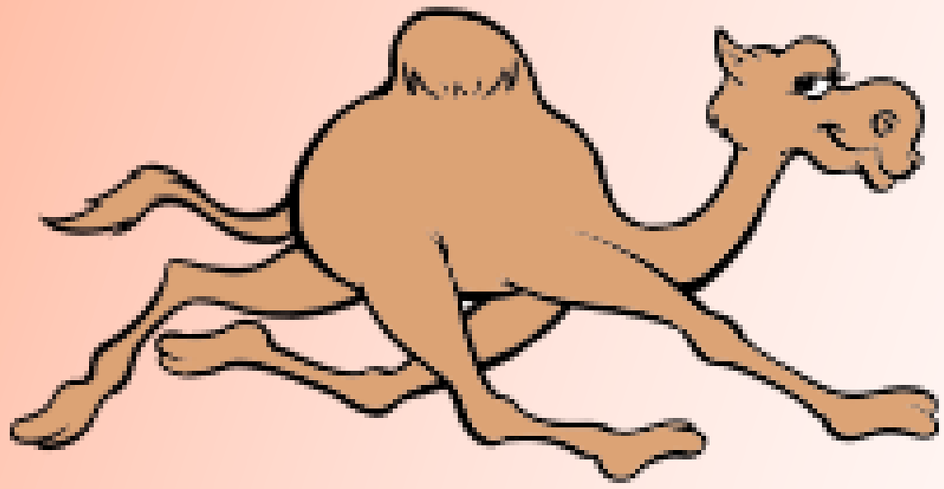
CamelOne 2013

June 10-11 2013

Boston, MA



“Banking” on Apache Camel

By Rohit Sitani

About the Speaker

Rohit Sitani

Senior Consultant with Capgemini

Instrumental in creating next-generation frameworks & libraries. Specializes in writing customer facing Enterprise applications.

rohit.sitani@capgemini.com

linkedin.com/in/rohitsitani

twitter.com/rohitsitani

Agenda

- **Digital Banking: One Bank view across channels**
- Why a Routing and Mediation engine
- Before & After Apache Camel
  - Simplification
  - Parallel Processing
  - Testing
- Some rough edges

