MICHAEL PLÖD

GRAILS 3 EVENTS FOR LOOSELY COUPLED AND REACTIVE APPLICATIONS

MICHAEL PLÖD

- Principal Consultant at innoQ
- Architecture and Software Development Consulting
- Areas of interest: Event Sourcing, Caching, Grails, Software Architecture
- Follow me on Twitter: @bitboss

WHAT DOES IT MEAN

REACTIVE?

- ? ***Responsive***
- ? ***Resilient***
- ? ***Elastic***
- ? ***Message driven***

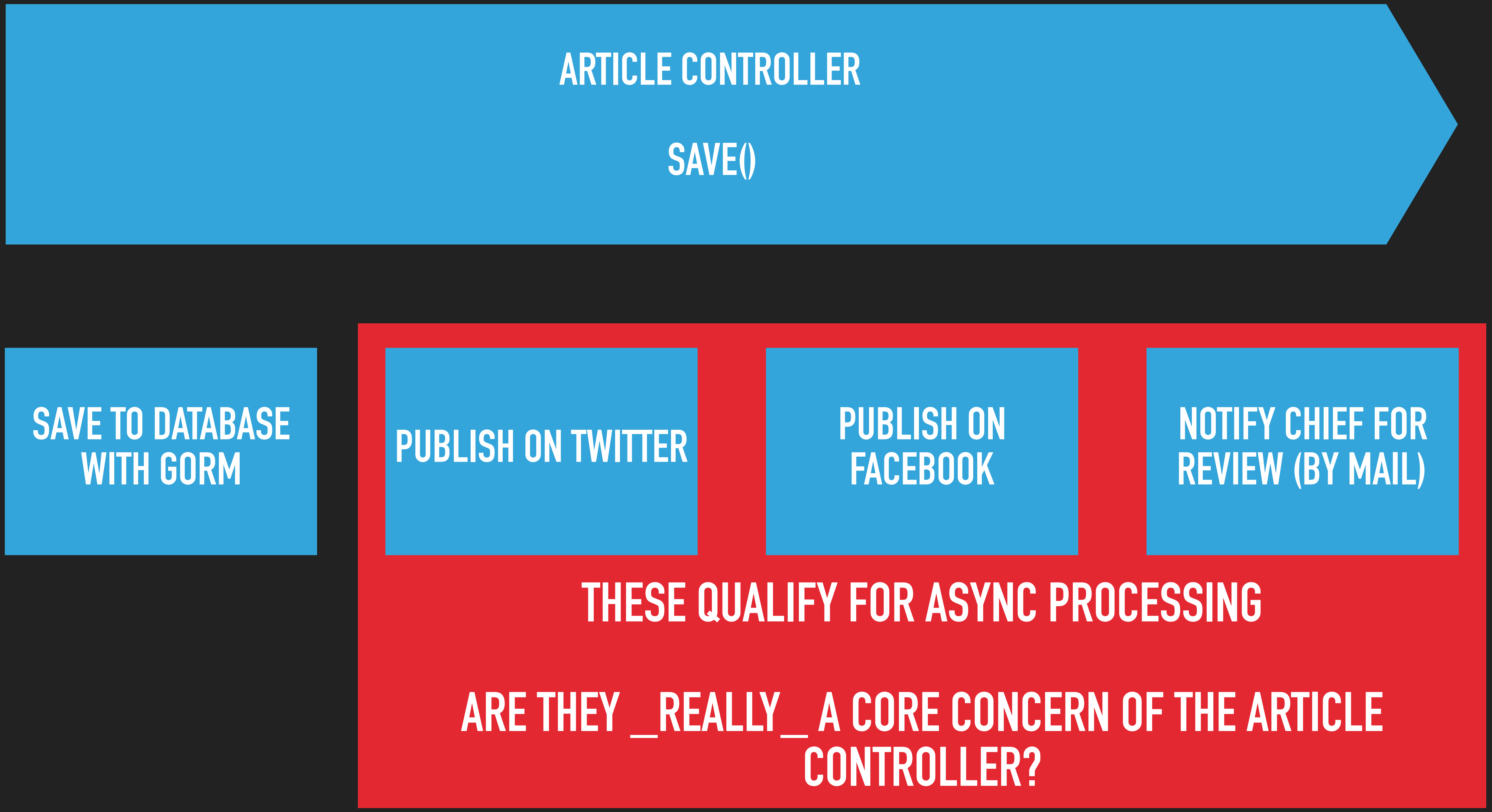
ASYNC FEATURES IN GRAILS 2.X

- Promises
- Async GORM
- Async Requens Handling
- Servlet 3.0 async

> I WANT TO CREATE A LOOSELY COUPLED APPLICATION WITH A HIGH COHESION

Some software craftsman

COUPLING REVISED



ARTICLE CONTROLLER

SAVE()

