LAGOM IN PRACTICE

THE NEW JAVA MICROSERVICES FRAMEWORK

JWORKS

POWERED BY ORDINA

ABOUT ME

Yannick De Turck
Java Developer
Scala and Play enthousiast
Ordina Belgium
@YannickDeTurck
https://github.com/YannickDeTurck

BLOGPOST LAGOM

Lagom: First Impressions and Initial Comparison to Spring Cloud

TOPICS

Introduction

Writing microservices

Demo

INTRODUCTION

MEET LAGOM

- Lightbend's microservices framework
- Focus on reactiveness
- MVP version
- Java API available, Scala API coming soon

DESIGN PHILOSOPHY

- Opinionated
- Message-Driven and Asynchronous
- Streaming first-class concept
- Distributed persistent patterns using ES and CQRS
- Embraces Domain-Driven Design

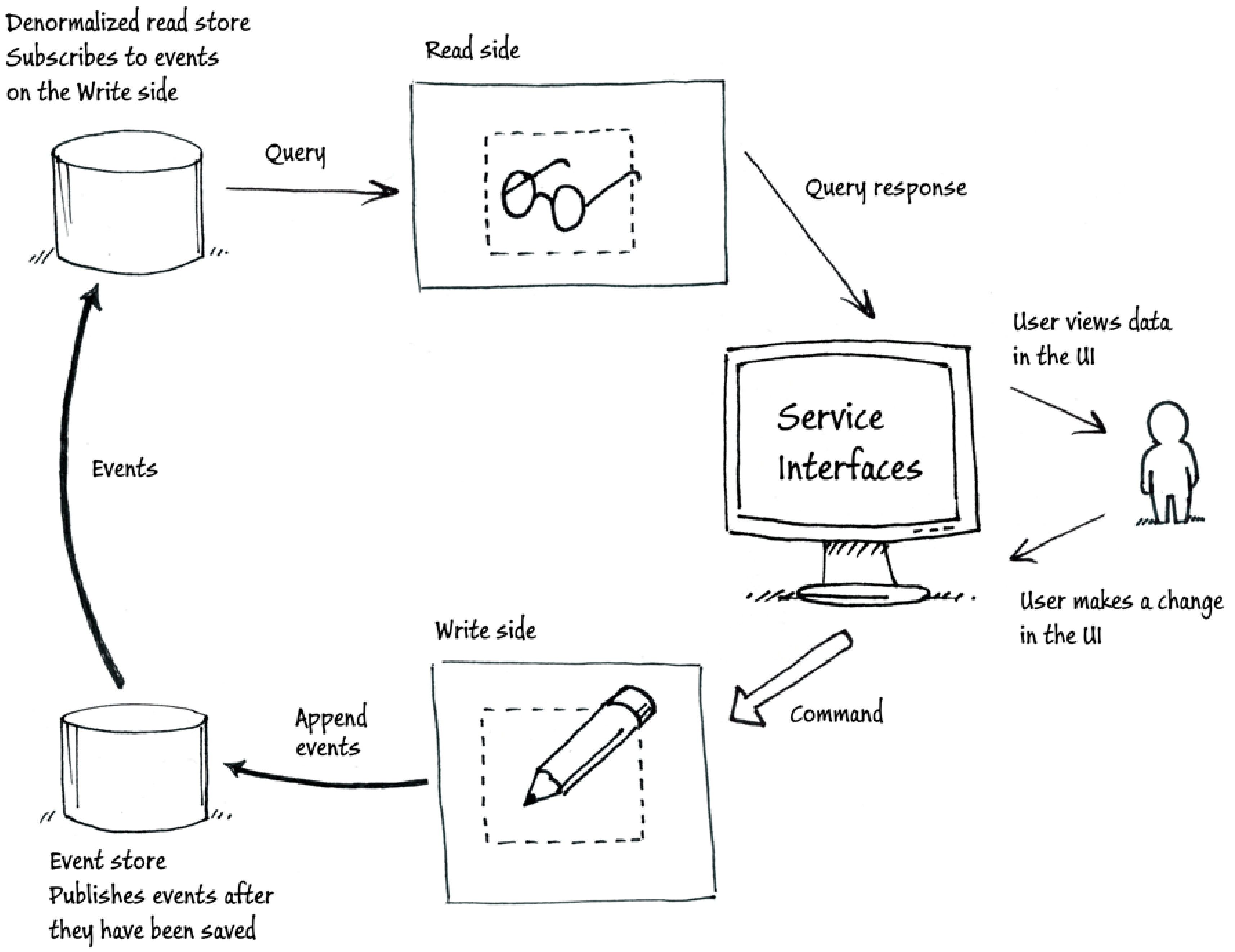
DEVELOPER PRODUCTIVITY

- Hot code reloading
- Start up with $ runAll
- Intra-service communication is managed for you

