车同轨科技有限公司接口开发规范及细则

> 当前版本：v1.0.0-alpha.1
> 该接口开发规范及细则为车同轨科技有限公司为配合开发项目的推进，所设计的接口规范初版。
> 以下内容具体可参考示例项目：https://github.com/hfangjian/template.git

一、接口返回参数

1. status : 代表返回状态，0：成功，1：失败。
2. msg : 返回时附带的消息。
3. data : 返回的主体数据内容

```
//例：登录成功时的返回示例：
//示例项目中访问路径 /user/login.do,
//访问参数：username = admin，password = admin
{
  "status": 0,
  "msg": "登录成功",
  "data":{
      "id": 1,
      "username": "admin",
      "password": null,
      "email": "admin@happymmall.com",
      "phone": "13800138000",
      "question": "问题",
      "answer": "答案",
      "role": 1,
      "create_time": 1478422605000,
      "update_time": 1491305256000
  },
  "success": true
}

//例：登录失败时的返回示例：
{
  "status": 1,
  "msg": "密码错误",
  "data": null,
  "success": false
}

//例：成功获取记录列表时的返回示例：
//示例项目中访问路径 /test/page_help.do,
{
"status": 0,
"msg": null,
"data":[
  {
    "id": 100001,
    "parentId": null,
    "name": "家用电器",
    "status": true,
    "sortOrder": null,
    "createTime": null,
    "updateTime": null
  },
  {
    "id": 100002,
    "parentId": null,
    "name": "数码3C",
    "status": true,
    "sortOrder": null,
    "createTime": null,
    "updateTime": null
  },
  {
    "id": 100003,
    "parentId": null,
    "name": "服装箱包",
    "status": true,
    "sortOrder": null,
```

```
    "createTime": null,
    "updateTime": null
  }
],
"success": true
}
```

二、通用响应类

> 具体源码可参见示例项目

```
@JsonSerialize(include =  JsonSerialize.Inclusion.NON_NULL)
//保证序列化json的时候,如果是null的对象,key也会消失
public class ServerResponse<T> implements Serializable{

    private int status;
    private String msg;
    private T data;

    private ServerResponse(int status){
        this.status = status;
    }
    private ServerResponse(int status,T data){
        this.status = status;
        this.data = data;
    }

    private ServerResponse(int status,String msg,T data){
        this.status = status;
        this.msg = msg;
        this.data = data;
    }

    private ServerResponse(int status,String msg){
        this.status = status;
        this.msg = msg;
    }

    @JsonIgnore
    //使之不在json序列化结果当中
    public boolean isSuccess(){
        return this.status == ResponseCode.SUCCESS.getCode();
    }

    public int getStatus(){
        return status;
    }
    public T getData(){
        return data;
    }
    public String getMsg(){
        return msg;
    }

    //返回 成功状态码（ status : 0 ）
    public static <T> ServerResponse<T> createBySuccess(){
        return new ServerResponse<T>(ResponseCode.SUCCESS.getCode());
    }

    //返回 成功状态码、附带信息 （ status : 0 , msg : ... ）
    public static <T> ServerResponse<T> createBySuccessMessage(String msg){
        return new ServerResponse<T>(ResponseCode.SUCCESS.getCode(),msg);
    }

    //返回 成功状态码、数据内容 （ status : 0 , data : ... ）
    public static <T> ServerResponse<T> createBySuccess(T data){
        return new ServerResponse<T>(ResponseCode.SUCCESS.getCode(),data);
    }

    //返回 成功状态码、附带信息、数据内容 （ status : 0 , msg : ... , data : ... ）
    public static <T> ServerResponse<T> createBySuccess(String msg,T data){
        return new ServerResponse<T>(ResponseCode.SUCCESS.getCode(),msg,data);
    }

    ////返回 失败状态码（ status : 1 )
```

```
    public static <T> ServerResponse<T> createByError(){
        return new ServerResponse<T>(ResponseCode.ERROR.getCode(),ResponseCode.ERROR.getDesc());
    }

    //返回 失败状态码、附带错误信息 （ status : 1 , msg : ... ）
    public static <T> ServerResponse<T> createByErrorMessage(String errorMessage){
        return new ServerResponse<T>(ResponseCode.ERROR.getCode(),errorMessage);
    }

    //返回 自定义的错误状态码、附带错误信息 （ status : errorCode , msg : ... ）
    public static <T> ServerResponse<T> createByErrorCodeMessage(int errorCode,String errorMessage){
        return new ServerResponse<T>(errorCode,errorMessage);
    }

}
```

上述ServerResponse类中的响应状态码在枚举类ResponseCode定义：

```
public enum ResponseCode {
    //0:成功;  描述:SUCCESS
    SUCCESS(0,"SUCCESS"),

    //1:错误（即失败）;  描述:ERROR
    ERROR(1,"ERROR"),

    //2: 非法参数; 描述: ILLEGAL_ARGUMENT
    ILLEGAL_ARGUMENT(2,"ILLEGAL_ARGUMENT");

    //10: 未登录; 描述: NEED_LOGIN
    NEED_LOGIN(10,"NEED_LOGIN"),

    private final int code;
    private final String desc;

    ResponseCode(int code,String desc){
        this.code = code;
        this.desc = desc;
    }

    public int getCode(){
        return code;
    }
    public String getDesc(){
        return desc;
    }
}
```

通用响应类的使用示例：

```
    //在某用户登录接口的实现类中:
    @Autowired
    private UserMapper userMapper;

    @Override
    public ServerResponse<User> login(String username, String password) {
        int resultCount = userMapper.checkUsername(username);
        //在用户登录时，当数据库中无此用户名信息时，返回登录失败信息
        if(resultCount == 0){
            return ServerResponse.createByErrorMessage("用户名不存在");
        }
        //密码登录 MD5
        String md5Pssword = MD5Util.MD5EncodeUtf8(password);
        User user = userMapper.selectLogin(username,md5Pssword);
        //密码错误时，返回失败信息
        if(user == null){
            return ServerResponse.createByErrorMessage("密码错误");
        }
        //登录成功，把密码信息去除
        user.setPassword(null);
        //登录成功后，返回成功信息和主体内容（登录用户的用户信息）
        return ServerResponse.createBySuccess("登录成功",user);
    }
```