DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

REDEC UNAM

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS
DE INFORMACIÓN Y COMUNICACIÓN

DIPLOMADO

Desarrollo de sistemas con tecnología Java

Módulo 10

API RESTful con Spring Boot

M. en C. Jesús Hernández Cabrera

Paginado

- El paginado permite dividir grandes conjuntos de datos en partes más pequeñas y manejables.
- Ayuda a reducir la carga del servidor y del cliente.
- Mejora la eficiencia y la experiencia del usuario al limitar la cantidad de datos devueltos en una sola respuesta.

Paginación

- Spring Boot ofrece soporte de paginado integrado a través de Spring Data JPA.
- Utiliza las clases Pageable y Page.
- **Pageable**: Representa la información sobre la página solicitada (número de página, tamaño de la página, y opcionalmente, orden).
- **Page**: Representa la respuesta paginada.

PageRequest

- Es una implementación concreta de la interfaz **Pageable** en **Spring Data**.
- Permite definir cómo deseas que Spring Data devuelva los resultados de una consulta de manera paginada. (page, size y sort.

```
Pageable pageable = PageRequest.of(0, 10);

Page<Alumno> alumnosPage = alumnoRepository.findAll(pageable);
```

Página 0

10 por página

```java
public interface IAlumnoService {

    List<AlumnoDto> getAlumnosList();
    List<AlumnoDto> getAlumnosListPageable(int page,
                                           int size,
                                           String sortDir,
                                           String sort);
    Alumno updateAlumno(Alumno alumno);
    Alumno createAlumno(Alumno alumno);
    void deleteAlumno(String matricula);
    Optional<Alumno> getAlumnoById(String matricula);
    List<Alumno> findAlumnosByEstado(String estado);
}
```

AlumnoService

```java
@Override
public List<AlumnoDto> getAlumnosListPageable(int page, int size
                            , String sortDir, String sort) {
    PageRequest pageRequest = PageRequest.of(page, size
                        , Sort.Direction.fromString(sortDir), sort);

    Page<Alumno> alumnos = alumnoRepository.findAll(pageRequest);
    return alumnos.getContent() List<Alumno>
            .stream() Stream<Alumno>
            .map(this::convertToDto) Stream<AlumnoDto>
            .collect(Collectors.toList());
}
```

Validaciones en una API REST

- Las validaciones son reglas que aseguran que los datos enviados por el cliente cumplen ciertos criterios antes de ser procesados.
- Las validaciones ayudan a prevenir la entrada de datos incorrectos o inconsistentes en la base de datos.
- En Spring Boot, las validaciones se manejan mediante **Bean Validation** y anotaciones como:
  - @NotNull, @Size, @Min, etc.

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

Métodos REST donde aplica validación

- **POST**: Para crear nuevos recursos (validar los datos de entrada antes de guardar).
- **PUT**: Para actualizar completamente un recurso (validar todos los campos requeridos).
- **PATCH**: Para actualizaciones parciales (puedes validar solo los campos que se van a modificar).

```
@PostMapping(path = "/")
public ResponseEntity<Alumno> createAlumno(@Valid @RequestBody AlumnoDto alumnoDto) {
    Alumno alumno = alumnoService.createAlumno(alumnoDto);
    return ResponseEntity.ok(alumno);
}
```

@Valid Put

```
@PutMapping(path = "/{id}")
public ResponseEntity<Alumno> updateAlumno(
    @PathVariable String id,
    @Valid @RequestBody AlumnoDto alumnoDto) {
    Alumno alumno = alumnoService.updateAlumno(id, alumnoDto);
    return ResponseEntity.ok(alumno);
}
```

DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

REDEC UNAM

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS
DE INFORMACIÓN Y COMUNICACIÓN

DIPLOMADO

Desarrollo de sistemas con tecnología Java

Manejo de Excepciones

Error Handling

@ExceptionHandler

- Se usa para manejar excepciones específicas.
- El método anotado se invoca cuando se lanzan las excepciones especificadas desde un @Controller

