Overview of the Project Reactor AsyncTaskBarrier Framework

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

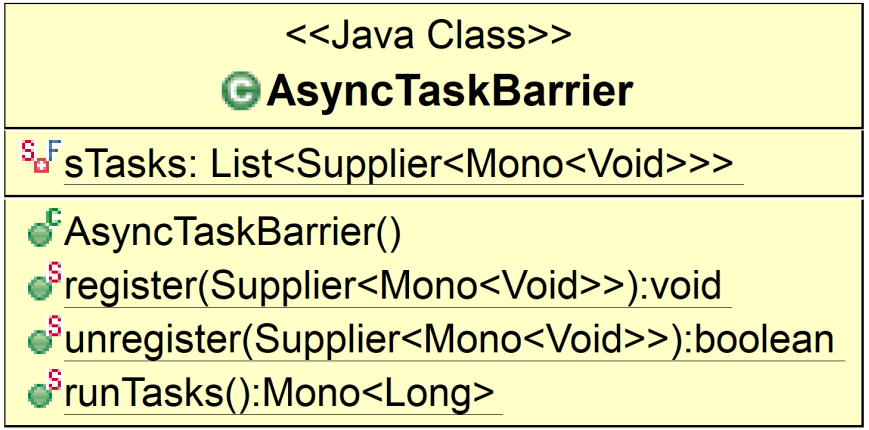
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand key classes in the Project Reactor API
- Be aware of the structure & functionality of the BigFraction case studies
- Recognize the capabilities of the AsyncTaskBarrier framework for Project Reactor



There are implementations for both Project Reactor & RxJava

Overview of the Project Reactor AsyncTaskBarrier Class

Overview of the Project Reactor AsyncTaskBarrier Class

- Most test methods in the BigFraction case studies run asynchronously via subscribeOn(), so these methods return before their computations complete

```
public static Mono<Void> testFractionReductionAsync() {
  BigFraction unreducedFraction = makeBigFraction(...);
  ...
  return Mono
    .fromCallable(() -> BigFraction.reduce(unreducedFraction))
    .subscribeOn(Schedulers.single())
    .map(result -> result.toMixedString())
    .doOnSuccess(result ->
                 System.out.println
                  ("big fraction = "
                  + result + "\n"))
    .then();
```

See Reactive/Mono/ex2/src/main/java/MonoEx.java

Overview of the Project Reactor AsyncTaskBarrier Class

- It's therefore helpful to define a single location in the main driver program that waits for all asynchronously executing test methods to complete

```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(MonoEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .block();
  ...
}
```

See Reactive/Mono/ex2/src/main/java/ex2.java

Overview of the Project Reactor AsyncTaskBarrier Class

- The AsyncTaskBarrier class provides an API to register non-blocking task methods that run *asynchronously*

We use the registered task methods to run async tests

```
public static void main (String[] argv)
  AsyncTaskBarrier
    .register(MonoEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .block();
  ...
}
```

Overview of the Project Reactor AsyncTaskBarrier Class

- The AsyncTaskBarrier class provides an API to register non-blocking task methods that run *asynchronously*

```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(MonoEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .block();
  ...
}
```

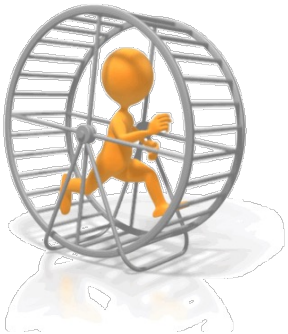
This framework also handles task methods that run and/or block *synchronously*

Overview of the Project Reactor AsyncTaskBarrier Class

- All of the registered task methods start running (a)synchronously when AsyncTaskBarrier.runTasks() is called

```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(MonoEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .block();
  ...
}
```



Overview of the Project Reactor AsyncTaskBarrier Class

- The driver program then calls block() on the Mono returned from runTasks() to wait for all asynchronous task processing to complete

```
public static void main (String[] argv) ... {
  AsyncTaskBarrier
    .register(MonoEx::testFractionReductionAsync);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable1);
  AsyncTaskBarrier
    .register(MonoEx::testFractionMultiplicationCallable2);

  long testCount = AsyncTaskBarrier
    .runTasks()
    .block();
  ...
}
```

Plays the role of a barrier synchronizer

See en.wikipedia.org/wiki/Barrier_(computer_science)

Overview of the Project Reactor AsyncTaskBarrier Class

- The AsyncTaskBarrier framework (a)synchronously registers/runs tasks & ensures the calling method doesn't exit until all async processing completes

Class AsyncTaskBarrier

```
public class AsyncTaskBarrier
extends java.lang.Object
```

This class asynchronously runs tasks that use the Project Reactor framework and ensures that the calling method doesn't exit until all asynchronous task processing is completed.

