

16 | 基础用户画像：如何用好用户的注册信息？

2023-05-22 黄鸿波 来自北京

《手把手带你搭建推荐系统》



你好，我是黄鸿波。

在前面的课程中我们讲了内容画像，讲了推荐系统的服务端搭建，本节课我们就在前面的基础上，讲解用户画像的知识。

我将本节课分为了下面三个重要部分。

1. 什么是用户画像，用户画像在推荐系统中的作用是什么。
2. 用户画像中的用户行为数据分析。
3. 如何利用用户信息搭建一个基础的用户画像。

用户画像及其作用

首先我们来聊一聊什么是用户画像。

先来区分一下内容画像和用户画像。内容画像是用来刻画内容的一系列特征组合，它只需要对内容服务，与用户的行为以及日志等相关信息并没有什么直接的联系。

但是用户画像就不一样了，一个优秀的用户画像不仅仅是对用户基本信息的特征提取，往往还包含着用户的行为信息、用户潜在行为信息以及用户的行为所关联的内容信息等。因此，相比于内容画像，用户画像更加复杂多变，所起的作用也更加庞杂。

通过用户画像，我们可以很容易地定位到一个人的喜好，并对喜好进行预测。要想给用户推荐更好的内容，就必须先对用户有着充分地了解。在很多公司中都会存在运营这个岗位，这个岗位有一个非常重要的岗位职责就是对用户进行定位，然后找到自己产品中的潜在用户，然后通过一系列的营销手段给用户进行产品的推荐。这个时候，运营人员就相当于一个推荐系统。

用户画像的好坏可以直接决定着推荐的质量。一个好的用户画像，不仅能够刻画出这个用户本身的基本属性，更能够让算法知道如何利用用户画像的特征来挖掘潜在的信息。

我们来假想这样一个场景，现在有以下两种用户画像。

第一种只是在注册的时候记录了基本的用户信息，这些用户信息假设包含了性别、年龄、所在地等。第二种是在第一种的基础上增加了用户的浏览记录，我们可以根据用户的浏览记录来获取到用户最近在看的是哪一类内容，有没有进行点赞、收藏，以及在这个页面停留多久。

很显然，第二种画像相比于第一种更能利用好历史信息，找到兴趣点进行推荐。

用户画像中的用户行为数据分析

用户的行为数据是非常关键，不同的用户行为完全不同。可以说，一万个人能有一万个完全不同的用户行为，我们之所以说推荐系统是千人千面的，就是因为有千变万化的行为信息。

在用户画像中，我们可以用多种方法来分析用户的行为数据。我们来看三个常见的用户行为数据分析的方法。

行为路径分析

行为路径分析是用户画像中最重要的一个部分，它可以帮助我们理解用户在产品中的行为。这些行为包含了一个用户从进入网站到关闭浏览器这个过程中的所有行为（比如搜索、浏览内容、点赞、收藏、评论、转发，在一些电商网站中还有添加到购物车和购买等），我们可以将每个用户从开始到结束的所有行为保存成一条记录，然后利用一些统计学原理进行行为分析，从而找到想要挖掘的点。

shikey.com转载分享

行为偏好分析

行为偏好就是利用用户画像来分析用户的个人习惯以及对内容的偏好，从而为其提供更好的个性化推荐服务。✎在[基于热度的召回](#)这一节中我们有讲过，一个内容的热度实际上跟这个内容收到多少个阅读、点赞、收藏、评论、转发等信息有关。我们可以为每一个行为设置一个权重（不同的权重代表重要程度不同），然后再用权重去乘以数量，最后把所有的权重求和就得到了内容的热度。这里内容的热度是作用于内容画像的，但是我们来反推这个经过，实际上这里的每一个步骤，就是一个用户画像中的行为。

我们可以通过用户对内容的操作知道用户对内容的喜好程度，这个喜好程度可以关联到内容的类别、关键词等信息中，从而实现对用户行为偏好的分析。几乎所有的推荐算法中都会用到行为偏好分析的数据，比如说后面会讲到的排序算法中的 GBDT+LR。

搜索行为分析

搜索行为分析实际上和✎[基于关键词的召回](#)相辅相成。搜索行为分析可以帮助我们了解用户的搜索行为，从而为他们提供更好的搜索结果和搜索体验。我们可以通过分析用户在搜索过程中的搜索词、搜索结果、点击率等行为，了解他们的搜索意图和偏好，从而更好地优化搜索引擎和推荐算法。

通过对用户行为的精准分析可以很好地提高推荐精度，同时还可以及时发现和修正推荐算法和模型中的不足之处，提高用户的黏性。有了用户行为分析，我们可以很好地发现内容的不足和产品的不足，从而改善产品的质量。

利用用户信息搭建一个基础用户画像

最后，我们来搭建一个基础的用户画像。

我们目前只是把召回集的内容以时间召回的形式推荐了出去，还没有完成用户的注册、登录等基本信息，因此还无法收集用户的行为记录。在本节课中，我们先来搭建一个以用户注册信息为导向的基础的用户画像。

一般来讲，用户的基础信息多为结构化的数据，并且这类数据一旦注册后就很难改变，因此在这里我建议使用结构化关系型数据库进行存储。在推荐系统或者说整个软件系统的架构中，常见的关系型数据库有 MySQL、Oracle、SqlServer 等，在这里，我们使用 MySQL 数据库作为用户信息的存储数据库。

在选择版本上面，我建议使用 MySQL Community Server 版本，尤其是开发时在 Windows 系统上，该版本非常适合。

我们可以使用 [这个网址](#) 来下载 MySQL Community Server 安装包，在这里我选择的版本为 5.7.41，选择 ZIP Archive 后面的 Download 进行下载，在 Windows 上可以直接下载 MSI 版本，如图所示。

MySQL Community Downloads

MySQL Community Server

[General Availability \(GA\) Releases](#)[Archives](#)[Info](#)

MySQL Community Server 8.0.32

Select Operating System:
Microsoft Windows

[Looking for previous GA versions?](#)

Recommended Download:


MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.


