JAVA II

Exception handling in Spring MVC

Exception Handling in Spring MVC

There are three options:

- per exception;
- per controller;
- globally;

Using HTTP Status Codes

By default spring return HTTP 500 response.

By using @ResponseStatus we can send different HTTP status code on response by implementing our exception class

```
@ResponseStatus(value=HttpStatus.NOT_FOUND, reason="No such Order")
// 404
public class OrderNotFoundException extends RuntimeException {
    // ...
}
```

Using HTTP Status Codes

In controller:

```
@RequestMapping(value="/orders/{id}", method=GET)
public String showOrder(@PathVariable("id") long id, Model model) {
    Order order = orderRepository.findOrderById(id);
    if (order == null) throw new OrderNotFoundException(id);
    model.addAttribute(order);
    return "orderDetail";
}
```

Controller Based Exception Handling

By using @ExceptionHandler we can:

1. Handle exceptions without the `@ResponseStatus` annotation (typically predefined exceptions
   that you didn't write)
2. Redirect the user to a dedicated error view
3. Build a totally custom error response

Controller Based Exception Handling

```
@Controller
public class ExceptionHandlingController {
  // Convert a predefined exception to an HTTP Status code
  @ResponseStatus(value=HttpStatus.CONFLICT, reason="Data integrity violation")  // 409
  @ExceptionHandler(DataIntegrityViolationException.class)
  public void conflict() {
    // Nothing to do
  }

  // Specify the name of a specific view that will be used to display the error:
  @ExceptionHandler({SQLException.class,DataAccessException.class})
  public String databaseError() {
    return "databaseError";
  }

  @ExceptionHandler(Exception.class)
  public ModelAndView handleError(HttpServletRequest req, Exception exception) {
    logger.error("Request: " + req.getRequestURL() + " raised " + exception);

    ModelAndView mav = new ModelAndView();
    mav.addObject("exception", exception);
    mav.addObject("url", req.getRequestURL());
    mav.setViewName("error");
    return mav;
  }
}
```

Global Exception Handling

Global exception handling is implemented by using @ControllerAdvce

```
@ControllerAdvice
class GlobalControllerExceptionHandler {
    @ResponseStatus(HttpStatus.CONFLICT)  // 409
    @ExceptionHandler(DataIntegrityViolationException.class)
    public void handleConflict() {
        // Nothing to do
    }
}
```

What to Use When?

1. For exceptions you write, consider adding `@ResponseStatus` to them.
2. For all other exceptions implement an `@ExceptionHandler` method on a `@ControllerAdvice` class
3. For Controller specific exception handling add `@ExceptionHandler` methods to your controller.
4. **Warning:** Be careful mixing too many of these options in the same application. If the same exception can be handed in more than one way, you may not get the behavior you wanted. `@ExceptionHandler` methods on the Controller are always selected before those on any `@ControllerAdvice` instance. It is *undefined* what order controller-advices are processed.