The Industry

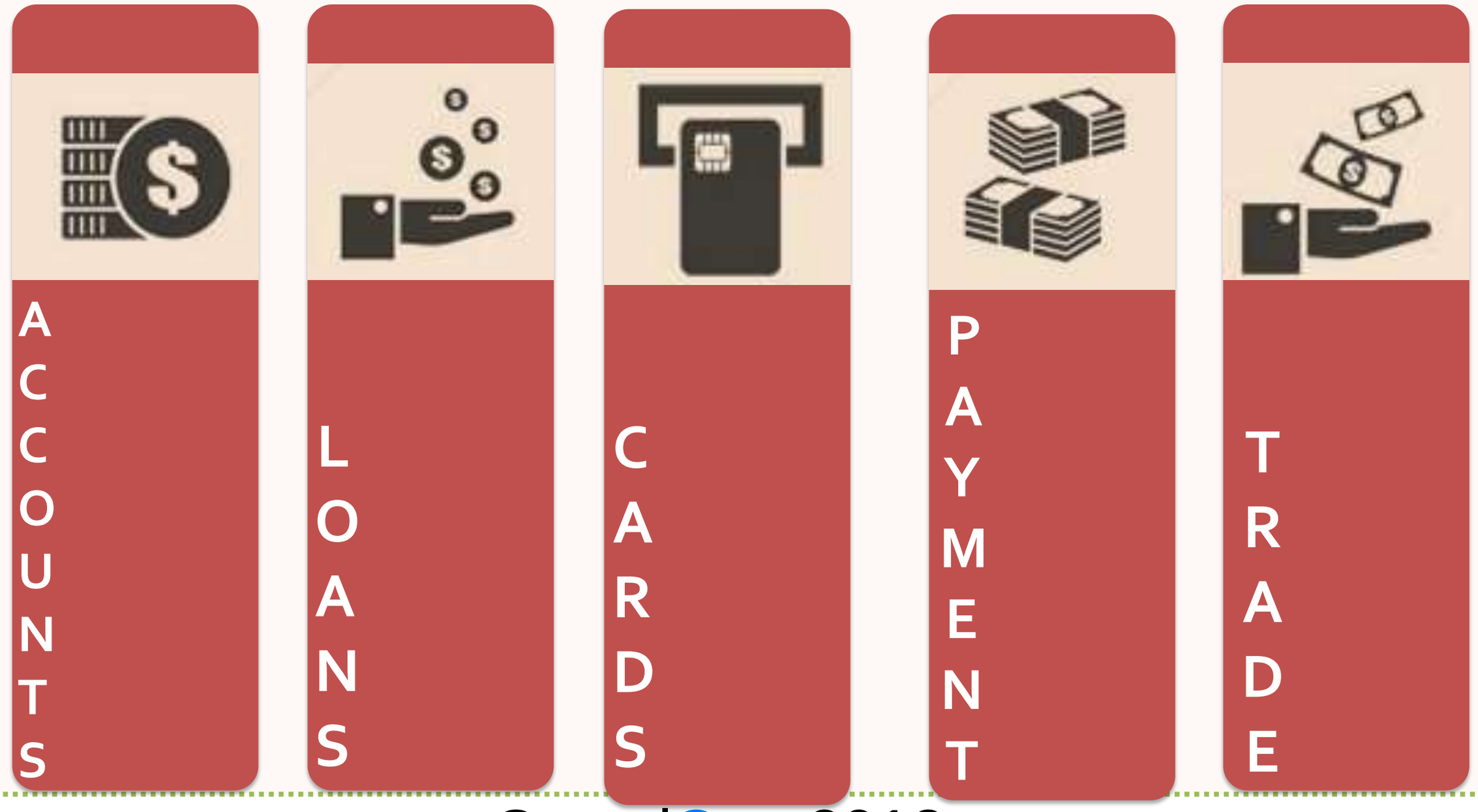
Traditional Banking

Internet Banking

Call center

ATM

Mobile Banking



One Bank View

Digital Banking

ACCOUNTS

LOANS

CARDS

PAYMENT

TRADE

CamelOne

CamelOne 2013

Digital Bank end-state Architecture

Native apps

Mobile Browsers

API facade

Digital Banking

JDBC

Local DB

LDAP

Authentication

SOAP

Smart Ads

JMS

SMTP

JMS

Billers

Message Q

SMTP

Message Q

Iso 8583

Payment

Accounts

Cards

Loans

Agenda