Windows (x86, 32 & 64-bit), MySQL Installer MSI

[Go to Download Page >](#)



Other Downloads:

Windows (x86, 64-bit), ZIP Archive (mysql-8.0.32-winx64.zip)	8.0.32	223.6M	Download
MD5: ef713001cfcee2e72c4de5f6c61db395 Signature			
Windows (x86, 64-bit), ZIP Archive Debug Binaries & Test Suite (mysql-8.0.32-winx64-debug-test.zip)	8.0.32	657.6M	Download
MD5: 0173906d48f23500be69663299b275f2 Signature			

 We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

然后选择使用之前的版本，如图所示。

MySQL Community Downloads

MySQL Installer

General Availability (GA) ReleasesArchives

MySQL Installer 8.0.32

Select Operating System:
Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.32.0.msi)	8.0.32	2.4M	Download
MD5: 0f882590f8338adc614e9dc5cb00ca0b Signature			
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.32.0.msi)	8.0.32	437.3M	Download
MD5: a29b5817cba2c7bc0e0b97e897c2591f Signature			

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

ORACLE © 2023 Oracle

[Privacy](#) / [Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) | [Cookie](#) 喜好设置

然后进行下载 MSI 版本的安装程序。

MySQL Community Downloads

MySQL Installer

General Availability (GA) ReleasesArchives

MySQL Installer 5.7.41

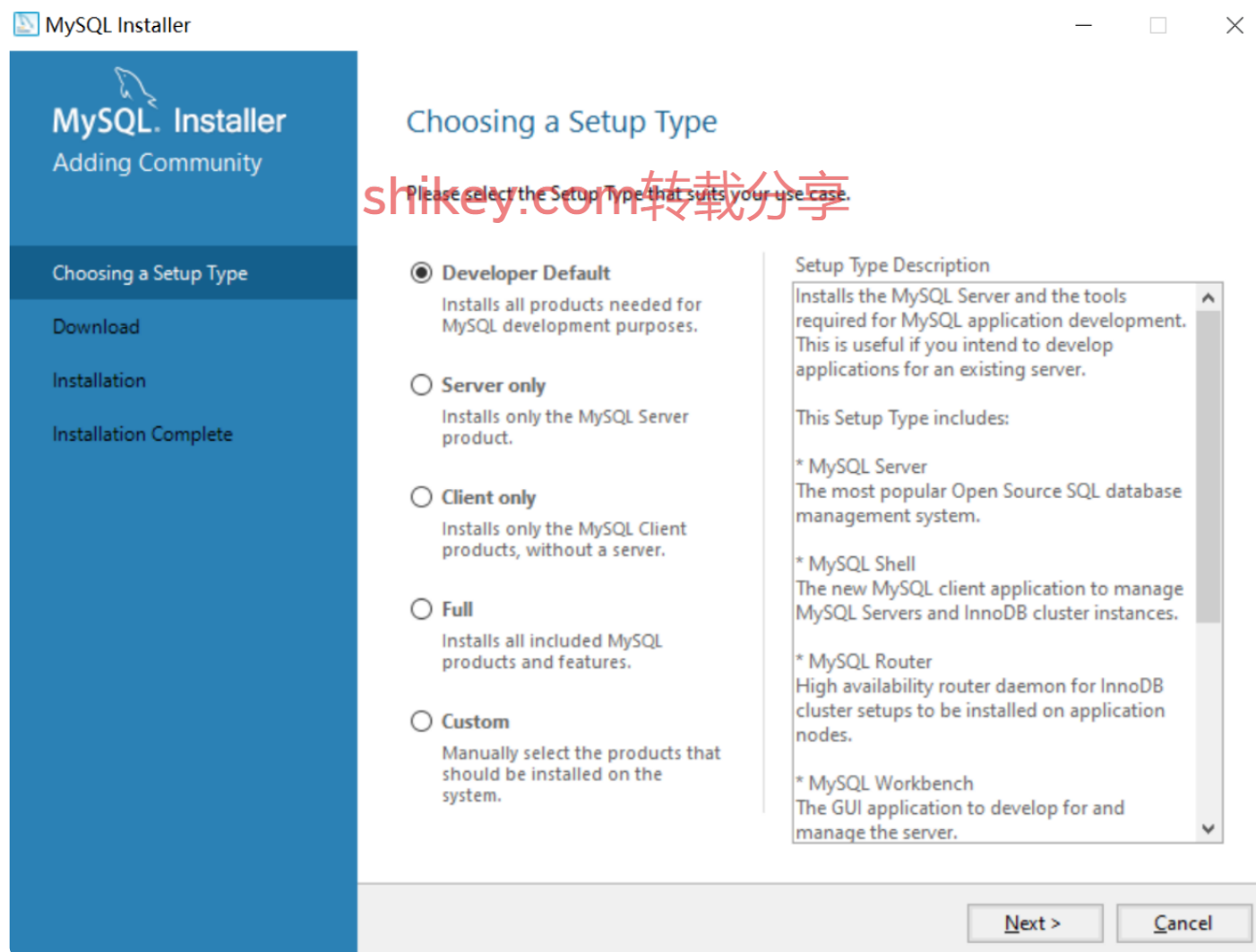
Select Operating System:
Microsoft Windows

Looking for the latest GA version?

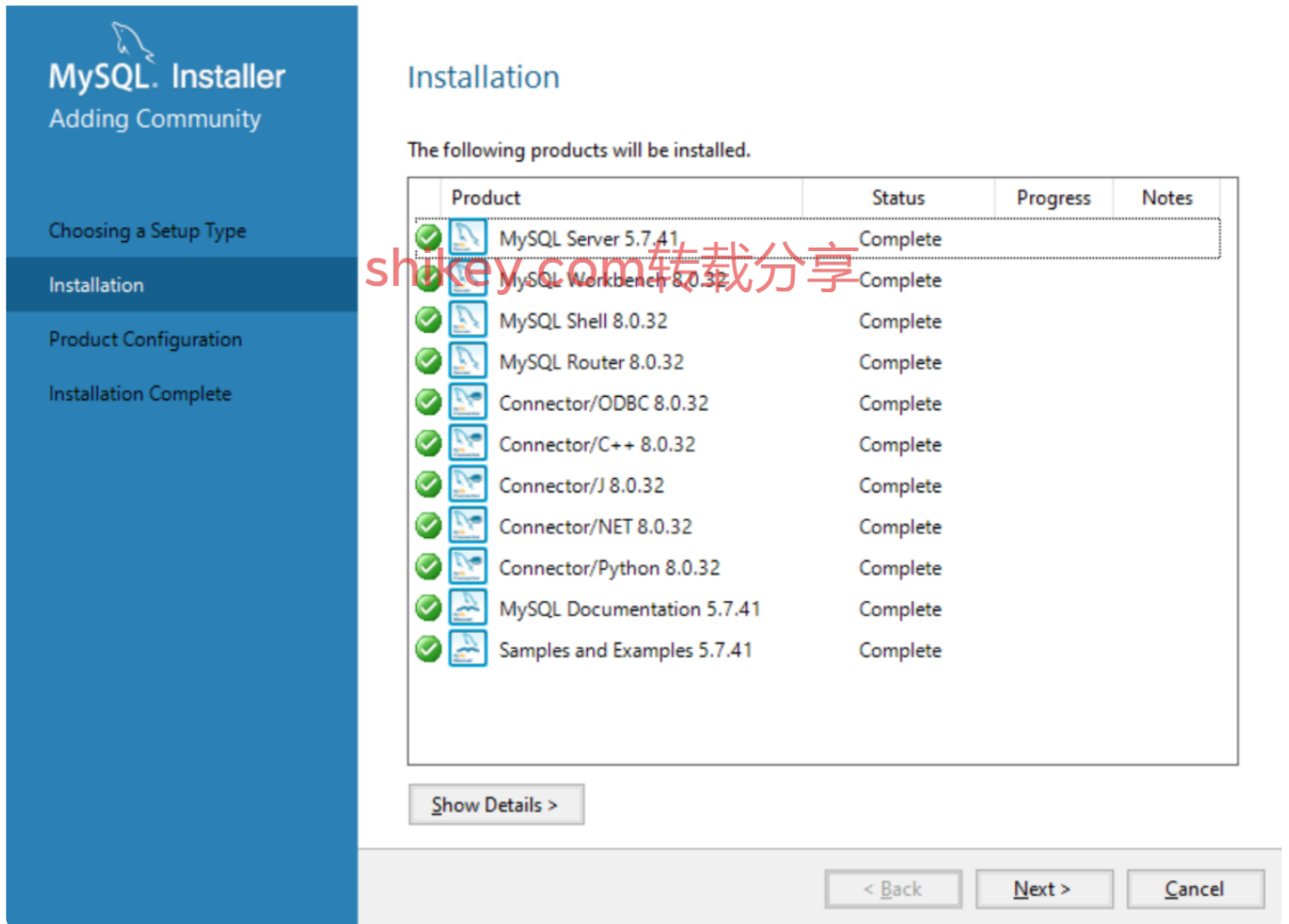
Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-5.7.41.0.msi)	5.7.41	2.4M	Download
MD5: 9588678a85bc34029f966fee7d0c5dab Signature			
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-5.7.41.0.msi)	5.7.41	517.1M	Download
MD5: 1ddcfd1b12fc12ab7a6a40fdb1188ef1 Signature			

