```
=====================================
Reactive Programming in Spring Boot
=====================================

1) What is Thread Per Request Model

                => In server Thread Pool will be available to handle incoming requests

                => For every request one thread will be used to handle that

                => Until The request processing got completed our thread will be blocked...

                => In this approach waiting period will be increased.

Note: To overcome this problem, we are using Reactive Programming


2) What is Reactive Programming

        => It is an approach that uses asynchronus & Non-Blocking execution

        => In this approch our threads will not be blocked

        => Reactive Programming will work based on Event driven approach

        => Reactive Programming Uses Back Pressure mechanism to ensure producers don't overburden
consumers.


=> Using reactive programming we can process multiple requests asynchronusly with Non-Blocking
technique.

==============================
Reactive Programming Features
==============================

1) New Programming paradigm

2) Asynchronus & Non-Blocking

3) Functional Style of code

4) Data Flow as event stream

5) Back Pressue


=========================
Spring Boot Web Flux
=========================

=> Spring web flux is a reactive programming model introduced in spring 5.x version

=> Spring Web Flux supports asynchrons and non-blocking event driven architecture for building web
application.

=> It enables developers to build scalable application which can handle loads of traffic without
compromising on performance.

=> Spring Web Flux will provide netty as default embedded container.

==================================
Reactive Programming Components
```

==================================

=> Reactive Programming works based on publisher and subscriber model

=> In Reactive programming mainley we have below 2 publishers

Mono : It can produce only one value at a time (0 or 1)

Flux : It can produce zero or more values (0 to N)

==

```
public class MonoFluxPublisherTest {

	// @Test
	public void testMono() {
		// publisher => publishes the content
		Mono<String> mono = Mono.just("ashokit").log();

		// subscriber => Will consume data from publisher
		mono.subscribe(System.out::println);
	}

	@Test
	public void testFlux() {
		// publisher => publishes the content
		Flux<String> flux = Flux.just("java", "programming", "language").log();

		// subscribe to publisher to get data
		flux.subscribe(System.out::println);
	}
}
```
===

===
Reactive Programming working based on Events
===

onSubscribe ( )

request ( )

onNext ( )

onComplete ( )

onError ( )

==

Spring Boot + Reactive App :: https://github.com/ashokitschool/Springboot_Reactive_App.git

==