- Digital Banking: One Bank view across channels
- **Why a Routing and Mediation engine**
- Before & After Apache Camel
  - Simplification
  - Parallel Processing
  - Testing
- Some rough edges

Why Routing & Mediation Engine

Write less code

- Less testing, Less maintenance, Lower cost
- Focus attention on business logic

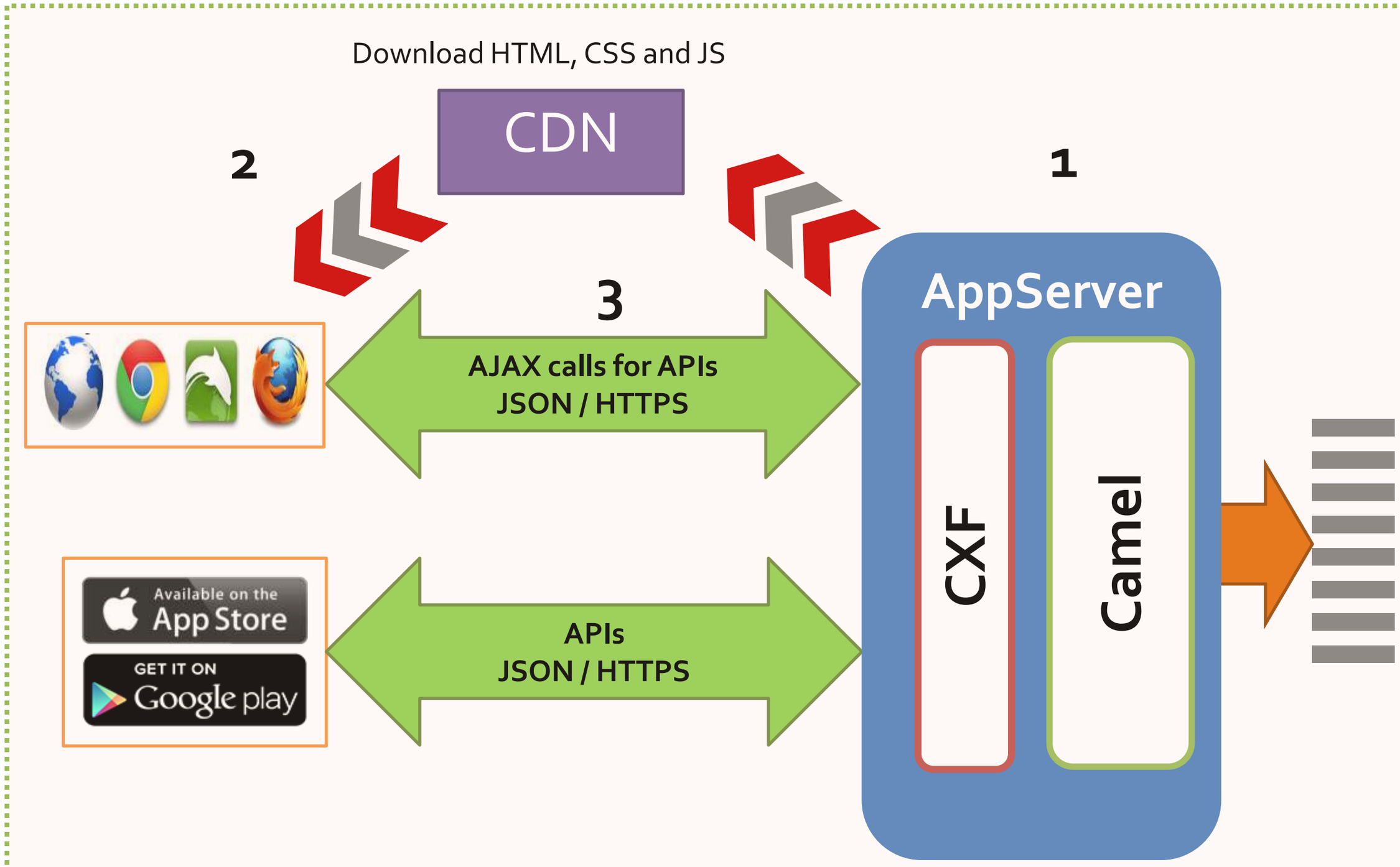
Leverage Best Practices

- Use existing API
- Stand on shoulders of Giants: "There is an API for (nearly) everything"
- Leverage EIP as a language