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

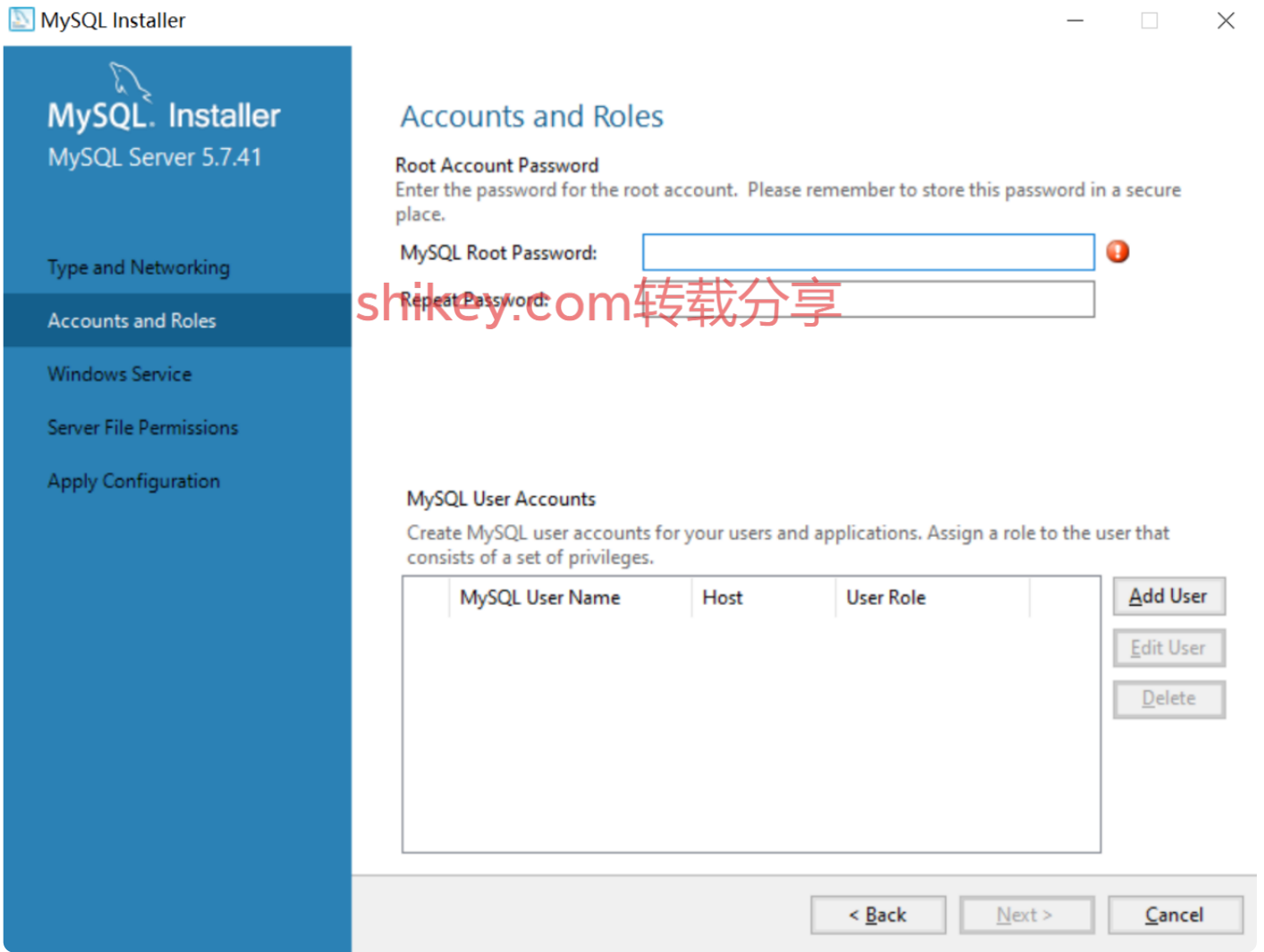
下载完成后，双击进入安装程序，这个时候，我们会进入到一个版本选择的界面。



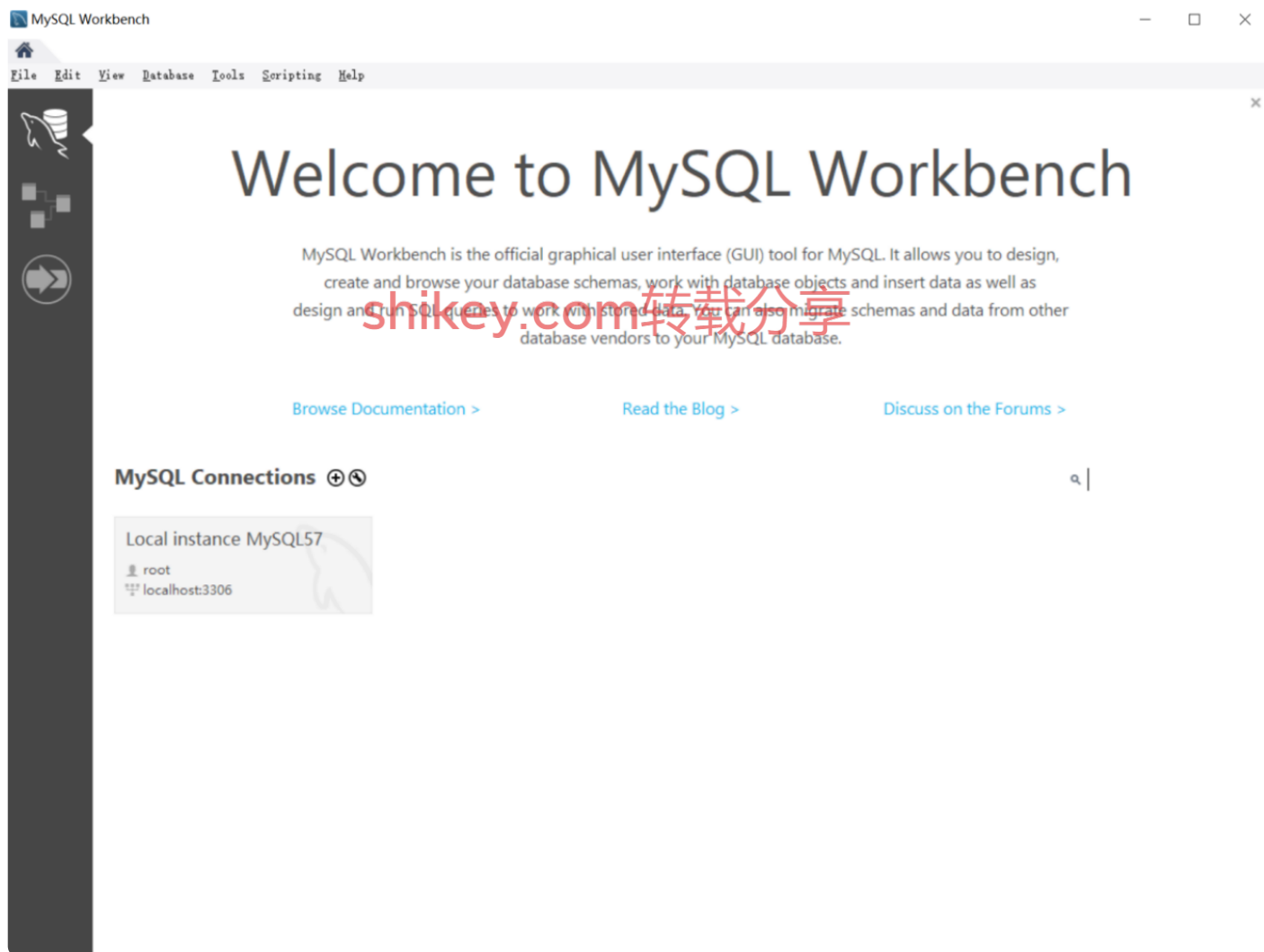
一般来讲，如果是用作本机开发，可以直接选择开发版，也就是 Developer Default，因为这个版本里除了包含 Server 之外，还包含了开发界面、远程调试的 shell 等等。如果是只用作服务的话，可以选择 Server only，这个时候，我们需要自己去下载一个其他的连接客户端用于 MySQL 数据库的连接。