@ExceptionHandler a nivel de controlador

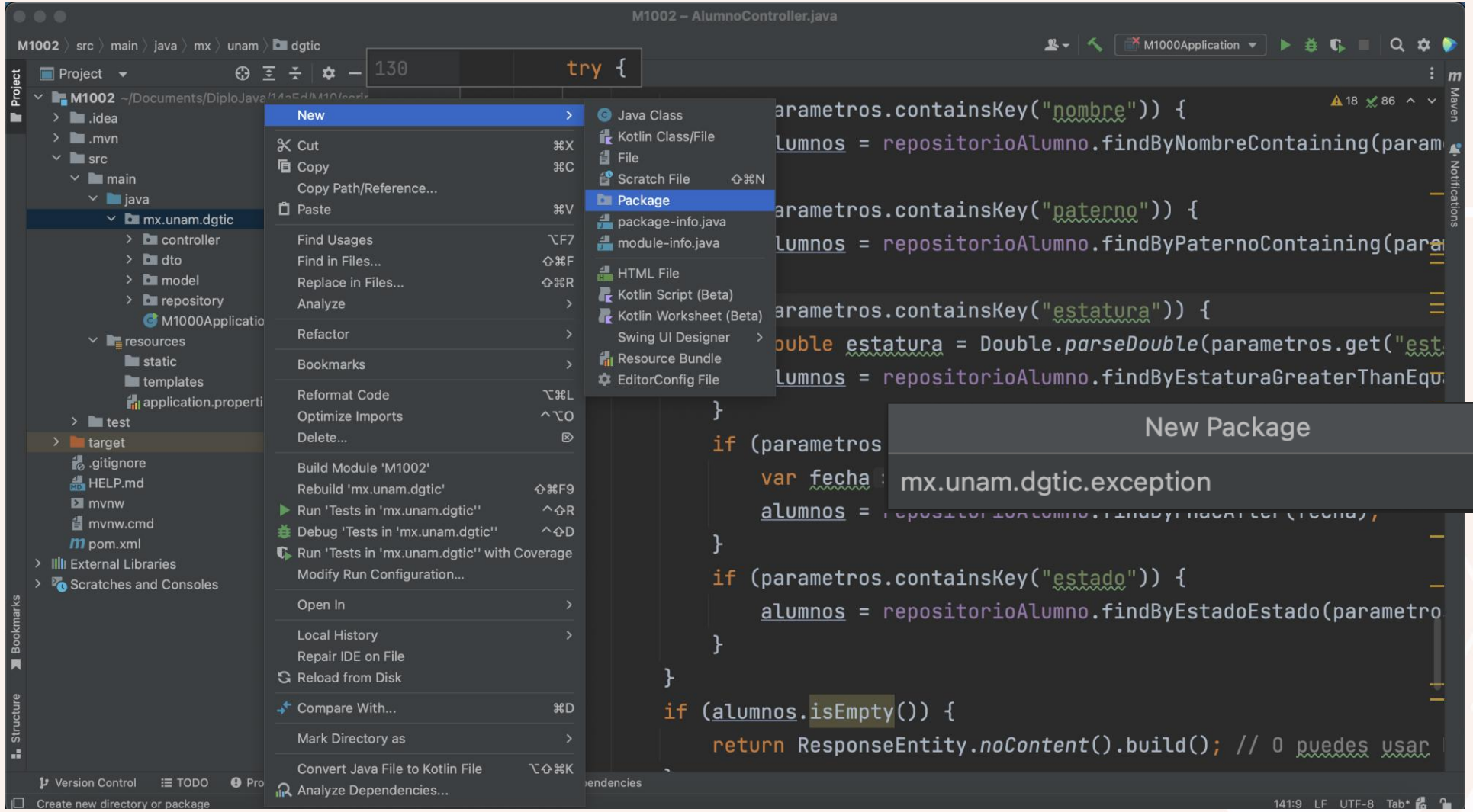
- En el controlador se define un método para manejar excepciones y se agrega la anotación ***@ExceptionHandler***

- Este enfoque tiene un inconveniente importante: el método anotado @ExceptionHandler solo está activo para ese controlador en particular, no globalmente para toda la aplicación.

