Not yet reactive but asynchronous MVC

Dmytro_Aleksandrov@epam.com

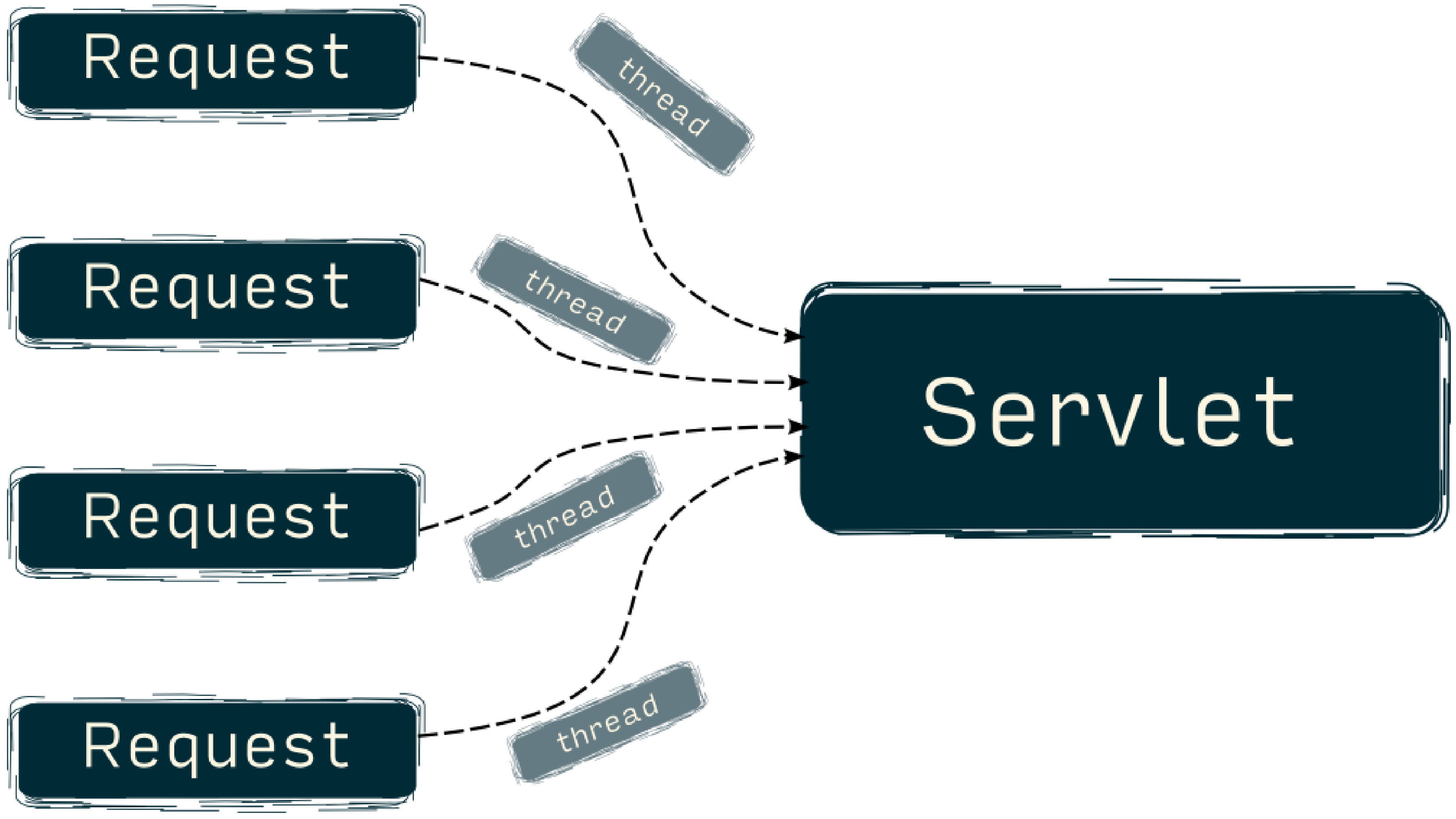
Signature of ‘typical’ @RequestMapping

```java
@RequestMapping(value = "/user", method = RequestMethod.GET)
public String getUser()
```

Link to source: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

How does it processed

- Threaded model



Link to source: http://www.oracle.com/technetwork/articles/grid/asynchronous-servlets-088344.html

