化；

[0112] 子步骤44，所述电脑判定所述业务逻辑发生变化。

[0113] 针对前文所述的步骤一～步骤六还需要说明的是，在步骤五中，所述电脑重新生成与待开发的计算机软件相关的内容，具体可以包括：根据从第二数据表中获取到的所述目标JSON Schema数据所描述的数据对象模型，重新生成与待开发的计算机软件相关的内容(包括接口文档和/或测试用例)。具体如何根据JSON Schema数据生成接口文档和测试用例可以参见前文的相关介绍，此处不再赘述。

[0114] 采用本申请实施例介绍的上述步骤一～步骤六，由于由所述电脑可以自动判断业务逻辑是否发生变化，并且在判定发生变化时，可以由所述电脑根据转换得到的JSON Schema数据所描述的数据对象模型，重新生成与待开发的计算机软件相关的内容，因此，相比于现有技术中软件开发人员口头通知根据变化后的业务逻辑进行相应的内容调整的方式相比，本申请实施例提供的方法可以节省口头通知所耗费的人力和时间，此外，还可以节省软件开发人员根据通知采用手动编辑的方式调整接口文档和/或测试用例等耗费的时间，从而可以提升软件开发过程的整体效率，并且降低人力资源的耗费。

[0115] 此外，由于所述转换得到的JSON Schema数据，也是由所述电脑根据控制层的接口返回的JSON格式的目标数据自动转换得到的，不需要软件开发人员手动编译，因此，可以进一步提升软件开发过程的整体效率，并且进一步降低人力资源的耗费。

[0116] 需要说明的是，虽然本申请实施例1中以方法的执行主体为某电脑为例，对该方法进行了介绍，但本领域技术人员可以理解，本申请实施例1所提供方法的各步骤的执行主体均可以是同一设备，或者，该方法也由不同设备作为执行主体。比如，步骤11和步骤12的执行主体可以为同一设备；又比如，步骤11的执行主体可以为设备1，步骤12的执行主体可以为设备2；等等。

[0117] 将本申请实施例提供的方法应用到实际中，可以产生如图2所示的可应用于计算机软件开发项目中的一种计算机软件开发流程。该流程具体可以包括下述步骤：

[0118] 步骤21，电脑获取用于描述待开发的计算机软件的数据对象模型的JSON Schema数据；

[0119] 步骤22，电脑根据获取到的JSON Schema数据，创建数据表；

[0120] 步骤23，电脑根据获取到的JSON Schema数据，生成基础代码；

[0121] 步骤24，电脑根据获取到的JSON Schema数据，生成接口文档；

[0122] 步骤25，电脑根据获取到的JSON Schema数据，生成测试用例；

[0123] 上述步骤21～步骤25的具体实现方式可以参见前文的描述，此处不再赘述。

[0124] 步骤26，电脑对业务逻辑是否发生变更进行监控；若监控到业务逻辑发生变更，执行步骤27，否则，执行步骤26；

[0125] 其中，电脑对业务逻辑是否发生变更进行监控，具体包括：电脑可以采用定期采用前文介绍的自动调用或者手动调用的方式，来判断业务逻辑是否发生变更。

[0126] 步骤27，电脑更新JSON Schema数据；

[0127] 比如，电脑可以根据控制层的接口返回的JSON格式的目标数据，更新第二数据表中保存的JSON Schema数据。

[0128] 步骤28，电脑根据更新后的JSON Schema数据，更新接口文档；

[0129] 即电脑根据更新后的JSON Schema数据，重新生成整个接口文档，或者重新生成接口文档中需要根据更新后的JSON Schema数据发生更新的部分；

[0130] 步骤29，电脑根据更新后的JSON Schema数据，更新测试用例，之后，可以跳转为执行步骤26。

[0131] 步骤29的具体过程，即电脑根据更新后的JSON Schema数据，重新生成整个测试用例，或者重新生成测试用例中需要根据更新后的JSON Schema数据发生更新的部分。

