Kaleidos

GeeCON 2015

# Grails and the real-time world

geecon

IVÁN LÓPEZ - @ilopmar

# Hello!

# I am Iván López

@ilopmar

Kaleidos
beautiful code

Groovy Madrid Groovy Users Group

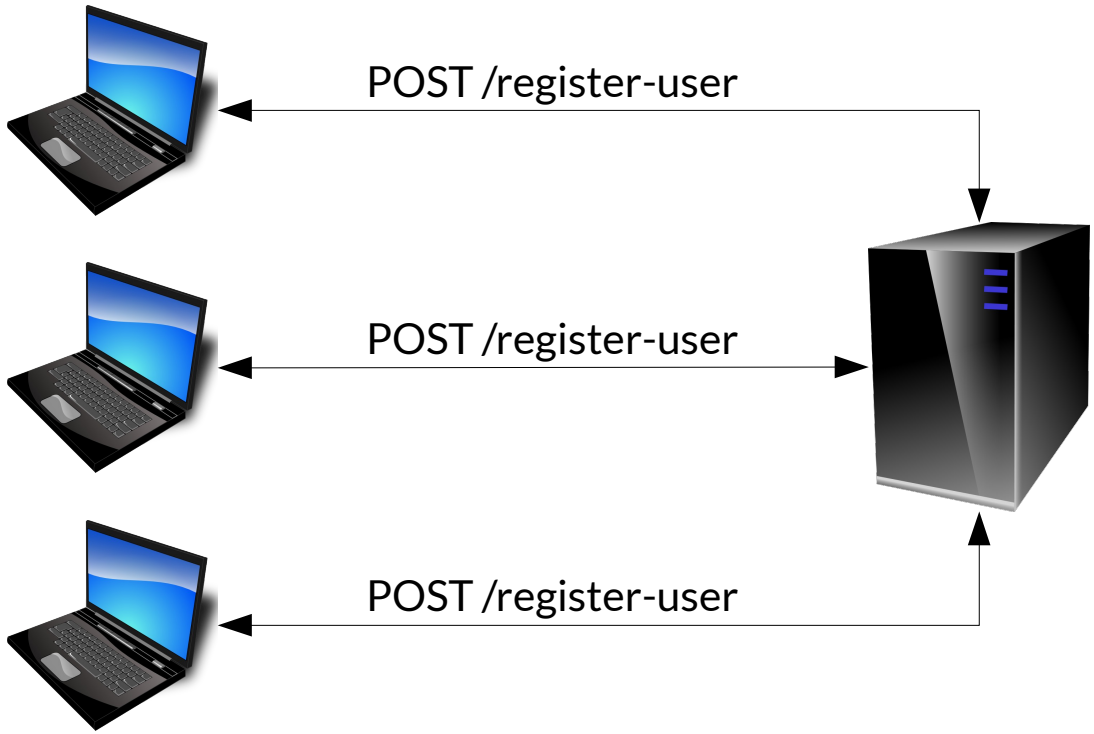
@madridgug

Greach
the Groovy spanish conf

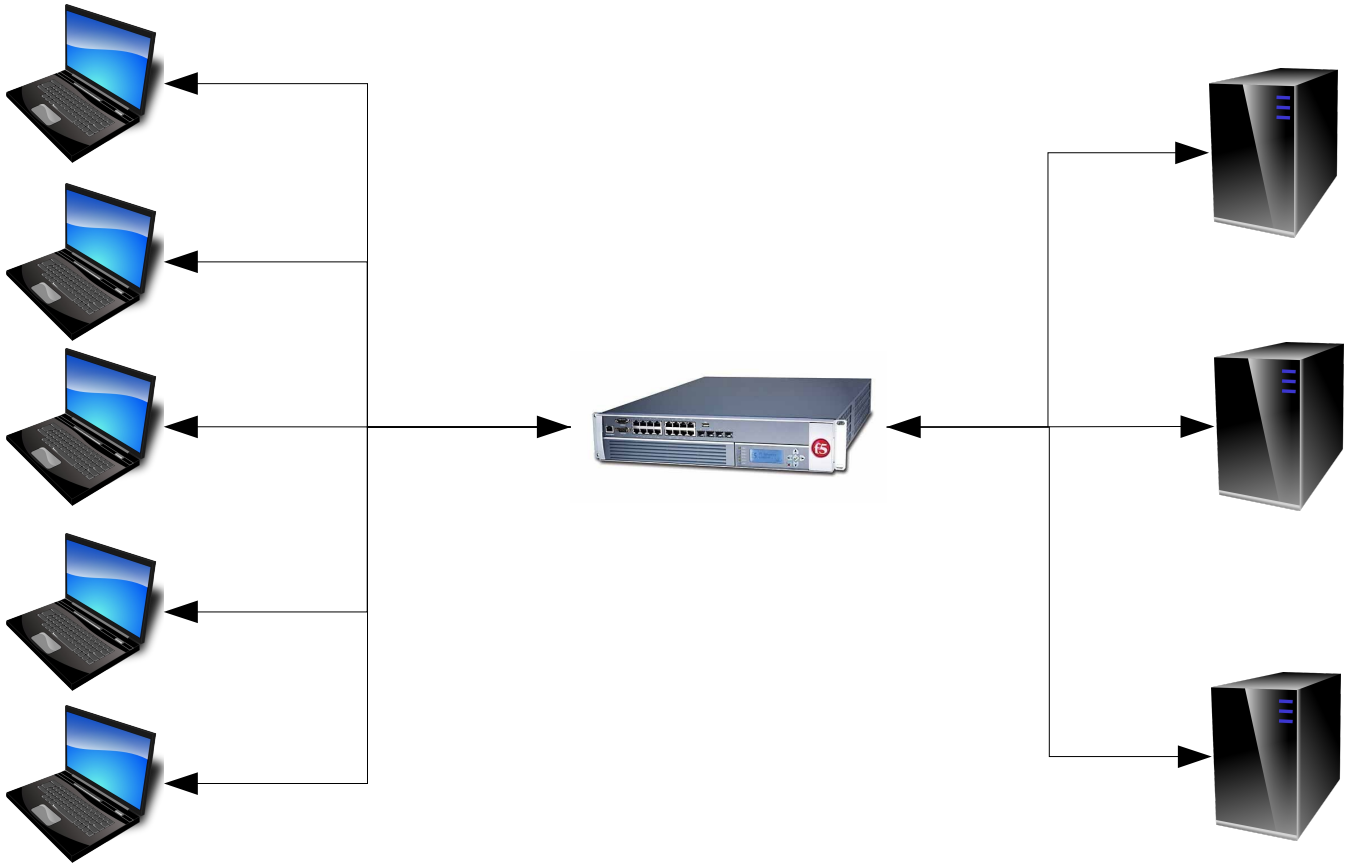
http://greachconf.com

# Traditional architectures

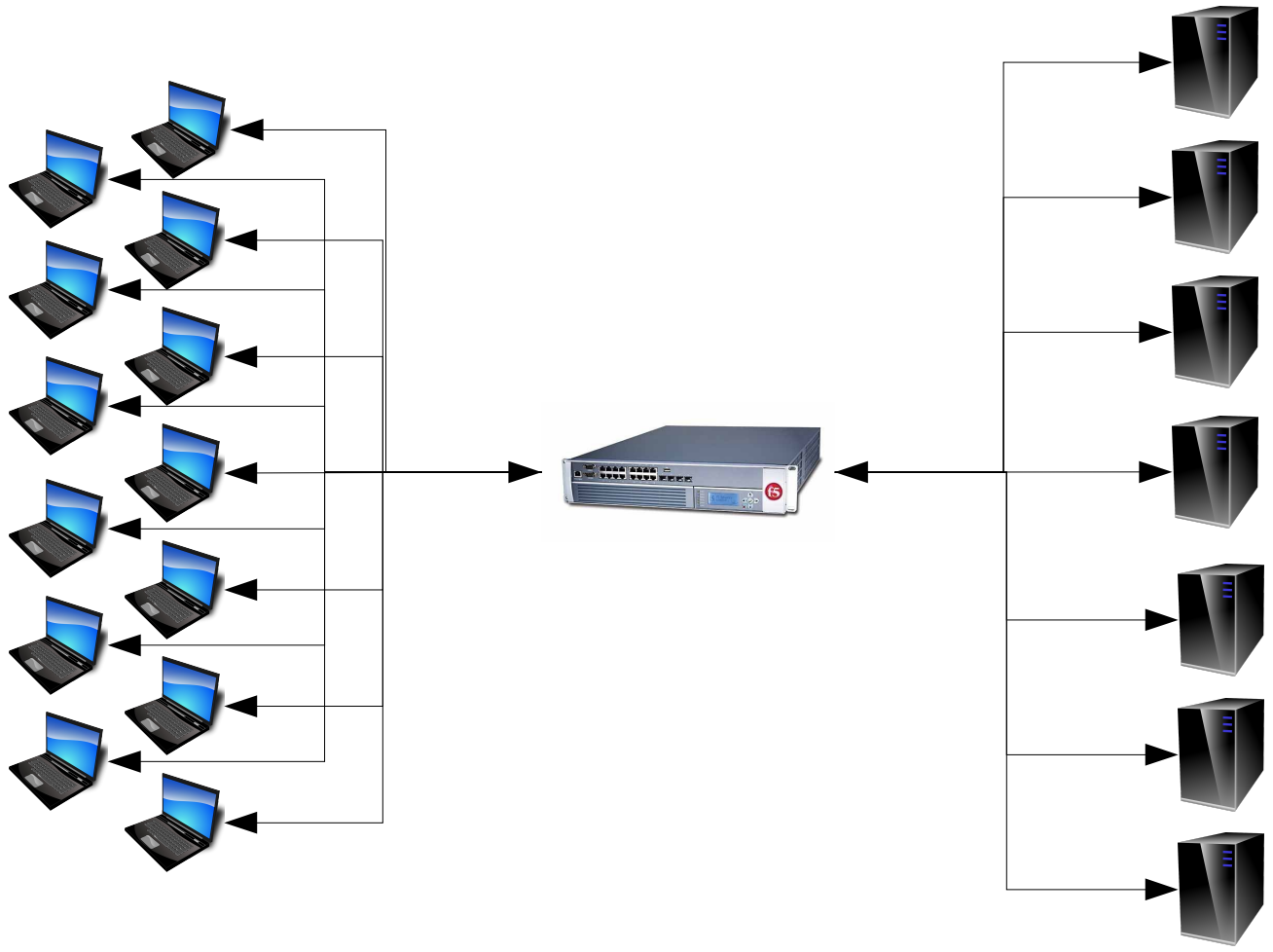
## Request-response



# Traditional architectures



# Traditional architectures



*A problem postponed is a problem solved.*

*Really?*