Divide Work

- Better collaboration
- Quicker turnaround

10K feet Architecture

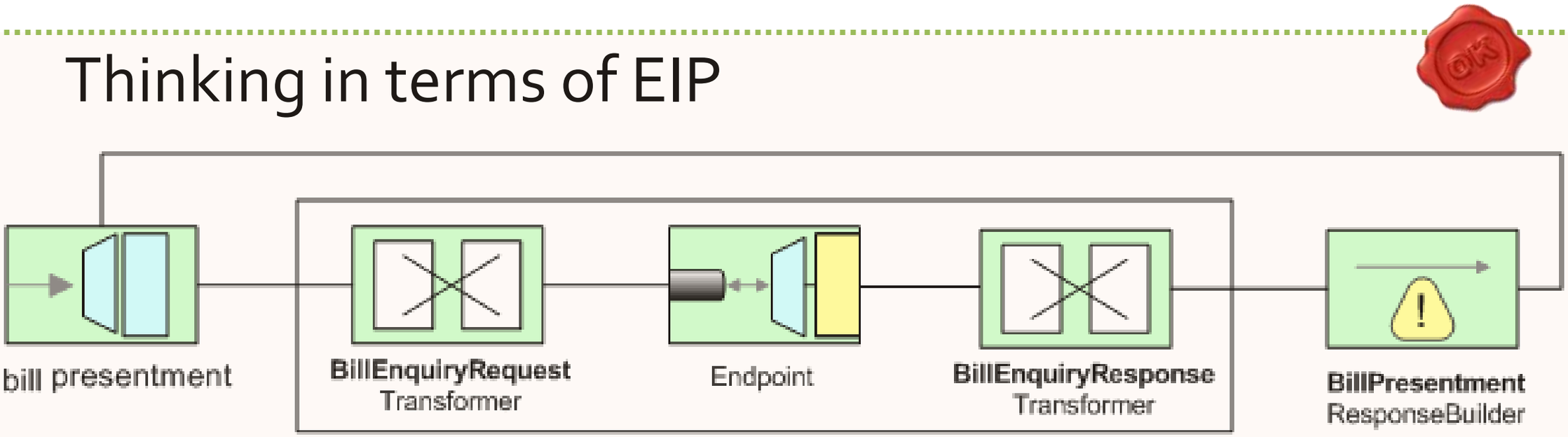


Agenda

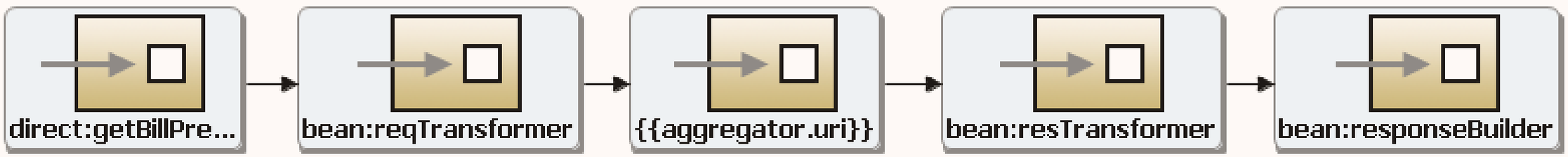
- Digital Banking: One Bank view across channels
- Why a Routing and Mediation engine
- **Before & After Apache Camel**
  - **Simplification**
  - Parallel Processing
  - Testing
- Some rough edges

