# 说明

版权归属：鑫学猿公司

微信：yuanhouduanyanfa

Java开发工程师转行辅导体系,保障拿到java工程师offer

图形用户界面, 文本, 应用程序

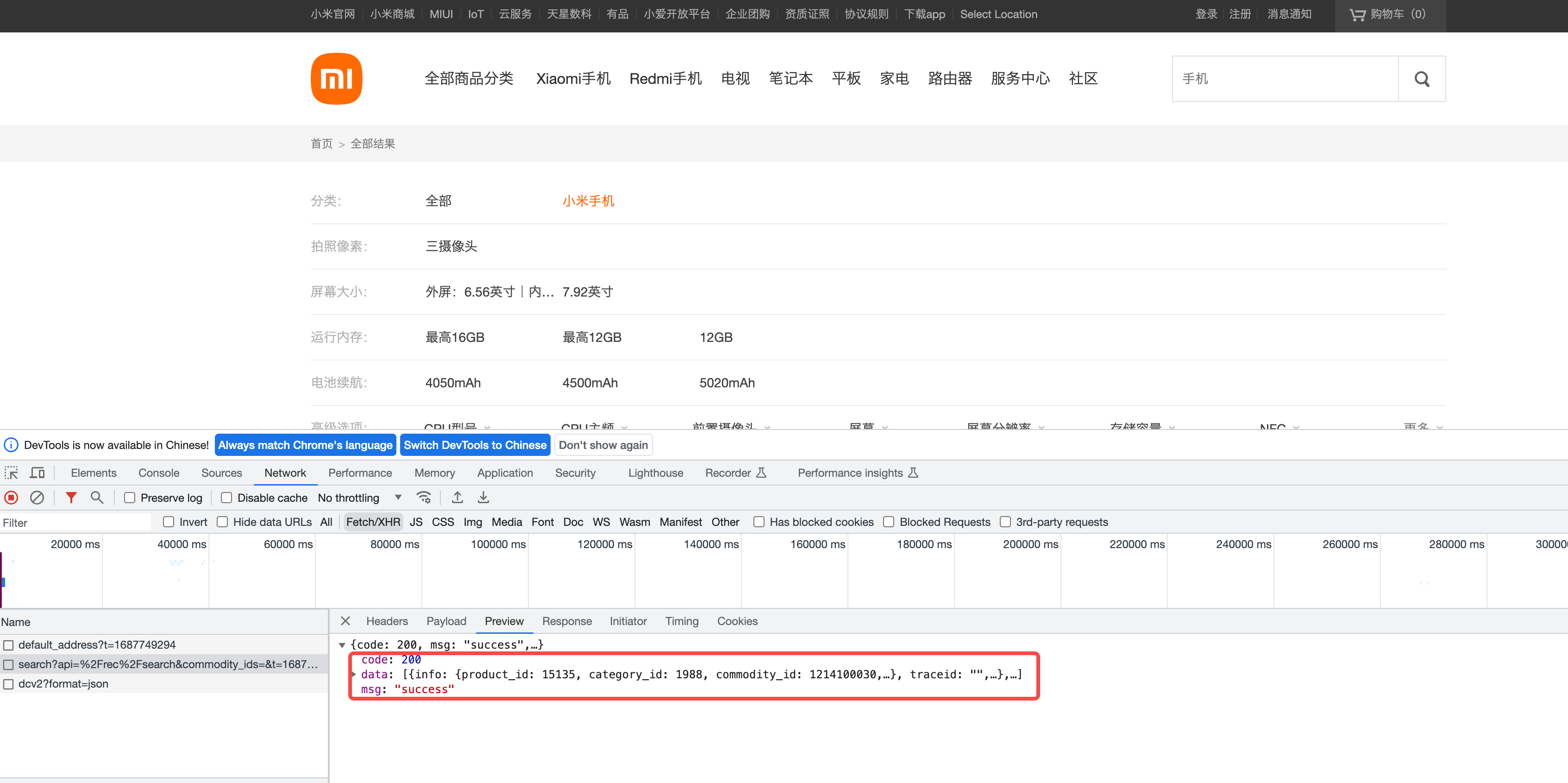
描述已自动生成

**Springboot项目返回数据统一封装**

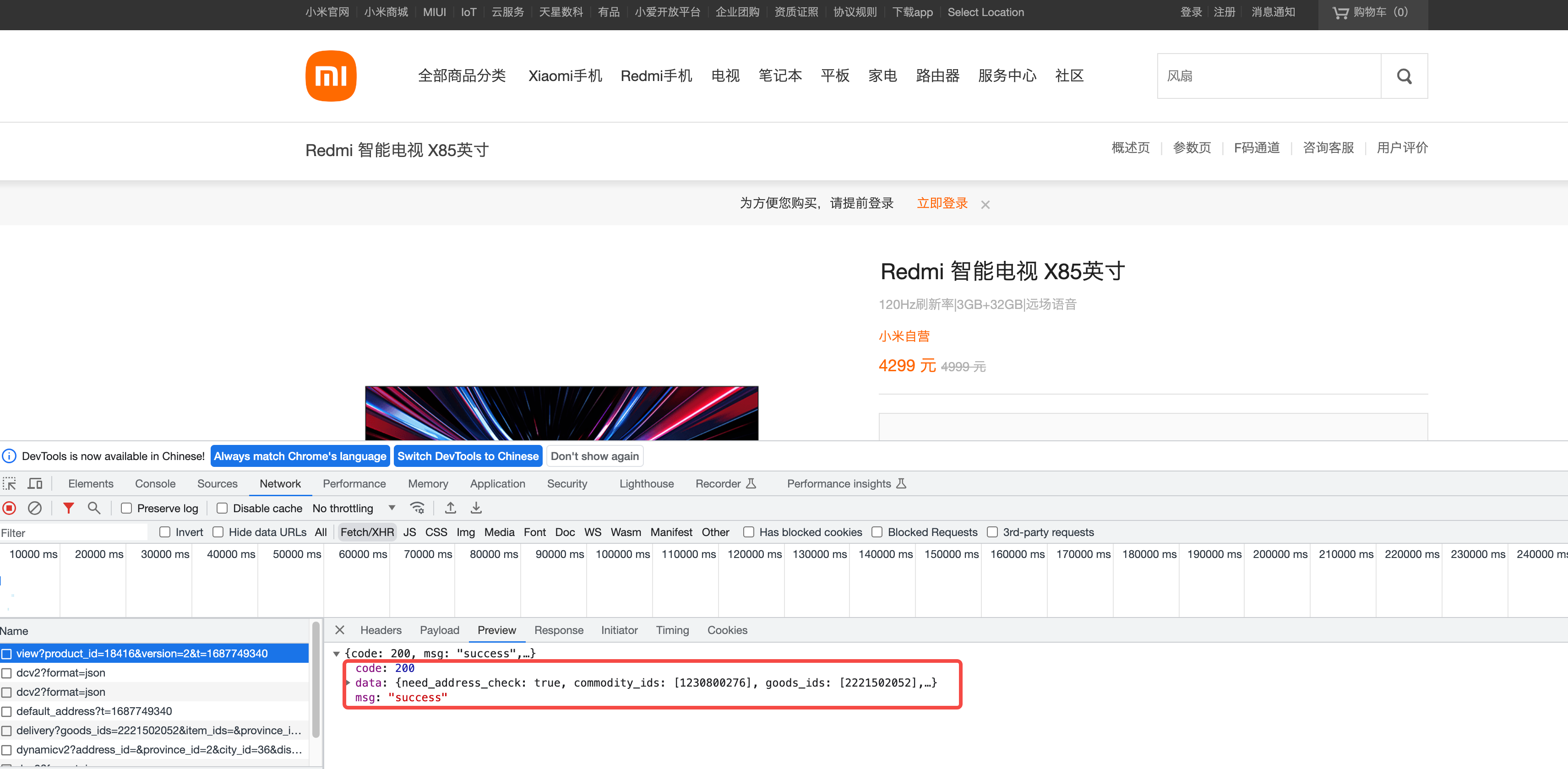
**背景**

不论是RPC还是http的交互方式，我们会发现在真实的网站中，每个请求的返回都是有固定格式的，比如小米商城的列表页和详情页

手机列表搜索页



电视详情页



**为什么？**

* 提高代码的可读性和维护性
* 便于前后端人员之间的协同开发和交流，降低出错的风险
* 如果项目中使用了多个框架或技术栈，通过统一响应格式也能够避免代码在不同框架和技术栈之间的差异

**教程**

1. 统一错误码

|  |
| --- |
| Java public interface ErrorCode {  /\*\*  \* 获取结果码  \*  \* @return  \*/  String getCode();   /\*\*  \* 获取结果码描述信息  \*  \* @return  \*/  String getMessage(); } |

|  |
| --- |
| Java public enum ErrorEnum implements ErrorCode {  /\*\*  \* 成功  \*/  SUCCESS("SUCCESS", "成功"),   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*系统异常\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  /\*\*  \* 请求失败(外域请求等)  \*/  REQUEST\_FAIL("REQUEST\_FAIL", "请求失败"),  /\*\*  \* 系统异常  \*/  SYSTEM\_ERROR("SYSTEM\_ERROR", "系统异常"),  /\*\*  \* 操作超时  \*/  OP\_TIMEOUT("OP\_TIMEOUT", "操作超时,请重试"),  /\*\*  \* 操作冲突(乐观锁、并发)  \*/  OP\_CONFLICT("OP\_CONFLICT", "操作冲突"),  /\*\*  \* 数据库执行错误  \*/  DB\_ERROR("DB\_ERROR", "数据库执行错误"),   //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*业务类异常\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  /\*\*  \* 参数错误  \*/  PARAMETER\_ERROR("PARAMETER\_ERROR", "参数错误"),  /\*\*  \* 没有权限  \*/  NO\_PRIVILEGE("NO\_PRIVILEGE", "没有权限"),  /\*\*  \* 数据异常(数据校验不通过等)  \*/  DATA\_ERROR("DATA\_ERROR", "数据异常"),  /\*\*  \* 数据不存在(数据校验等)  \*/  DATA\_NOT\_FOUND("DATA\_NOT\_FOUND", "数据不存在"),  /\*\*  \* 数据已存在(数据校验等)  \*/  DATA\_EXIST("DATA\_EXIST", "数据已存在");    /\*\*  \* 结果码  \*/  private String code;   /\*\*  \* 结果信息  \*/  private String message;   ErrorEnum(String code, String message) {  this.code = code;  this.message = message;  }   @Override  public String getCode() {  return this.code;  }   @Override  public String getMessage() {  return this.message;  } } |