[0132] 实施例2

[0133] 以上为本申请实施例提供的计算机软件开发方法，基于同样的思路，本申请实施例还提供了相应的计算机软件开发装置，如图3所示。

[0134] 图3为本申请实施例提供的计算机软件开发装置结构示意图，具体包括：

[0135] JSON Schema数据获取模块31，用于获取待开发的计算机软件对应的JSON Schema数据；所述待开发的计算机软件对应的JSON Schema数据，用于描述所述待开发的计算机软件的数据对象模型；

[0136] 第一生成模块32，用于根据所述JSON Schema数据所描述的数据对象模型，生成与所述待开发的计算机软件相关的内容；

[0137] 所述内容，包括下述至少一项：

[0138] 基础项目框架；接口文档；测试用例。

[0139] 若所述内容包括所述基础项目框架；所述基础项目框架包括第一数据表；

[0140] 则所述第一生成模块32包括：

[0141] 转换单元，用于按照预定转换规则，将所述JSON Schema数据所描述的数据对象模型转换为相应的数据定义语言DDL；

[0142] 第一生成单元，用于根据转换得到的DDL，生成所述第一数据表；

[0143] 其中，所述预定转换规则，包括下述规则项中的至少一种：

[0144] 将获取到的JSON Schema数据中包含的数据对象模型的标题，转换为用于定义第一数据表的表名的DDL；

[0145] 将获取到的JSON Schema数据中包含的数据对象模型的描述，转换为用于定义第一数据表的备注的DDL；

[0146] 将获取到的JSON Schema数据中包含的数据对象模型的属性名称，转换为用于定义第一数据表中字段的名称的DDL。

[0147] 若所述内容包括所述基础项目框架；所述基础项目框架还包括模型层、服务层和

控制层；

[0148] 则所述第一生成模块32还包括：

[0149] 第二生成单元，用于根据所述JSON Schema数据所描述的数据对象模型，按照模型层、服务层和控制层的生成规则，生成模型层、服务层和控制层；其中，模型层、服务层和控制层的生成规则，具体包括：根据所述JSON Schema数据所描述的数据对象模型的标题，分别生成模型层、服务层和控制层的类的名称；根据所述JSON Schema数据所描述的数据对象模型的属性，分别生成模型层、服务层和控制层的类的类属性和类属性的访问方法；

[0150] 所述模型层，用于提供对所述第一数据表中的数据进行操作的操作函数；

[0151] 所述服务层，包括业务逻辑代码框架；所述业务逻辑代码框架，用于软件开发人员在所述业务逻辑代码框架下编写用于实现业务逻辑的代码；

[0152] 所述控制层，包括用于调用所述服务层的业务逻辑的接口的代码。

[0153] 若所述对所述第一数据表中的数据进行操作的操作函数，包括对所述第一数据表中的数据进行查询的查询函数；在所述业务逻辑代码框架中设置有业务逻辑的情况下，所述装置还包括：

[0154] 调用模块33，用于通过调用所述控制层中包含的用于调用所述服务层的业务逻辑的接口，使得所述服务层被调用的业务逻辑调用所述模型层中的所述查询函数；

[0155] 目标数据获取模块34，用于获得所述接口返回的、由所述查询函数从所述第一数据表中查询到的目标数据；

[0156] 转换模块35，用于将所述目标数据转换为用于描述与所述目标数据的标识对应的数据对象模型的JSON Schema数据；

[0157] 判断模块36，用于根据转换得到的JSON Schema数据，判断所述业务逻辑是否发生变化；

[0158] 第二生成模块37，用于若判断所述业务逻辑发生变化，则重新生成与待开发的计算机软件相关的内容；

[0159] 重新生成的内容包括接口文档和/或测试用例。

[0160] 所述判断模块36包括：

[0161] 目标JSON Schema数据获取单元，用于从存储有JSON Schema数据的第二数据表中，获取目标JSON Schema数据；其中，所述目标JSON Schema数据为：用于描述与所述目标数据的标识相对应的数据对象模型的JSON Schema数据中，最近一次更新的JSON Schema数据；