当然，如果你的内存和电脑的空间很大，也可以无脑选 Full。Custom 是自定义，就是根据你自己的需要进行组件的安装。在这里我选择 Developer Default，然后点击 Next。



当全部安装完成之后，一路点击 Next，来到下面这个界面。

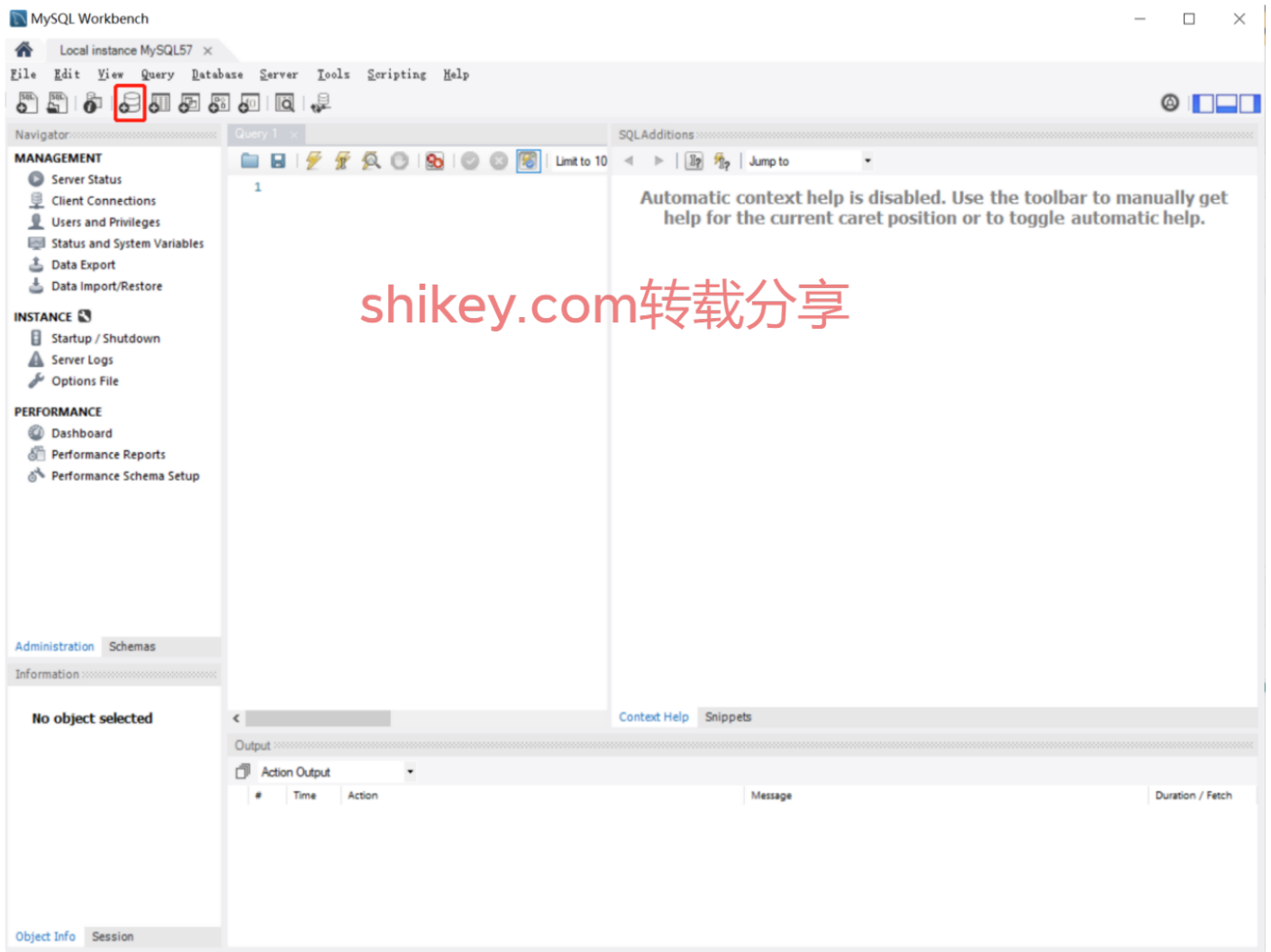


在这里我们需要为 MySQL 的 root 用户设置一个密码，由于是开发机，我直接设置成 123456，然后一路点击下一步，直到安装完成。如果是在服务器上，建议可以设置比较复杂的密码，并且在下面新建一个单独的 MySQL 账户。安装完成后，弹出如下界面。