# 1.
# Event Driven Architectures

# Event Drive Architectures

- Fire & Forget
- Decouple producer and consumer
- Immediate action in the consumer
- Real-time

# Traditional architecture

POST /purchase
- Receive request 5 ms
- Data validation 20 ms
- Save 40 ms
- PDF generation 200 ms
- Send email 80 ms
- Render response 50 ms

Total: 395 ms

*Can we do it better?*

# Event driven architecture

POST /purchase

| | |
|---|---|
| - Receive request 5 ms | No |
| - Data validation 20 ms | No |
| - Save 40 ms | No |
| - PDF generation 200 ms | Yes |
| - Send email 80 ms | Yes |
| - Render response 50 ms | No |

Total: 115 ms ~ 70% better

Can anyone else do it?

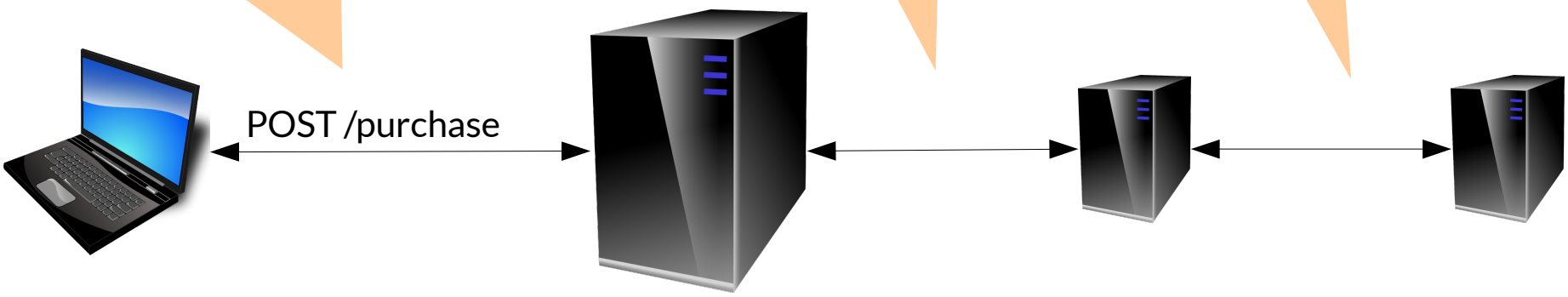
# Event driven architecture

POST /purchase
- Receive request 5 ms
- Data validation 20 ms
- Save 40 ms
- Render response 50 ms

Total: 115 ms

PDF Generation

Send email



*Don't keep your clients waiting unnecessarily!*

*Can I defer it?*

# Goals

- Loosely coupled architecture, easy to extend and evolve.
- Build high performance and scalable systems
- Keep the business logic where "*it belongs*"