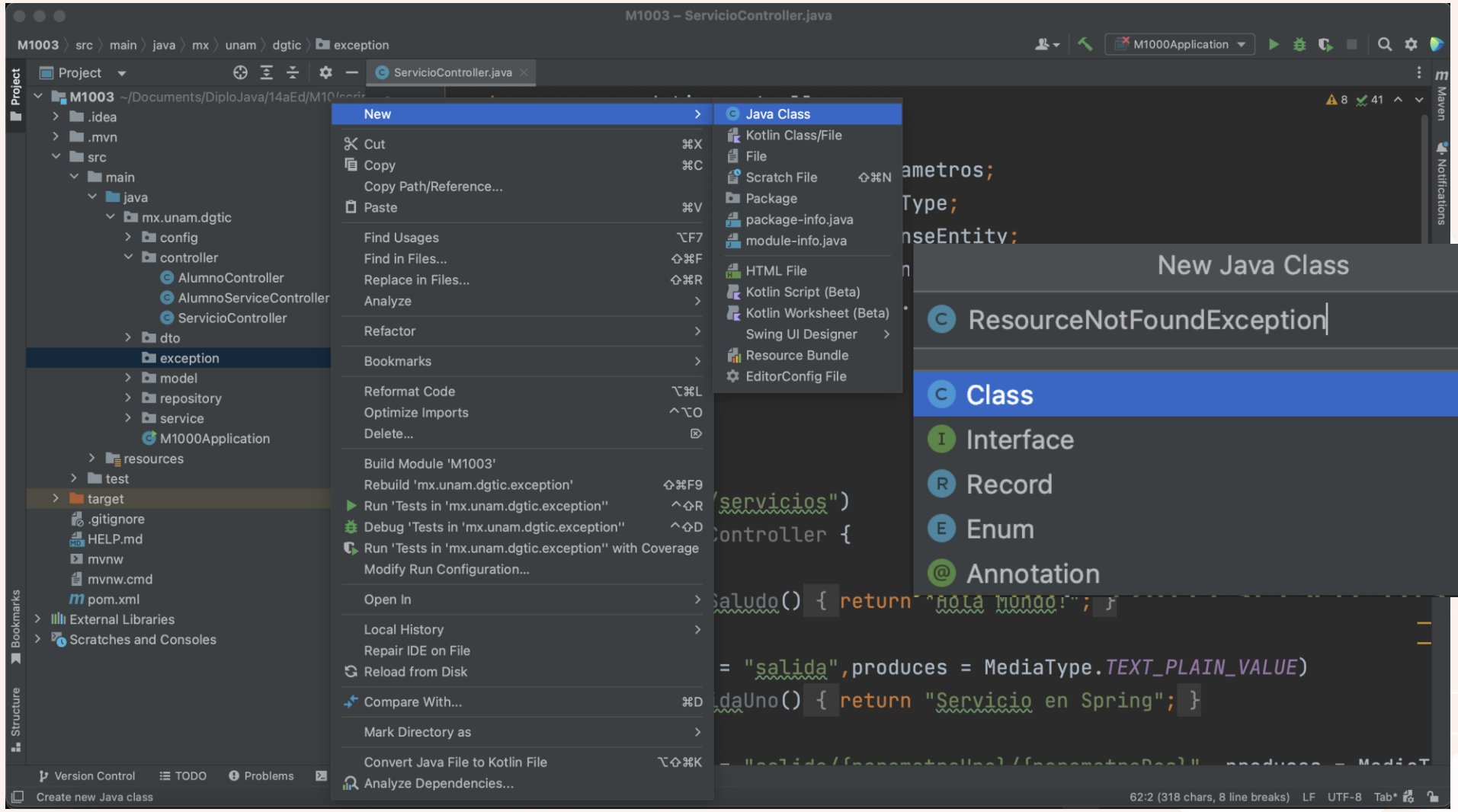
@ExceptionHandler a nivel de controlador

```
@RestController
@RequestMapping("/api/servicios")
public class ServicioController {
   @GetMapping
   public String getSaludo() {
      return "Hola Mundo!";
   }
   @ExceptionHandler
   public String errorParametro(MethodArgumentTypeMismatchException ex){
      String nombre=ex.getName();
      String tipo=ex.getRequiredType().getSimpleName();
      Object valor=ex.getValue();
      return String.format("El parámetro '%s' es tipo '%s' y el valor '%s' no es '%s'"
            ,nombre,tipo,valor,tipo);
   }
}
```

