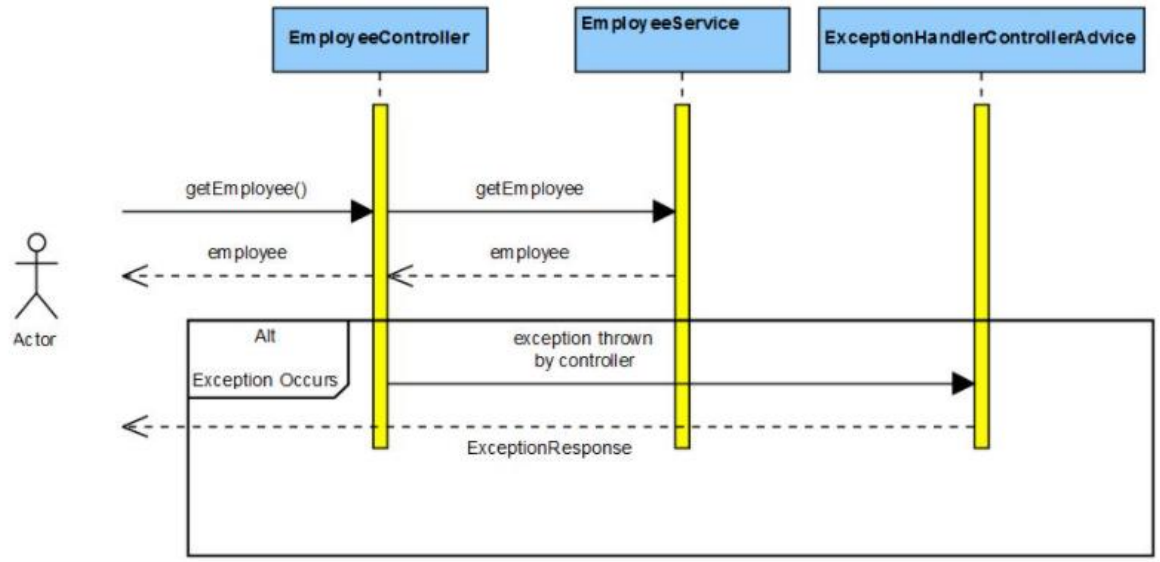
**GESTTION DES EXCEPTIONS SOUS SPRING BOOT EN UTILISANT @ControllerAvice : Cas pratique sur le CRUD d'un employé.**

**@ControllerAdvice** est une puissante annotation de Spring permettant la gestion des exceptions. Nous implémenterons dans cet article, une classe ControlerAdvice qui manipulera toutes les exceptions renvoyées par la classe Contoller.

L'exception renvoyée par une **méthode du Controller** est mappée par **une méthode du ControllerAdvice** utilisant **l'annotation @ExceptionHandler.**




Dans le pom.xml, nous avons besoin de la dépendance web :

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

Définir la classe EmployeeServiceException à lancer par la classe EmployeeService en cas d'exception.

```
package cm.belrose.SpringBootExceptionHandle.exception;

public class EmployeeServiceException extends Exception {
    private static final long serialVersionUID = -470180507998010368L;

    public EmployeeServiceException() {
        super();
    }

    public EmployeeServiceException(final String message) {
        super(message);
    }
}
```

```java
package cm.belrose.SpringBootExceptionHandle.services;

import cm.belrose.SpringBootExceptionHandle.entities.Employe;
import
cm.belrose.SpringBootExceptionHandle.exception.EmployeServiceException;
import
cm.belrose.SpringBootExceptionHandle.exception.ResourceNotFoundException;

import java.util.Optional;

public interface EmployeService {
    public Optional<Employe> findEmployeById(Long id) throws
ResourceNotFoundException, EmployeServiceException;
}
```

```java
package cm.belrose.SpringBootExceptionHandle.services;

import cm.belrose.SpringBootExceptionHandle.dao.EmployeDao;
import cm.belrose.SpringBootExceptionHandle.entities.Employe;
import
cm.belrose.SpringBootExceptionHandle.exception.ResourceNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class EmployeServiceImpl implements EmployeService {
    @Autowired
    EmployeDao employeDao;

    @Override
    public Optional<Employe> findEmployeById(Long id) throws
ResourceNotFoundException {

            Optional<Employe> employeFound = employeDao.findById(id);
            if (employeFound.isEmpty()) {
                throw new ResourceNotFoundException("Employee not found");
            }
            return employeFound;

    }
}
```

Définir la classe **ResourceNotFoundException**. Cette exception sera lancée par le **Contrôleur** lorsqu'aucun employé à renvoyer, n'est trouvée.

```java
public class ResourceNotFoundException extends Exception {
    private static final long serialVersionUID = -9079454849611061074L;

    public ResourceNotFoundException() {
        super();
    }

    public ResourceNotFoundException(final String message) {
        super(message);
    }
}
```

Créer la classe Controller **(EmployeController)**. Cette classe contient la requête @GetMapping qui map **/employe/{id}** sur la méthode **findEmployeById(@PathVariable("id") Long id).** Cette dernière retourne un objet **Employe,** un **null** ou **lève une exception.**

```
package cm.belrose.SpringBootExceptionHandle.Controllers;

import cm.belrose.SpringBootExceptionHandle.entities.Employe;
import
cm.belrose.SpringBootExceptionHandle.exception.EmployeServiceException;
import
cm.belrose.SpringBootExceptionHandle.exception.ResourceNotFoundException;
import cm.belrose.SpringBootExceptionHandle.services.EmployeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.util.Optional;

@RestController
public class EmployeeController {
    @Autowired
    private EmployeService employeService;

    @GetMapping(value = "/employe/{id}")
    public ResponseEntity<Employe> findEmployeById(@PathVariable("id") Long
id) throws ResourceNotFoundException, EmployeServiceException {
        Optional<Employe> employeFound = employeService.findEmployeById(id);
        return new ResponseEntity<>(employeFound.get(), HttpStatus.FOUND);

    }

}
```

Définir la classe POJO qui sera envoyée en réponse par la classe ControllerAdvice.

```
package cm.belrose.SpringBootExceptionHandle.exception;

public class ExceptionResponse {
    private String errorMessage;
    private String requestedURI;

    public String getErrorMessage() {
        return errorMessage;
    }
    public void setErrorMessage(final String errorMessage) {
        this.errorMessage = errorMessage;
    }
    public String getRequestedURI() {
        return requestedURI;
    }
    public void callerURL(final String requestedURI) {
        this.requestedURI = requestedURI;
    }
}
```

Le **ExceptionHandlerControllerAdvice** capte les exceptions lancées par la méthode du contrôleur, et nous envoyons une réponse plus appropriée à l'appelant. Par exemple, si une exception se produit, nous ne voulons pas que l'appelant reçoive toute la trace de la pile impliquant des détails techniques. Nous pouvons également renvoyer le statut de réponse approprié en fonction de la logique métier.

```
package cm.belrose.SpringBootExceptionHandle.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import javax.servlet.http.HttpServletRequest;

@RestControllerAdvice
public class ExceptionHandlerControllerAdvice {

    @ExceptionHandler(ResourceNotFoundException.class)
    @ResponseStatus(value = HttpStatus.NOT_FOUND)
    public @ResponseBody ExceptionResponse handleResourceNotFound(final
ResourceNotFoundException exception, final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());
        return error;
    }

    @ExceptionHandler(Exception.class)
    @ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
    public @ResponseBody ExceptionResponse handleException(Exception ex,
                                                           HttpServletRequest req) {

        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(ex.getMessage());
        error.callerURL(req.getRequestURI());

        return error;
    }
}
```