[0162] 比较单元，用于比较所述目标JSON Schema数据与转换得到的JSON Schema数据是否一致；

[0163] 第一判定单元，用于若所述目标JSON Schema数据与转换得到的JSON Schema数据一致，则判定所述业务逻辑没有发生变化；

[0164] 第二判断单元，用于若所述目标JSON Schema数据与转换得到的JSON Schema数据不一致，则判定所述业务逻辑发生变化。

[0165] 所述第二生成模块包括：

[0166] 第三生成单元，用于根据所述目标JSON Schema数据所描述的数据对象模型，重新生成与待开发的计算机软件相关的内容。

[0167] 为了实现调用服务层的业务逻辑的接口，在一种实施方式中，所述调用模块33可以包括：地址获取单元，用于通过扫描所述控制层，获取所述控制层包含的所述接口的地址；第一调用单元，用于根据所述地址，调用所述接口。

[0168] 为了实现调用服务层的业务逻辑的接口，在另一种实施方式中，所述调用模块33可以包括：指令接收单元，用于接收针对所述接口的指令；第二调用单元，用于当确定出所述指令中存在特定参数值时，调用所述接口。其中，所述指令中包含所述特定参数值，表示所述指令用于触发调用所述接口。

[0169] 本申请通过根据用于描述待开发的计算机软件的数据对象模型的JSON Schema数据所描述的数据对象模型，生成与待开发的计算机软件相关的内容(具体而言是基础项目框架、接口文档和测试用例中的至少一项)，因此，无需依赖软件开发人员采用编写代码的方式逐一产出与待开发的计算机软件相关的内容，从而可以提升计算机软件开发过程的整体效率。

[0170] 本领域内的技术人员应明白，本发明的实施例可提供为方法、系统、或计算机程序产品。因此，本发明可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本发明可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0171] 本发明是参照根据本发明实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0172] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0173] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0174] 在一个典型的配置中，计算设备包括一个或多个处理器(CPU)、输入/输出接口、网络接口和内存。

[0175] 内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器(RAM)和/或非易失性内存等形式，如只读存储器(ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。

[0176] 计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存(PRAM)、静态随机存取存储器(SRAM)、动

态随机存取存储器(DRAM)、其他类型的随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带，磁带磁磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括暂存电脑可读媒体(transitory media)，如调制的数据信号和载波。

[0177] 还需要说明的是，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个……”限定的要素，并不排除在包括所述要素的过程、方法、商品或者设备中还存在另外的相同要素。

[0178] 以上所述仅为本申请的实施例而已，并不用于限制本申请。对于本领域技术人员来说，本申请可以有各种更改和变化。凡在本申请的精神和原理之内所作的任何修改、等同替换、改进等，均应包含在本申请的权利要求范围之内。

