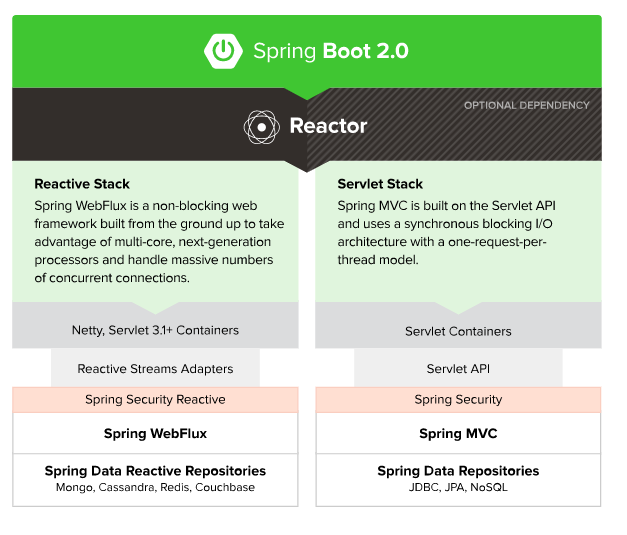
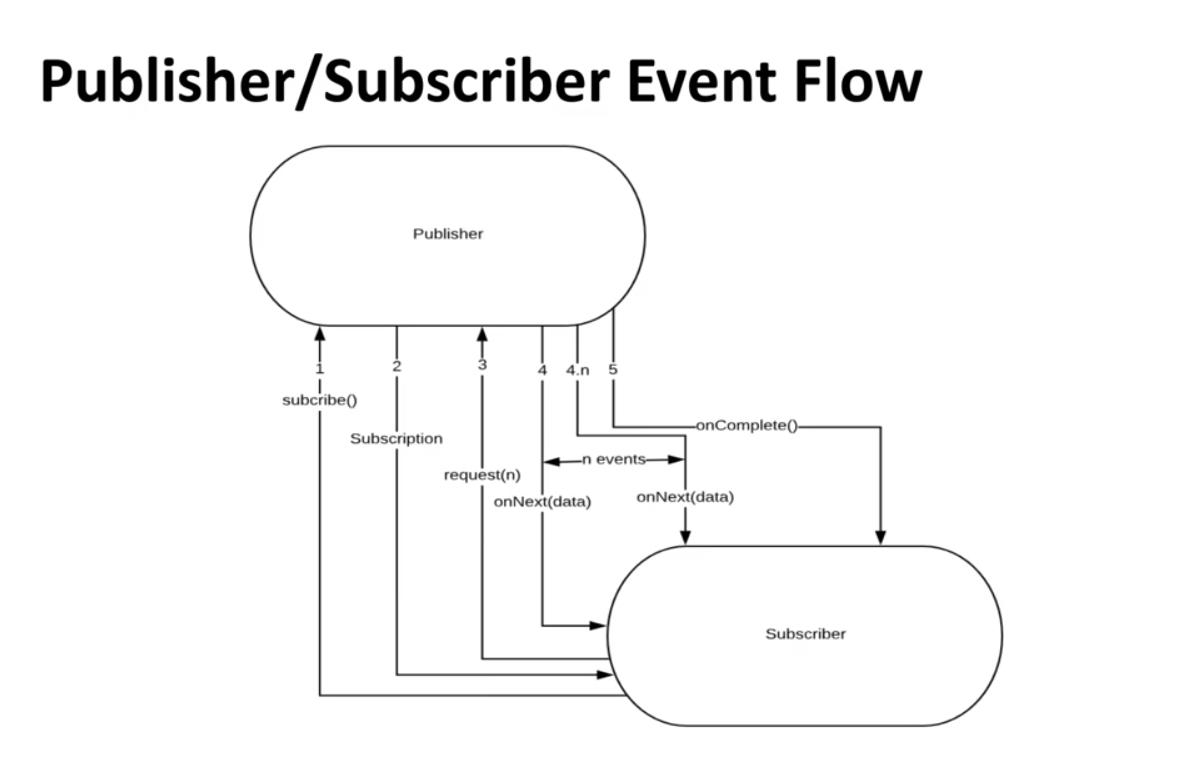
**REACTIVE STREAM SPECIFICATION **