@RestControllerAdvice



@ RestControllerAdvice



Utilizar excepción personalizada.

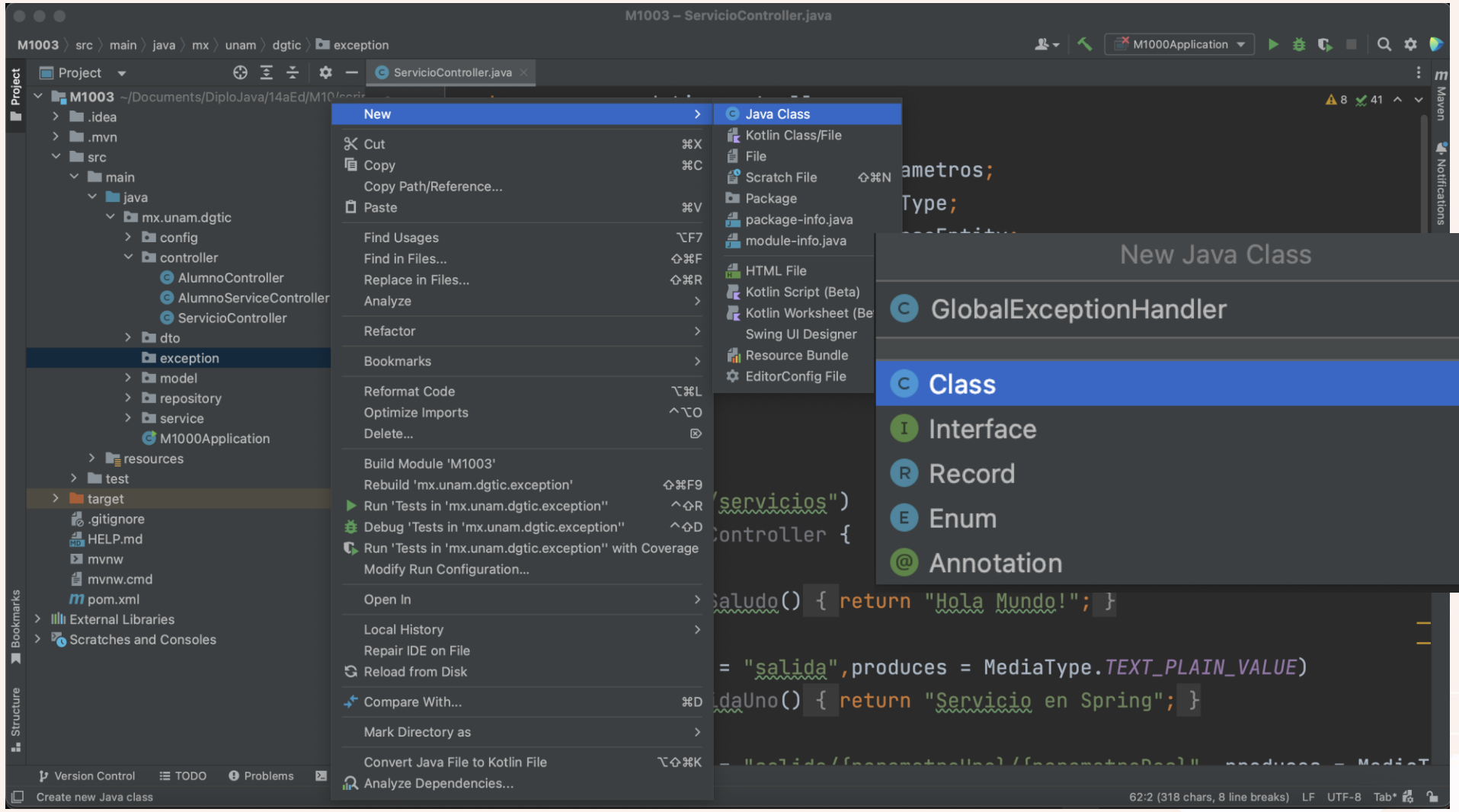
```
@GetMapping(path = "/{id}")
public ResponseEntity<AlumnoDto> getAlumnoById(@PathVariable String id){
    Optional<AlumnoDto> alumno = alumnoService.getAlumnoById(id);
    if (alumno.isPresent()){
        return ResponseEntity.ok(alumno.get());
    }else{
        //return ResponseEntity.notFound().build();
        throw  new ResourceNotFoundException("El Alumno con ID: " + id + " No existe");
    }

}
```