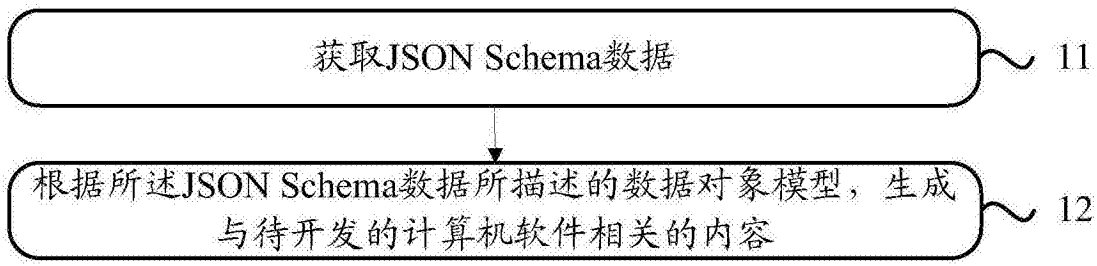

获取JSON Schema数据
11
根据所述JSON Schema数据所描述的数据对象模型，生成与待开发的计算机软件相关的内容
12

图1a

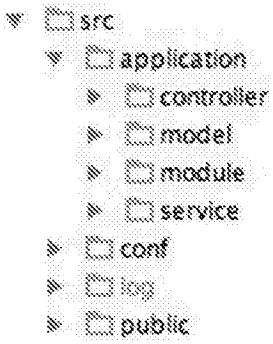

src
application
controller
model
module
service
conf
log
public

图1b

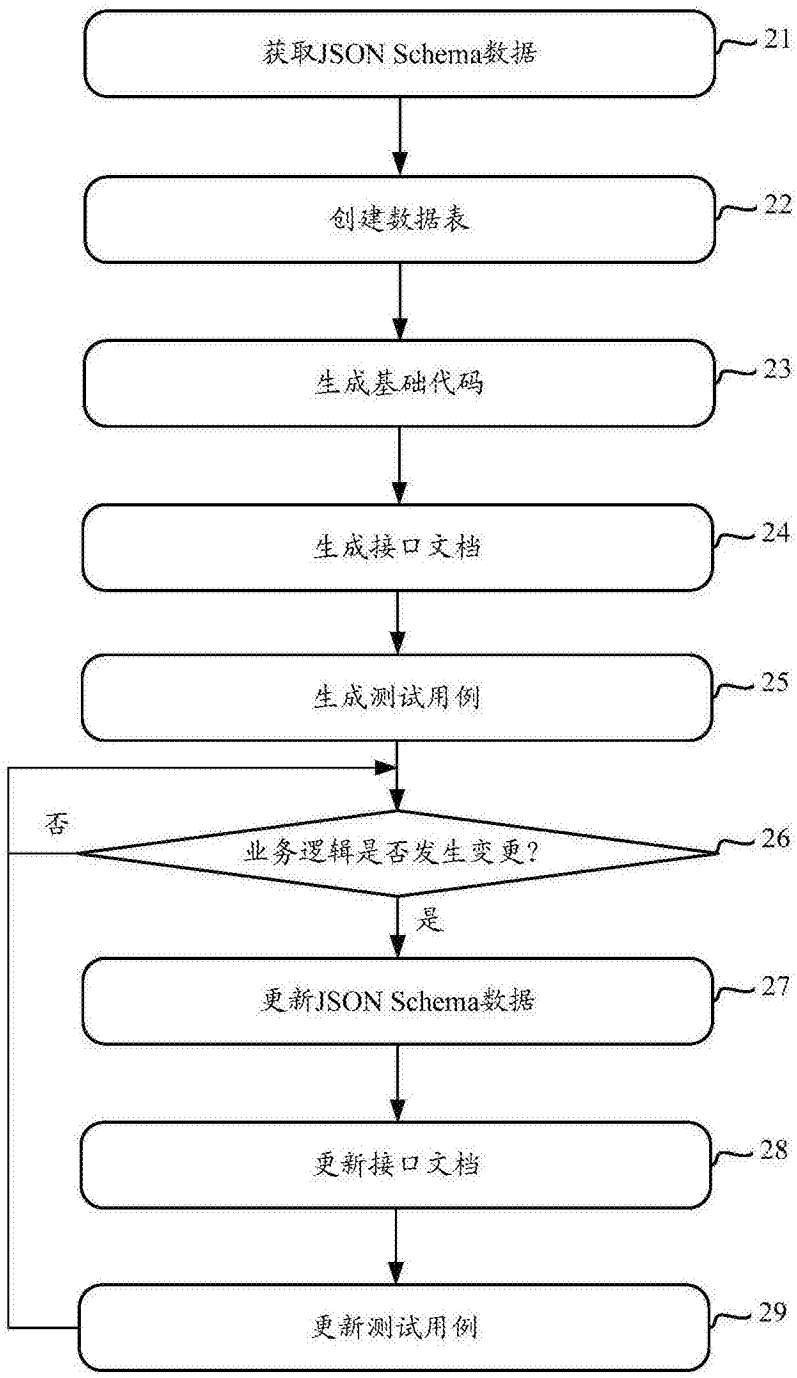

获取JSON Schema数据
21
创建数据表
22
生成基础代码
23
生成接口文档
24
生成测试用例
25
否
业务逻辑是否发生变更?
26
是
更新JSON Schema数据
27
更新接口文档
28
更新测试用例
29

