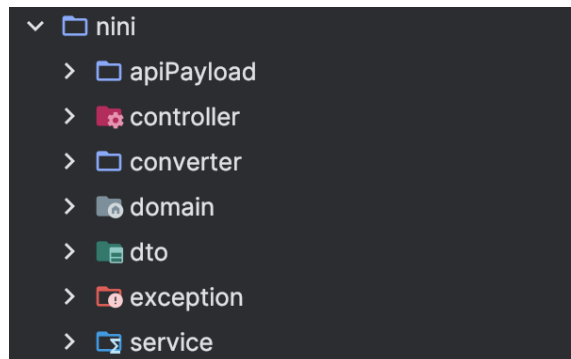
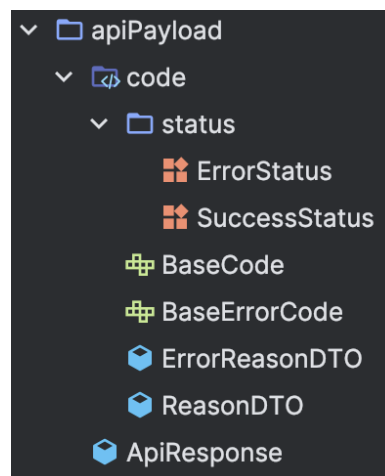


# mission 8

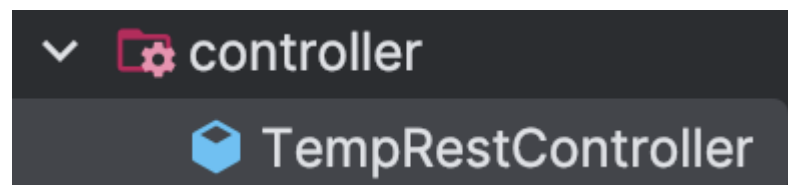
- 디렉터리 구조



## ▼ apiPayload



## ▼ controller



```

@RestController
@RequestMapping("/temp")
@RequiredArgsConstructor
public class TempRestController {
    private final TempQueryService tempQueryService;

    @GetMapping("/test")
    public ApiResponse<TempResponse.TempTestDTO> testAPI(){
        return ApiResponse.onSuccess(TempConverter.toTempTestDTO());
    }

    @GetMapping("/exception")
    public ApiResponse<TempResponse.TempExceptionDTO> exceptionAPI(@RequestParam Integer flag){
        tempQueryService.CheckFlag(flag);
        return ApiResponse.onSuccess(TempConverter.toTempExceptionDTO(flag));
    }
}

```

## ▼ converter

```

public class TempConverter {
    1 usage
    public static TempResponse.TempTestDTO toTempTestDTO(){
        return TempResponse.TempTestDTO.builder()
            .testString("This is Test!")
            .build();
    }

    1 usage
    public static TempResponse.TempExceptionDTO toTempExceptionDTO(Integer flag){
        return TempResponse.TempExceptionDTO.builder()
            .flag(flag)
            .build();
    }
}

