Enhancing Java Completable Futures Framework Extensibility (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

• Evaluate the pros of using the Java completable futures framework
• Evaluate the cons of using the Java completable futures framework
• Understand enhancements to the Java completable futures framework
  • Enhanced timeout handling
  • Enhancing extensibility
Enhancing Java Completable Future Extensibility
Enhancing Java CompletableFuture Extensibility

- Java 9 enhances the Java 8 completable future framework with new methods that support custom Executor implementations

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default Executor</td>
<td>()</td>
<td>Returns default Executor used for methods that don’t specify an Executor</td>
</tr>
<tr>
<td>new Incomplete Future</td>
<td>()</td>
<td>Returns a new CompletableFuture that will be returned by a CompletionStage method</td>
</tr>
<tr>
<td>complete Async</td>
<td>Supplier&lt;T&gt;</td>
<td>Complete the CompletableFuture asynchronously using the value given by the Supplier</td>
</tr>
</tbody>
</table>
Enhancing Java CompletableFuture Extensibility

- Java 9 enhances the Java 8 completable future framework with new methods that support custom Executor implementations

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>default Executor</td>
<td>()</td>
<td>Returns default <code>Executor</code> used for methods that don’t specify an <code>Executor</code></td>
</tr>
<tr>
<td>new Incomplete Future</td>
<td>()</td>
<td>Returns a new <code>CompletableFuture</code> that will be returned by a <code>CompletionStage</code> method</td>
</tr>
<tr>
<td>complete Async</td>
<td><code>Supplier&lt;T&gt;</code></td>
<td>Complete the <code>CompletableFuture</code> asynchronously using the value given by the <code>Supplier</code></td>
</tr>
</tbody>
</table>
Enhancing Java CompletableFuture Extensibility

- Java 9 enhances the Java 8 completable future framework with new methods that support custom Executor implementations

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>default Executor</code></td>
<td>() Executor</td>
<td>Returns default <code>Executor</code> used for methods that don’t specify an <code>Executor</code></td>
</tr>
<tr>
<td><code>new Incomplete Future</code></td>
<td>() Completable Future&lt;T&gt;</td>
<td>Returns a new <code>CompletableFuture</code> that will be returned by a <code>CompletionStage</code> method</td>
</tr>
<tr>
<td><code>complete Async</code></td>
<td>Supplier&lt;T&gt; Completable Future&lt;T&gt;</td>
<td>Complete the <code>CompletableFuture</code> asynchronously using the value given by the <code>Supplier</code></td>
</tr>
</tbody>
</table>

See [docs.oracle.com/javase/9/docs/api/java/util/concurrent/CompletableFuture.html#newIncompleteFuture](docs.oracle.com/javase/9/docs/api/java/util/concurrent/CompletableFuture.html#newIncompleteFuture)
Enhancing Java CompletableFuture Extensibility

- Java 9 enhances the Java 8 completable future framework with new methods that support custom Executor implementations

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>default Executor</code></td>
<td>()</td>
<td>Returns default <code>Executor</code> used for methods that don’t specify an <code>Executor</code></td>
</tr>
<tr>
<td><code>new Incomplete Future</code></td>
<td>()</td>
<td>Returns a new <code>CompletableFuture</code> that will be returned by a <code>CompletionStage</code> method</td>
</tr>
<tr>
<td><code>complete Async</code></td>
<td><code>Supplier&lt;T&gt;</code></td>
<td>Complete the <code>CompletableFuture</code> asynchronously using the value given by the <code>Supplier</code></td>
</tr>
</tbody>
</table>

See [docs.oracle.com/javase/9/docs/api/java/util/concurrent/CompletableFuture.html#completeAsync](http://docs.oracle.com/javase/9/docs/api/java/util/concurrent/CompletableFuture.html#completeAsync)
Extending the Java Completable Future Framework
Extending the Java CompletableFuture Framework

