

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

Methods	Params		
<code>default Executor</code>	<code>()</code>	<code>Executor</code>	Returns default <i>Executor</i> used for methods that don't specify an <i>Executor</i>
<code>new Incomplete Future</code>	<code>()</code>	<code>Completable Future<T></code>	Returns a new <i>CompletableFuture</i> that will be returned by a <i>CompletionStage</i> method
<code>complete Async</code>	<code>Supplier<T></code>	<code>Completable Future<T></code>	Complete the <i>CompletableFuture</i> asynchronously using the value given by the <i>Supplier</i>

See www.baeldung.com/java-9-completablefuture

Enhancing Java CompletableFuture Extensibility

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

Methods	Params		
default Executor	()	Executor	Returns default <i>Executor</i> used for methods that don't specify an <i>Executor</i>
new Incomplete Future	()	Completable Future<T>	Returns a new <i>CompletableFuture</i> that will be returned by a <i>CompletionStage</i> method
complete Async	Supplier<T>	Completable Future<T>	Complete the <i>CompletableFuture</i> asynchronously using the value given by the <i>Supplier</i>

See docs.oracle.com/javase/9/docs/api/java/util/concurrent/CompletableFuture.html#defaultExecutor

Enhancing Java CompletableFuture Extensibility

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

Methods	Params		
default Executor	()	Executor	Returns default <i>Executor</i> used for methods that don't specify an <i>Executor</i>
new Incomplete Future	()	Completable Future<T>	Returns a new <i>CompletableFuture</i> that will be returned by a <i>CompletionStage</i> method
complete Async	Supplier<T>	Completable Future<T>	Complete the <i>CompletableFuture</i> asynchronously using the value given by the <i>Supplier</i>

Enhancing Java CompletableFuture Extensibility

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

Methods	Params		
default Executor	()	Executor	Returns default <i>Executor</i> used for methods that don't specify an <i>Executor</i>
new Incomplete Future	()	Completable Future<T>	Returns a new <i>CompletableFuture</i> that will be returned by a <i>CompletionStage</i> method
complete Async	Supplier<T>	Completable Future<T>	Complete the <i>CompletableFuture</i> asynchronously using the value given by the <i>Supplier</i>

See 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

```
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);  
        } ...
```

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();
```

*Customization requires
the use of inheritance*

```
    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);  
        } ...
```

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();
```

*This Executor creates a
virtual thread per task*

```
    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);  
        } ...
```

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();
```

*Return the
default Executor*

```
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);  
} ...
```

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);  
        } ...
```

*Factory method that
creates the subclass*

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);  
    } ...
```

*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);  
        } ...
```

*Arrange to run the
supplier in the
default Executor*

End of Enhancing Java Completable Futures Framework Extensibility (Part 1)