```

## ▼ dto

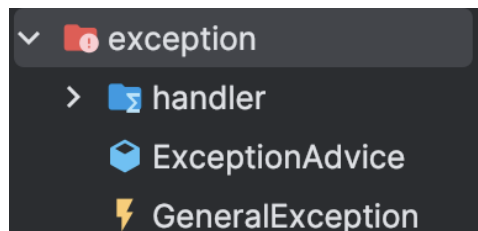
```

public class TempResponse {
    3 usages
    @Builder
    @Getter
    @NoArgsConstructor
    @AllArgsConstructor
    public static class TempTestDTO{
        String testString;
    }

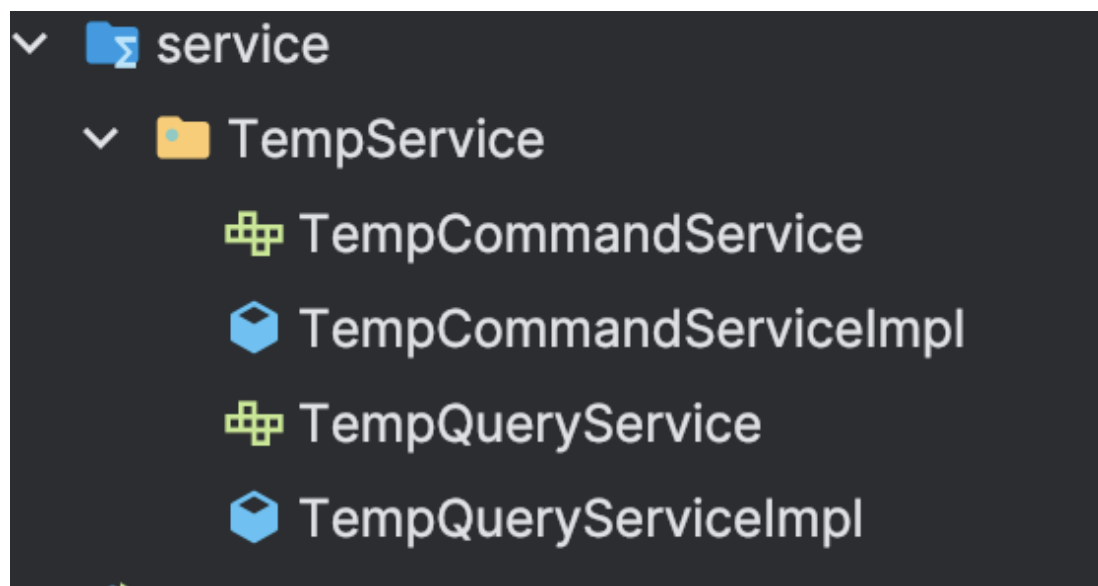
    3 usages
    @Builder
    @Getter
    @NoArgsConstructor
    @AllArgsConstructor
    public static class TempExceptionDTO{
        Integer flag;
    }
}

```

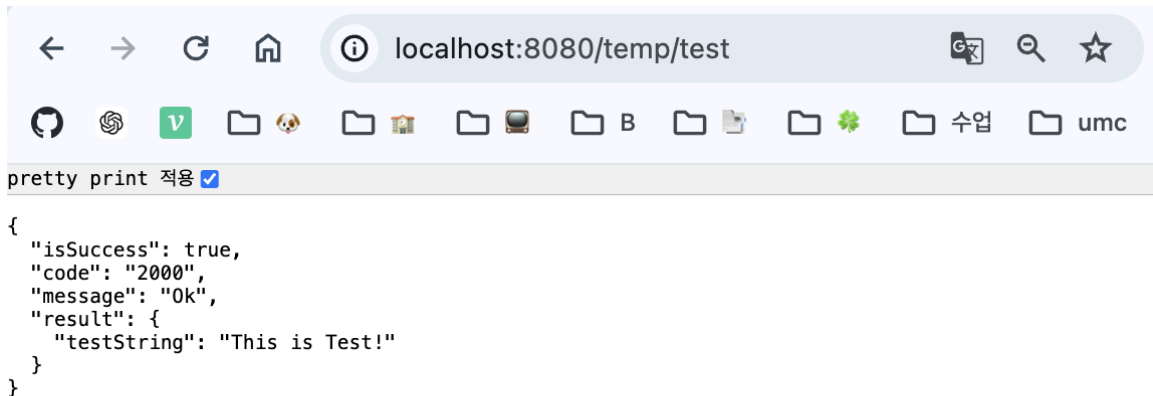
#### ▼ exception



#### ▼ service



- <http://localhost:8080/temp/test> 접속시의 response확인



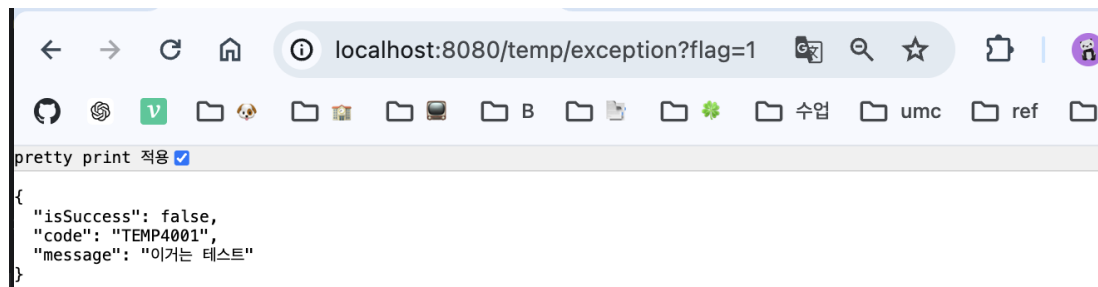
```

{
  "isSuccess": true,
  "code": "2000",
  "message": "0k",
  "result": {
    "testString": "This is Test!"
  }
}