图2

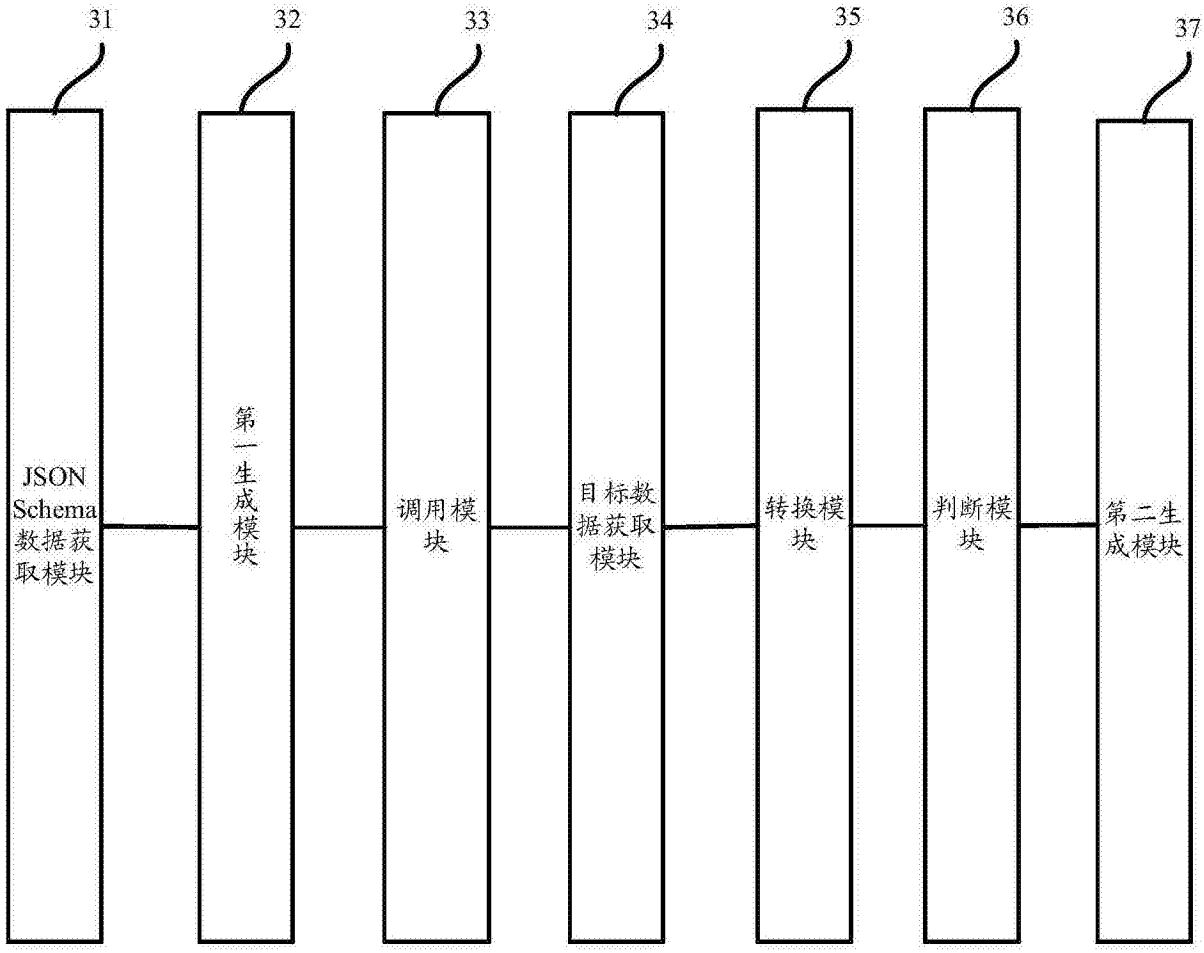

31
32
33
34
35
36
37
JSON Schema 数据获取模块
第一生成模块
调用模块
目标数据获取模块
转换模块
判断模块
第二生成模块

图3