Error Handling

@RestControllerAdvice

- Es un mecanismo de manejo de excepciones global en Spring de manera **UNIFORME**.
- Define un lugar central para manejar las excepciones que ocurren en la aplicación.
- Esta clase puede contener varios métodos, cada uno de los cuales está anotado con la anotación ***@ExceptionHandler*** y es responsable de manejar una excepción específica.

@RestControllerAdvice



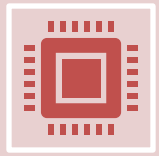
@RestControllerAdvice

- Spring brinda soporte para un ***@ExceptionHandler*** global con la anotación ***@RestControllerAdvice***.
- La anotación @ RestControllerAdvice permite consolidar múltiples @ExceptionHandlers dispersos en un único componente de manejo de errores global.
- El mecanismo es extremadamente simple pero también muy flexible
  - Brinda control total sobre el cuerpo de la respuesta, así como sobre el código de estado.
  - Proporciona mapeo de varias excepciones al mismo método, para ser manejadas juntas.
  - Hace un buen uso de la respuesta RESTful ResposeEntity.

@RestControllerAdvice

```
@GetMapping(value = "/{id}")
public ResponseEntity<?> getAlumno(@PathVariable("id") String id) {
   Optional<Alumno> alumno = alumnoService.getAlumnoById(id);
   if (alumno.isPresent()) {
      AlumnoDto alumnoDto = convertToDto(alumno.get());
      return ResponseEntity.ok(alumnoDto); // Devuelve el alumno DTO con estado 200 OK
   } else {
      //return ResponseEntity.notFound().build(); // Devuelve estado 404 NOT FOUND
      throw new ResourceNotFoundException("Recurso con ID " + id + " no encontrado.");
   }
}
```

ResponseStatusException



La excepción ***ResponseStatusException*** en Spring es una manera conveniente de lanzar una excepción desde un controlador con un código de estado HTTP específico y un mensaje de error opcional.

Se introdujo en Spring 5 y ofrece una forma programática de indicar problemas de la aplicación con más control sobre la respuesta HTTP que se enviará al cliente.

Es especialmente útil en situaciones donde necesitas una manera rápida de manejar errores HTTP sin configurar manejadores de excepciones globales o anotaciones adicionales en tus clases de excepción.

ResponseStatusException

- Existen tres constructores para generar un ***ResponseStatusException***
  - ResponseStatusException(HttpStatus status)
  - ResponseStatusException(HttpStatus status, java.lang.String reason)
  - ResponseStatusException(
    HttpStatus status,
    java.lang.String reason,
    java.lang.Throwable cause
)

Contacto

M. En C. Jesús Hernández Cabrera

Profesor de carrera

jesushc@unam.mx

Redes sociales:

www.linkedin.com/in/hcjesus