October 2nd, 2015
Jiri Jetmar : jiri@apache.org
Paul Merlin : paulmerlin@apache.org

Agenda

- **What is Apache Zest ?**
- Zest Libraries & App Building Strategies
- Sample App – Live Coding Session
- Zest Community, Real World Apps & Outlook

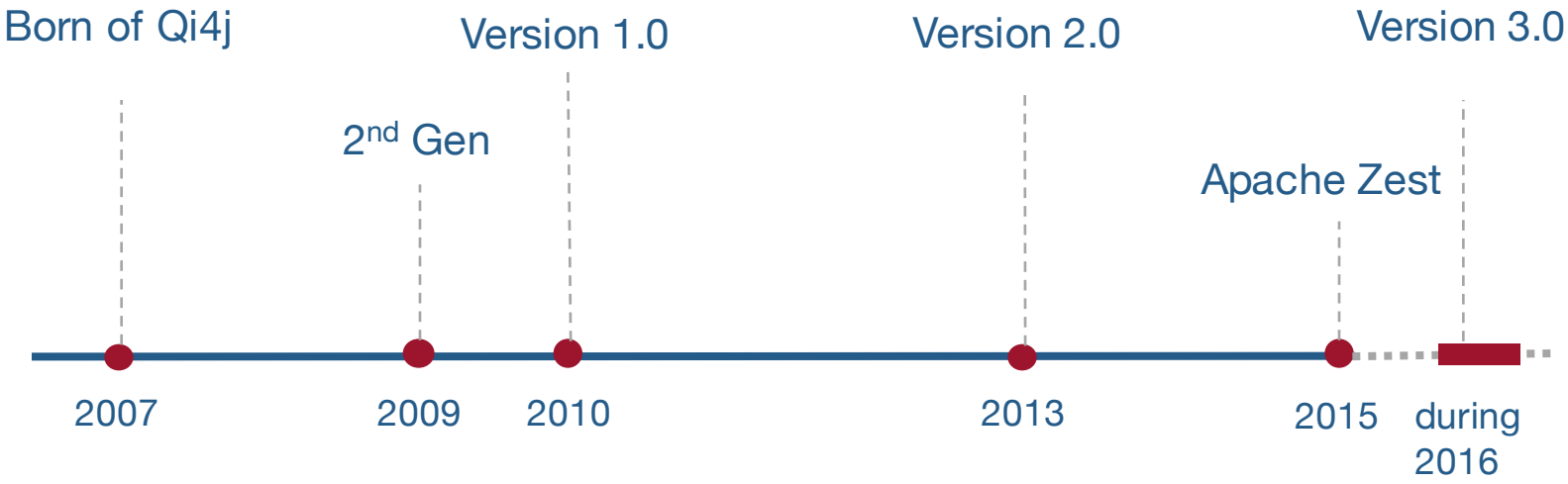
"Genius is one percent inspiration and ninety–nine percent perspiration."

Thomas A. Edison

Apache Zest™ Fact Sheet

:: History

- Apache Zest was born as **Qi4j** in 2007
- Founded by **Richard Öberg** (EJBoss, later JBoss) and **Niclas Hedhman**
- Version 1.0, 1.1 and 1.2 in 2010
- Version 2.0 in 2013
- Since beginning of 2015 a Top Level Apache Foundation Project
- Current Version 2.1, first release of the Qi4j codebase under ASF. Still uses the org.qi4j.* for backward compability
- Version 3.0 in 2016 with lots of new features



Apache Zest™ – It is about solving (business) problems

:: (Re-) Usage of proven patterns

- Zest is nothing new. It is a logical next step based on existing and proved patterns and ideas
- Zest is a Java implementation of the **Composite Oriented Programming** (COP) pattern
- COP is a programming model that allows the creation of rich domain models

:: Business problems centric approach

1) Start with the business problem
2) Use the terminology from Domain Driven Design (DDD)
3) Allow developers to implement the Domain Model directly in code using that Terminology
4) Plug infrastructural building blocks that reflects the needs and requirements