Concept to Delivery: Verified

Thinking in terms of EIP



Fuse IDE representation of actual route



Decompose Complexity

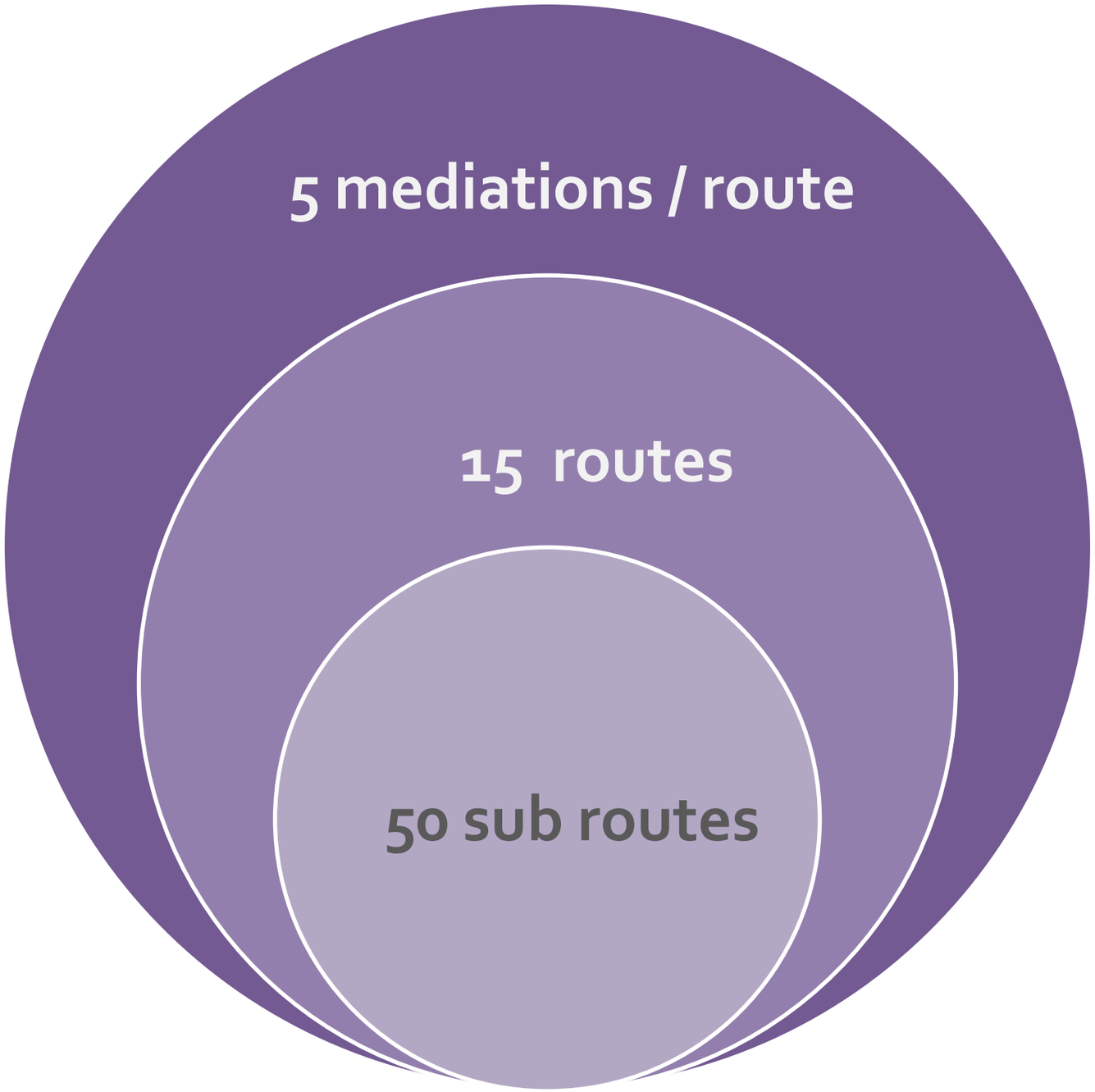
```
<route id="common">
    <from uri="direct:common"/>
    <to uri="bean:ftCasaReqTransformer"/>
    <to uri="{{account.jms.uri}}"/>
    <to uri="bean:ftCasaResTransformer"/>
</route>
```

```
<route id="invokeBillPayment">
    <from uri="direct:invokeBillPayment"/>
    . . .
    <to uri="direct:common"/>
    . . .
</route>
```

```
<route id="invokeFundTransfer">
    <from uri="direct:invokeFundTransfer"/>
    . . .
    <to uri="direct:common"/>
    . . .
</route>
```

Enable Re-use | Reduce Redundancy

Smarter Connections



- **LDAP**: Authenticate
- **SOAP**: Display Smart ad
- **JMS**: Account
- **JMS**: Card
- **JDBC**: Update database
- **SMTP**: Send email

Flexible Data Carrier

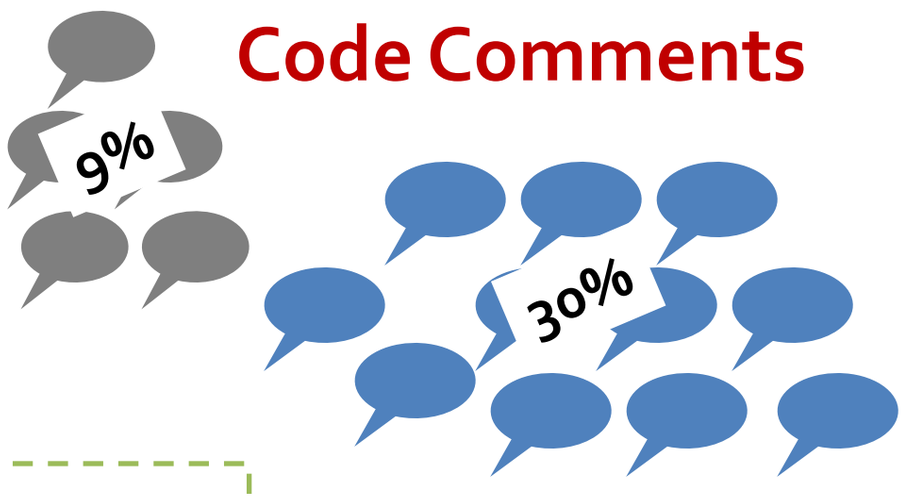
MessageContentsList

```
public void process(Exchange ex) throws Exception {

    //The data carrier in routes is of type MessageContentsList
    MessageContentsList mcl = (MessageContentsList) ex.getIn().getBody();

    //BillPaymentRequest is searched by type in the MessageContentsList
    ex.getIn().getBody(BillPaymentRequest.class);

    //AdditionalData is searched by type in the MessageContentsList
    ex.getIn().getBody(AdditionalData.class);

    //futher processing
}
```