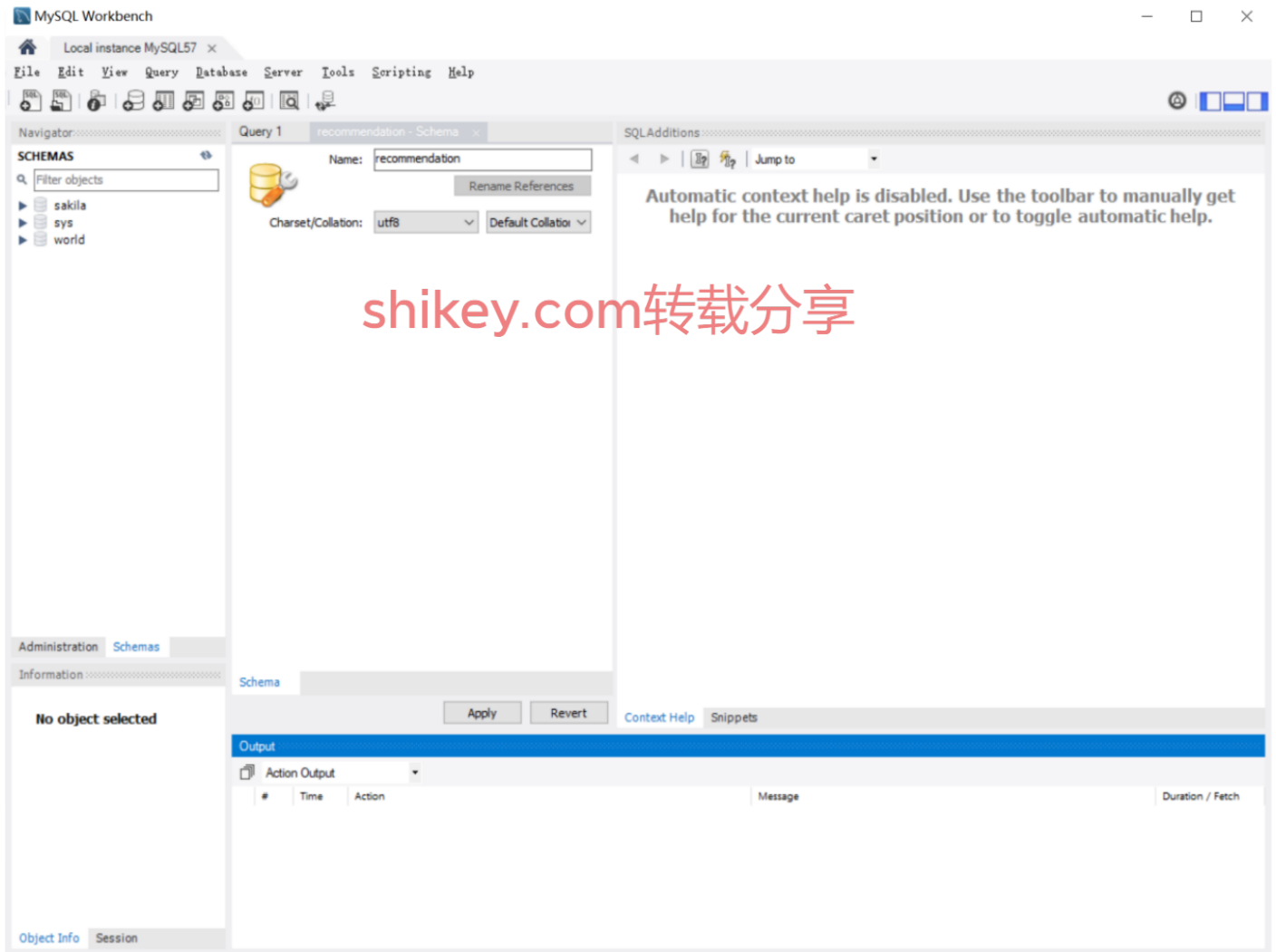


我们可以点击下面默认的连接，如果能正常进入，则说明安装服务正常。

接下来，我们尝试在这里新建一个数据库，名为 recommendation，我们可以直接在连接里点击上面的 Create a new schema in the connected Server 创建新的数据库，如图所示。



然后输入数据库名进行创建，这里我们用的数据库名叫做 recommendation。



这里必须要注意的一点就是，字符集一定要改为 utf8，否则中文会报错。然后点击 Apply 进行创建。

下面就是要做用户注册的服务端了，我把事情分成了四步。


1. 在我们的服务端程序里写 MySQL 连接的程序，然后连接数据库。
2. 建立数据表和实体，这里我们用的是 SQLAlchemy。
3. 写注册和登录的 service 服务，并将注册的用户信息存储到 MySQL 数据库中。
4. 使用 Postman 试一下。

