互动百科架构设计说明书

目录

[一． 简介 1](#_Toc471397272)

[二． 整体说明 2](#_Toc471397273)

## 简介

###### 编写目的

互动百科是由用户贡献内容，具有知识性的词条。词条要求准确、清晰、无歧义，一般由不同部分组成。本文档详细描述了互动百科的构架，根据需求定义和架构约束，给出解决方案，用于指导项目的执行。

###### 文档范围

本架构文档适合于项目开发人员、测试人员以及需要了解项目总体架构的其他人员

###### 术语定义

1. 词条（wiki）：由用户创建、具有知识性、准确性等执行，对各种事物、现象进行解释
2. 模板：对应词条的组成部分。不同词条对应有不同的模板，如人物词条模板、重大事件词条模板。在本架构设计中，为方便管理，我们只使用一个模板，包括词条简介、词条对应图片和词条内容
3. 分类：词条对应所属类别，在本架构设计中，共有8类，分别为：人物、自然、文化、体育、社会、历史、娱乐和教育
4. MVC模式：用于web框架设计的架构模式，其中M代表模型（model），是数据的表示形式；V代表视图（view），与用户进行交互；C代表控制器（controller），负责控制模型与视图的交互
5. MVT模式：著名的web框架django框架中使用的模式，其中M代表模型（model），是数据访问层；T代表模板（template），是数据表示层；V代表视图（view），是商业逻辑层。
6. ORM：对象关系映射，将基于SQL的关系模型数据库结构映射到使用对象模型的对象，常用于各web框架中。使用ORM后，用户只需要对对象进行操作，对应的操作就会在数据库更新。
7. C/S架构：即客户机/服务器架构。服务器提供服务，客户机通过客户端访问服务器使用服务。我们使用C/S架构来实现我们的系统，后台部署在服务端，提供与前端的接口；而用户可以在客户机用浏览器访问前端页面，使用我们的系统。

###### 参考资料

1. [Model-view-controller pattern](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller)
2. [Object-relational mapping](https://en.wikipedia.org/wiki/Object-relational_mapping)
3. [The Model-View-Controller Design Pattern](http://djangobook.com/model-view-controller-design-pattern/)
4. [Django documentation](https://docs.djangoproject.com/en/1.10/)

# 整体说明

###### 简介

在本节中，我们将描述系统架构目标和约束、质量属性并最终给出解决方案等

###### 架构目标和约束

本互动百科架构目标和约束主要有记下几点：

1. 提供用户模块，包括提供用户登录、验证和更新个人信息等功能
2. 提供词条模块，包括创建词条、更新词条和查看词条等功能
3. 提供权限控制功能，只有管理员权限才能审核词条

###### 质量属性要求

基于需求文档和上述架构目标，我们在这一节考虑本互动百科需要考虑的质量属性，并以具体场景来定义。

* 1. 可得性（Availability）

场景一

|  |  |
| --- | --- |
| 刺激源 | 外部系统 |
| 刺激 | 系统被攻击，导致崩溃 |
| 环境 | 崩溃状态 |
| 制品 | 整个系统 |
| 响应 | 记录日志，重新部署服务 |
| 响应测量 | 系统崩溃到重新上线的时间，受损程度 |

* 1. 可修改性（Modifiability）

场景一

|  |  |
| --- | --- |
| 刺激源 | 系统开发人员 |
| 刺激 | 前端人员希望更改UI |
| 环境 | 系统处于关闭状态 |
| 制品 | 前端页面 |
| 响应 | 修改UI并进行测试 |
| 响应测量 | 完成修改到重新上线所需时间 |

* 1. 性能（Performance）

|  |  |
| --- | --- |
| 刺激源 | 用户 |
| 刺激 | 用户搜寻某一词条 |
| 环境 | 系统处于正常运行环境 |
| 制品 | 整个系统 |
| 响应 | 根据用户输入进行搜索并返回相应结果 |
| 响应测量 | 搜索所有时间以及结果相关性 |

4. 安全性（Security）

|  |  |
| --- | --- |
| 刺激源 | 用户 |
| 刺激 | 外部人员尝试强行进入管理员界面/内部人员需要登入管理员界面 |
| 环境 | 系统处于正常运行环境 |
| 制品 | 用户系统 |
| 响应 | 拒绝未授权用户登入/运行合法用户登入 |
| 响应测量 | 当未授权用户登入时系统受损程度 |

5. 可测试性（Testability）

|  |  |
| --- | --- |
| 刺激源 | 测试人员 |
| 刺激 | 对系统发起单元测试、集成测试等 |
| 环境 | 系统处于测试状态 |
| 制品 | 整个系统 |
| 响应 | 执行测试 |
| 响应测量 | 未通过测试数量和测试所用时间 |