SAVE TO DATABASE WITH GORM

PUBLISH ON TWITTER

PUBLISH ON FACEBOOK

NOTIFY CHIEF FOR REVIEW (BY MAIL)

THESE QUALIFY FOR ASYNC PROCESSING

ARE THEY _REALLY_ A CORE CONCERN OF THE ARTICLE CONTROLLER?

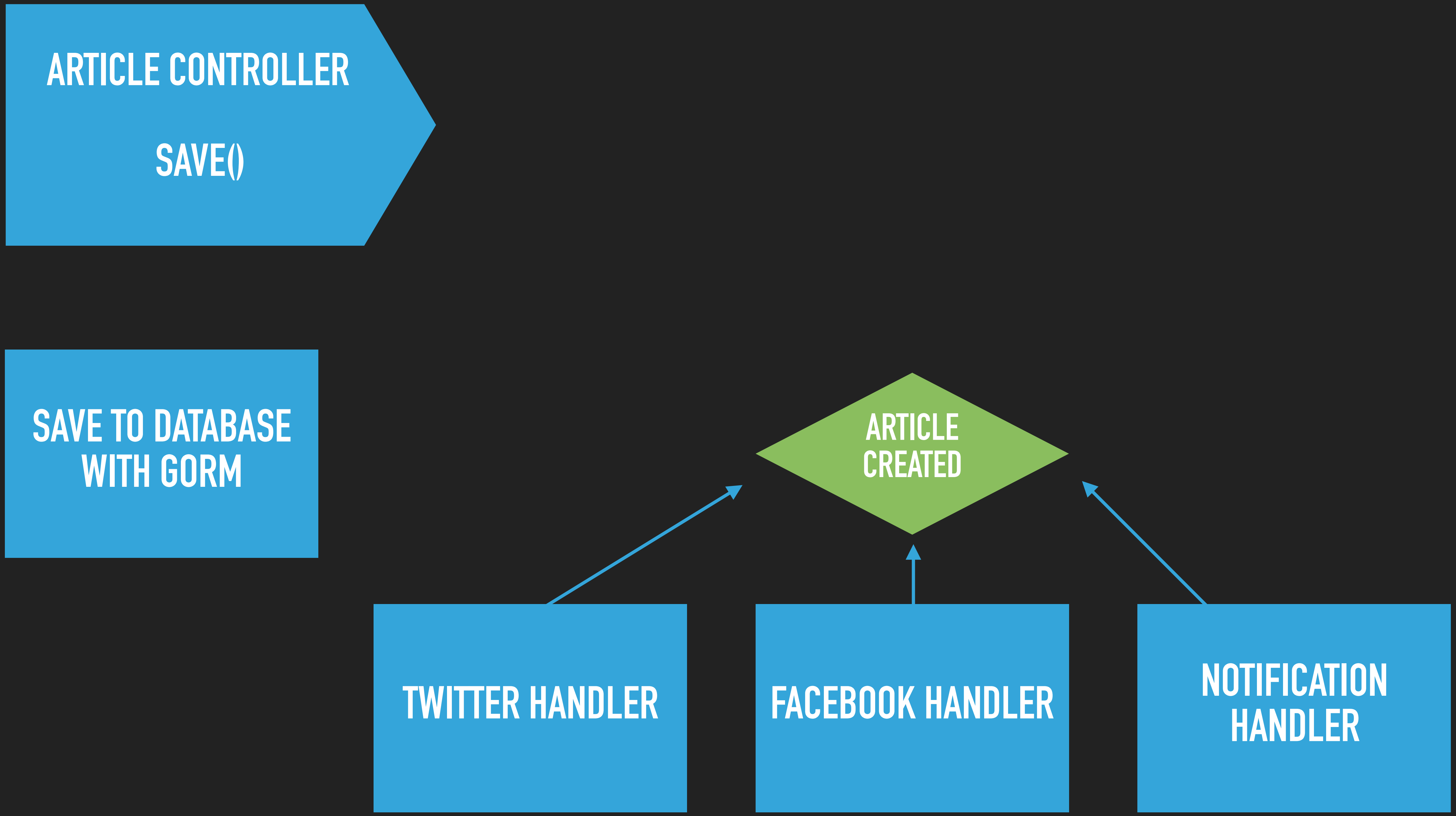
> EVEN BY USING PROMISES WE TIGHTLY COUPLE OUR COMPONENTS

Some software craftsman

LET’S BE HONEST

< THIS
IS THE ANATOMY OF A TYPICAL
SYSTEM AFTER A FEW YEARS

EVENTS: AN INTERESTING ALTERNATIVE



ARTICLE CONTROLLER

SAVE()

SAVE TO DATABASE WITH GORM

ARTICLE CREATED

TWITTER HANDLER

FACEBOOK HANDLER

NOTIFICATION HANDLER

WHAT IS REACTOR

- Foundational Framework for asynchronous applications on the JVM
- Abstractions for Java, Groovy and other JVM languages
- Aims at making event and data-driven applications easier
- High performance (15.000.000 events / second with the fastest non-blocking dispatcher)
- https://github.com/reactor/reactor.

```
IMPORT STATIC REACTOR.FN.$;

REACTOR REACTOR = R.CREATE();

REACTOR.ON($("PARSE"), NEW
CONSUMER<EVENT<STRING>>() {
    PUBLIC VOID CALL(EVENT<STRING> EV) {
      SERVICE.HANDLEEVENT(EV);
    }
});

REACTOR.NOTIFY("PARSE", FN.EVENT("HELLO
WORLD!"));
```

THE BASIC BUILDING BLOCKS ARE:

SELECTORS, CONSUMERS AND EVENTS

REACTOR IN GRAILS 3

- Configuration
- Consuming Events
- Sending Events
- Events trait for Services and Controllers
- Spring Annotations
- Events trait for other classes (that aren’t Services or Controllers)
- GORM Events
- Spring Events

CONFIGURATION IN APPLICATION.YML

```
reactor:
    dispatchers:
        default: myExecutor
        myExecutor:
            type: threadPoolExecutor
            size: 5
            backlog: 2048
```

CONSUMING EVENTS WITH CONSUMERS AND SELECTORS

```
on("myEvent") {
    println "Event fired!"
}