下面我们一步一步来。

第一步是要连接 MySQL，我们要先在环境中安装 SQLAlchemy 库，在 Python 中想调用 MySQL 的话，还需要安装一个 PyMySQL 的库。


首先可以通过控制台使用如下命令激活虚拟环境。

shikey.com转载分享

 复制代码


```
1 activate recommendation-service
```

然后使用下面的命令来安装 PyMySQL 库。

 复制代码

```
1 pip install pymysql
```


然后再通过如下命令安装 SQLAlchemy 库。

 复制代码

```
1 pip install sqlalchemy
```

等待安装完成即可。

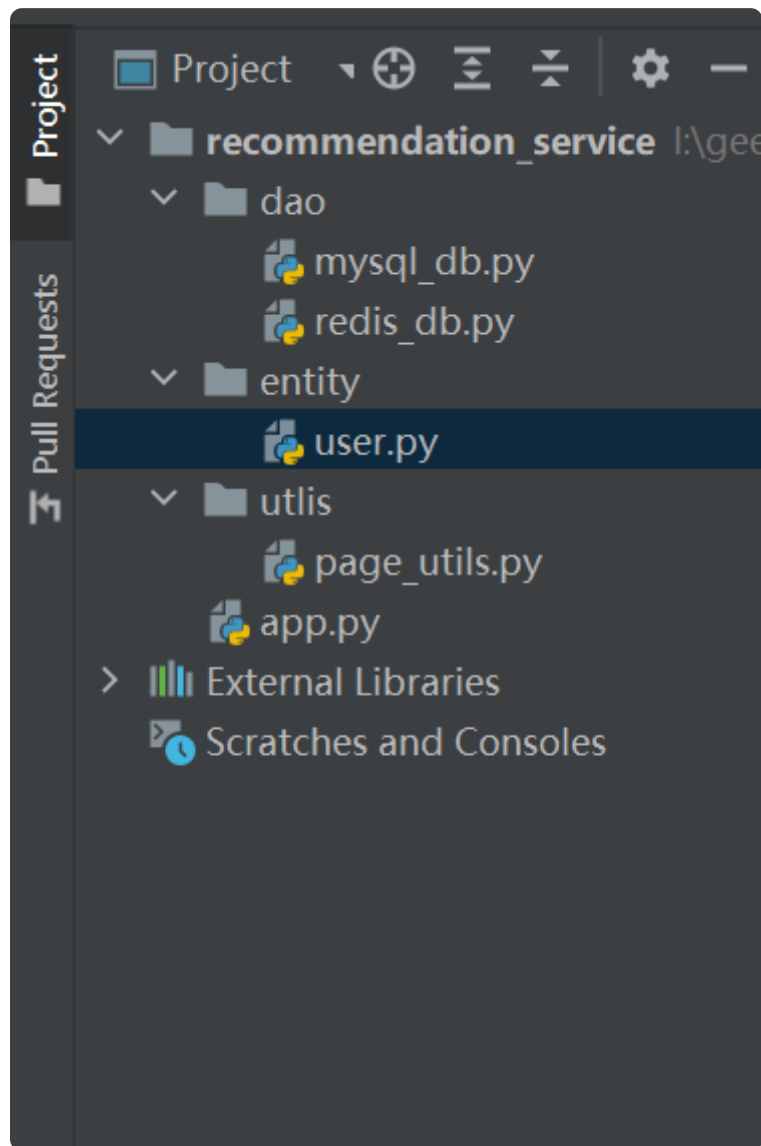
接着打开 Webservice 程序，即 recommendation-service 项目，然后在 dao 目录中新建 mysql_db.py 文件，并向里面写入下面的代码。

 复制代码

```
1 from sqlalchemy import create_engine
2 from sqlalchemy.orm import sessionmaker
3
4 class Mysql(object):
5     def __init__(self):
6         self.engine = create_engine('mysql+pymysql://root:123456@127.0.0.1:3306/r
7         self._DBSession = sessionmaker(bind=self.engine)
```

这段代码实际上和我们之前写的 Redis 数据库以及 MongoDB 数据库的连接方法非常类似，就是创建了一个连接的方法，然后将用户名、密码、Host、数据库名等信息加入进去，然后进行连接。

在 Python 中，向 MySQL 插入数据有一个非常简单的方法，就是利用 ORM 映射。我们在根目录下新建一个目录，名为 `entity`，专门用来存放各种实体。在这里我们需要存放的是用于 MySQL 的用户表的实体，因此，我们在 `entity` 目录下再建立一个名为 `user.py` 的文件，此时的目录结构如下。



然后再在里面写入以下代码。


```
1 from sqlalchemy import Column, String, Integer
2 from sqlalchemy.ext.declarative import declarative_base
3 from dao.mysql_db import Mysql
4
5
6 Base = declarative_base()
7 class User(Base):
8     __tablename__ = 'User'
9     id = Column(Integer(), primary_key=True)
10    username = Column(String(20))
11    password = Column(String(500))
12    nick = Column(String(20))
13    gender = Column(String(10))
14    age = Column(String(2))
15    city = Column(String(10))
16
17    def __init__(self):
18        mysql = Mysql()
19        engine = mysql.engine
20        Base.metadata.create_all(engine)
```

在这段代码中，我们引入了 sqlalchemy 和刚刚创建的 MySQL 连接程序，然后使用 declarative_base() 函数创建了一个 BaseModel 类，这个类的子类可以自动与一个表关联。

在 User 类中，我们首先使用 __tablename__ = 'User' 创建了一个表，表的名字叫做 User，然后再在 User 表里面定义了一些字段，如下表格所示。

字段名	类型	说明
id	Integer	id字段，表示唯一的索引
username	String (20)	用户名，长度为20字符
password	String (500)	密码，长度为500字符
nick	String (20)	昵称，长度为20字符
gender	String (10)	性别，长度为10字符
age	String (2)	年龄，长度为2字符
city	String (10)	城市，长度为10字符



这里需要注意的是，password 之所以设置了很长的字符，是因为我们要对密码进行加密。根据加密方式的不同长度也会不同，在这里我就设置的相对长一点。age 这个字段其实可以用 int 来表示，但是为了方便后面的处理，我这里直接使用了 String 类型。

最后，使用 `__init__` 函数将 MySQL 类和引擎进行初始化。

到现在为止，可以说是万事俱备只欠东风，接下来我们来写服务。服务依然写在 `app.py` 这个文件中，我们在这个文件中加入两个函数，分别是 `register()` 和 `login()`。

`register()` 函数。

复制代码


```
1 @app.route("/recommendation/register", methods=['POST'])
2 def register():
3     if request.method == 'POST':
4         req_json = request.get_data()
```

```

5     rec_obj = json.loads(req_json)
6     user = User()
7     user.username = rec_obj['username']
8     user.nick = rec_obj['nick']
9     user.age = rec_obj['age']
10    user.gender = rec_obj['gender']
11    user.city = rec_obj['city']
12    user.password = str(hashlib.md5(rec_obj['password'].encode()).hexdigest())
13    try:
14        mysql = Mysql()
15        sess = mysql._DBSession()
16        if sess.query(User.id).filter(User.username == user.username).count() > 0:
17            return jsonify({"code": 1000, "msg": "用户已存在"})
18        sess.add(user)
19        sess.commit()
20        sess.close()
21
22        result = jsonify({"code": 0, "msg": "注册成功"})
23        return result
24    except Exception as e:
25        print(str(e))
26        return jsonify({"code": 2000, "msg": "error"})

```

login() 函数。

 复制代码

```

1 @app.route("/recommendation/login", methods=['POST'])
2 def login():
3     if request.method == 'POST':
4         req_json = request.get_data()
5         rec_obj = json.loads(req_json)
6         username = rec_obj['username']
7         password = str(hashlib.md5(rec_obj['password'].encode()).hexdigest())
8     try:
9         mysql = Mysql()
10        sess = mysql._DBSession()
11        res = sess.query(User.id).filter(User.username == username, User.password
12        if res.count() > 0:
13            for x in res.all():
14                data = {"userid": str(x[0])}
15                info = jsonify({"code": 0, "msg": "登录成功", "data": data})
16                return info
17        else:
18            return jsonify({"code": 1000, "msg": "用户名或密码错误"})
19    except Exception as e:

```

```
20         print(str(e))
21         return jsonify({"code": 2000, "msg": "error"})
```

我们来解析下这两个函数。

这两个函数分别是注册和登录。在注册函数中，首先确定注册所接收的请求为 POST 请求，于是，我们从前端接收注册时所传来的参数。参数有用户名、密码、昵称、年龄、性别和城市，其中密码使用 MD5 的方式进行加密，然后把所有接收到的数据传到 MySQL 数据库中。

在传入 MySQL 数据库之前，我们首先会通过用户名来进行查询，看看数据库里是不是已经有这个用户名了。如果查到的用户名数量大于 0，说明已经有这个用户了，这个时候就会返回用户已存在，否则就正常注册提交提示注册成功。这个时候，我们所执行的 `session.commit()` 方法就会把用户信息插入到 MySQL 数据库中，当然，还会出现服务端报错的现象，这个时候就要抛出异常。

再来说说登录函数。登录函数实际上就是利用注册函数插入数据库中的内容进行校验，将用户名和密码在插入的 `user` 表中进行查询，匹配成功则正常登录，否则就会提示用户名和密码错误。

在登录中要注意，我们会同时校验用户名和密码，由于 MD5 是不可逆的加密方式，因此校验方法是将用户输入的密码再一次进行 MD5 加密，然后再和数据库中的密码进行比对。如果比对成功，并且用户名比对也是成功的，则说明登录正常，否则就是登录异常。

运行程序后，得到如下界面。



```
Run: app
C:\Users\admin\.conda\envs\recommendation-service\python.exe I:\geekbang\recommendation_service\app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:10086
* Running on http://192.168.1.18:10086
Press CTRL+C to quit
```

接下来，用 Postman 请求一下注册端口，试试能否注册成功。

POST 127.0.0.1:10086/recommendation/login Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "username": "huanghongbo",
3   "password": "huanghongbo123"
4 }
```

shikey.com转载分享

Body Cookies Headers (8) Test Results Status: 200 OK Time: 23 ms

Pretty Raw Preview JSON

```
1 {
2   "code": 0,
3   "data": {
4     "userid": "1"
5   },
6   "msg": "登录成功"
7 }
```

S 中 中 中 中 中 中 中 中

可以发现，现在已经能够正常地使用接口进行注册了。接下来同样的参数，我们再提交一次看看会发生什么？

The screenshot shows a REST client interface with a POST request to `127.0.0.1:10086/recommendation/register`. The request body is a JSON object with the following fields: `username`, `password`, `nick`, `age`, `gender`, and `city`. The response status is `200 OK` with a time of `27 ms`. The response body is a JSON object with `code` `1000` and `msg` `"用户已存在"`.

```
POST 127.0.0.1:10086/recommendation/register
```

Authorization Headers (1) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "username": "huanghongbo",
3   "password": "huanghongbo123",
4   "nick": "黄鸿波",
5   "age": "18",
6   "gender": "male",
7   "city": "zhuhai"
8 }
```

Body Cookies Headers (8) Test Results Status: 200 OK Time: 27 ms

Pretty Raw Preview JSON

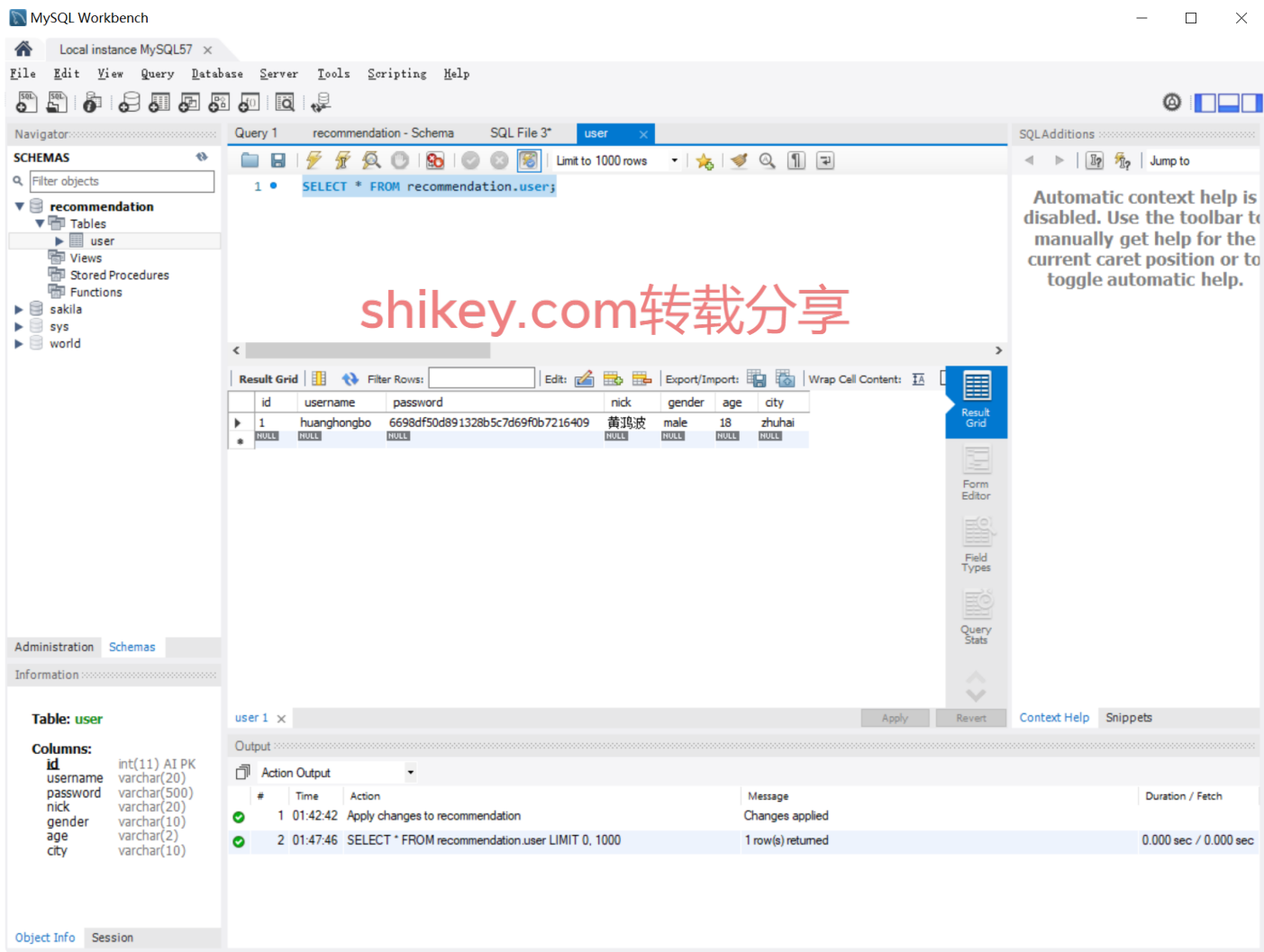
```
1 {
2   "code": 1000,
3   "msg": "用户已存在"
4 }
```