6．可用性（Usability）

场景一

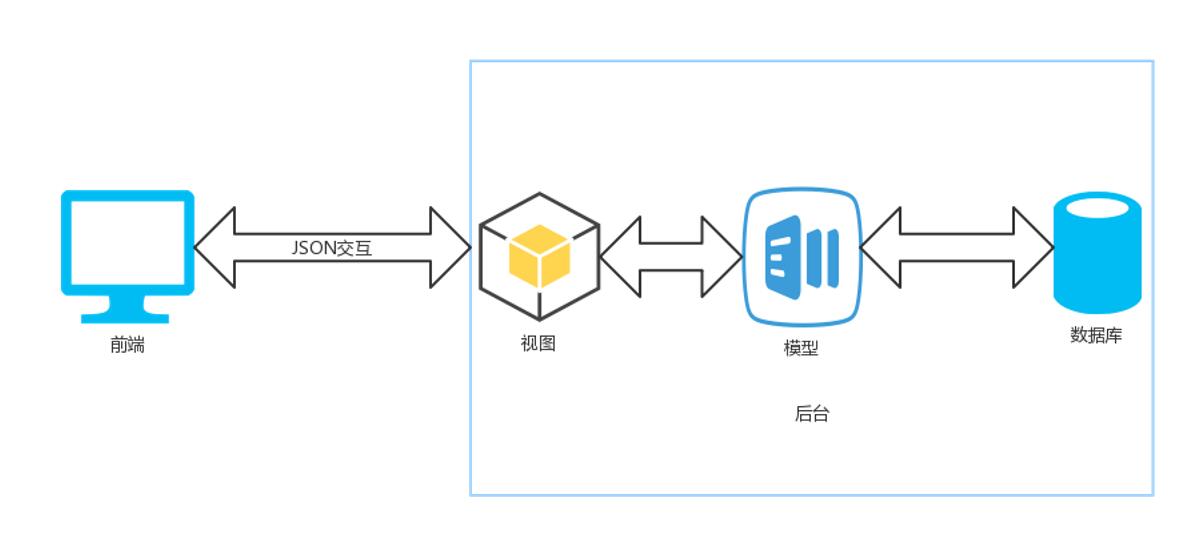
|  |  |
| --- | --- |
| 刺激源 | 用户 |
| 刺激 | 用户需要修改个人信息、wiki等 |
| 环境 | 系统处于正常运行环境 |
| 制品 | 整个系统 |
| 响应 | 拒绝非法修改、接受合法修改 |
| 响应测量 | 响应用户修改所用时间 |

场景二

|  |  |
| --- | --- |
| 刺激源 | 用户 |
| 刺激 | 用户需要查看wiki以及其他信息等 |
| 环境 | 系统处于正常运行环境 |
| 制品 | 整个系统 |
| 响应 | 返回用户需要的个人信息 |
| 响应测量 | 查询数据库所用时间 |

###### 解决方案

基于上述约束和质量属性分析，我们决定使用基于MVC的web架构实现。为方便测试，我们将架构分为前端和后台两部分，于是对前端的修改不影响后台部分。前端使用HTML、CSS和JS编写，负责与用户交互。后台使用Python语言的Django框架，部署在服务器上，提供数据库访问、提供服务等。前后台交互使用JSON作为交互数据格式。于是总体设计如下图



