
SPRING FRAMEWORK
GURU

Exception Handling in Spring MVC

HTTP Status Codes

- HTTP 5XX Server Error
  - HTTP 500 - Internal Sever Error
    - Generally, any unhandled exception
  - Other 500 errors are generally not used with Spring MVC

HTTP Status Codes

- HTTP 4XX Client Errors - Generally Checked Exceptions
  - 400 Bad Request - Cannot process due to client error
  - 401 Unauthorized - Authentication required
  - 404 Not Found - Resource Not Found
  - 405 Method Not Allowed - HTTP method not allowed

HTTP Status Codes

- HTTP 4XX Client Errors
  - 409 - Conflict - Possible with simultaneous updates
  - 417 Expectation Failed - Sometimes used with RESTful interfaces
  - 418 - I’m a Teapot - April Fools Joke from IETF (Internet Engineering Task Force) in 1998.

@ResponseStatus

- Allows you to annotate custom exception classes to indicate to the framework the HTTP status you want returned when that exception is thrown.
- Global to the application

@ExceptionHandler

- @ExceptionHandler works at the controller level
- Allows you to define custom exception handling
  - Can be used with @ResponseStatus for just returning a http status
  - Can be used to return a specific view
  - Also can take total control and work with the Model and View
    - ‘Model’ cannot be a parameter of an ExceptionHandler method

HandlerExceptionResolver

- HandlerExceptionResolver is an interface you can implement for custom exception handling
- Used Internally by Spring MVC
- Note Model is not passed

```
public interface HandlerExceptionResolver {

    /**...*/
    @Nullable
    ModelAndView resolveException(
            HttpServletRequest request, HttpServletResponse response, @Nullable Object handler, Exception ex);

}
```

Internal Spring MVC Exception Handlers

- Spring MVC has 3 implementations of HandlerExceptionResolver
- ExceptionHandlerExceptionResolver - Matches uncaught exceptions to @ExceptionHandler
- ResponseStatusExceptionResolver - Looks for uncaught exceptions matching @ResponseStatus
- DefaultHandlerExceptionResolver - Converts standard Spring Exceptions to HTTP status codes (Internal to Spring MVC)

Custom HandlerExceptionResolver

- You can provide your own implementations of HandlerExceptionResolver
- Typically implemented with Spring's Ordered interface to define order the handlers will run in
- Custom implementations are uncommon due to Spring robust exception handling

SimpleMappingExceptionResolver

- A Spring Bean you can define to map exceptions to specific views
- You only define the exception class name (no package) and the view name
- You can optionally define a default error page

Which to Use When?

- Depends on your specific needs
  - If just setting the HTTP status - use @ReponseStatus
  - If redirection to a view, Use SimpleMappingExceptionResolver
  - If both, consider @ExceptionHandler on the controller


SPRING FRAMEWORK
GURU