

restful 全局异常捕获处理

作者: [crick77](#)

原文链接: <https://ld246.com/article/1511795152698>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

全局异常处理

本文项目已发布到github，后续学习项目也会添加到此工程下，欢迎fork点赞。

<https://github.com/wangyuheng/spring-boot-sample>

偷懒代码

偷懒是程序员的美德，但是有些偷懒是为了少写代码，有些则是少思考，直接copy。

不知道你有没有见过这种代码，在最外层**catch Exception**，避免抛出异常信息。。。

```
@RestController
@RequestMapping("/user")
public class UserApi {

    @Autowired
    private UserService userService;

    @GetMapping("/{id}")
    public Object getInfo(@PathVariable("id") int id) {
        try {
            return userService.getUsernameById(id);
        } catch (UserException) {
            return "用户id异常";
        } catch (Exception e) {
            return "服务异常,请稍后再试!";
        }
    }
}
```

UserService用于模拟异常抛出

```
@Service
public class UserService {

    public String getUsernameById(long id) {
        if (0 == id % 2) {
            throw new IllegalArgumentException("error param!");
        } else {
            throw new UserException("custom exception!");
        }
    }
}
```

全局异常处理

自定义异常

通过自定义异常，区分业务异常，并增加errorCode支持，返回给接口调用方。

```

public enum ErrorCode {
    Error(10000, "服务异常"),
    UserIdError(10001, "用户id异常");

    private int code;
    private String message;

    ErrorCode(int code, String message) {
        this.code = code;
        this.message = message;
    }

    public int getCode() {
        return code;
    }

    public String getMessage() {
        return message;
    }
}

public class CustomException extends RuntimeException {

    private int errorCode;

    public CustomException(int errorCode, String message) {
        super(message);
        this.errorCode = errorCode;
    }

    public int getErrorCode() {
        return errorCode;
    }
}

public class UserException extends CustomException {
    public UserException(String message) {
        super(ErrorCode.UserIdError.getCode(), message);
    }

    public UserException() {
        super(ErrorCode.UserIdError.getCode(), ErrorCode.UserIdError.getMessage());
    }
}

```

@ControllerAdvice

会应用到所有的Controller中的@RequestMapping注解的方法中，通过**annotations = RestController.class**指定代理的Controller类中。

@ExceptionHandler

用于捕获异常，@ExceptionHandler本身只能捕获当前类的异常信息，结合@ControllerAdvice可

捕获全部controller异常。

```
@ControllerAdvice(annotations = RestController.class)
@ResponseBody
public class GlobalExceptionHandler {

    private Map<String, Object> getErrorObject(int code, String message) {
        Map<String, Object> error = new HashMap<>();
        error.put("code", code);
        error.put("message", message);
        return error;
    }

    @ExceptionHandler(Exception.class)
    public Object exceptionHandler() {
        return getErrorObject(ErrorCode.Error.getCode(), ErrorCode.Error.getMessage());
    }

    @ExceptionHandler(CustomException.class)
    public Object customException(CustomException e) {
        return getErrorObject(e.getErrorCode(), e.getMessage());
    }
}
```

优化代码

优化后的代码简洁了，无需每次try...catch，统一管理异常信息。

```
@RestController
@RequestMapping("/user")
public class UserApi {

    @Autowired
    private UserService userService;

    @GetMapping("/{id}")
    public Object getInfo(@PathVariable("id") int id) {
        return userService.getUsernameById(id);
    }
}
```

测试

mockMvc请求api接口，判断返回值

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class UserApiTest {

    private MockMvc mockMvc;
```

```
@Autowired
private WebApplicationContext context;

@Autowired
private UserApi userApi;

@Before
public void setup() {
    mockMvc = MockMvcBuilders.webAppContextSetup(context).build();
}

private JSONObject getUserApiResult(int id) throws Exception {
    String path = "/user/" + id;
    return new JSONObject(mockMvc.perform(MockMvcRequestBuilders.get(path))
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andDo(MockMvcResultHandlers.print())
        .andReturn()
        .getResponse()
        .getContentAsString());
}

@Test
public void test_exception_handler() throws Exception {
    int i = 1;
    assertTrue(ErrorCode.UserIdError.getCode() == getUserApiResult(i).getInt("code"));
    i++;
    assertTrue(ErrorCode.Error.getCode() == getUserApiResult(i).getInt("code"));
}
}
```