Spring Boot API Response and Error Handling

1. Standard Response Model

```
public class ApiResponse<T> {
    private boolean success;
    private String message;
    private T data;
    private ApiError error;

    // Getters and Setters

    public boolean isSuccess() {
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
```

```
    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

    public ApiError getError() {
        return error;
    }

    public void setError(ApiError error) {
        this.error = error;
    }
}
```

2. Error Model

```
public class ApiError {
    private String errorCode;
    private String errorMessage;
    private List<String> details;
```

```
// Getters and Setters


public String getErrorCode() {

    return errorCode;

}


public void setErrorCode(String errorCode) {

    this.errorCode = errorCode;

}


public String getErrorMessage() {

    return errorMessage;

}


public void setErrorMessage(String errorMessage) {

    this.errorMessage = errorMessage;

}


public List<String> getDetails() {

    return details;

}


public void setDetails(List<String> details) {

    this.details = details;

}
```

```
}
```

3. Global Exception Handler

```
@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(Exception.class)
    public ResponseEntity<ApiResponse<Object>> handleException(Exception ex) {
        ApiError apiError = new ApiError();
        apiError.setErrorCode("500");
        apiError.setErrorMessage(ex.getMessage());
        apiError.setDetails(Collections.singletonList(ex.toString()));

        ApiResponse<Object> response = new ApiResponse<>();
        response.setSuccess(false);
        response.setMessage("An error occurred");
        response.setError(apiError);

        return new ResponseEntity<>(response, HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @ExceptionHandler(ResourceNotFoundException.class)
                                    public          ResponseEntity<ApiResponse<Object>>
handleResourceNotFoundException(ResourceNotFoundException ex) {
```

```
        ApiError apiError = new ApiError();
        apiError.setErrorCode("404");
        apiError.setErrorMessage(ex.getMessage());
        apiError.setDetails(Collections.singletonList(ex.toString()));

        ApiResponse<Object> response = new ApiResponse<>();
        response.setSuccess(false);
        response.setMessage("Resource not found");
        response.setError(apiError);

        return new ResponseEntity<>(response, HttpStatus.NOT_FOUND);
    }

    // Add more handlers for other specific exceptions as needed
}
```

4. Controller Example

```
@RestController
@RequestMapping("/api")
public class ExampleController {

    @GetMapping("/test")
    public ResponseEntity<ApiResponse<String>> testApi() {
        ApiResponse<String> response = new ApiResponse<>();
```

```
        response.setSuccess(true);
        response.setMessage("Request was successful");
        response.setData("Test Data");

        return ResponseEntity.ok(response);
    }

    @GetMapping("/error")
    public ResponseEntity<ApiResponse<Object>> testError() throws Exception {
        throw new Exception("Test exception");
    }

    @GetMapping("/not-found")
                public    ResponseEntity<ApiResponse<Object>>    testNotFound()    throws
ResourceNotFoundException {
        throw new ResourceNotFoundException("Resource not found");
    }
}
```

5. Custom Exception

```
public class ResourceNotFoundException extends RuntimeException {
    public ResourceNotFoundException(String message) {
        super(message);
    }
```

```
}
```