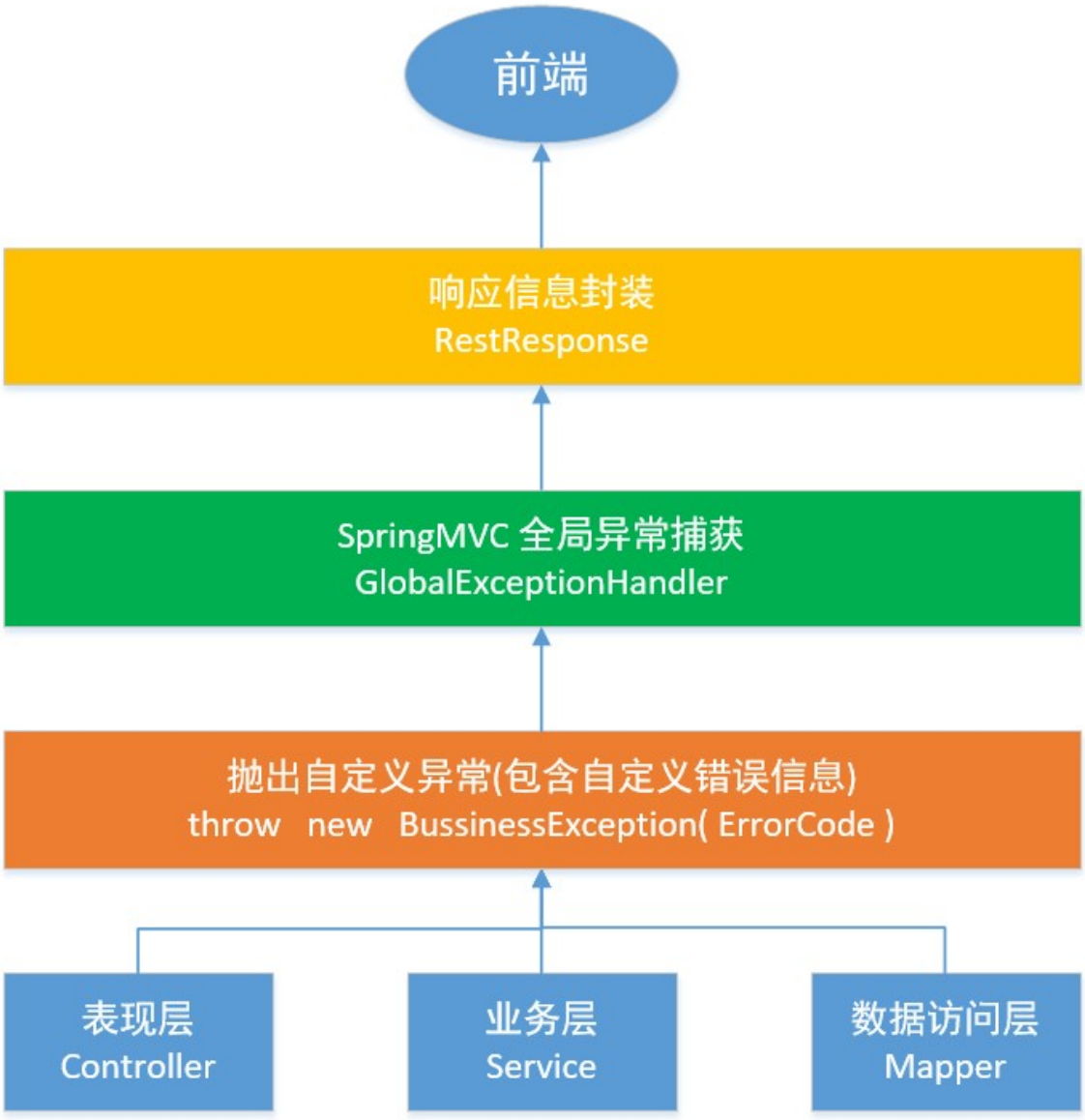
异常处理机制

一. 概述

我们在编码过程中一定是要处理异常的，传统异常处理方式存在一些问题：

1. 项目中会有很多地方都要进行异常处理，编码麻烦，代码冗余，不易维护
2. 异常处理的方式不统一
3. 错误提示信息不统一，不友好

为了解决上述问题，我们在P2P项目中设计了一套好用的异常处理机制：



1. 自定义异常类，自定义错误代码和提示信息(统一且友好)
2. 各层只抛异常(建议在业务层)，不做异常处理
3. 异常捕获和处理统一交给SpringMVC的全局异常捕获类
4. 响应给前端的错误提示信息进行统一封装

二. API设计

1. 自定义错误代码和提示信息

   接口：ErrorCode.java

```
public interface ErrorCode {

    int getCode(); //错误代码

    String getDesc(); //错误提示信息

}
```