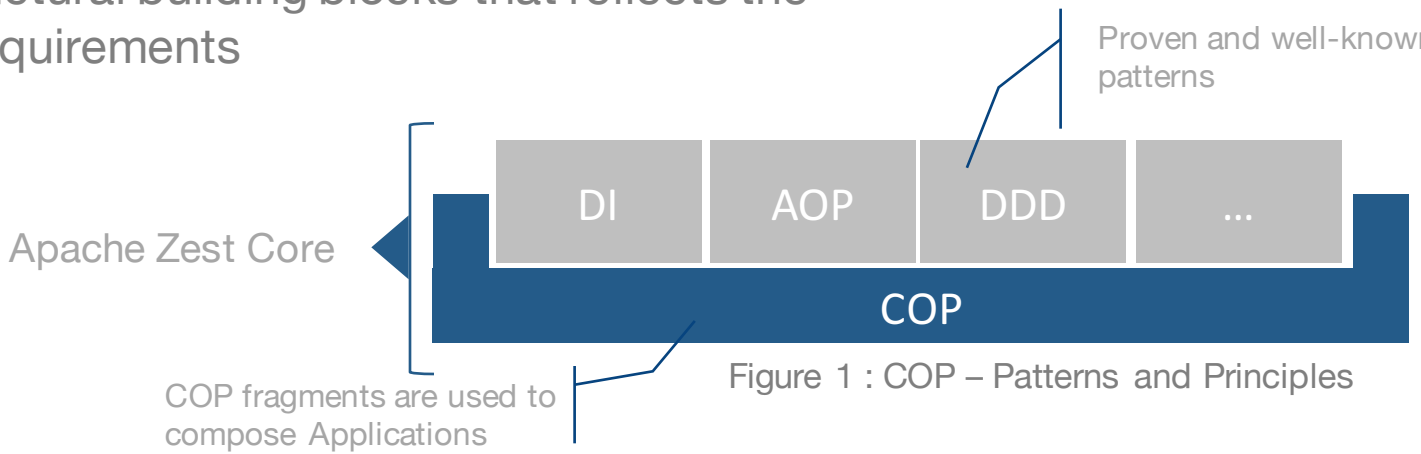


Figure 1 : COP – Patterns and Principles

Apache Zest™ – What kind of problems does Zest solve ?

:: Rich Domain Models

- When it is required to manage a high number of states
- Avoids growing codebase complexity
- Separation of storage and index

:: Domain Model evolution

- When the ratio and frequency of Domain Model structural modifications is high
- Simplifies the maintenance of code that interacts with the Domain Model
- The Domain Model becomes “elastic”, refactorings are done on the source code level by refactorable artifacts (e.g. Entities, Commands, Queries)

:: API & Data centric, integrative and technology independent Apps

- When it is required to provide a API (e.g. REST)
- When there is a need to integrate external, 3th party Services on different levels (API, RPC, REST, ..)
- It is required to deal with large amounts of data

Apache Zest™ & COP Overview

:: COP in one sentence

"Composite Oriented Programming allows developers to work with 'fragments', smaller than classes, and 'compose' fragments into larger 'composites' which acts like the regular objects."

Niclas Hedhman

:: COP & Apache Zest

- The most basic element in Zest is the Fragment
- A Composite is created by composing a number of Fragments
- Mixins are Fragments that can handle method invocations
- Modifiers are Fragments that modify method invocations (Decorator pattern) – Constraints, Concerns, SideEffects

Apache Zest™ Basic Fragments

:: Fragments

- **@Concerns** intercept method calls
  - Allowed to modify arguments and i.) return values, ii.) return with calling in chain, iii.) throws exceptions
- **@Constraint** validates method arguments
  - Can have many Constraints per argument
  - Uses annotations to trigger
  - Cooperate with concern for failure actions
- **@SideEffects** are called after a method call has finished
  - Cannot change method arguments or return value
  - Cannot throw exceptions
  - Can inspect exceptions and return values
  - May be asynchronous
- **@Mixins** implements Composite interfaces
  - A Mixin may implement one interface, many interfaces or just some methods
  - May contain Composite state, such as Property and Association instances
  - May be Composite private – not exposed in Composite interface

