# 1. نصائح التحكم (Controller Advise )

## 1.1 مفهوم الاستثناءات في جافا :

الاستثناء (**Exceptions**) هو عبارة عن خطأ يحدث أثناء تشغيل البرنامج يؤدي إلى إيقافه بشكل غير طبيعى.

ظهور خطأ يؤدي إلى إيقاف البرنامج هو أمر سيئ جداً لأنه يؤدي إلى نفور عدد كبير من المستخدمين وعدم رغبتهم في العودة إلى استخدام هذا البرنامج مجدداً، ولابد أن نجعل البرنامج يلتقط هذه الأخطاء فى حال حدثت ، لجعل البرنامج شغال دائماً في نظر المستخدم و لا يظهر له أي أخطاء.

## 1.2 بعض الأسباب لحدوث استثناء في البرنامج:

- في حال إدخال قيمة لا تتطابق مع نوع المتغير الذي ستخزن فيه.
- في حال إدخال رقم index غير موجود في مصفوفة أو متغير نوعه String.
- في حال كان البرنامج يتصل بالشبكة وفجأة انقطع الاتصال.
- في حال كان البرنامج يحاول قراءة معلومات من ملف نصي وكان هذا الملف غير موجود.

## 1.3 بناء الاستثناءات في جافا:

تم تقسيم الاستثناءات في جافا إلى عدة أنواع وكل نوع تم تمثيله في كلاس منعزل، وجميع هذه الكلاسات ترث من كلاس أساسي اسمه (Exception) .

## 1.4 الاستثناءات في Spring Boot:

لكي يستطيع السيرفر التعامل مع هذه الاستثناءات لابد من إنشاء كلاس يسمى Controller Advice وهو طبقة التحكم المسؤولة عن التعامل مع الأخطاء التي قد تحدث أثناء عمل السيرفر لتجنب توقفه من خلال استقباله لهذه الأخطاء وإرسال نتيجة بنوع الخطأ ومكانه ، ولابد من تعريف Methods داخل هذا الكلاس لتتعامل مع كل نوع من أنواع هذه الأخطاء . وسنقوم بإضافة التعليق التوضيحي **@RestControllerAdvise** ليأخذ الكلاس خاصية التعامل مع الاستثناءات .

```java
// Our Exception
@ExceptionHandler(value = ApiException.class)
public ResponseEntity ApiException(ApiException e){
    String message=e.getMessage();
    return ResponseEntity.status(400).body(message);
}

// Server Validation Exception
@ExceptionHandler(value = MethodArgumentNotValidException.class)
public ResponseEntity<ApiResponse>
MethodArgumentNotValidException(MethodArgumentNotValidException e) {
    String msg = e.getFieldError().getDefaultMessage();
    return ResponseEntity.status(400).body(new ApiResponse(msg));
}

// SQL Constraint Exception
@ExceptionHandler(value =
SQLIntegrityConstraintViolationException.class)
public ResponseEntity<ApiResponse>
SQLIntegrityConstraintViolationException(SQLIntegrityConstraintViolat
ionException e){
    String msg=e.getMessage();
    return ResponseEntity.status(400).body(new ApiResponse(msg));
}

// Method not allowed Exception
@ExceptionHandler(value =
HttpRequestMethodNotSupportedException.class)
public ResponseEntity<ApiResponse>
HttpRequestMethodNotSupportedException(HttpRequestMethodNotSupportedE
xception e) {
    String msg = e.getMessage();
    return ResponseEntity.status(400).body(new ApiResponse(msg));
}

// Json parse Exception
@ExceptionHandler(value = HttpMessageNotReadableException.class)
public ResponseEntity<ApiResponse>
HttpMessageNotReadableException(HttpMessageNotReadableException e){
    String msg = e.getMessage();
    return ResponseEntity.status(400).body(new ApiResponse(msg));
}

// TypesMissMatch Exception
@ExceptionHandler(value = MethodArgumentTypeMismatchException.class)
public ResponseEntity<ApiResponse>
MethodArgumentTypeMismatchException(MethodArgumentTypeMismatchExcepti
on e) {
    String msg = e.getMessage();
    return ResponseEntity.status(400).body(new ApiResponse(msg));
}
```

### 1.5 إنشاء استثناء Exception :

يمكننا كذلك انشاء Exception خاص وبداخله رسالتنا الخاصة من خلال إنشاء كلاس اسمه كما هو واضح في الأعلى

ApiException ويرث من كلاس RuntimeException

```
public class ApiException extends RuntimeException {
    public ApiException(String message){
        super(message);
    }
}
```

وسيمكننا الآن أن نرمي استثناء في أي مكان نتوقع فيه الخطأ مع رسالتنا الخاصة.

**مثال على ذلك:** نريد أن نبحث عن Blog في قاعدة البيانات بواسطة ال id الخاص بها وفي حال لم تكن هذا المدونة

موجودة سنقوم برمي استثناء :

```
public Blog getBlogbyId(Integer id){
   Blog blog=blogRepository.findBlogById(id);
   if(blog==null){
       throw new ApiException("wrong ID");
   }
   return blog;

}
```