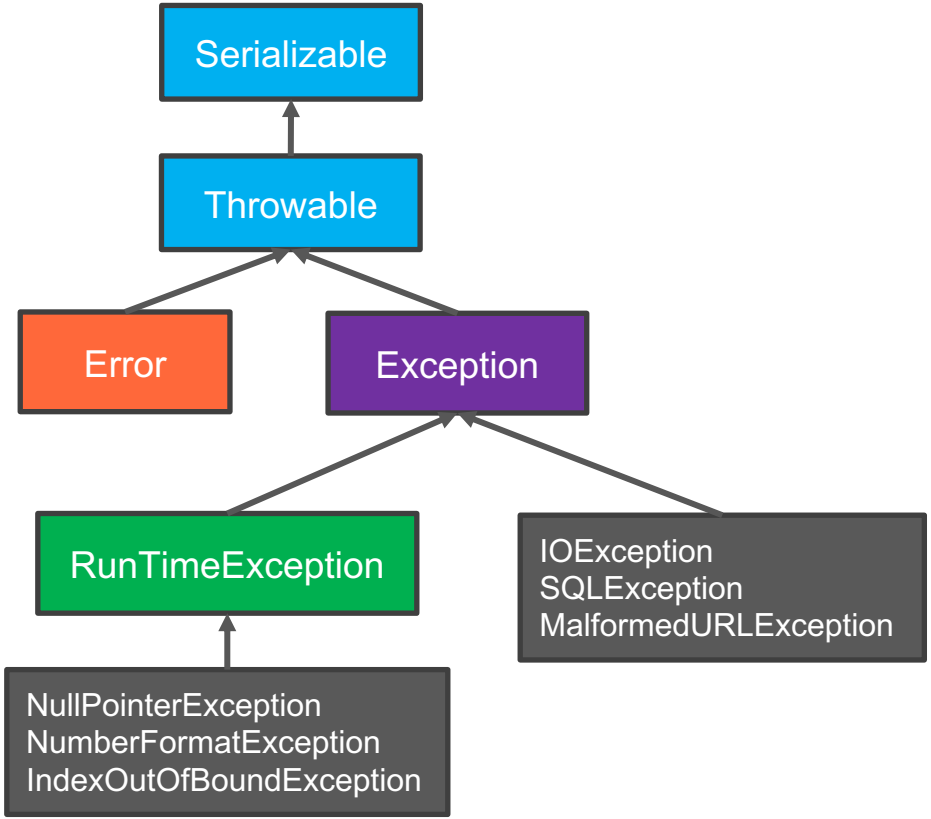
Exception Handling

cuong@techmaster.vn


Serializable
Throwable
Error
Exception
RunTimeException
IOException
SQLException
MalformedURLException
NullPointerException
NumberFormatException
IndexOutOfBoundException

Checked Exception

- Phải dùng try catch để bắt exception hoặc khai báo throws để ném tiếp Exception ra caller
- Được kiểm tra bởi Java compiler
- Checked Exception thường là lỗi vào ra file, kết nối CSDL, kết nối service. Nguyên nhân gây lỗi không phải do code mà do yếu tố bên ngoài.

Unchecked Exception

- Không cần dùng try catch hay throws
- Không kiểm tra bởi Java compiler. Ném ra và xử lý lúc runtime
- Những lỗi do logic code gây ra, không liên quan yếu tố bên ngoài : NullPointerException, NumberFormatException

Phân biệt giữa Checked Exception và Unchecked (Run time exception)