Now imagine a service

```
@RestController
class EventsController {

   def blockingOp(): Int = {…}


   def fastEvent(): Event =
   {…}


}
```

Now imagine a service

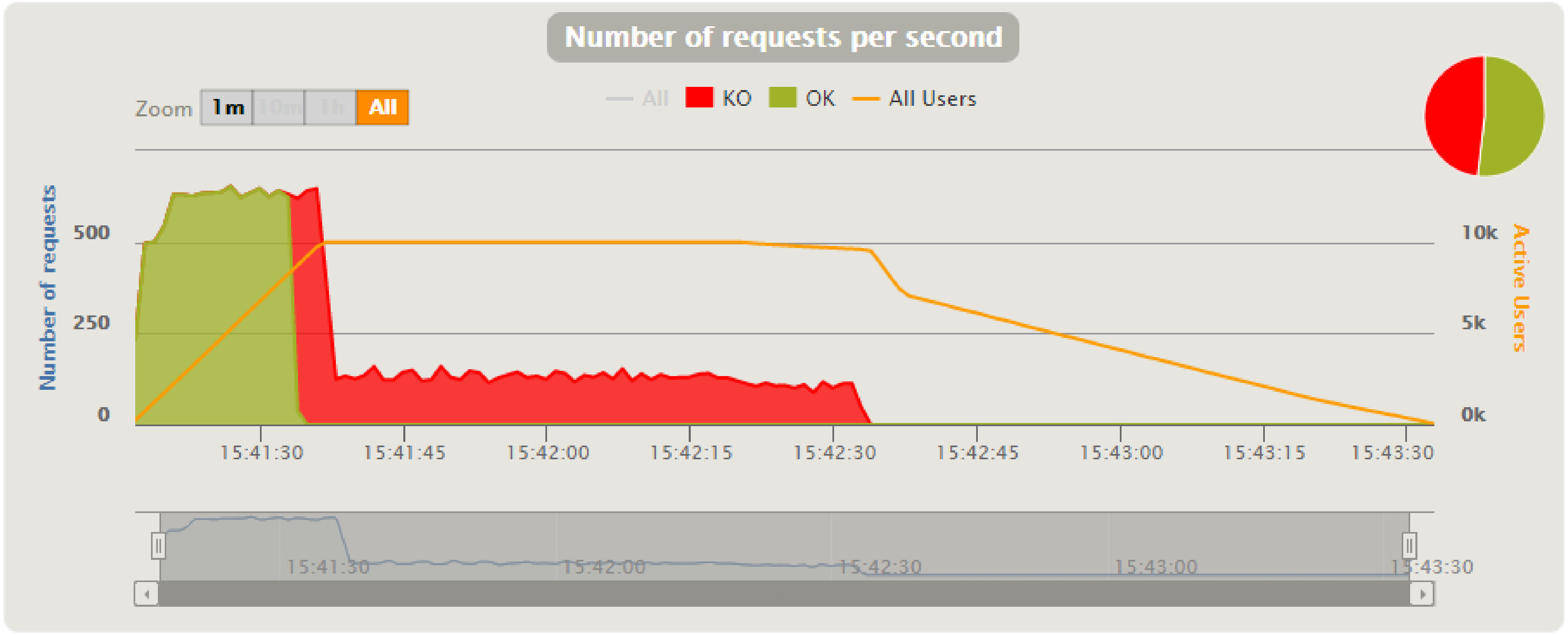
With one “heavy” operation

```
def blockingOp(): Int = {
  val processingTime = ThreadLocalRandom.current.nextInt(1000, 2000)
  // IO bounded operation intended here
  Thread.sleep(processingTime)
  processingTime
}
```

And a light one

```
def fakeEvent(): Event = {
  Event("fakeId", System.currentTimeMillis, Array.empty)
}
```

Load test sample



Link to source: http://goo.gl/abYAO8

Side note

Load test are written in Gatling

Why not Jmeter?