**There are 2 ways of creating Reactive API**

1. Using RestController
2. Using Functional programming of URL hander mapping

**There are 2 types of Reactor**

1. Mono for API that expose 0 or 1 element. Eg: Just only one Employee object then choose MONO.
2. Flux for API that expose 0 to N elements (i.e) more than one elements. Eg: List of Employee objects then choose FLUX.



Publisher is the data producer or Data emitter.

Subscriber is the consumer (consuming the API) that is going to read the data from publisher. Basically Subscriber is also called as Consumer.

No in the arrow mark represents the flow of Events, so above diagram shows 5 events for successfully consuming of Data

Subscriber is going to invoke the subscribe method of the publisher.

All of these below 4 interface talk to each other to form the REACTIVE stream flow

1. **Publisher (**[**Code**](https://github.com/reactive-streams/reactive-streams-jvm/blob/v1.0.3/api/src/main/java/org/reactivestreams/Publisher.java)**)**

* Emits a sequence of events to subscribers according to the demand received from its subscribers. A publisher can serve multiple subscribers.

It has a single method:

public interface Publisher<T> {

public void subscribe(Subscriber<? super T> s);

}

1. Publisher is also called as Producer.
2. Publisher are generally the **Data source producers**.
   1. Data may be from Data Base
   2. Or data may be from External service
3. If we hit the producer URL in browser, internally Subscribe method of producer is called.
4. If we want to hit the publisher method Flux or Mono from code, we manually had to call the Subscribe method.

#### Subscriber ([Code](https://github.com/reactive-streams/reactive-streams-jvm/blob/v1.0.3/api/src/main/java/org/reactivestreams/Subscriber.java))

* Subscriber receives and processes events emitted by a Publisher. Please note that no notifications will be received until Subscription#request(long) is called to signal the demand.
* It has four methods to handle various kind of responses received.

public interface Subscriber<T> {

public void onSubscribe(Subscription s);

public void onNext(T t);

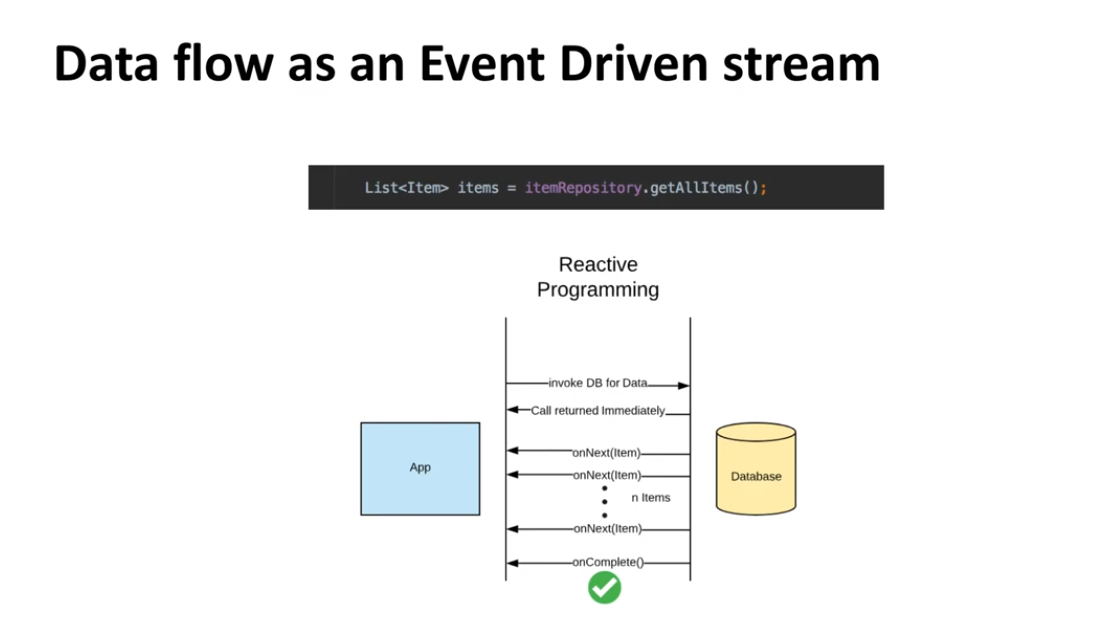
public void onError(Throwable t);