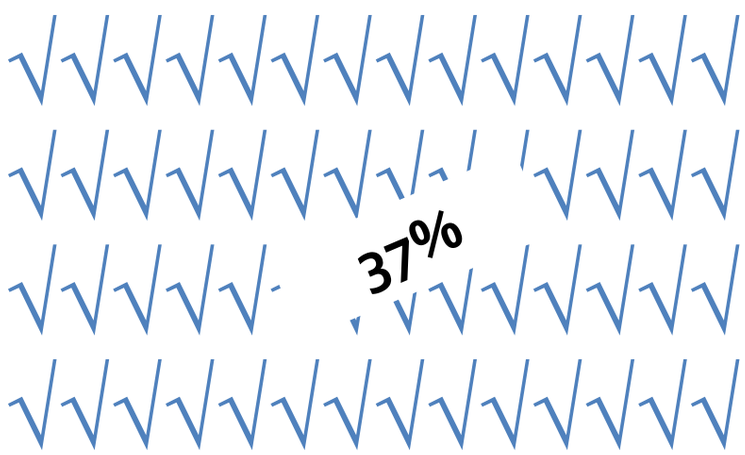
Reduce multiple data transfer objects

Simplify: Reduce & Magnify

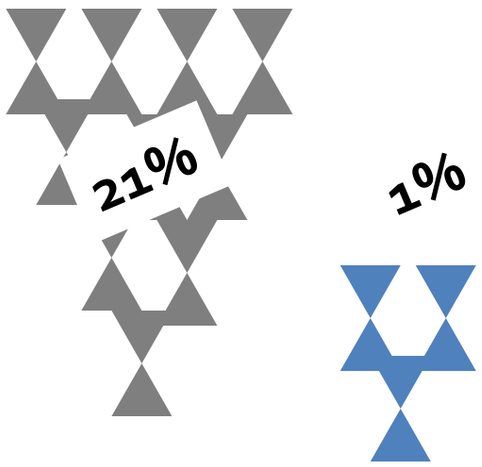
Lines of code



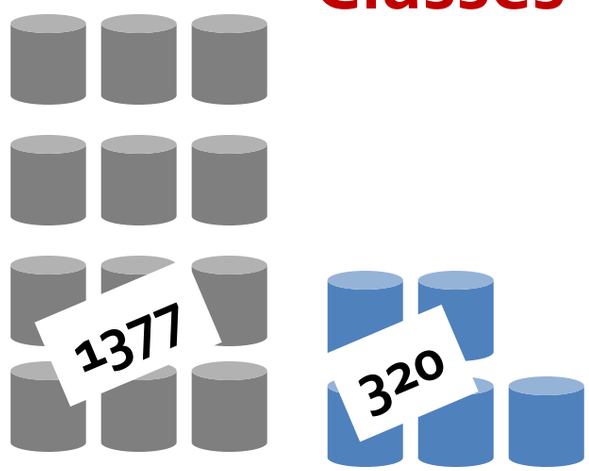
Code Comments



Unit test coverage



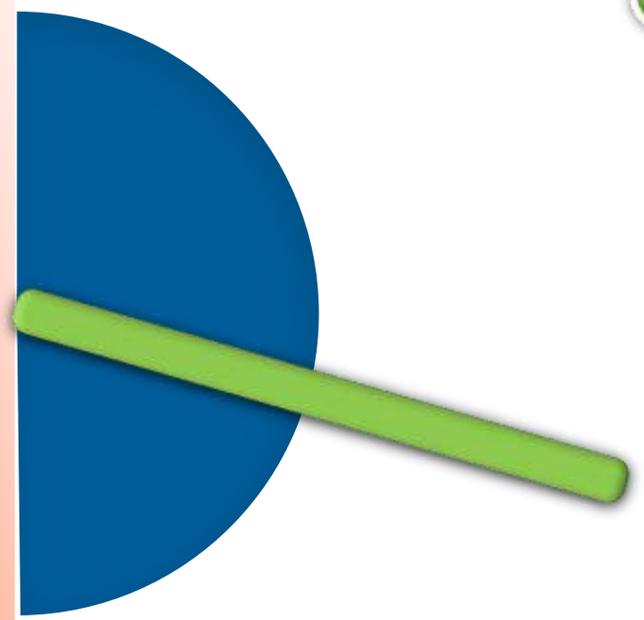
Duplication



Classes

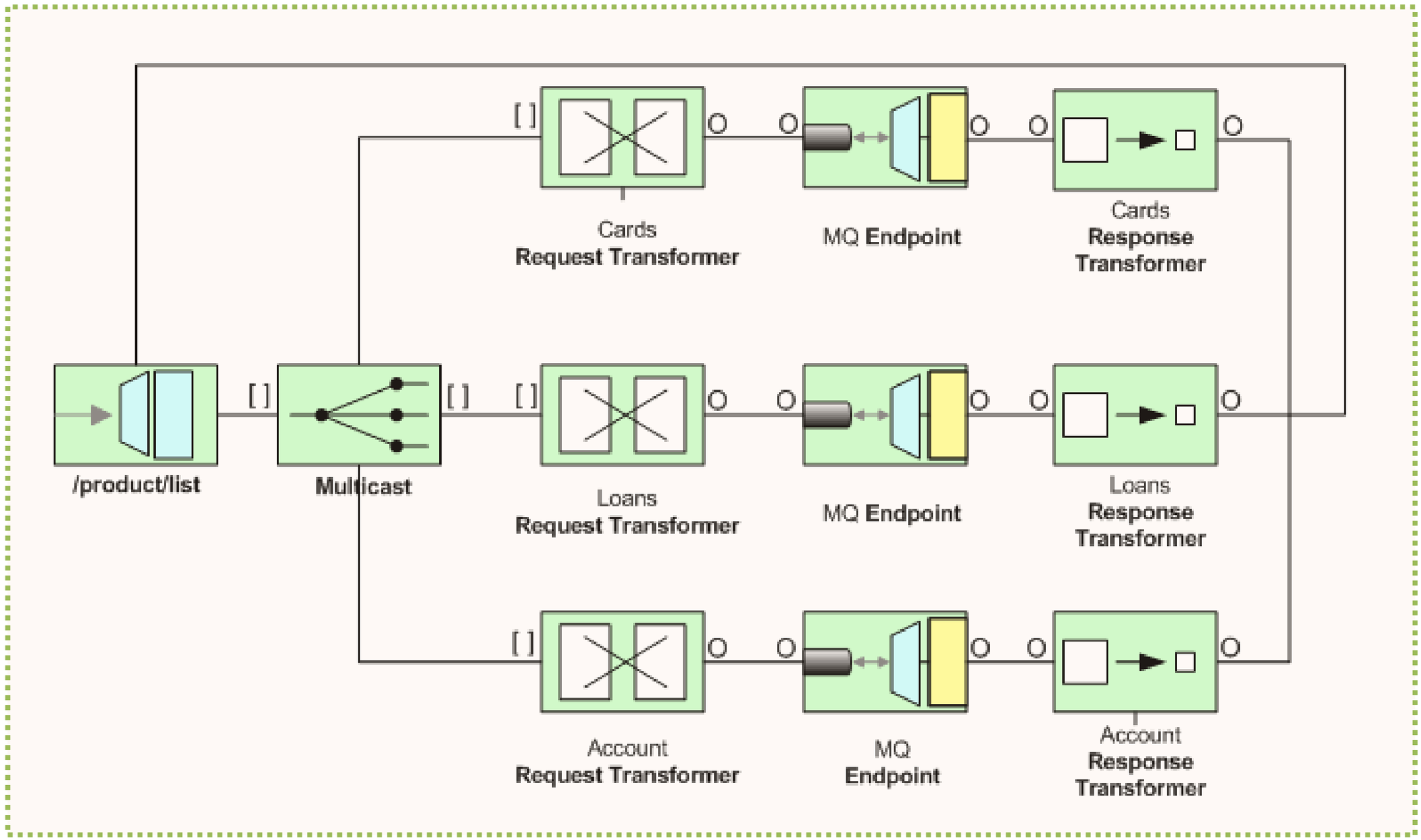
Agenda

- Digital Banking: One Bank view across channels
- Why a Routing and Mediation engine
- **Before & After Apache Camel**
  - Simplification
  - **Parallel Processing**
  - Testing
- Some rough edges