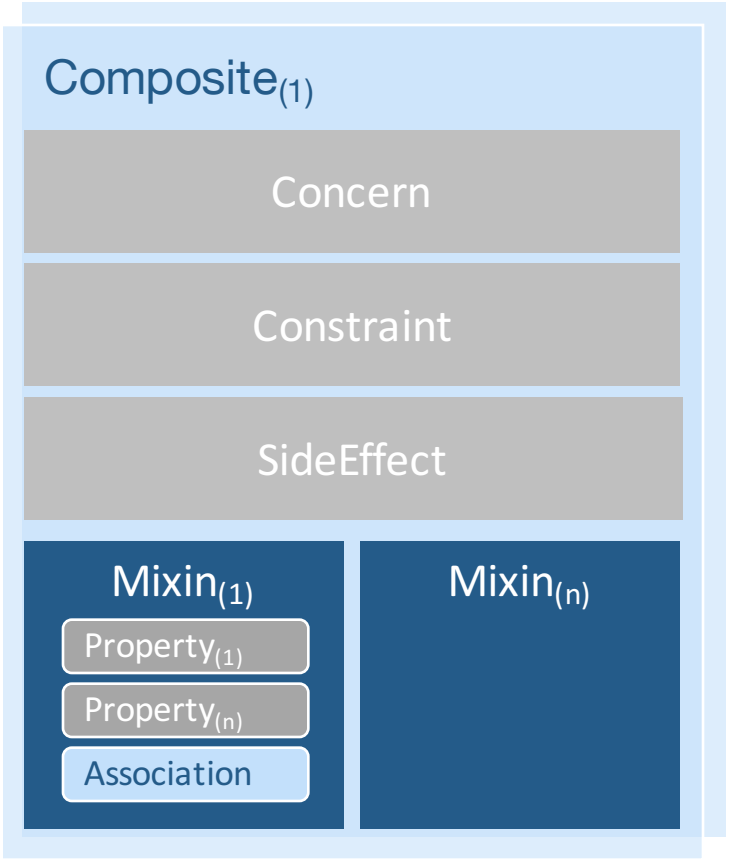


Figure 2 : Composite Fragments

Apache Zest™ Structure Composition

:: Application Structure

- Composites define the internals of objects
  - Explicit Composite meta types : Value, Entity, Service and Transient
- Composites reside in Modules
- Modules can be grouped into Layers
  - One or more Layers per Application
  - Zero, one or more Modules per Layer
- Visibility and dependency of Composites between structures is controlled
  - Declaration of Visibility of Composites