public void onComplete();

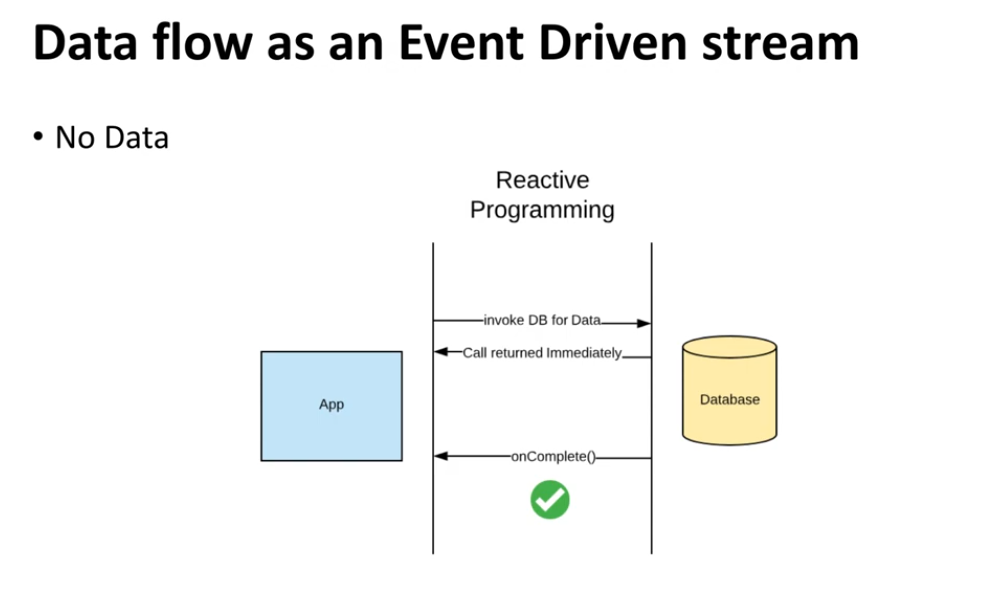
}

1. Subscriber is also called as Consumer.
2. When the **Subscribe** method of producer is invoked(called), then it internally calls the **Subscriber** to execute the event method.

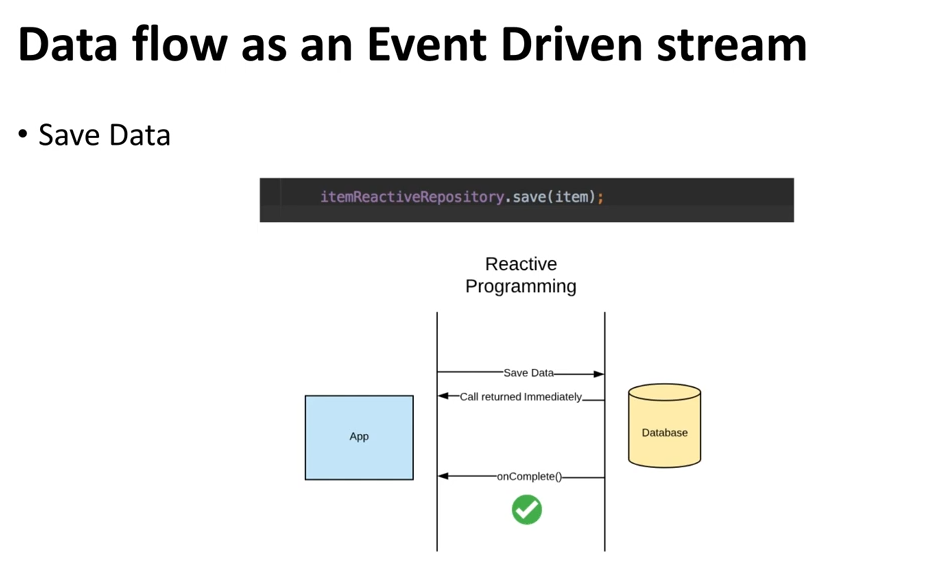
**When there is list of employee object is called, then below is the Event triggered.**