Try to simulate C10K with JMeter

threaded model, again

Link to source: http://gatling.io/

Link to source: https://github.com/alkersan/timebench/blob/master/modules/loadtest/src/test/scala/SpringBootSimulation.scala

Asynchronous completion

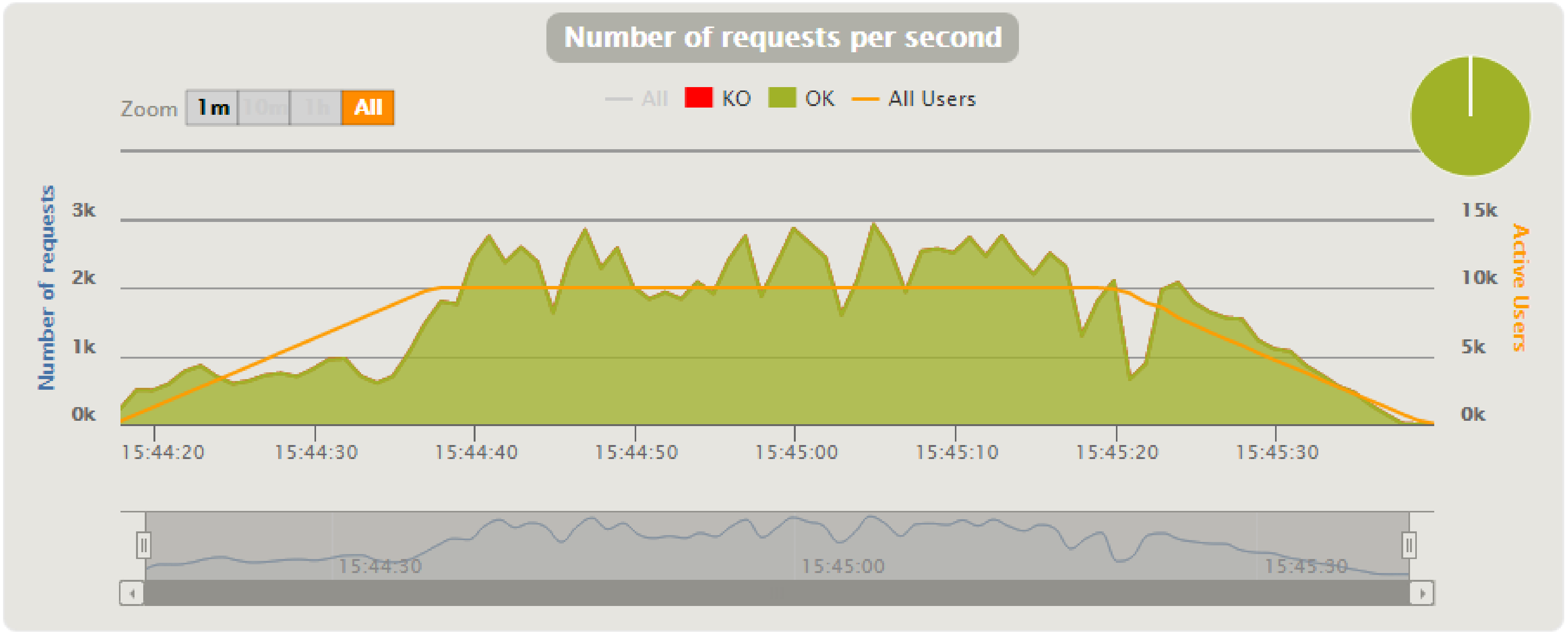
- Servlet 3.0 and 3.1
- Spring 3.2

Simply change signature to utilize this:

```
@RequestMapping(value = "/user", method = RequestMethod.GET)
public Callable<String> getUser()

or     Future<T>

or     DeferredResult<T>
```

Link to source: http://docs.spring.io p.17.3.4

What’s changed under load



Link to source: http://goo.gl/hYl0Ie

SLA

Lightweight operations:

```
def fakeEvent(): Event = {
  Event("fakeId", System.currentTimeMillis, Array.empty)
}
```

Can be served under load

Why this still is not “reactive”?