ARCHITECTURE AND TECHNOLOGIES

- Scala
- Java
- Play Framework
- Akka Cluster & Akka Persistence
- sbt
- Cassandra
- Guice
- ConductR

CQRS AND ES


Denormalized read store
Subscribes to events
on the Write side
Query
Read side
Query response
User views data
in the UI
Service
Interfaces
User makes a change
in the UI
Command
Write side
Append
events
Events
Event store
Publishes events after
they have been saved

EVENT SOURCING

- Capture all changes as domain events
- Aggregate Root
- Replies to queries for an identifier
- Cannot reply to queries spanning multiple aggregates

EVENT SOURCING BENEFITS

- All events are stored in an event store
- No object-relational impedance mismatch
- Built-in audit mechanism and historical tracing
- Performance, simplification and scalability
- Testability
- Debugging by replaying the event log

CQRS

- Separation of write- and read-side
- Scalability
- Different models for write- and read-side
- Eventual consistency

CQRS AND ES INTRODUCTION

- https://msdn.microsoft.com/en-us/library/jj591573.aspx

WRITING MICROSERVICES

PROJECT STRUCTURE

```
helloworld-api                  → Microservice API submodule
 └ src/main/java                → Java source code interfaces with model objects
helloworld-impl                 → Microservice implementation submodule
 └ logs                         → Logs of the microservice
 └ src/main/java                → Java source code implementation of the API submodule
 └ src/main/resources           → Contains the microservice application config
 └ src/test/java                → Java source code unit tests
logs                            → Logs of the Lagom system
project                         → Sbt configuration files
 └ build.properties             → Marker for sbt project
 └ plugins.sbt                  → Sbt plugins including the declaration for Lagom itself
.gitignore                      → Git ignore file
build.sbt                       → Application build script
```

API INTERFACE

```java
public interface HelloService extends Service {
  ServiceCall<NotUsed, String> hello(String name);

  ServiceCall<GreetingMessage, String> useGreeting(String id);

  @Override
  default Descriptor descriptor() {
    return named("helloservice").with(
        restCall(Method.GET,  "/api/hello/:name", this::hello),
        restCall(Method.POST, "/api/hello/:id", this::useGreeting)
      ).withAutoAcl(true);
  }
}
```

API IMPLEMENTATION

```java
public class HelloServiceImpl implements HelloService {
  List<String> savedGreetings = new ArrayList<>();

  @Override
  public ServiceCall<NotUsed, String> hello(String name) {
    return (request) -> {
      CompletableFuture.completedFuture("Hello, " + name);
    };
  }

  @Override
  public ServiceCall<GreetingMessage, String> useGreeting() {
    return (request) -> {
      String greeting = request.getGreeting();
      savedGreeting.add(greeting);
      CompletableFuture.completedFuture("Greeting '" + greeting + "' saved!");
    };
  }
}
```

API MODULE

```
public class HelloServiceModule extends AbstractModule
                                implements ServiceGuiceSupport {
  @Override
  protected void configure() {
    bindServices(serviceBinding(HelloService.class, HelloServiceImpl.class));
    bindClient(OtherService.class);
  }
}
```

API MODULE

The module is defined in the **application.config**

```
play.modules.enabled += sample.helloworld.impl.HelloServiceModule
```

REGISTERING THE MICROSERVICE

build.sbt

```
lazy val helloworldApi = project("helloworld-api")
  .settings(
    version := "1.0-SNAPSHOT",
    libraryDependencies += lagomJavadslApi
  )

lazy val helloworldImpl = project("helloworld-impl")
  .enablePlugins(LagomJava)
  .settings(
    version := "1.0-SNAPSHOT",
    libraryDependencies ++= Seq(
      lagomJavadslPersistence,
      lagomJavadslTestKit
    )
  )
  .settings(lagomForkedTestSettings: _*)
  .dependsOn(helloworldApi)
```