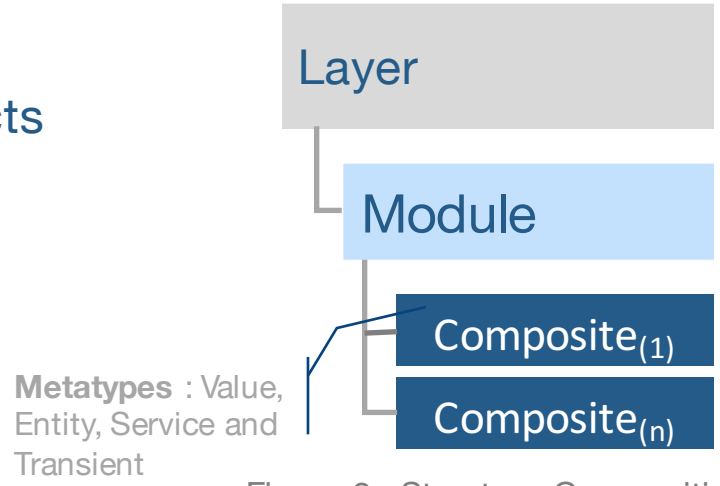


Figure 3 : Structure Composition

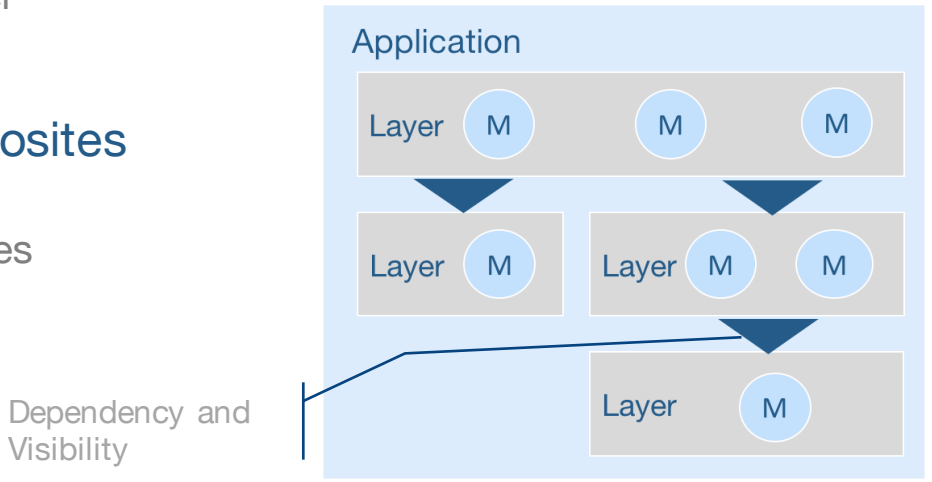


Figure 4 : Application Structuring

Agenda

- What is Apache Zest ?
- **Apache Zest Libraries & App Building Strategies**
- Sample App – Live Coding Session
- Zest Community, Real World Apps & Outlook

„We learn something every day, and lots of times it's that what we learned the day before was wrong."