1. 统一返回类

|  |
| --- |
| Java @Data public class Result<T> {  private String code;  private String msg;  private T data;   @JsonIgnore  public boolean getSuccess() {  return this.code.equals(ErrorEnum.SUCCESS.getCode());  }   private Result(String code, String msg, T data) {  this.code = code;  this.msg = msg;  this.data = data;  }   private Result(ErrorEnum errorEnum, T data) {  this.code = errorEnum.getCode();  this.msg = errorEnum.getMessage();  this.data = data;  }   public static <T> Result<T> Result(ErrorEnum errorEnum, T data) {  return new Result<>(errorEnum, data);  }   public static <T> Result<T> Result(String code, String msg, T data) {  return new Result<>(code, msg, data);  } } |

1. Service和DAO层

|  |
| --- |
| Java @Mapper public interface UserMapper {  /\*\*  \* 根据姓名查找  \*  \* @param name  \* @return  \*/  UserDO findUserByName(String name);   /\*\*  \* 根据id查找  \*  \* @param id  \* @return  \*/  @Select("SELECT \* FROM user WHERE id = #{id}")  UserDO findUserById(@Param("id") Long id);   /\*\*  \* 查询所有  \*  \* @return  \*/  @Select("SELECT \* FROM user")  List<UserDO> findAll();   /\*\*  \* 插入用户  \*  \* @param name  \*/  @Insert("INSERT INTO user (name) VALUES (#{name})")  void insertUser(@Param("name") String name); } |

|  |
| --- |
| Java public interface UserService {  void createUser(String name);   UserDTO queryUserByName(String name);   List<UserDTO> queryAll(); } |

|  |
| --- |
| Java @Service public class UserServiceImpl implements UserService {  @Autowired  private UserMapper userMapper;   @Override  public void createUser(String name) {  userMapper.insertUser(name);  }   @Override  public UserDTO queryUserByName(String name) {  UserDO userDO = userMapper.findUserByName(name);  if (Objects.isNull(userDO)) {  return null;  }  return new UserDTO(userDO.getName(), userDO.getId().intValue());  }   @Override  public List<UserDTO> queryAll() {  List<UserDO> userList = userMapper.findAll();  if (CollectionUtils.isEmpty(userList)) {  return null;  }  return userList.stream()  .map(u -> new UserDTO(u.getName(), u.getId().intValue()))  .collect(Collectors.toList());  } } |

1. Controller层

|  |
| --- |
| Java @RestController @Api(tags = "用户数据接口") public class UserController {  @Resource  private UserService userService;   @ApiOperation(value = "查询用户", notes = "根据姓名查询用户")  @GetMapping("/user/{name}")  public Result<UserDTO> queryUserByName(@PathVariable String name) {  UserDTO user = userService.queryUserByName(name);  if (Objects.isNull(user)) {  return Result.Result(ErrorEnum.DATA\_NOT\_FOUND, null);  }  return Result.Result(ErrorEnum.SUCCESS, user);  }   @ApiOperation(value = "查询所有用户", notes = "查询所有用户")  @GetMapping("/user/all")  public Result<List<UserDTO>> queryAllUser() {  List<UserDTO> users = userService.queryAll();  if (CollectionUtils.isEmpty(users)) {  return Result.Result(ErrorEnum.DATA\_NOT\_FOUND, null);  }  return Result.Result(ErrorEnum.SUCCESS, users);  }   @ApiOperation(value = "新增用户", notes = "根据用户名和ID添加新用户")  @ApiImplicitParams({  @ApiImplicitParam(paramType = "query", name = "userName", value = "用户名",  required = true),  @ApiImplicitParam(paramType = "query", name = "userID", value = "用户ID",  required = true)  })  @PostMapping("/user")  public String addUser(@RequestParam String userName, @RequestParam Integer userID) {  userService.createUser(userName);  return "新增用户：" + userName + "，ID：" + userID;  }   @ApiOperation(value = "修改用户信息", notes = "更新用户信息")  @PutMapping("/user")  public String updateUser(@RequestBody UserDTO userDTO) {  return userDTO.toString();  } } |

1. 测试

