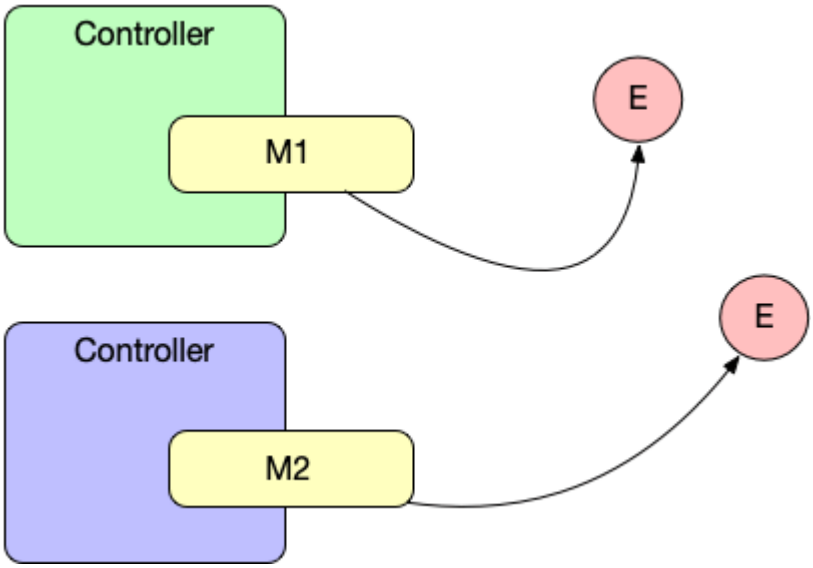
Cuando desarrollamos aplicaciones en Spring Boot, es clave gestionar bien los errores para ofrecer una mejor experiencia al usuario y mantener un código limpio. En lugar de manejar excepciones dentro de cada controlador, lo ideal es usar @ControllerAdvice, una anotación que nos permite centralizar el manejo de errores en un solo lugar.



¿Qué es @ControllerAdvice?

@ControllerAdvice es una anotación de Spring que nos ayuda a capturar y procesar excepciones desde un solo punto, evitando repetir código en cada controlador.

¿Por qué usar @ControllerAdvice?

- Centraliza la gestión de errores.
- Separa responsabilidades (SoC – Separation of Concerns).
- Hace el código más limpio y fácil de mantener.
- Permite personalizar las respuestas de error de manera uniforme.

Implementación de @ControllerAdvice en Spring

Boot

Veamos cómo usar @ControllerAdvice para manejar excepciones de forma global en una app Spring Boot.

Paso 1: Crear una excepción personalizada

Para capturar errores específicos, definimos una excepción propia:

```
public class RecursoNoEncontradoException extends RuntimeException {
    public RecursoNoEncontradoException(String mensaje) {
        super(mensaje);
    }
}
```

Paso 2: Crear un manejador global de excepciones

Ahora, creamos una clase con @ControllerAdvice para capturar y responder a las excepciones:

```
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
public class ManejadorGlobalExcepciones {
    @ExceptionHandler(RecursoNoEncontradoException.class)
    public ResponseEntity<String>
manejarRecursoNoEncontrado(RecursoNoEncontradoException ex) {
        return new ResponseEntity<>(ex.getMessage(),
HttpStatus.NOT_FOUND);
```

```
    }
    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> manejarExcepcionGeneral(Exception
ex) {
        return new ResponseEntity<>("Ocurrió un error inesperado",
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

¿Qué hace este código?

1. @ControllerAdvice: Indica que esta clase manejará errores de forma global.
2. @ExceptionHandler(RecursoNoEncontradoException.class): Captura y maneja específicamente la excepción RecursoNoEncontradoException, devolviendo un código HTTP 404.
3. @ExceptionHandler(Exception.class): Captura cualquier otra excepción no controlada y devuelve un código HTTP 500.

Paso 3: Lanzar la excepción en un controlador

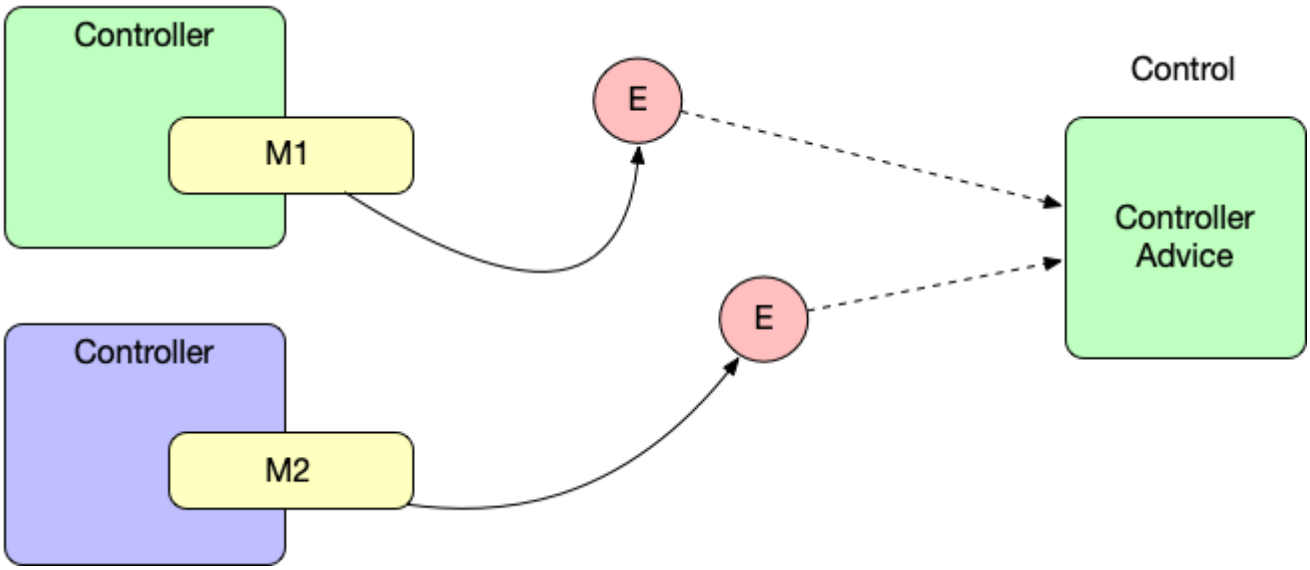
Ahora, en un controlador podemos lanzar la excepción para que se maneje globalmente:

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ProductoController {
    @GetMapping("/producto")
    public String obtenerProducto(@RequestParam(required = false)
String id) {
        if (id == null) {
            throw new RecursoNoEncontradoException("El producto no fue
```

```
encontrado");
        }
        return "Producto encontrado";
    }
}
```

Si alguien intenta acceder a /producto sin proporcionar un id, la excepción RecursoNoEncontradoException será capturada por ManejadorGlobalExcepciones, devolviendo un error 404 con el mensaje correspondiente.



Conclusión

@ControllerAdvice en Spring Boot nos permite gestionar excepciones de manera eficiente y centralizada. Con esta técnica, podemos dar respuestas más claras y personalizadas, hacer el código más mantenible y seguir buenas prácticas en el desarrollo de aplicaciones.

- JQuery Ajax y Eventos Globales
- Java Try Catch y su manejo
- Java 7
- JPA Persist y buenas prácticas
- Java RuntimeException y su utilidad