Bill Vaughan

Apache Zest™ Core, Libraries and Extensions

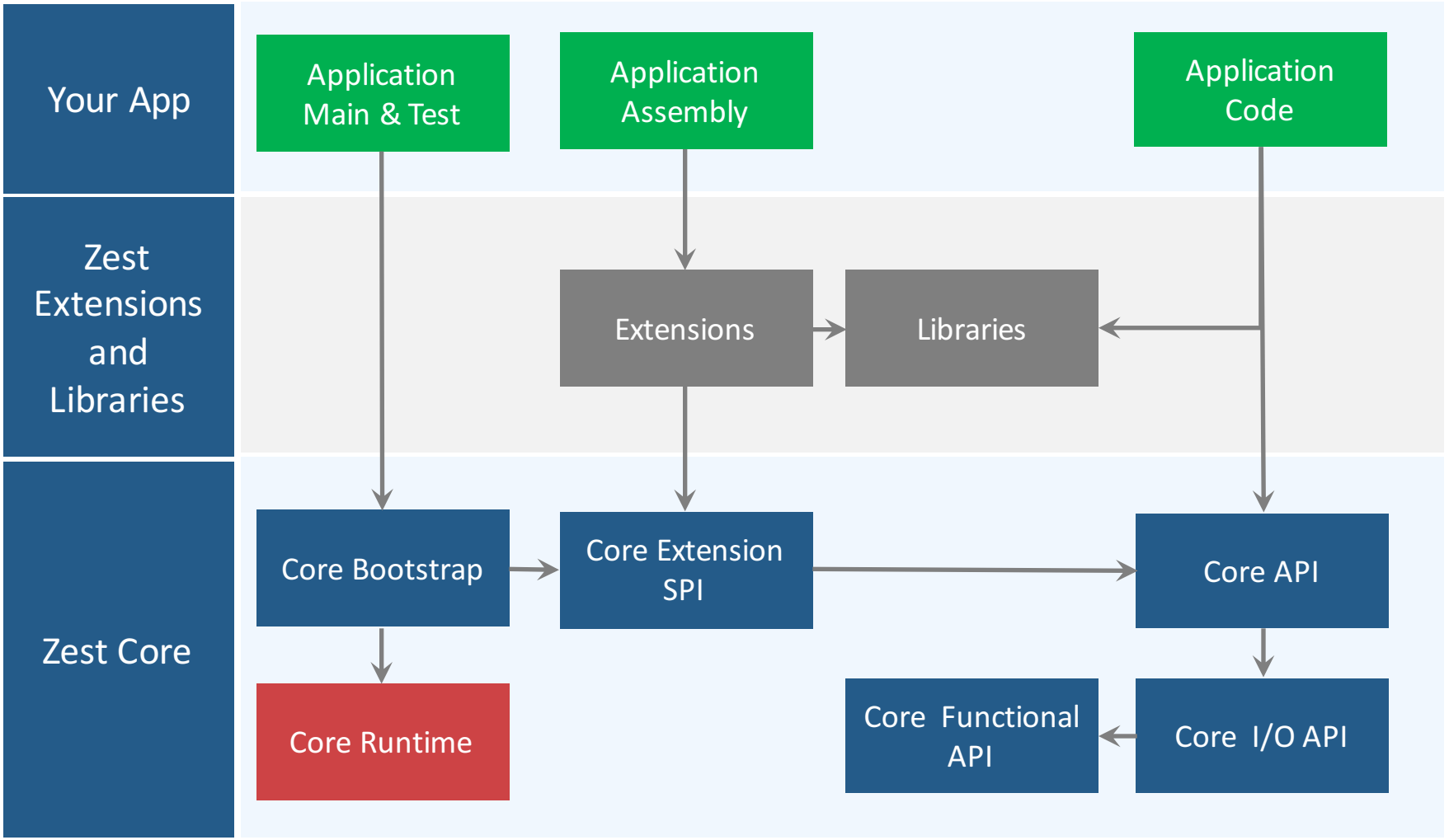