当我们再点击一次，会发现服务端已经返回用户已存在，说明注册是成功的。我们来看看数据库里有没有一条新的数据增加，可以使用如下命令来查询一下。

```
1 SELECT * FROM recommendation.user;
```

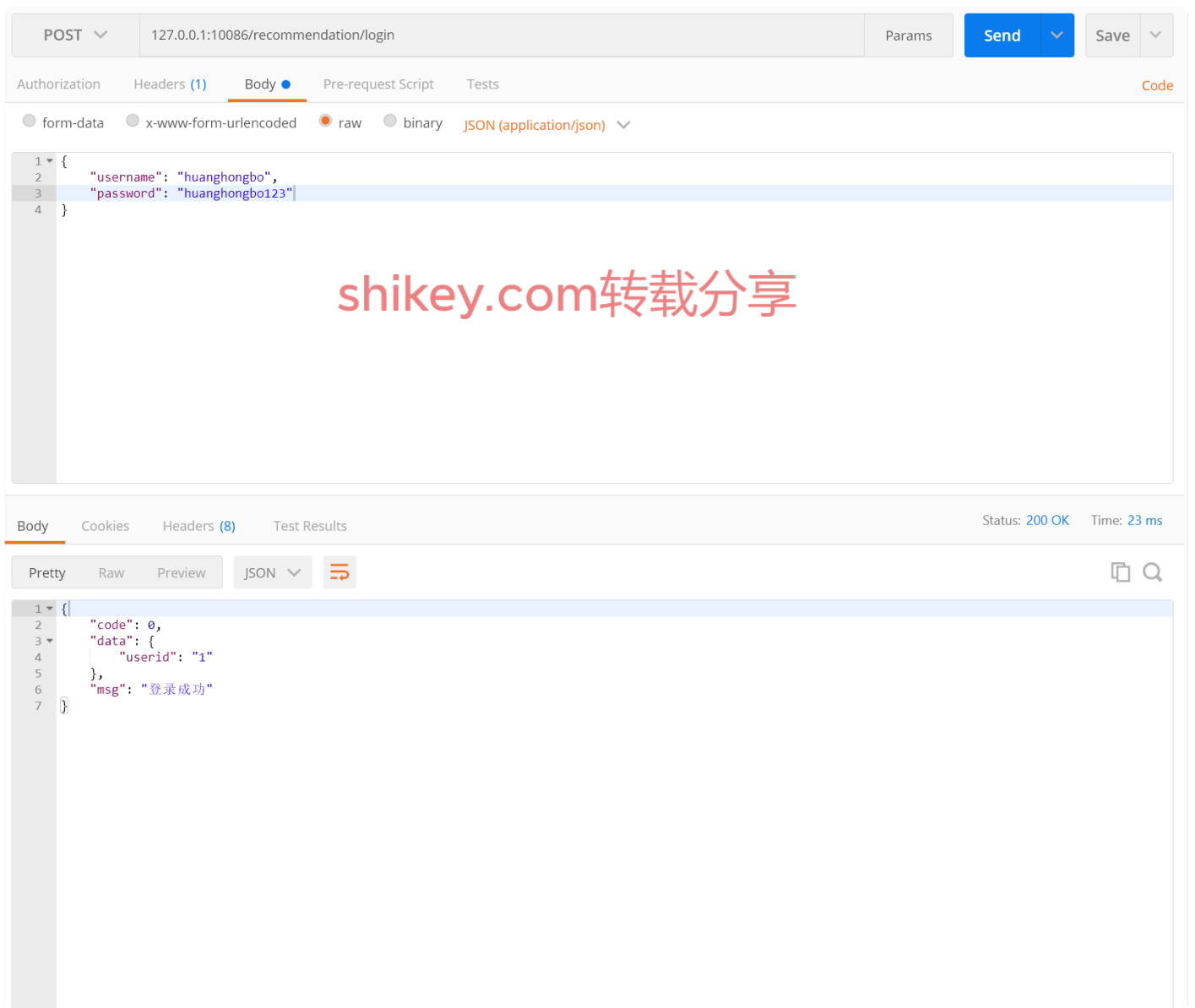
复制代码

得到了下面的内容。



可以看到，用户数据已经插入到数据库中了，并且已经是我们希望得到的值了。

接下来再试试登录接口，同样是利用 Postman 来测试一下。



调用一下登录接口，输入用户名和密码，返回登录成功。这里需要注意的是，在登录时返回了一个 `userid`，这样以后再发其他请求的时候，就可以用 `userid` 来作为唯一的标识了。

到这里，注册和登录就已经全部搞定了。

虽然这只是一个用户的基本信息，但是也确实是一个最基本的用户画像了。在接下来的课程中，我们会逐渐对这个用户画像进行内容的填充，使其更加丰富。

总结

到这里，我们本节课的内容就已经全部讲完了，我来对这节课的内容做一个总结，学完本节课你应该知道下面五个要点。

1. 什么是用户画像，以及用户画像在推荐系统中的作用和地位是什么？
2. 对用户行为分析的方法有很多种，比较常见的有行为路径分析、行为偏好分析和搜索行为分析。
3. 熟悉如何使用 Python 搭建一个 Webservice 服务，通过注册的方式收集最基础的用户画像。
4. 知道如何安装 MySQL 数据库，以及如何利用 Python 来操作它。
5. 我们可以通过 Python 来创建 MySQL 数据库、插入数据，并且写一个 Webservice 接口，使用 Postman 进行模拟调用。

shikey.com转载分享

课后题

学完本节课，给你布置两个小作业。

1. 完成本节课的环境搭建以及代码编写。
2. 想一想，还可以收集哪些用户信息？欢迎你在评论区留言，也可以通过代码实现它们。

欢迎你在留言区与我交流讨论，如果这节课对你有帮助，也欢迎你推荐给朋友一起学习。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (2)



Geek_ccc0fd

2023-05-23 来自广东

"将每个用户从开始到结束的所有行为保存成一条记录，然后利用一些统计学原理进行行为分析，从而找到想要挖掘的点" 老师这块可以举例详细说说嘛，基于哪些统计学原理，能够挖掘什么数据





peter

2023-05-22 来自北京

Q1: 用户画像, 从数据结构的角度看, 是怎么存储的?

Q2: 文中提到的“webservice”是什么意思?

是普通的“网站服务”, 没有特殊含义吗? 还是指一种特殊的协议? 我记得有一种协议叫“webservice”, 好像挺复杂的。

共 1 条评论 >

shikey.com 转载分享