CommonErrorCode.java提供通用错误编码和提示信息：

```
public enum CommonErrorCode implements ErrorCode {

    ////////////////////////////////////公用异常编码 ////////////////////////////

    SUCCESS(0, "成功"),
    /**
     * 传入参数与接口不匹配
     */
    E_100101(100101,"传入参数与接口不匹配"),
    /**
     * 验证码错误
     */
    E_100102(100102,"验证码错误"),
    /**
     * 验证码为空
     */
    E_100103(100103,"验证码为空"),
    /**
     * 查询结果为空
     */
    E_100104(100104,"查询结果为空"),
    /**
     * ID格式不正确或超出Long存储范围
     */
    E_100105(100105,"ID格式不正确或超出Long存储范围"),

    E_100106(100106,"请求失败"),

    /**
     * 未知错误
     */
    UNKOWN(999999,"未知错误");

    private int code;
    private String desc;

    public int getCode() {
        return code;
    }
```

```
    public String getDesc() {
        return desc;
    }

    private CommonErrorCode(int code, String desc) {
        this.code = code;
        this.desc = desc;
    }
}
```

各个微服务可以根据自身业务提供错误编码和提示信息类，例如：统一账户微服务的AccountErrorCode.java

```
public enum AccountErrorCode implements ErrorCode {

    E_130101(130101, "用户名已存在"),
    E_130104(130104, "用户未注册"),
    E_130105(130105, "用户名或密码错误"),
    E_140141(140141, "注册失败"),
    E_140151(140151, "获取短信验证码失败"),
    E_140152(140152, "验证码错误");

    private int code;
    private String desc;

    public int getCode() {
        return code;
    }

    public String getDesc() {
        return desc;
    }

    private AccountErrorCode(int code, String desc) {
        this.code = code;
        this.desc = desc;
    }
}
```

2. 自定义异常类：BusinessException.java

```
public class BusinessException extends RuntimeException {

    private static final long serialVersionUID = 5565760508056698922L;

    private ErrorCode errorCode;  //自定义错误代码和提示信息接口

    public BusinessException(ErrorCode errorCode) {
        super();
        this.errorCode = errorCode;
    }

    public BusinessException() {
        super();
    }
```

```
    public BusinessException(String arg0, Throwable arg1, boolean arg2, boolean
arg3) {
        super(arg0, arg1, arg2, arg3);
    }

    public BusinessException(ErrorCode errorCode, String arg0, Throwable arg1,
                                                    boolean arg2, boolean arg3)
{
        super(arg0, arg1, arg2, arg3);
        this.errorCode = errorCode;
    }

    public BusinessException(String arg0, Throwable arg1) {
        super(arg0, arg1);
    }

    public BusinessException(ErrorCode errorCode, String arg0, Throwable arg1) {
        super(arg0, arg1);
        this.errorCode = errorCode;
    }

    public BusinessException(String arg0) {
        super(arg0);
    }

    public BusinessException(ErrorCode errorCode, String arg0) {
        super(arg0);
        this.errorCode = errorCode;
    }

    public BusinessException(Throwable arg0) {
        super(arg0);
    }

    public BusinessException(ErrorCode errorCode, Throwable arg0) {
        super(arg0);
        this.errorCode = errorCode;
    }

    public ErrorCode getErrorCode() {
        return errorCode;
    }

    public void setErrorCode(ErrorCode errorCode) {
        this.errorCode = errorCode;
    }

}
```

3. SpringMVC全局异常捕获类：GlobalExceptionHandler.java

```
@ControllerAdvice
@Slf4j
public class GlobalExceptionHandler {

    @ExceptionHandler(value = Exception.class)
    @ResponseBody
```

```
    public RestResponse<Nullable> exceptionGet(HttpServletRequest req,
                                    HttpServletResponse response , Exception e) {
        if (e instanceof BusinessException) {
            BusinessException be = (BusinessException) e;
            if(CommonErrorCode.CUSTOM.equals(be.getErrorCode())){
                return new RestResponse<Nullable>(be.getErrorCode().getCode(),
                                                        be.getMessage());
            }else{
                return new RestResponse<Nullable>(be.getErrorCode().getCode(),
                                                    be.getErrorCode().getDesc());
            }
        }else if(e instanceof NoHandlerFoundException){
            return new RestResponse<Nullable>(404, "找不到资源");
        }else if(e instanceof HttpRequestMethodNotSupportedException){
            return new RestResponse<Nullable>(405, "method 方法不支持");
        }else if(e instanceof HttpMediaTypeNotSupportedException){
            return new RestResponse<Nullable>(415, "不支持媒体类型");
        }

        log.error("【系统异常】" + e.getMessage());
        return  new RestResponse<Nullable>(CommonErrorCode.UNKOWN.getCode(),
                                            CommonErrorCode.UNKOWN.getDesc());
    }
}
```

上述这些API在项目中会直接提供给大家使用。