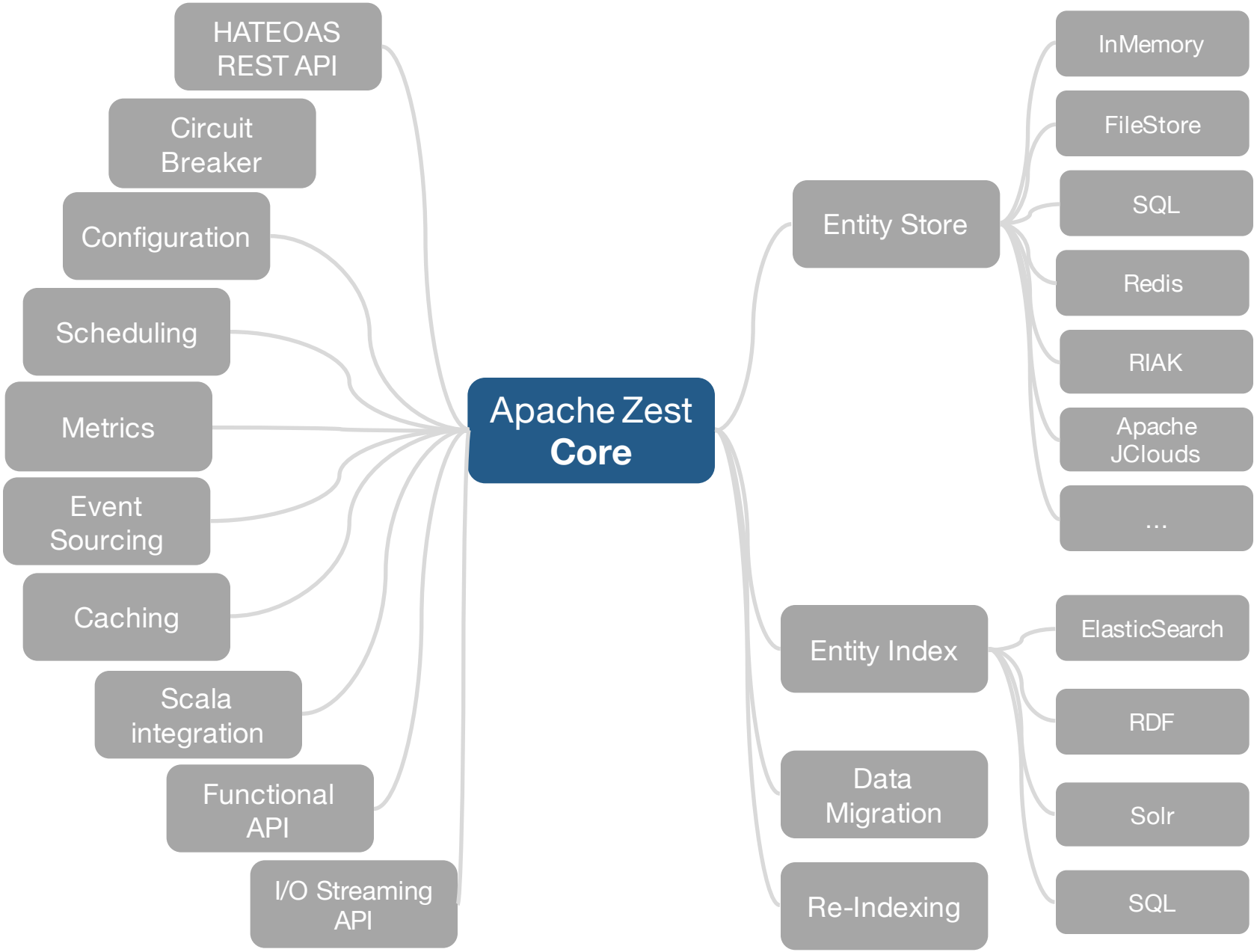
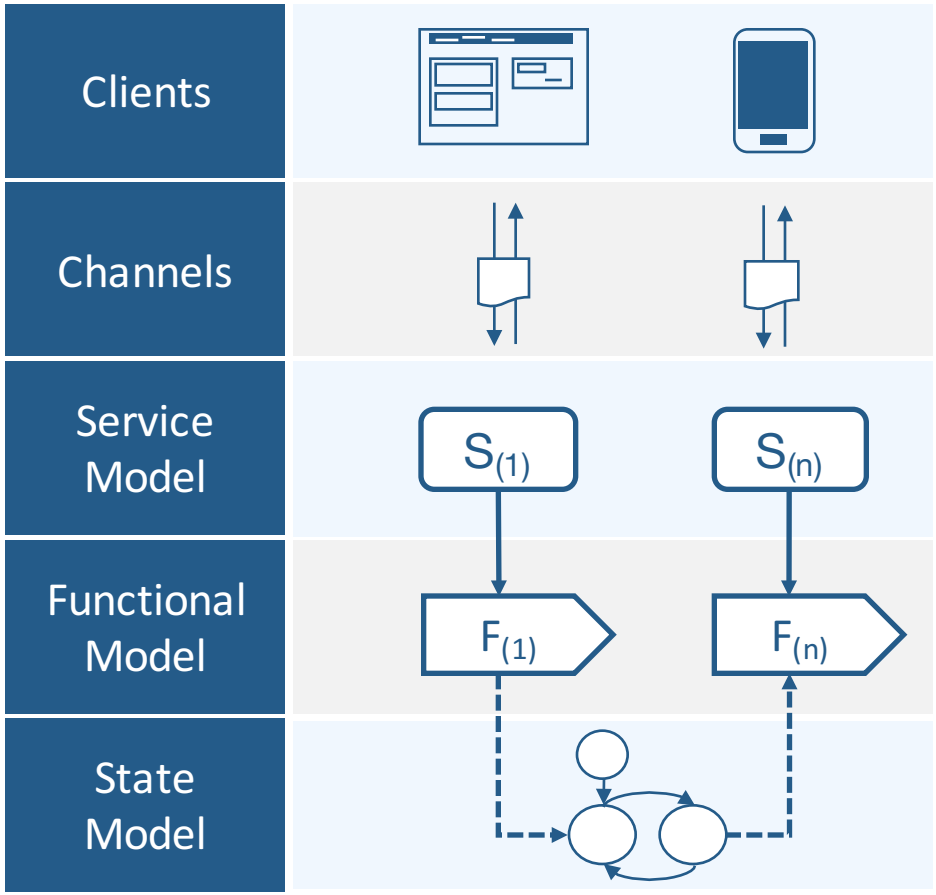


