链滴

接触企业级框架 SpringBoot(二)

作者：Gaoshengyue

原文链接：https://ld246.com/article/1589362997661

来源网站：链滴

概述

很久没更了，这段时间各方面都比较繁忙，没拿出那么多的时间去更新一些文档。本站的本意是从基上去理解一些框架，提供一些框架的基础构建思路。很多框架文档没有写那么多复杂的东西，也是为可以让大家快速应用。可能一些初学者会对里面的一些概念模糊，希望大家都可以找一些方面的基础档去学习下，这里不说太多影响其他人思路的东西，因为本身也看过太多引人走弯路的文档了。。

回顾

上一篇文档讲解了一些SpringBoot的基本结构，从项目启动、目录到入口、路由、表模型映射等等这篇就舒展开来对具体业务框架有一个简单的描述。

业务结构

DataRepository

DAO这里的概念就直接复制一些资料上的。

DAO(Data Access Object) 数据访问对象是一个面向对象的数据库接口。

总结一下，用大白话来讲，springboot面向接口，DataRepository数据的提供，也就是数据的增删改，可以封装成一个个的数据接口进行调用。

例如我们想要查询一个用户，有很多条件，那么这一条查询就可以封装为一个数据接口

```
   @Query(value = "select * from user where username= :username or real_name= :real_name or phone_number= :phone_number or id_number= :id_number", nativeQuery = true)
    List<User> findUser(@Param("username") String username, @Param("real_name") String real_name, @Param("phone_number") String phone_number, @Param("id_number") String id_number);
```

在框架中，我们想要调用这个接口去查询用户的数据，只需要调用接口就可以。定义DAO代码如下

```
package com.pyramid.loansupermarket.modelRepository;

import com.pyramid.loansupermarket.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.List;
//用户数据接口定义
public interface UserRepository extends JpaRepository<User, Integer> {
    //sql编写，数据操作编写，可扩展
    @Query(value = "select * from user where username= :username or real_name= :real_name or phone_number= :phone_number or id_number= :id_number", nativeQuery = true)
    List<User> findUser(@Param("username") String username, @Param("real_name") String real_name, @Param("phone_number") String phone_number, @Param("id_number") String id_number);

    @Query(value = "select * from user where id = :id", nativeQuery = true)
    User findOneUserById(@Param("id") Integer id);
```

```
}
```

然而使用封装好的数据接口，我们需要把定义好的接口注入到我们的Service中。

```
@Autowired
private UserRepository userRepository;
```

那么，Service的作用是什么?

Service

Service在SpringBoot中的定义是服务接口，后端普遍的大白话解释，就是实际业务逻辑接口。

这里的Service定义主要是为了标准化,如果觉得自己写的代码质量无所谓的话，直接写在Controller没什么问题，但是代码结构会非常乱，而且后期扩展困难。

```
package com.pyramid.loansupermarket.Service;

import com.alibaba.fastjson.JSONObject;
import com.pyramid.loansupermarket.ServiceRepository.UserAdminRepository;
import com.pyramid.loansupermarket.messageLib.UserInsertFaieldMessage;
import com.pyramid.loansupermarket.messageLib.UserInsertSucessMessage;
import com.pyramid.loansupermarket.model.User;
import com.pyramid.loansupermarket.modelRepository.UserRepository;
import com.pyramid.loansupermarket.status.UserResultStatus;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
//UserAdmin用户管理业务接口实体
@Service
public class UserAdmin implements UserAdminRepository {

    @Autowired
    private UserRepository userRepository;

    public UserResultStatus register(JSONObject jsonObject) {
        UserResultStatus rs_obj = new UserResultStatus();
        User user_obj;
        user_obj = createUserObj(jsonObject);
        List<User> user_list;
        user_list = getUser(user_obj);
        if (user_list.size() <= 0) {
            UserInsertSucessMessage message_obj = new UserInsertSucessMessage();
            rs_obj.setUser(userRepository.save(user_obj));
            rs_obj.setMessage(message_obj.message);
            rs_obj.setStatusCode(message_obj.code);
        } else {
            UserInsertFaieldMessage message_obj = new UserInsertFaieldMessage();
            rs_obj.setMessage(message_obj.message);
            rs_obj.setStatusCode(message_obj.code);
        }
        return rs_obj;
```

```
    }


    //用户操作方法
    public User createUserObj(JSONObject jsonObject) {
        User userObj = new User();
        userObj.setUsername(jsonObject.getString("username"));
        userObj.setPassword(jsonObject.getString("password"));
        userObj.setPhoneNumber(jsonObject.getString("phoneNumber"));
        userObj.setIdNumber(jsonObject.getString("idNumber"));
        userObj.setRealName(jsonObject.getString("realName"));
        return userObj;
    }

    public List<User> getUser(User user_obj) {
        List<User> user_list;
        user_list = userRepository.findUser(user_obj.getUsername(), user_obj.getRealName(), use
_obj.getPhoneNumber(), user_obj.getIdNumber());
        return user_list;
    }


}
```

这里的Service已经注入了数据接口，可以我们直接调用。

代码中已经定义好了几个方法，这里的Service还没发直接被控制器调用，我们需要定义好业务接口库属性。

ServiceRepository

这里就没什么好说了，定义好我们写好的业务逻辑Service,注入到控制器里面使用

```
package com.pyramid.loansupermarket.ServiceRepository;

import com.alibaba.fastjson.JSONObject;
import com.pyramid.loansupermarket.model.User;
import com.pyramid.loansupermarket.status.UserResultStatus;
import org.springframework.stereotype.Repository;

import java.util.List;
// UserAdmin用户管理接口定义
@Repository("userAdminRepository")
public interface UserAdminRepository {

    UserResultStatus register(JSONObject jsonObject);

    User createUserObj(JSONObject jsonObject);

    List<User> getUser(User user_obj);

}
```

在控制器中的使用

```
package com.pyramid.loansupermarket.ApiLib;


import com.alibaba.fastjson.JSONObject;
import com.pyramid.loansupermarket.ServiceRepository.UserAdminRepository;
import com.pyramid.loansupermarket.model.User;
import com.pyramid.loansupermarket.modelRepository.UserRepository;
import com.pyramid.loansupermarket.status.UserResultStatus;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;


@RestController
@ResponseBody
@RequestMapping("/user")
public class UserApi {
    @Autowired
    private UserRepository respository;

    @RequestMapping(value = "/admin", method = RequestMethod.GET)
    public List<User> userList() {
        return respository.findAll();
    }

    @Autowired
    private UserAdminRepository userAdminRepository;
    @RequestMapping(value = "/admin", method = RequestMethod.POST)
    public UserResultStatus InsertUser(@RequestBody JSONObject jsonObject) {
        UserResultStatus userResultStatus;
        userResultStatus=userAdminRepository.register(jsonObject);
        return userResultStatus;
    }
}
```

这里数据接口和业务接口都有应用。

以下是详细目录结构，直接传到github了，还是很初级的一个基本结构，有兴趣的可以看一下

https://github.com/Gaoshengyue/SpringBootDemo