OpenLMIS Error Handling Scheme

Spring conventions

Spring MVC provides several approaches to exception handling. It is possible to use both described options or to choose only one of them.

Controller based exception handling
In this approach exception handler methods should be defined in controller class by annotating them with @ExceptionHandler annotation. Such methods apply to exceptions raised by @RequestMapping methods of that controller (or any of its subclasses). An exception handler method can be also annotated with @ResponseStatus annotation. Then the status code is applied to the HTTP response when the handler method is invoked and overrides status information set by other means.
Good solution for this approach is to create one abstract controller per module that would define exceptions handlers. Other controllers from given module would extend this abstract controller.

Global Exception Handling
This approach allows to create an exception handler methods that will apply to exceptions raised across the whole application, not just by an individual controller. Any class with @ControllerAdvice annotation becomes a controller-advice. Exception handling methods are annotated with @ExceptionHadler and optionally with @ResponseStatus.
The main difference from the first approach is that using @ControllerAdvice allows to have a centralized way of handling exceptions, because it applies to all defined controllers.

My suggestion is to mix those two approaches. It might be a good solution to create one abstract controller per module to handle exceptions raised by controllers from certain module and use @ControllerAdvice to handle exceptions globally. @ExceptionHandler methods in the controller are always selected before those in @ControllerAdvice instance (meaning, that it is possible to override the global behaviour).

Basic coding practices related with Java exception handling

Throw early, catch late
Throw exception as soon as you can, and catch it late as much possible. You should wait until you have all the information to handle it properly.

Avoid empty catch blocks
Leaving an empty catch block leads to situation where a program behaves oddly or incorrectly but there is no sign that anything went wrong. Any catch block should always attempt to log an exception or throw the exception up to the caller of your method.

Either log the exception or throw it but never do both
Logging and throwing an exception will result in multiple log messages in log files for a single problem in the code. Exceptions should be logged only when they are handled.

Do not throw and catch generic Exception or Throwable class
Always throw and catch specific exception classes so that caller will know the root cause of exception easily. This makes debugging easy and makes it easier to handle exceptions appropriately.

Correctly wrap the exceptions in custom exceptions
Wrap the exceptions in a correct way so that the stack trace is not lost. When wrapping an exception, pass the original exception as a *cause* of a new exception.

Use custom exceptions
Create a specific exception with meaningful name for a certain problem, instead of using the same exception for multiple problems. It is also good to create a simple hierarchy - create an exception that other exceptions from the certain module can extend.

Automatic style checking

Checks that should be applied to Checkstyle to support error handling standard:

- EmptyBlock - checks for empty blocks.
- IllegalThrows - checks that certain exception types are not declared to be thrown.
- IllegalCatch - checks that certain exception types do not appear in catch statement.

My suggestion is to also use PMD plugin to perform additional quality checks.
Rules that should be applied to support our error handling standard:

- AvoidThrowingRawExceptionTypes: Avoid throwing certain exception types. Rather than throw a raw RuntimeException, Throwable, Exception, or Error, use a subclassed exception or error instead.
- PreserveStackTrace: Throwing a new exception from a catch block without passing the original exception into the new exception will cause the true stack trace to be lost, and can make it difficult to debug effectively.