Because in reality it doesn’t solve the reason of blocking

```
def blockingOp(): Int = {
// JDBC call
// or Remote WS call
// or File IO
// or remote RPC call
}
```

For example: SpringData-Elastic call

- Intuitive interface
- Reduces boilerplate

```
public interface UserRepository extends
                        CrudRepository<User, Long> {

  Long deleteByLastname(String lastname);

  List<User> removeByLastname(String lastname);

}
```

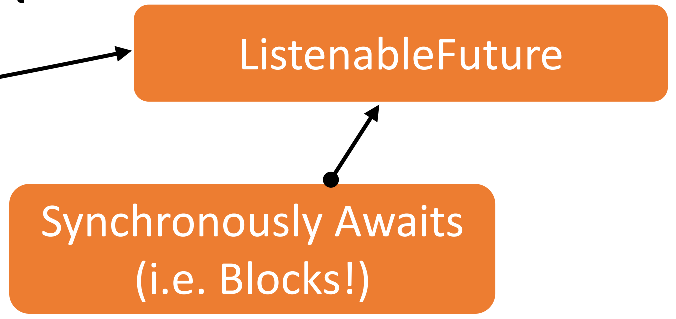
Link to source: http://docs.spring.io/spring-data/elasticsearch/docs/1.1.0.M1/reference/htmlsingle/

“Doubt everything we know”
- Descartes

Basic construct is:

ElasticsearchTemplate

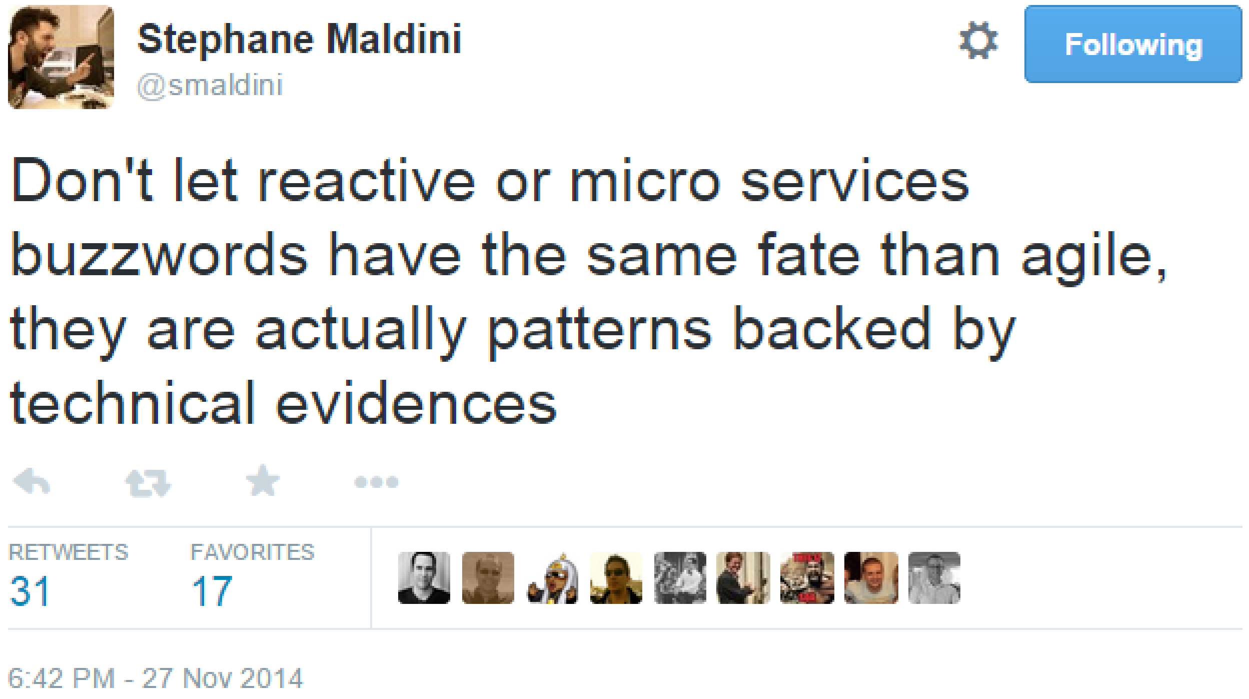
```
public <T> Page<T> queryForPage (StringQuery query,  …args…, ..args…) {
  SearchResponse response = repareSearch(query,clazz)
    .setQuery(query.getSource())
    .execute()
    .actionGet();

  return mapper.mapResults(response, clazz, query.getPageable());
}
```



Link to source: https://github.com/spring-projects/spring-data-elasticsearch

Conclusion

Always look under the hood



Stephane Maldini
@smaldini

Don't let reactive or micro services buzzwords have the same fate than agile, they are actually patterns backed by technical evidences

RETWEETS 31 FAVORITES 17

6:42 PM - 27 Nov 2014