- **Checked Exception** thường là những ngoại lệ cần phải kiểm tra để khắc phục được, thường là lỗi liên quan đến tài nguyên file, database
- **Unchecked Exception** không buộc phải kiểm tra, xảy lúc chạy, khó hoặc không thể khắc phục
  - Quên không khoá cửa dẫn đến trộm vào nhà. Đây rõ ràng là một lỗi nghiệm trọng, hoàn toàn có thể kiểm tra và tránh được. Vậy nó nên là Checked Exception: ForgetLockDoorException.
  - Sửa chữa điện bị điện giật. Rõ ràng làm việc với điện có nguy cơ giật rất cao. Luôn phải kiểm tra bằng bút thử điện. Vậy nó nên là Checked Exception: ElectricShockException
  - Đang ăn thì cắn phải lưỡi. Chúng ta ăn cả mấy năm có mấy khi cắn vào lưỡi. Đang ăn cứ phải kiểm tra để tránh cắn vào lưỡi cũng kỳ cục. Nên ăn chậm rãi là được. Vậy đây có thể gọi là RuntimeException: BiteTongeException
  - Đang đi thì đập đầu vào cột điện. Bạn có thể để ý hơn để tránh va vào cột điện RunTimeException: BirdShitDropToYourHeadException

Runtime Exception có thể loại bỏ nhờ viết code tốt hơn

```
try{
  _map.put(myKey, myValue);
} catch(NullPointerException e){
  _map = new HashMap<String, String>();
}
```

Lạm dụng try catch

```
if(_map == null){
  _map = new HashMap<String, String>();
}
_map.put(myKey, myValue);
```

Sử dụng if kiểm tra gọn và nhanh hơn nhiều

Khác biệt giữa Error và Exception

Java còn có Error theo như bảng so sánh dưới đây thì Error khá giống với Runtime Exception

Sr. No.	Key	Error	Exception
1	Type	Classified as an unchecked type	Classified as checked and unchecked
2	Package	It belongs to java.lang.error	It belongs to java.lang.Exception
3	Recoverable/ Irrecoverable	It is irrecoverable	It is recoverable
4		It can't be occur at compile time	It can occur at run time compile time both
5	Example	OutOfMemoryError ,IOError	NullPointerException , SqlException

Recoverable vs Irrecoverable Exception

- **Recoverable Exception**: ngoại lệ có thể khắc phục được. Thường là nó xảy ra thường xuyên, phổ biến nên cần phải lưu ý để xử lý. Ví dụ kết nối đến server. Lỗi kết nối, server không tồn tại là rất cao. Do đó cần phải chuẩn bị ứng phó trước.
- **Irrecoverable Exception**: ngoại lệ rất ít khi xảy ra, rất khó lường trước (chim ỉa vào đầu) hoặc bất chợt xảy ra do bất cẩn của lập trình viên (cắn vào lưỡi - NullPointerException). Nếu ngoại lệ xảy ra rất khó khôi phục

try catch hay throws tiếp ra ngoài?