# What about Grails?

- Platform core plugin
- Events plugin
- Executor plugin
- Grails 2.3+ async

# Synchronous example

```
// Send confirmation email
def user = new User(params).save()
emailService.sendRegistrationMail(user)
render view:'registerOk'
```

# Synchronous example

```
// Send confirmation email
def user = new User(params).save()
emailService.sendRegistrationMail(user)
render view:'registerOk'



class EmailService {
    public void sendRegistrationMail(User user) {
        sendMail {
            to user.email
            subject "Confirm your account"
            html g.render(template: "userEmailConfirmation")
        }
    }
}
```

# Asynchronous example

```
// Platform core
def user = new User(params).save()
event 'sendRegistrationMail', user
render view:'registerOk'
```

# Asynchronous example

```
// Platform core
def user = new User(params).save()
event 'sendRegistrationMail', user
render view:'registerOk'



class EmailService {
    @grails.events.Listener
    public void sendRegistrationMail(User user) {
        sendMail {
            to user.email
            subject "Confirm your account"
            html g.render(template: "userEmailConfirmation")
        }
    }
}
```

# Asynchronous example

```
// Executor
def user = new User(params).save()
runAsync {
    emailService.sendRegistrationMail(user)
}
render view:'registerOk'


class EmailService {
    public void sendRegistrationMail(User user) {
        sendMail {
            to user.email
            subject "Confirm your account"
            html g.render(template: "userEmailConfirmation")
        }
    }
}
```

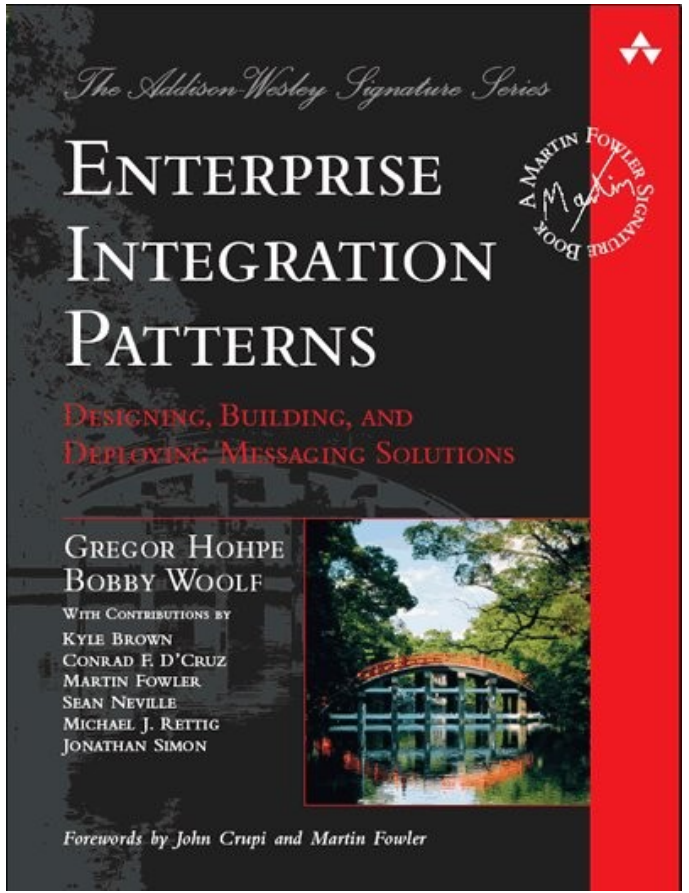
*I love the smell of code in the morning*

# What if we don't want this?

- Extract “dependencies” to configuration
- Change application behaviour modifying the configuration

# Spring Integration

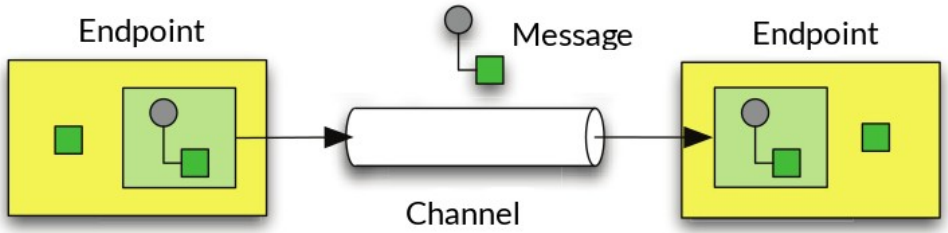
Use inside Spring the well-know *Enterprise Integration Patterns*