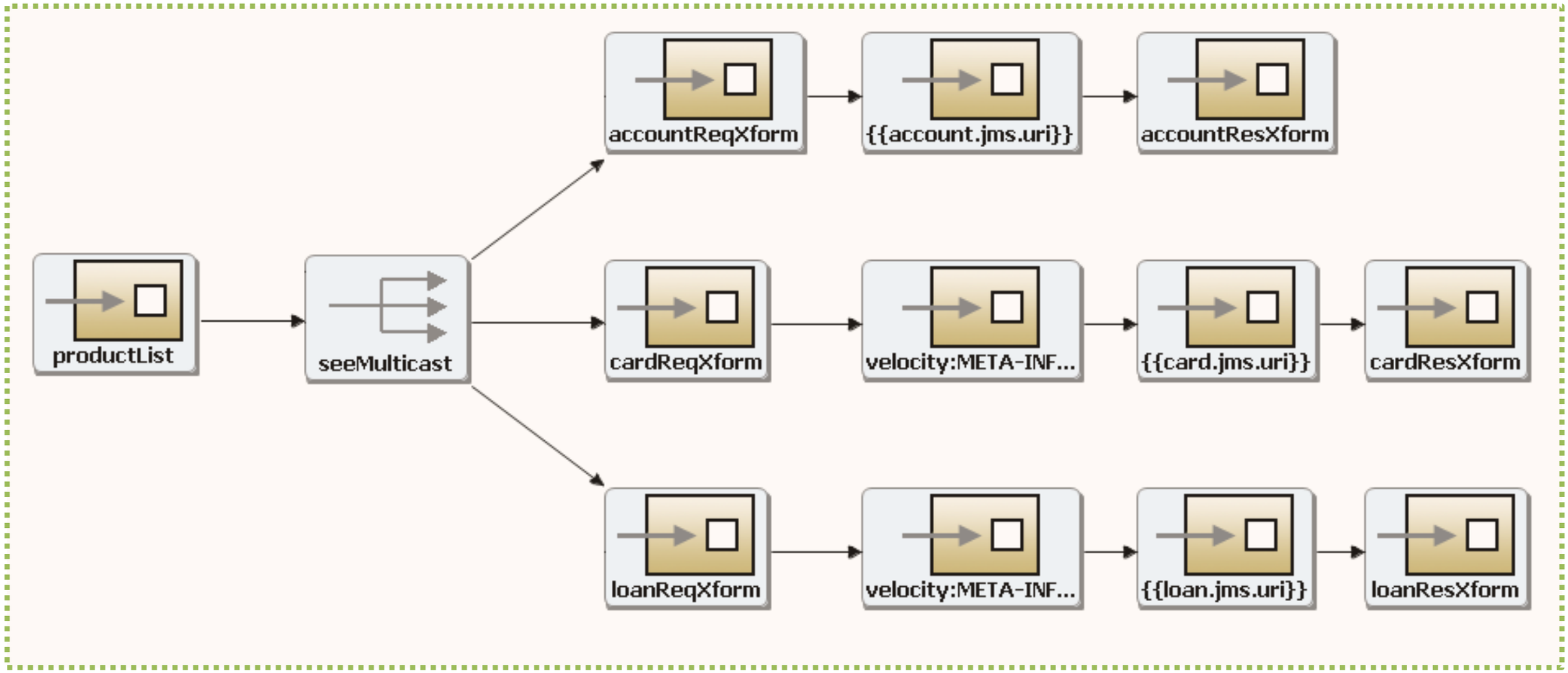
Parallel Processing: Out of Box

Concept › Delivery › Configuration



Parallel Processing: Out of Box

Concept → **Delivery** → Configuration



Parallel Processing: Out of Box

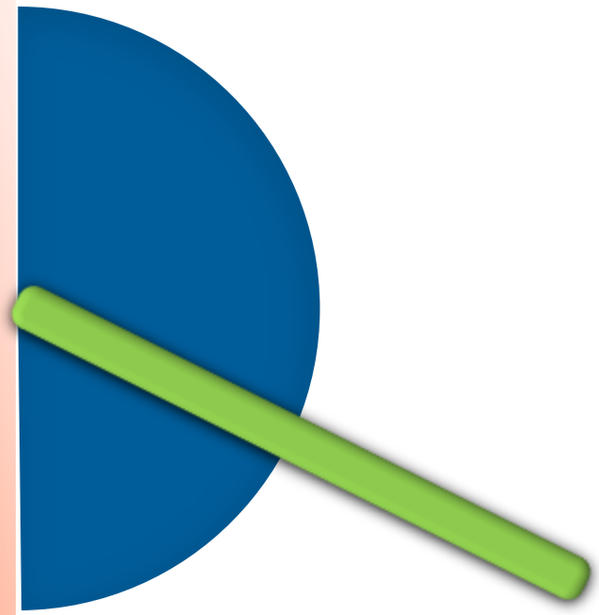
Concept → Delivery → **Configuration**

```
<multicast parallelProcessing="true"
           strategyRef="aggrStrategy"
          executorServiceRef="lowPool"
           timeout="20000">
    <to uri="direct:getAccountList"/>
    <to uri="direct:getCardList"/>
    <to uri="direct:getLoanList"/>
</multicast>
```

No explicit threads

Agenda

- Digital Banking: One Bank view across channels
- Why a Routing and Mediation engine
- **Before & After Apache Camel**
  - Simplification
  - Parallel Processing
  - **Testing**
- Some rough edges