- Customize the completable futures framework to use virtual threads by default

```java
class CompletableFutureEx<T> extends CompletableFuture<T> {
    private static Executor sEXEC = Executors.
        newVirtualThreadPerTaskExecutor();

    public Executor defaultExecutor() { return sEXEC; }

    public <T> CompletableFuture<T> newIncompleteFuture()
        { return new CompletableFutureEx<>(); }

    public static <T> CompletableFuture<T>
        supplyAsync(Supplier<T> supplier) {
        return new CompletableFutureEx<T>().completeAsync(supplier);
    }
}
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Loom/ex6](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Loom/ex6)
Extending the Java CompletableFuture Framework

- Customize the completable futures framework to use virtual threads by default with the following class definition:

```java
class CompletableFutureEx<T> extends CompletableFuture<T> {
    private static Executor sEXEC = Executors.newVirtualThreadPerTaskExecutor();

    public Executor defaultExecutor() { return sEXEC; } 

    public <T> CompletableFuture<T> newIncompleteFuture() { return new CompletableFutureEx<>(); }

    public static <T> CompletableFuture<T> supplyAsync(Supplier<T> supplier) { 
        return new CompletableFutureEx<T>().completeAsync(supplier); 
    }
}
```

Customization requires the use of inheritance.
Extending the Java CompletableFuture Framework

- Customize the completable futures framework to use virtual threads by default

```java
class CompletableFutureEx<T> extends CompletableFuture<T> {
    private static Executor sEXEC = Executors.newVirtualThreadPerTaskExecutor();

    public Executor defaultExecutor() { return sEXEC; }

    public <T> CompletableFuture<T> newIncompleteFuture() {
        return new CompletableFutureEx<>();
    }

    public static <T> CompletableFuture<T> supplyAsync(Supplier<T> supplier) {
        return new CompletableFutureEx<T>().completeAsync(supplier);
    }
    ...
}
```

This Executor creates a virtual thread per task
Extending the Java CompletableFuture Framework

- Customize the completable futures framework to use virtual threads by default

```java
class CompletableFutureEx<T> extends CompletableFuture<T> {
    private static Executor sEXEC = Executors.newVirtualThreadPerTaskExecutor();

    public Executor defaultExecutor() { return sEXEC; }

    public <T> CompletableFuture<T> newIncompleteFuture()
    { return new CompletableFutureEx<>(); }

    public static <T> CompletableFuture<T>
        supplyAsync(Supplier<T> supplier) {
        return new CompletableFutureEx<T>().completeAsync(supplier);
    }
}
```

Return the default Executor
Extending the Java CompletableFuture Framework

- Customize the completable futures framework to use virtual threads by default.

```java
class CompletableFutureEx<T> extends CompletableFuture<T> {
    private static Executor sEXEC = Executors.newVirtualThreadPerTaskExecutor();

    public Executor defaultExecutor() { return sEXEC; }

    public <T> CompletableFuture<T> newIncompleteFuture()
    { return new CompletableFutureEx<>(); }

    public static <T> CompletableFuture<T> supplyAsync(Supplier<T> supplier) {
        return new CompletableFutureEx<T>().completeAsync(supplier);
    }
    ...
}
```

Factory method that creates the subclass
Extending the Java CompletableFuture Framework

- Customize the completable futures framework to use virtual threads by default

```java
class CompletableFutureEx<T> extends CompletableFuture<T> {
    private static Executor sEXEC = Executors.newVirtualThreadPerTaskExecutor();

    public Executor defaultExecutor() { return sEXEC; }

    public <T> CompletableFuture<T> newIncompleteFuture() {
        return new CompletableFutureEx<>();
    }

    public static <T> CompletableFuture<T> supplyAsync(Supplier<T> supplier) {
        return new CompletableFutureEx<T>().completeAsync(supplier);
    }
}
```

Submit supplier to run asynchronously
Extending the Java CompletableFuture Framework

- Customize the completable futures framework to use virtual threads by default

class CompletableFutureEx<T> extends CompletableFuture<T> {
    private static Executor sEXEC = Executors.
        newVirtualThreadPerTaskExecutor();

    public Executor defaultExecutor() { return sEXEC; }

    public <T> CompletableFuture<T> newIncompleteFuture()
    { return new CompletableFutureEx<>(); }

    public static <T> CompletableFuture<T> supplyAsync(Supplier<T> supplier) {
        return new CompletableFutureEx<T>().completeAsync(supplier);
    } ...
End of Enhancing Java Completable Futures Framework Extensibility (Part 1)