on(„articlePublishedEvent") { Article article ->
    //do something with the article
}
```

SERVICES AND CONTROLLERS IMPLEMENT THE EVENTS TRAIT

```
class EventController {
    @PostConstruct
    void init()
    {
        on("articlePublished") { payload ->
            println "article published "
        }
    }
    …
}
```

SERVICES MUST BE ANNOTATED WITH @CONSUMER

```
@Transactional
@Consumer
class TwitterService{
    @PostConstruct
    void init()
    {
        on("articlePublished") { payload ->
            println "article published"
        }
    }
}
```

YOU CAN ALSO USE SPRING ANNOTATIONS FOR SELECTORS

```
@Transactional
@Consumer
class TwitterService{
    @Selector('articlePublished')
    void publishArticleOnTwitter()
    {
    .....
    }
}
```

EVENTS OF NON-SERVICE OR -CONTROLLER CLASSES

```
@Component
@Consumer
class TwitterPublisher implements Events {
    @PostConstruct
    void init()
    {
        on("articlePublished") { payload ->
            println "article published"
        }
    }
}
```

YOU CAN EVEN REACT TO EVENTS FROM GORM

- gorm:preInsert
- gorm:postInsert
- gorm:preDelete
- gorm:postDelete
- gorm:preLoad
- gorm:postLoad
- gorm:preUpdate
- gorm:postUpdate
- gorm:saveOrUpdate
- gorm:datastoreInitialized
- gorm:validation

```
on(„gorm:preInsert“) { PreInsertEvent e ->
    //do something with the event
}
```

GORM EVENTS ARE ASYNCHRONOUS: THEY CAN'T MODIFY PERSISTENCE OPERATIONS

SPRING ALSO FIRES EVENTS

```
on("spring:applicationStarted") { ApplicationStartedEvent event ->
    // ....
}

on("spring:servletRequestHandled") { RequestHandledEvent event ->
    // ....
}
```

DEMO

LET'S GET OUR HANDS DIRTY

THANK YOU!

MICHAEL PLOED - INNOQ

@BITBOSS

Example: https://github.com/mploed/grails-event-example