**Exception Handling in Web API 2**

* Some unhandled exceptions can be processed via [exception filters](https://www.asp.net/web-api/overview/web-api-routing-and-actions/exception-handling), but there are a number of cases that exception filters can’t handle. For example:

1. Exceptions thrown from controller constructors.
2. Exceptions thrown from message handlers.
3. Exceptions thrown during routing.
4. Exceptions thrown during response content serialization

* There are two major cases for handling exceptions, the case where we are able to send an error response and the case where all we can do is log the exception. An example for the latter case is when an exception is thrown in the middle of streaming response content; in that case it is too late to send a new response message since the status code, headers, and partial content have already gone across the wire, so we simply abort the connection. Even though the exception can’t be handled to produce a new response message, we still support logging the exception.
* We provide two new user-replaceable services, [IExceptionLogger](https://www.asp.net/web-api/overview/releases/whats-new-in-aspnet-web-api-21) and IExceptionHandler, to log and handle unhandled exceptions. The services are very similar, with two main differences:

1. We support registering multiple exception loggers but only a single exception handler.

2. Exception loggers always get called, even if we’re about to abort the connection. Exception handlers only get called when we’re still able to choose which response message to send.

* Referenced URLS :

1. <https://www.asp.net/web-api/overview/error-handling/web-api-global-error-handling>
2. <https://www.exceptionnotfound.net/the-asp-net-web-api-exception-handling-pipeline-a-guided-tour/>