Testing

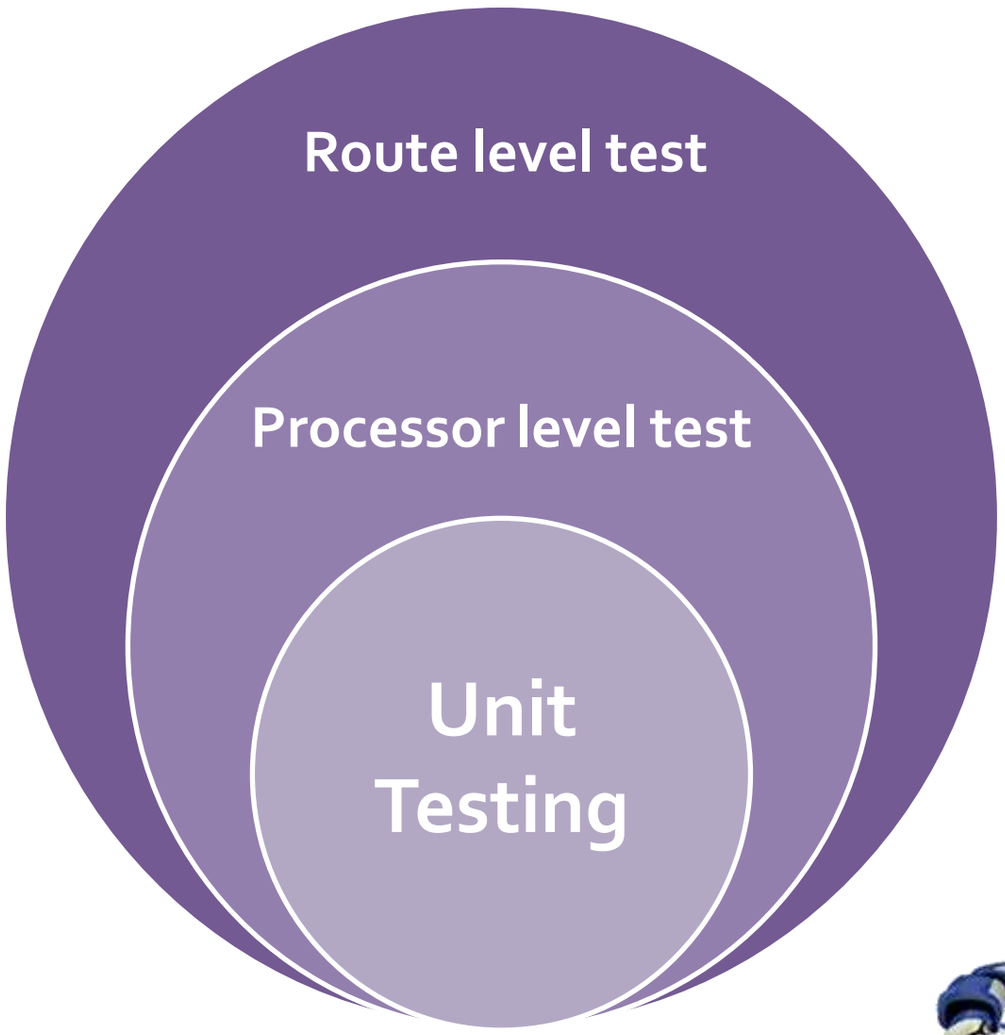
Before

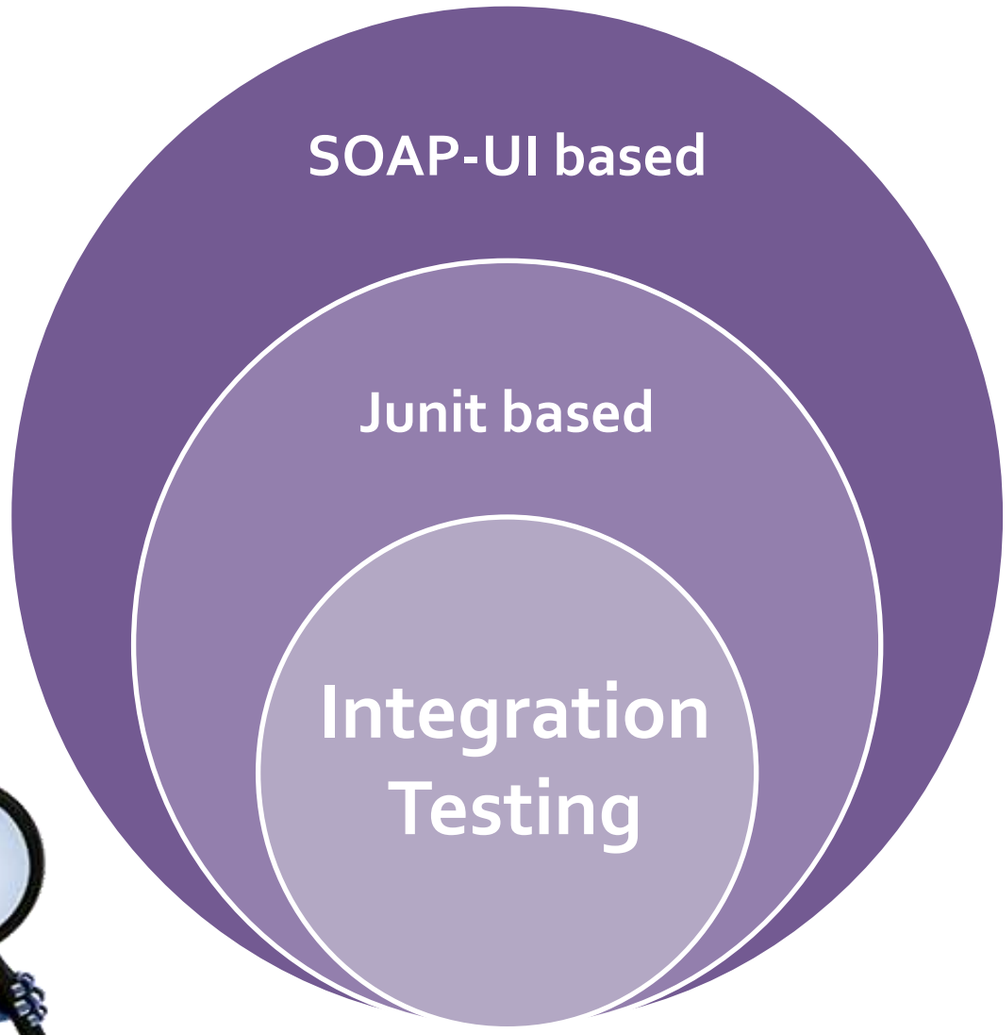
- Unit test limited to utilities
- Manual System testing after completing coding
- Bug fixing and regression is cumbersome
- Difficult to test & code when sub systems are down

After

- Now follow TDD, with focus on mocking sub systems
- Uninterrupted testing and development via mocking objects

Testing inside out: Cover Everything





Testing: the Camel way

Producer templates

```
template.requestBody("direct:invokeBillPayment", request);
```

Advice with

```
private void weaveJMS() throws Exception {
   context.getRouteDefinition(“common.ftCasaRoute")
   .adviceWith(contex, new AdviceWithRouteBuilder() {
         public void configure() throws Exception {
               weaveById(“jms.uri").replace()
               .bean(new SimulatedJMSResponseProcessor());
   }});}
```

Mock Components

```
public void configure() throws Exception {
   weaveById("emailProcessor").after().to("mock:result”);
}
```

Agenda

- Digital Banking: One Bank view across channels
- Why a Routing and Mediation engine
- Before & After Apache Camel
- Simplification
- Parallel Processing
- Testing
- **Some rough edges**

Rough Edges

- List & Camel JMS
- List & Camel Simple expression
- CXF annotations & Camel
- Aggregation Strategy & Exceptions
- Graphical tooling

… The road ahead

- Hawt.io
- SEDA
- Exception Handling with re-tries
- Camel-blureprint
- Micro service - scale the services that are more used than others

“Banking” on Apache Camel

rohit.sitani@capgemini.com

linkedin.com/in/rohitsitani

twitter.com/rohitsitani

QUESTIONS