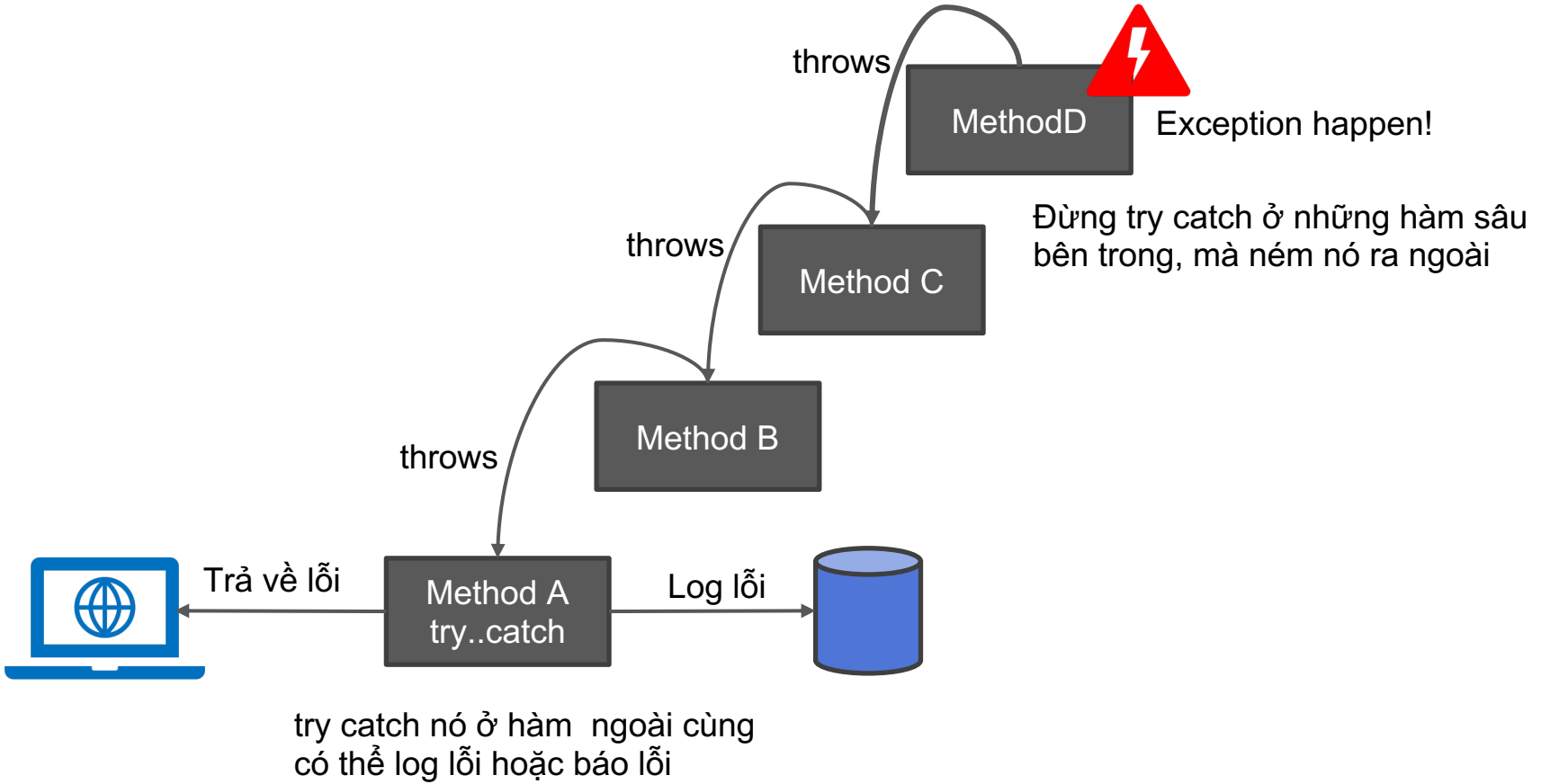
Đối với checked exception sẽ có 2 khả năng: try ... catch để xử lý ngay trong phương thức hoặc throws: ném ngoại lệ ra phương thức ngoài (caller).

Dùng try ... catch khi:

- Đó là phương thức ngoài cùng hoặc giao tiếp với một module khác hoặc trả về cho client
- Phương thức này có thể báo lỗi hoặc có cơ chế báo lỗi ví dụ controller trả về lỗi cho end user

Dùng throws khi:

- Đây chỉ là phương thức con bên trong, cần ném ngoại lệ ra ngoài để tập trung xử lý ở một chỗ.


throws
MethodD
Exception happen!
Đừng try catch ở những hàm sâu bên trong, mà ném nó ra ngoài
throws
Method C
throws
Method B
Trả về lỗi
Method A
try..catch
Log lỗi
try catch nó ở hàm ngoài cùng có thể log lỗi hoặc báo lỗi

Có nên tạo custom exception? Và khi nào tạo?