```

- 에러 핸들러

- flag=1인 경우

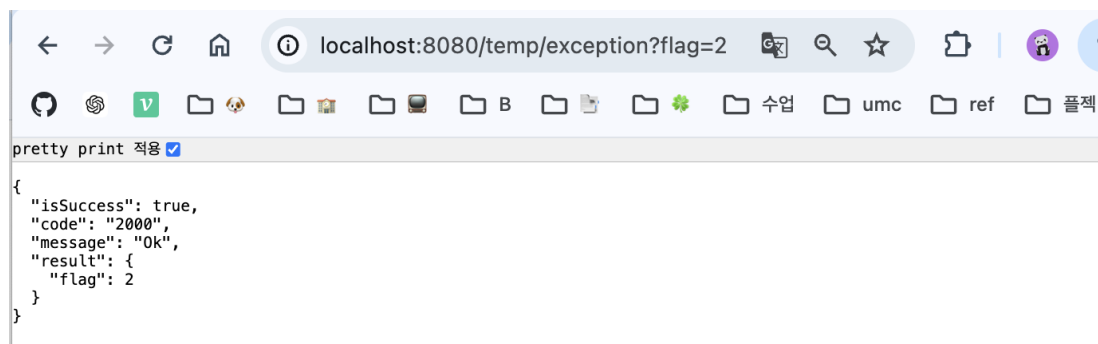


```

{
  "isSuccess": false,
  "code": "TEMP4001",
  "message": "이거는 테스트"
}

```

- flag=2인 경우



```

{
  "isSuccess": true,
  "code": "2000",
  "message": "0k",
  "result": {
    "flag": 2
  }
}

```

- RestControllerAdvice의 장점, 그리고 없을 경우 어떤 점이 불편한가?

- **RestControllerAdvice의 장점**

- 1) 중앙에서 예외 처리

- : 여러 컨트롤러에서 발생하는 예외를 한곳에서 처리할 수 O

- 2) 일관된 응답 포맷

- : 예외 발생시 일관된 형태로 응답을 제공해줄 수 O

- **RestControllerAdvice가 없는 경우 어떤 점이 불편한가?**

- 1) 코드 중복 증가

- : 컨트롤러 마다 예외 처리 로직을 작성해야하기 때문

- 2) 일관성 없는 응답

- : 컨트롤러 마다 다른 형태의 예외 응답을 제공할 수 있음

- 3) 유지 보수 어려움

- : 새로운 예외를 처리하려면 모든 관련 컨트롤러를 찾아 수정해야함

- 
- 6주차 워크북 미션 부분 참고하여 미션 목록 조회(진행중, 진행 완료) API 명세서 작성하기

Headers - Authorization : accessToken (String)

- **진행중인 미션 조회**

GET /users/{userId}/missions/pending

- Response

```
{
  "isSuccess ": true,
  "code" : "MISSION200_1",
  "message" : "진행중인 미션 조회",
  "result" :
```

```

{
  "userId":1234
  "missions": [
    {
      "missionId": 3,
      "title": "Mission 3",
      "description": "음식점 1",
      "status": "pending",
      "completedDate": "2024-06-01"
    },
    {
      "missionId": 4,
      "title": "Mission 4",
      "description": "음식점2",
      "status": "pending",
      "completedDate": "2024-06-02"
    }
  ]
}

```

#### ◦ 진행 완료된 미션 조회

GET /users/{userId}/missions/completed

#### ■ Response

```

{
  "isSuccess ": true,
  "code" : "MISSION200_2",
  "message" : "완료한 미션 조회",
  "result" :
    {
      "userId":1234
      "missions": [
        {
          "missionId": 15,

```

```
        "title": "Mission 15",
        "description": "음식점 1",
        "status": "completed",
        "completedDate": "2024-06-01"
    },
    {
        "missionId": 16,
        "title": "Mission 16",
        "description": "음식점2",
        "status": "completed",
        "completedDate": "2024-06-02"
    }
]
}
```