TESTING THE MICROSERVICE

```
$ curl localhost:24266/api/hello/World
Hello, World!

$ curl -H "Content-Type: application/json" -X POST -d \
    '{"message": "Hello "}' http://localhost:24266/api/hello/World
Greeting 'Hello' was saved!
```

TESTING THE MICROSERVICE

```java
public class HelloServiceTest {
  private static ServiceTest.TestServer server;

  @Test
  public void shouldRespondHello() throws Exception {
    withServer(defaultSetup(), server -> {
      HelloService service = server.client(HelloService.class);
      String hello = service.hello("Yannick")
        .invoke(NotUsed.getInstance()).toCompletableFuture().get(5, SECONDS);
      assertEquals("Hello, Yannick", hello);
    });
  }
}
```

ES AND CQRS - PERSISTENTENTITY

```java
public class HelloWorld extends PersistentEntity {
  @Override
  public Behavior initialBehavior(Optional snapshotState) {
    BehaviorBuilder b = newBehaviorBuilder(
        snapshotState.orElse(new WorldState("Hello", LocalDateTime.now().toString()))

    b.setCommandHandler(UseGreetingMessage.class, (cmd, ctx) ->
        ctx.thenPersist(new GreetingMessageChanged(cmd.message),
        evt -> ctx.reply(Done.getInstance()))
    );

    b.setEventHandler(GreetingMessageChanged.class,
        evt -> new WorldState(evt.message, LocalDateTime.now().toString()));

    b.setReadOnlyCommandHandler(Hello.class,
        (cmd, ctx) -> ctx.reply(state().message + ", " + cmd.name + "!")
    );
  }
}
```

ES AND CQRS - STATE

```java
@Immutable
@JsonDeserialize
public final class WorldState implements CompressedJsonable {

  public final String message;
  public final String timestamp;

  @JsonCreator
  public WorldState(String message, String timestamp) {
    this.message = Preconditions.checkNotNull(message, "message");
    this.timestamp = Preconditions.checkNotNull(timestamp, "timestamp");
  }

  // equals, hashcode, toString, ...
}
```

ES AND CQRS - COMMAND

```java
public interface HelloCommand extends Jsonable {
  @Immutable
  @JsonDeserialize
  public final class UseGreetingMessage implements HelloCommand,
    CompressedJsonable, PersistentEntity.ReplyType {
    public final String message;

    @JsonCreator
    public UseGreetingMessage(String message) {
      this.message = Preconditions.checkNotNull(message, "message");
    }

    // equals, hashcode, toString,...
  }

  @Immutable
  @JsonDeserialize
  public final class Hello implements HelloCommand,
    CompressedJsonable, PersistentEntity.ReplyType {
    public final String name;
    public final Optional<String> organization;
```

ES AND CQRS - EVENT

```java
public interface HelloEvent extends Jsonable {

  @Immutable
  @JsonDeserialize
  public final class GreetingMessageChanged implements HelloEvent {
    public final String message;

    @JsonCreator
    public GreetingMessageChanged(String message) {
      this.message = Preconditions.checkNotNull(message, "message");
    }

    // equals, hashCode, toString
}
```

ES AND CQRS - SERVICEIMPL

```
public class HelloServiceImpl implements HelloService {
  private final PersistentEntityRegistry persistentEntityRegistry;

  @Inject
  public HelloServiceImpl(PersistentEntityRegistry persistentEntityRegistry) {
    this.persistentEntityRegistry = persistentEntityRegistry;
    persistentEntityRegistry.register(HelloWorld.class);
  }

  @Override
  public ServiceCall hello(String name) {
    return (request) -> {
      PersistentEntityRef ref =
        persistentEntityRegistry.refFor(HelloWorld.class, name);
      return ref.ask(new Hello(id, Optional.empty()));
    };
  }

  @Override
  public ServiceCall useGreeting(String id) {
    return (request) -> {
```

DEMO

LAGOM SHOP

- Item Service: Create and lookup items
- Order Service: Create and lookup orders for items
- Play front-end

QUESTIONS?

Resources: Github repository

Blogpost: Lagom: First Impressions and Initial Comparison to Spring Cloud

Podcast: Lightbend Podcast Ep. 09: Andreas Evers test drives Lagom in comparison with Spring Cloud

http://bit.ly/1RWmTeQ

THANKS FOR WATCHING!