Bạn cần tạo custom exception khi:

- Custom exception giúp debug, khoanh vùng, kiểm soát lỗi dễ dàng
- Ngoại lệ đó liên quan đến nghiệp vụ cụ thể : busines domain exception
- Không thể tìm ngoại lệ tương đương, có sẵn trong Java
- Cần bổ xung thuộc tính, phương thức để mô tả ngoại lệ chi tiết hơn, xử lý ngoại lệ gọn gàng hơn ví dụ: host name, server name, time, userid…

```
public class FilmException extends Exception {
  public String hostname;
  public String userid;
  public String filmid;
  public FilmException(String message, String hostname, String userid, String
filmid) {
    super(message);
    this.hostname = hostname;
    this.userid = userid;
    this.filmid = filmid;
  }
  @Override
  public String toString() {
    return "FilmException [filmid=" + filmid + ", hostname=" + hostname + ",
userid=" + userid + "]";
  }
}
```

Nên và không nên khi tạo Custom Exception

- Tạo vừa đủ, không quá nhiều custom exception. Mỗi một business domain / microservice tạo một custom exception. Ví dụ FilmException.
- Không nên tạo quá nhiều custom exception, mỗi exception ứng với một entity ví du: FilmException, GenreException, DirectorException, ActorException. Quá vụn vặt, chả giúp việc debug dễ hơn, mà chỉ làm code rối hơn. Hãy bổ xung property vào custom exception.
- Nên thêm tham số kiểu Throwable để gắn kèm căn nguyên gây lỗi root cause vào Custom Exception

Custom checked exception hay custom run time exception ?

- Tạo custom exception kế thừa từ RunTimeException rõ ràng là thuận tiện hơn cho lập trình viên. Lập trình viên chủ động đặt try catch ở chỗ họ cần và không cần phải bổ xung throws ở khai báo hàm. Các business logic exception thuộc loại này: FilmNotFoundException, InsufficientBalanceException.
- Tạo custom exception kế thừa từ Exception khi cần tạo lỗi liên quan đến I/O, tài nguyên. Luôn phải kiểm tra và đóng connection.
- Nếu các Exception có sẵn của Java đã có đủ thì không cần tạo Custom Exception. Don’t reinvent the wheel.

Đọc thêm

- https://reflectoring.io/business-exceptions/
- https://stackify.com/best-practices-exceptions-java/
- https://howtodoinjava.com/best-practices/java-exception-handling-best-practices/#1
- Write Useful Exceptions (if you have to write your own Exceptions, make sure they provide useful information about the problem that occurred)

Xử lý Exception trong REST

Nếu có lỗi cần trả về gì cho REST client?

- Status code: dạng số nguyên dương 40x, 50x, …
  400: bad request, 401: unauthorized, 402: payment required, 403: forbidden, 404: not found, 408 request timeout

  https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#4xx_client_errors
- Message: cần bản địa hoá nếu là lỗi trả về người dùng cuối: tài khoản không tồn tại, không đủ tiền trong ví, không có quyền thực hiện,
- Cause: nguyên nhân gây lỗi nếu có
- Package . Module . Server: địa chỉ nơi lỗi phát sinh.

Whitelabel Error Page

Phù hợp cho dev để debug nhưng quá nhiều thông tin cho end-user

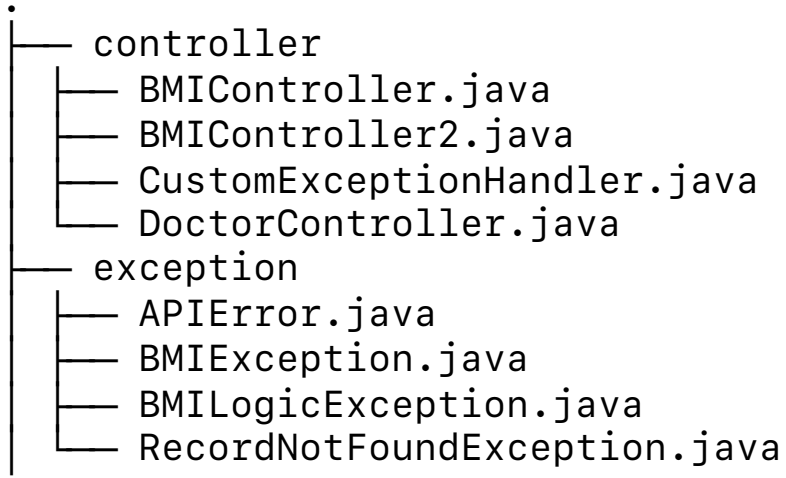
This application has no explicit mapping for /error, so you are seeing this as a fallback.

