HandlerExceptionResolver Explanation

give me full details about this Handle Exception Resolver in details so that i can able to understand it well donot have any doubt on this

Below is a **complete, professional-level, no-doubt explanation** of **HandlerExceptionResolver** in **Spring MVC / Spring Boot**.
I'll explain **what it is, why it exists, how it works internally, execution flow, types, priority order, examples, and best practices**.

1️⃣ What is HandlerExceptionResolver?

Definition (simple but accurate):

> **HandlerExceptionResolver** is a Spring MVC mechanism that **intercepts exceptions thrown during request handling and converts them into an appropriate HTTP response** (view or JSON), instead of letting the application crash.

In short:
👉 **Controller throws exception → HandlerExceptionResolver handles it → Proper response is sent to client**

2️⃣ Why do we need HandlerExceptionResolver?

Without it:

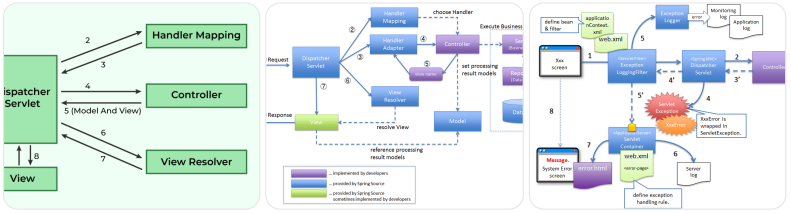
- Any exception in controller would:
  - Break request flow
  - Return ugly stack traces
  - Expose internal details
  - Be inconsistent across APIs

With it:

- Centralized exception handling
- Clean HTTP status codes (`400` , `404` , `500`)
- Clean JSON error responses
- Separation of **business logic** and **error handling**

3️⃣ Where does it fit in Spring MVC flow?

Let's first see the **big picture**.



🔁 Normal Request Flow

```
arduino

Client
  ↓
DispatcherServlet
  ↓
HandlerMapping
  ↓
Controller
  ↓
Response
```

❌ When Exception Occurs

```
javascript

Controller
  ↓ (Exception)
DispatcherServlet
  ↓
HandlerExceptionResolver
  ↓
Error Response
```

4️⃣ How DispatcherServlet uses HandlerExceptionResolver

Internally, Spring Framework works like this:

```java
try {
    handler.handle(request, response);
} catch (Exception ex) {
    for (HandlerExceptionResolver resolver : resolvers) {
        ModelAndView mav = resolver.resolveException(request, response,
handler, ex);
        if (mav != null) {
            return mav;
        }
    }
    throw ex;
}
```

🔍 Key Points

- DispatcherServlet maintains a **list of HandlerExceptionResolvers**
- They are executed **in order**
- First resolver that handles the exception **wins**
- If none handle it → exception propagates

5️⃣ Core Method: `resolveException()`

```java
ModelAndView resolveException(
    HttpServletRequest request,
    HttpServletResponse response,
    Object handler,
    Exception ex
);
```

What each parameter means:

Parameter	Meaning
`request`	Incoming HTTP request

Parameter	Meaning
`response`	HTTP response to write into
`handler`	Controller method that failed
`ex`	The actual exception

Return Value:

- **ModelAndView** → exception handled
- **null** → try next resolver

6️⃣ Built-in HandlerExceptionResolvers (IMPORTANT)

Spring Boot automatically registers **three main resolvers** 👇

1️⃣ ExceptionHandlerExceptionResolver

✔ **MOST IMPORTANT**

Handles:

- `@ExceptionHandler`
- `@ControllerAdvice`
- `@RestControllerAdvice`

Example:

```java
@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(NullPointerException.class)
    public ResponseEntity<String> handleNPE() {
        return ResponseEntity
                .status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body("Null pointer error");
```

```
    }
}
```

➡️ This resolver:

- Finds matching `@ExceptionHandler`
- Builds response
- Stops further processing

2️⃣ ResponseStatusExceptionResolver

Handles:

- `@ResponseStatus`
- `ResponseStatusException`

Example:

```java
@ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeException {}
```

or

```java
throw new ResponseStatusException(
        HttpStatus.BAD_REQUEST,
        "Invalid input"
);
```

➡️ Automatically maps exception → HTTP status

3️⃣ DefaultHandlerExceptionResolver

Handles **framework-level exceptions**:

Exception	HTTP Status
`HttpRequestMethodNotSupportedException`	405

Exception	HTTP Status
`HttpMediaTypeNotSupportedException`	415
`MissingServletRequestParameterException`	400
`TypeMismatchException`	400

➡️ You rarely interact with this directly

7️⃣ Execution Order (VERY IMPORTANT)

Spring executes resolvers in this order:

1. `ExceptionHandlerExceptionResolver`
2. `ResponseStatusExceptionResolver`
3. `DefaultHandlerExceptionResolver`

⚠️ **Why order matters?**

Because once one resolver handles the exception, others are skipped.

8️⃣ Custom HandlerExceptionResolver (Advanced)

You can write your own resolver:

```java
@Component
@Order(1)
public class CustomExceptionResolver implements HandlerExceptionResolver {

    @Override
    public ModelAndView resolveException(
            HttpServletRequest request,
            HttpServletResponse response,
            Object handler,
```

```
            Exception ex) {

        if (ex instanceof IllegalArgumentException) {
            response.setStatus(HttpStatus.BAD_REQUEST.value());
            return new ModelAndView("error/400");
        }
        return null;
    }
}
```

When to use this?

- ✔ Cross-cutting legacy handling
- ✔ Custom logging
- ✔ Special routing logic

❌ **Not recommended for REST APIs** (use `@RestControllerAdvice` instead)

9️⃣ REST APIs vs MVC Views

For REST APIs (JSON)

✔ Use:

- `@RestControllerAdvice`
- `@ExceptionHandler`
- `ResponseEntity`

For MVC (HTML)

✔ Use:

- `ModelAndView`
- Error pages

🔟 Relationship with Filters & Interceptors

Component	Handles Exceptions?
Filter	❌ (before DispatcherServlet)

Component	Handles Exceptions?
Interceptor	❌ (must rethrow)
Controller	❌
HandlerExceptionResolver	✅

⚠️ **Important:**

Exceptions thrown in **Filters** are NOT handled by HandlerExceptionResolver.

1️⃣1️⃣ Best Practices (Industry Level)

✅ Use **@RestControllerAdvice** for APIs

✅ Define **custom exception classes**

✅ Return **structured error response**

```json
{
  "timestamp": "2026-01-21",
  "status": 400,
  "error": "Bad Request",
  "message": "Invalid email format"
}
```

✅ Never expose stack traces in production

✅ Log exception internally, return safe message externally

1️⃣2️⃣ Summary (One-Glance)

Concept	Meaning
HandlerExceptionResolver	Central exception handler
Triggered by	DispatcherServlet

Concept	Meaning
Purpose	Convert exception → response
Order matters	Yes
Best tool for REST	`@RestControllerAdvice`
Handles filter errors	❌ No

🔥 Final Mental Model

> **HandlerExceptionResolver is the safety net of Spring MVC**
> It ensures that **every exception becomes a meaningful HTTP response instead of a crash**.

If you want, I can next explain:

- **Difference between `@ControllerAdvice` vs `HandlerExceptionResolver`**
- **Real-world production error handling design**
- **How Spring Boot auto-configures exception handling**

Just tell me 👍