Method Summary

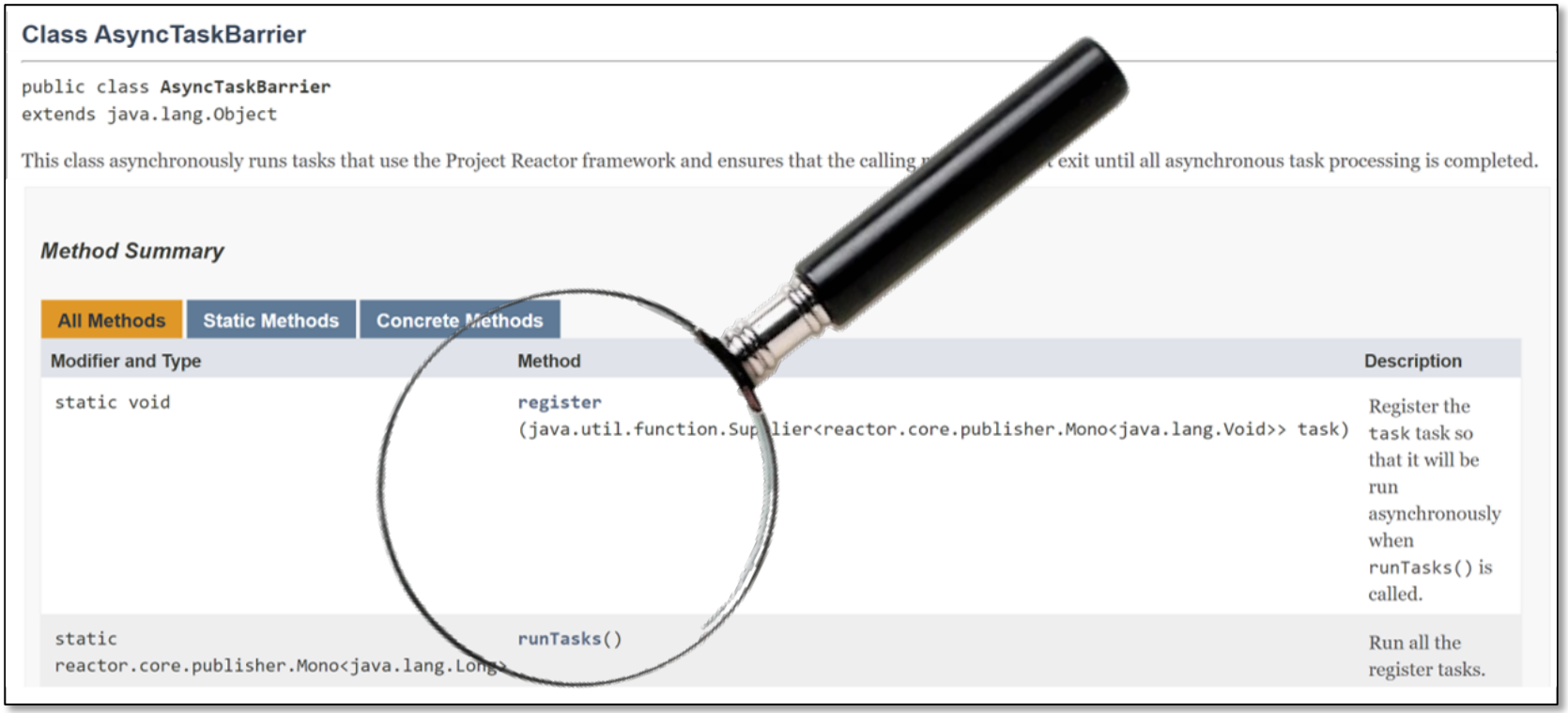
All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	register (java.util.function.Supplier<reactor.core.publisher.Mono<java.lang.Void>> task)	Register the task task so that it will be run asynchronously when runTasks() is called.
static reactor.core.publisher.Mono<java.lang.Long>	runTasks()	Run all the register tasks.

See Reactive/mono/ex1/src/main/java/utils/AsyncTaskBarrier.java

Overview of the Project Reactor AsyncTaskBarrier Class

- The AsyncTaskBarrier framework (a)synchronously registers/runs tasks & ensures the calling method doesn't exit until all async processing completes



Class AsyncTaskBarrier

public class **AsyncTaskBarrier**
extends java.lang.Object

This class asynchronously runs tasks that use the Project Reactor framework and ensures that the calling … exit until all asynchronous task processing is completed.

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	register (java.util.function.Supplier<reactor.core.publisher.Mono<java.lang.Void>> task)	Register the task task so that it will be run asynchronously when runTasks() is called.
static reactor.core.publisher.Mono<java.lang.Long>	runTasks()	Run all the register tasks.

We'll explore AsyncTaskBarrier's implementation after covering Reactor in detail

End of Overview of the Project Reactor Async TaskBarrier Framework