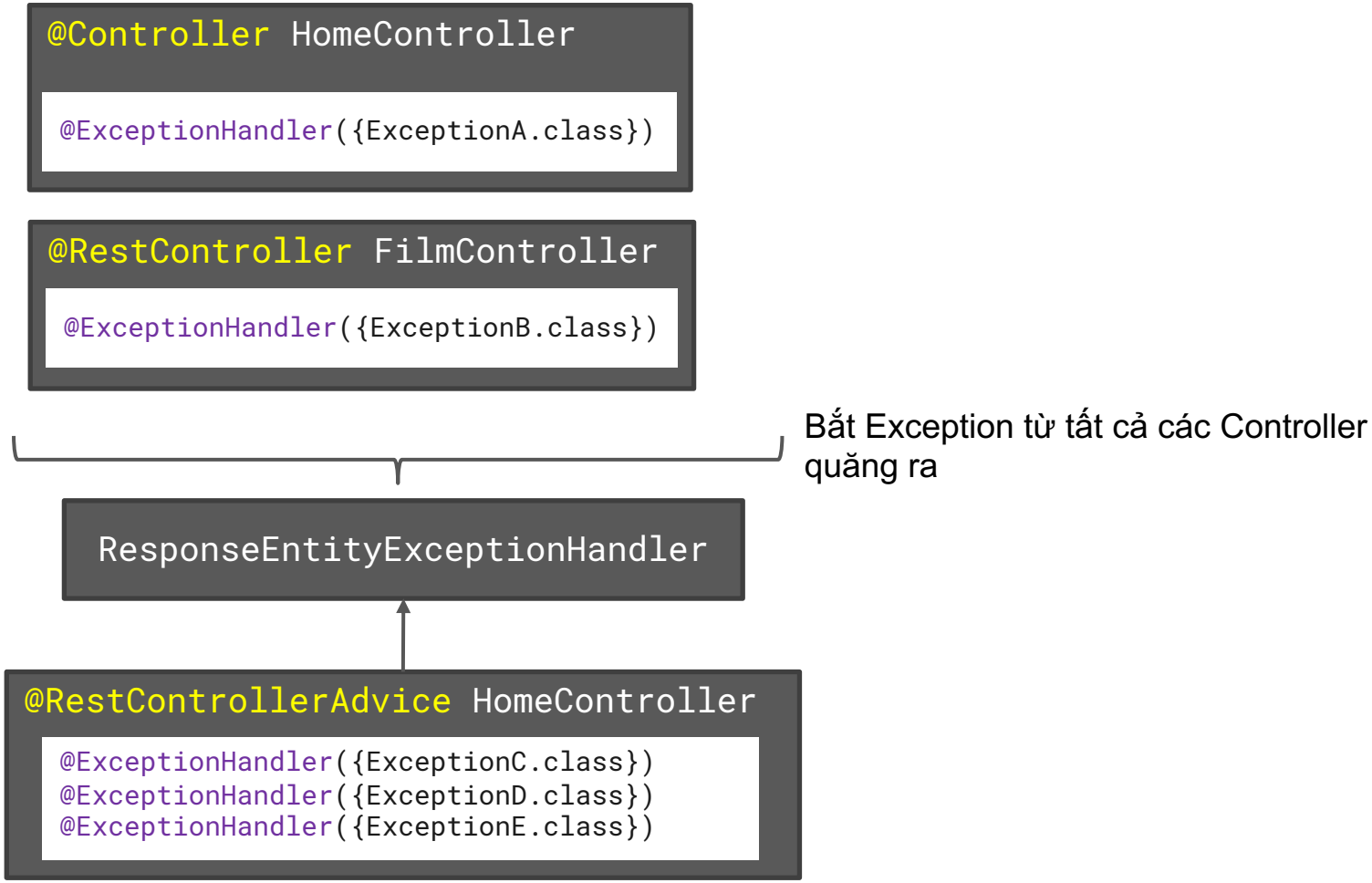
Fri Dec 03 23:39:55 ICT 2021
There was an unexpected error (type=Not Found, status=404).
Actor Not Found
org.springframework.web.server.ResponseStatusException: 404 NOT_FOUND "Actor Not Found"; nested exception is java.lang.RuntimeException: Film not found
    at vn.techmaster.bmiservice.controller.BMIController2.demoThrow4(BMIController2.java:28)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.base/java.lang.reflect.Method.invoke(Method.java:568)
    at org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:205)
    at org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:150)
    at org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:117)
    at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:895)
    at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)
    at org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)
    at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1067)
    at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:963)
    at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)
    at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:655)
    at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:764)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:227)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
    at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:189)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
    at org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:100)
    at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)