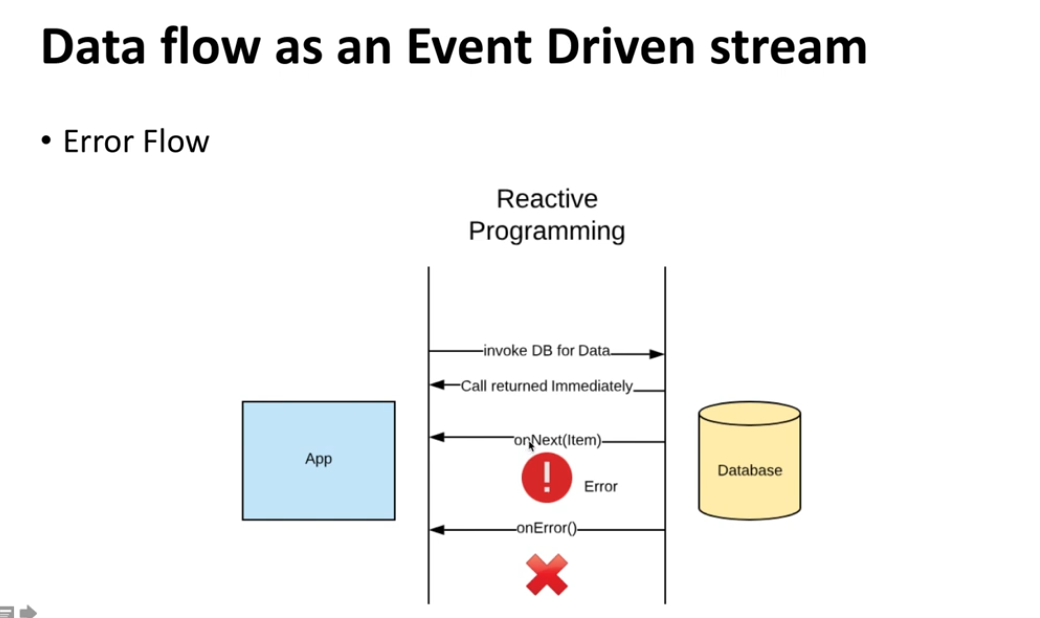
**When there no Data found for executed query, then below are the events triggered.**



**When we save the data below are the events triggered.**



**When there is an error occurred below is the event triggered.**



When there is an error occurred in middle of Publisher emitting the data, then publisher stops sends the data once the error occurred.

#### Subscription ([Code](https://github.com/reactive-streams/reactive-streams-jvm/blob/v1.0.3/api/src/main/java/org/reactivestreams/Subscription.java))

* Defines a one-to-one relationship between a Publisher and a Subscriber. It can only be used once by a single Subscriber. It is used to both signal desire for data and cancel demand (and allow resource clean-up).

public interface Subscription {

public void request(long n);

public void cancel();

}

#### 4.Processor ([Code](https://github.com/reactive-streams/reactive-streams-jvm/blob/v1.0.3/api/src/main/java/org/reactivestreams/Processor.java))

* Represents a processing stage consisting of both a Subscriber and a Publisher and obeys the contracts of both.

public interface Processor<T, R> extends Subscriber<T>, Publisher<R> {

}