Advanced Java CompletableFuture Features:
Handling Runtime Exceptions (Part 1)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

• Understand advanced features of completable futures, e.g.
  • Factory methods initiate async computations
  • Completion stage methods chain together actions to perform async result processing & composition
    • Method grouping
    • Single stage methods
    • Two stage methods (and)
    • Two stage methods (or)
    • Apply these methods
  • Handle runtime exceptions (Overview)
Overview of Handling Exceptions in Completion Stages
Overview of Handling Exceptions in Completion Stages

- Exception handling is more complex for asynchronous computations than for synchronous computations

See blog.lightstreamer.com/2014/07/exception-handling-in-asynchronous-java.html
Exception handling is more complex for asynchronous computations than for synchronous computations, e.g.,

- The conventional exception handling model propagates exceptions up the runtime call stack synchronously.

See [en.wikipedia.org/wiki/Exception_handling](en.wikipedia.org/wiki/Exception_handling)
Overview of Handling Exceptions in Completion Stages

- Exception handling is more complex for asynchronous computations than for synchronous computations, e.g.,
  - The conventional exception handling model propagates exceptions up the runtime call stack synchronously
  - However, completable futures that run asynchronously don’t conform to a conventional call stack model
Overview of Handling Exceptions in Completion Stages

- Exception handling is more complex for asynchronous computations than for synchronous computations, e.g.,
  - The conventional exception handling model propagates exceptions up the runtime call stack synchronously
  - However, completable futures that run asynchronously don’t conform to a conventional call stack model
    - e.g., completion stage methods can run in different worker threads!

See suryanarayanjena.wordpress.com/async-methods-in-completablefuture
Overview of Handling Exceptions in Completion Stages

- Completion stage methods handle runtime exceptions that occur asynchronously

**Overview of Handling Exceptions in Completion Stages**

- Completion stage methods handle runtime exceptions that occur asynchronously.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params</th>
<th>Returns</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>when Complete</code></td>
<td>Bi Consumer</td>
<td>Completable Future with result of earlier stage or throws exception</td>
<td>Handle outcome of a stage, whether a result value or an exception</td>
</tr>
<tr>
<td><code>(Async)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>handle</code></td>
<td>Bi Function</td>
<td>Completable Future with result of BiFunction</td>
<td>Handle outcome of a stage &amp; return new value</td>
</tr>
<tr>
<td><code>(Async)</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>exceptionally</code></td>
<td>Function</td>
<td>Completable Future&lt;T&gt;</td>
<td>When exception occurs, replace exception with result value</td>
</tr>
</tbody>
</table>

Help make programs more *resilient* by handling erroneous computations gracefully.
### Overview of Handling Exceptions in Completion Stages

- Completion stage methods handle runtime exceptions that occur asynchronously.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params</th>
<th>Returns</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>when Complete</td>
<td>Bi Consumer</td>
<td>Completable Future</td>
<td>Handle outcome of a stage, whether a result value or an exception.</td>
</tr>
<tr>
<td>(Async)</td>
<td></td>
<td>with result of earlier stage</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>or throws exception.</td>
<td></td>
</tr>
<tr>
<td>handle</td>
<td>Bi Function</td>
<td>Completable Future</td>
<td>Handle outcome of a stage &amp; return new value.</td>
</tr>
<tr>
<td>(Async)</td>
<td></td>
<td>with result of BiFunction</td>
<td></td>
</tr>
<tr>
<td>exceptionally</td>
<td>Function</td>
<td>Completable Future&lt;T&gt;</td>
<td>When exception occurs, replace exception with result value.</td>
</tr>
<tr>
<td>(Async)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Added in Java 12*

See [www.logicbig.com/tutorials/core-java-tutorial/java-12-changes/completion-stage-new-methods.html](http://www.logicbig.com/tutorials/core-java-tutorial/java-12-changes/completion-stage-new-methods.html)