Cách phổ biến để trả về lỗi

- Cách 1: @ExceptionHandler trực tiếp trong controller
- Cách 2: @RestControllerAdvice kết hợp với ResponseEntityExceptionHandler

Xem dự án mẫu bmiservice




@Controller HomeController
@ExceptionHandler({ExceptionA.class})
@RestController FilmController
@ExceptionHandler({ExceptionB.class})
Bắt Exception từ tất cả các Controller quăng ra
ResponseEntityExceptionHandler
@RestControllerAdvice HomeController
@ExceptionHandler({ExceptionC.class})
@ExceptionHandler({ExceptionD.class})
@ExceptionHandler({ExceptionE.class})

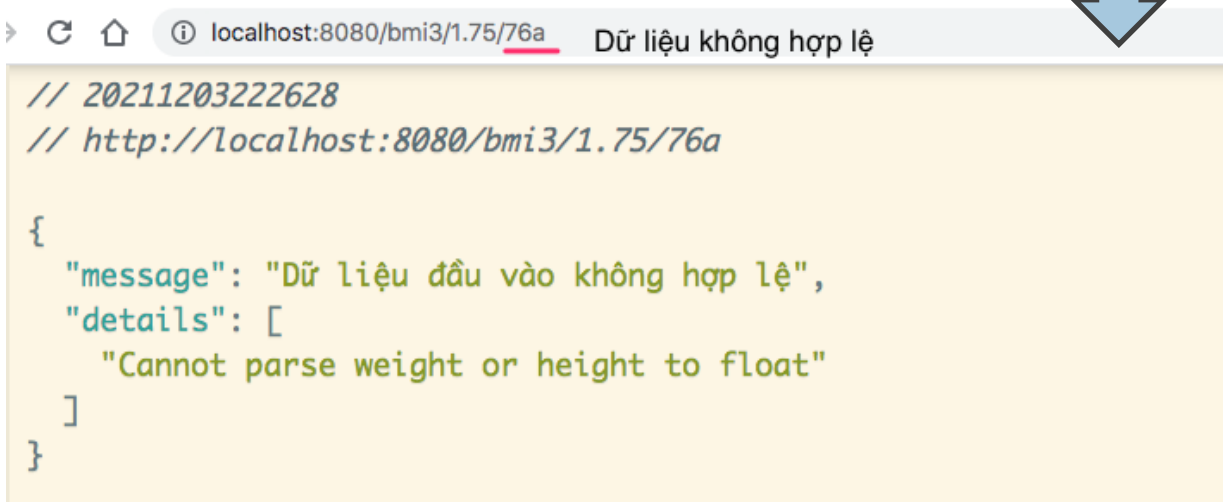
C1: thêm @ExceptionHandler vào Controller

```
@ExceptionHandler({ BMIException.class })
public ResponseEntity<APIError> handleException(BMIException ex) {
 return ResponseEntity.badRequest().body(new APIError("Dữ liệu đầu vào
không hợp lệ", ex.getMessage()));
}

@ExceptionHandler({ BMILogicException.class })
public ResponseEntity<APIError> handleException2(BMILogicException ex)
{
  return ResponseEntity.badRequest().body(new APIError("Logic tính BMI
sai", ex.getMessage()));
}
```