###### 总体设计

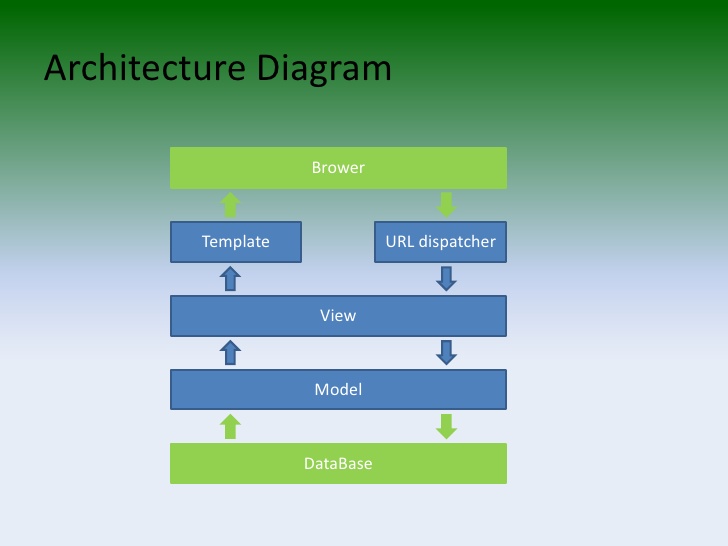
这节描述总体设计。分为前端、后台以及数据库设计三部分

**前端**

前端采用纯HTML、CSS和JS编写，用户可以通过浏览器直接访问前端页面。

**后台**

后台采用Python语言的Django框架。Django框架使用基于MVC模式的MVT模式。整体架构如下图。



Django是基于MVC的web框架，提供了model、view和template功能。其中，model与数据有关，包含了如何访问数据，如何验证数据，数据行为，数据更新和创建等。view包含基本逻辑，包含访问model以及渲染合适的模板。template则是与用户交互部分，用于展示数据和UI。由于这里我们将前后台分离，于是我们只使用了Django框架提供的model和view功能。除此之外，Django框架提供ORM功能，将数据库表结构映射到Django的model类。而在view中，我们定义各项操作，包括调用model来获取数据，向前端返回数据等。使用Django框架，我们可以很好的满足上述质量属性。

在后台设计中，我们将其主要分为两个模块，分别为用户模块，wiki模块。用户模块负责处理与用户有关的事务，包括用户注册，验证以及用户授权等；wiki模块负责与wiki有关的事务，包括wiki创建、编辑等功能。我们在model定义用户类、wiki类等，负责与数据库交互；而在views层，我们则接受来自前端的请求，执行操作并返回结果

**数据库设计**

在这一节，我们描述我们数据库定义，分别有四个表，wiki, user, user\_wiki\_relationship, comment。

其中，wiki表存储wiki信息，user表存储用户信息，comment表存储用户对某一个wiki发表的评论信息，而user\_wiki\_relationship则描述了用户与某一wiki之间的关系。详细数据字典参见下面

**表wiki:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据名称 | 数据业务描述 | 业务数据类型 | 对应的字段 | 字段类型 |
| id | 标识wiki，一条wiki只对应一个唯一id。 | 整型，长度最长为11位 | id | int(11) |
| 标题 | wiki对应标题 | 字符串，最长为45个字符 | title | varchar(45) |
| 简介 | wiki对应简介 | 文本，长度不限 | introduction | text |
| 内容 | wiki对应内容 | 文本，长度不限 | content | text |
| 状态 | wiki的状态，其中0表示正在审核，-1表示审核不通过，1表示审核通过 | 整型，长度为1 | status | int(11) |
| 时间 | wiki 创建时间 | 时间戳类型 | time | datetime |
| 图片地址 | wiki 对应图片地址 | 字符串，最长为45个字符 | img\_url | varchar(45) |
| 类别 | wiki 对应类别 | 字符串，最长为45个字符 | category | varchar(45) |
| 点击量 | wiki 对应点击量 | 整型，长度最长为11位 | hits | int(11) |

**表user**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据名称 | 数据业务描述 | 业务数据类型 | Field | Type |
| 用户名 | 用户名，唯一标识一名用户 | 字符串，最长为45个字符 | account | varchar(45) |
| 密码 | 用户帐号对应密码 | 字符串，最长为250个字符 | password | varchar(250) |
| 上一次登录时间 | 用户上一次登录时间 | 时间戳类型 | last\_login | datetime |
| 头像地址 | 用户头像所在地址 | 字符串，最长为45个字符 | portrait\_url | varchar(45) |
| 是否为管理员 | 标识用户是否是管理员。1表示是管理员，0表示不是 | 整型，最长为1位 | is\_admin | tinyint(1) |
| 创建词条数 | 用户所创建的词条数 | 整型，最长为11位 | num\_of\_wiki | int(11) |

**表user\_wiki\_relationship**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据名称 | 数据业务描述 | 业务数据类型 | Field | Type |
| wiki id | 对应wiki的id，外键约束 | 整型，最长为11位 | wiki | int(11) |
| 用户名 | 对应用户名，外键约束 | 字符串，最长为45个字符 | user\_account | varchar(45) |
| 关系 | 用户与该wiki对应关系。0表示 | 整型，最长为11位 | relationship | int(11) |

**表comment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据名称 | 数据业务描述 | 业务数据类型 | Field | Type |
| id | 标识评论，一条评论唯一对应一个id | 整型，最长为11位 | comment\_id | int(11) |
| 用户名 | 该评论对应用户 | 字符串，最长为45个字符 | user\_account | varchar(45) |
| 内容 | 对应评论内容 | 文本，长度不限 | content | text |
| 时间 | 对应评论时间 | 时间戳类型 | time | datetime |
| 标题 | 对应所评论wiki的标题 | 字符串，最长为45个字符 | wiki\_title | varchar(45) |

了解更详细的数据库设计和详细实现方案，请参考文档数据库设计和详细设计。