http://www.enterpriseintegrationpatterns.com/

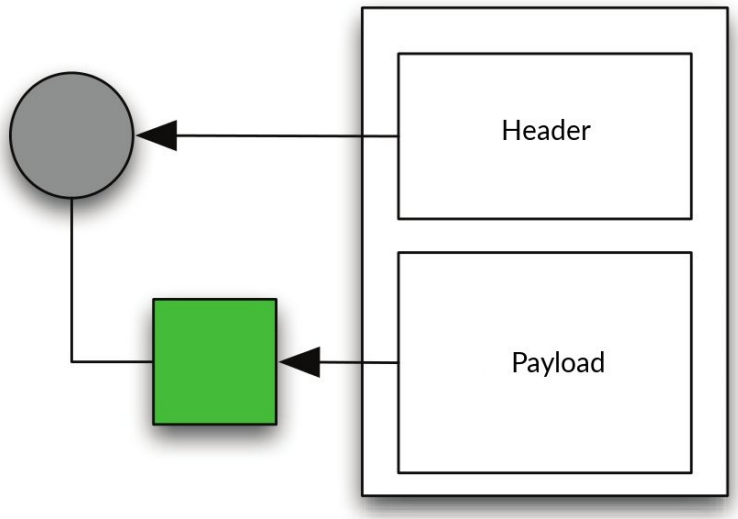
# Spring Integration

- Lightweight messaging mechanism for Spring apps
- High level abstraction for messaging
- External systems integration declaring adapters



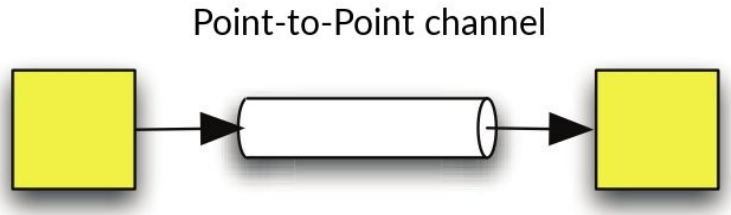
# Message

- Payload
- Headers
- Immutable

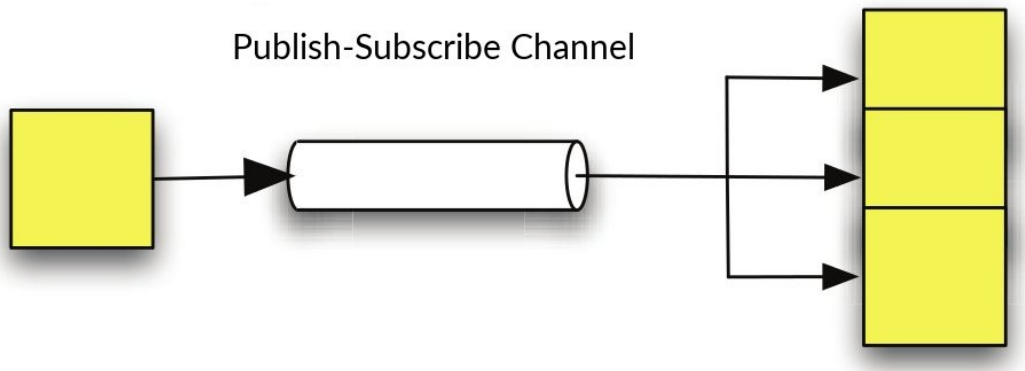


# Channels

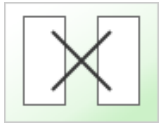
- Point-to-point

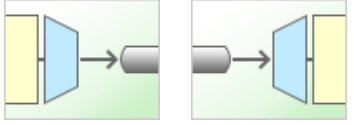



- Publish-Subscribe



# Endpoints

- Transformer 
- Filter 
- Router 
- Splitter 
- Aggregator 

- Service activator 
- Channel-adapter 
- Enricher 
- Bridge 
- ... 

# Adapters

- JMS
- AMQP
- TCP
- UDP
- File
- FTP
- RMI
- HTTP (Rest)
- WS
- Mail
- JDBC
- XMPP
- Twitter
- RSS
- MongoDB
- Redis
- Gemfire
- Stream

*Talk is cheap. Show me the code.*

# 2.
# Demo

# 3. Summary

# Summary

- Grails standard architecture fits in most of the cases
- It does not scale to infinite (and beyond!)
- Think about the application you are building
- Keep the information flow in mind

# Thanks!

## Any questions?

Iván López

@ilopmar

lopez.ivan@gmail.com

https://github.com/lmivan

http://kcy.me/22ze3