BMIException

BMILogicException

```java
public class APIError {
  public String message;
  public List<String> details;

  public APIError(String message, String ...
details) {
   this.message = message;
   this.details = Arrays.asList(details);
   }
}
```



Nhược điểm C1 thêm trực tiếp @ExceptionHandler vào Controller

- Việc báo lỗi chỉ giới hạn trong từng Controller
- Nếu mỗi controller sẽ có một custom exception riêng thì cách này hợp lý.
- Nhưng nếu nhiều controller sử dụng chung một custom exception, thì cách này sẽ phải lặp lại code ở mỗi controller

C2: @RestControllerAdvice + ResponseEntityExceptionHandler

- @RestControllerAdvice = @RestControllerAdvice + @ResponseBody
- @RestControllerAdvice và @ControllerAdvice sẽ can thiệp tới tất cả các controller
- ResponseEntityExceptionHandler là abstract class chứa các mẫu hàm để xử lý ngoại lệ
- Cách này phù hợp khi nhiều controller sẽ ném ra chung vài kiểu custom exception

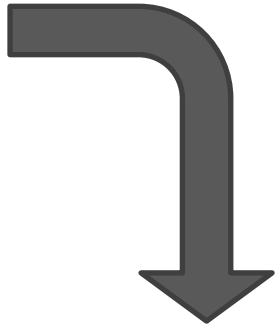
```java
@Order(Ordered.HIGHEST_PRECEDENCE)
@RestControllerAdvice
public class CustomExceptionHandler extends ResponseEntityExceptionHandler {

  @ExceptionHandler(value = {Exception.class, RuntimeException.class})
  public final ResponseEntity<APIError> handleAllExceptions(Exception ex, WebRequest
request) {
    APIError apiError = new APIError("Generic Exception", ex.getLocalizedMessage());
    return new ResponseEntity<>(apiError, HttpStatus.INTERNAL_SERVER_ERROR);
  }


  @ExceptionHandler(RecordNotFoundException.class)
  public final ResponseEntity<APIError>
handleUserNotFoundException(RecordNotFoundException ex, WebRequest request) {
    APIError apiError = new APIError("Record Not Found", ex.getLocalizedMessage(),
ex.getModel(), "Google Cloud : 122.222.11.105");
    return new ResponseEntity<>(apiError, HttpStatus.NOT_FOUND);
  }
```

Dùng ResponseStatusException thay cho quăng Exception

Bạn có thể không cần tạo CustomException mà chỉ cần dùng ResponseStatusException kết hợp hàm bắt GenericException

```java
@GetMapping("/bmi6")
public String demoThrow4() {
  throw new ResponseStatusException(
    HttpStatus.NOT_FOUND, "Actor Not Found");
}
```

Quăng ra



Bắt lấy

```java
@ExceptionHandler(value = {Exception.class, RuntimeException.class})
public final ResponseEntity<APIError> handleAllExceptions(Exception ex, WebRequest request) {
    APIError apiError = new APIError("Generic Exception", ex.getLocalizedMessage());
    return new ResponseEntity<>(apiError, HttpStatus.INTERNAL_SERVER_ERROR);
}
```