Figure 5 : Zest Core, Libraries and Extensions

Apache Zest™ Library ecosystem (.. there is much more !)



Application Building – A bit „grey“ theory



- Clients consumes the application state and
- triggers state mutations (e.g. new Order)

- Defines the Message Exchange Patterns (MEP‘s) e.g. In/Out
- Technology independent but highly influences the used technology

- Defines a Service API (Contract)
- Specifies the Data Exchange Format (e.g. JSON) and communication schemas (REST/RPC)

- Provides a implementation for a Usecase
- Interacts (CRUDF) with the state model

- Application related Domain Model (DM)
- Represents a state of a Usecase (e.g. a Reservation has a state flow like new, pending, canceled, expired)

Figure 6 : Top-Down Application Model

Application Building – The Zest way !

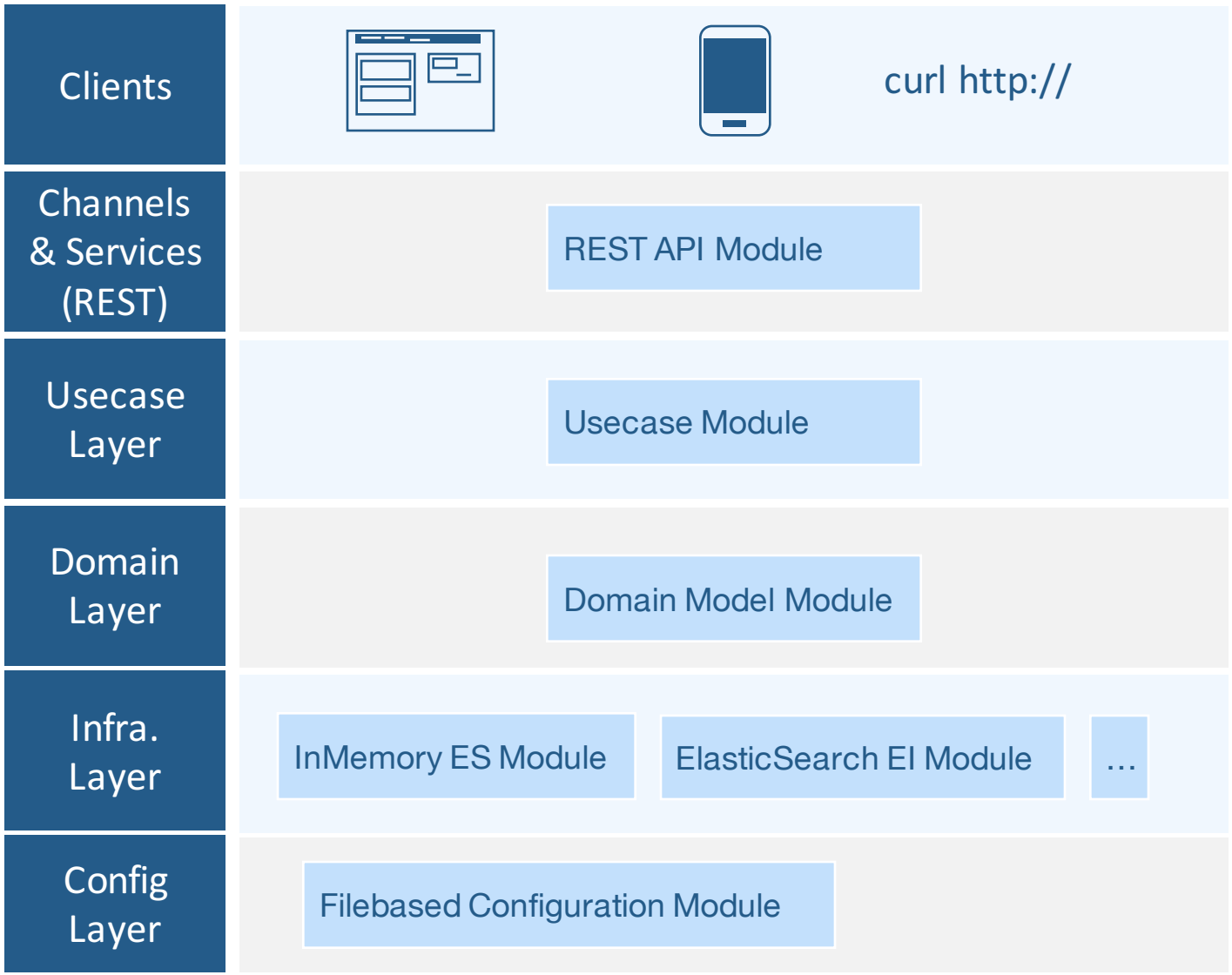


Figure 7 : Concrete Zest-based Application structure

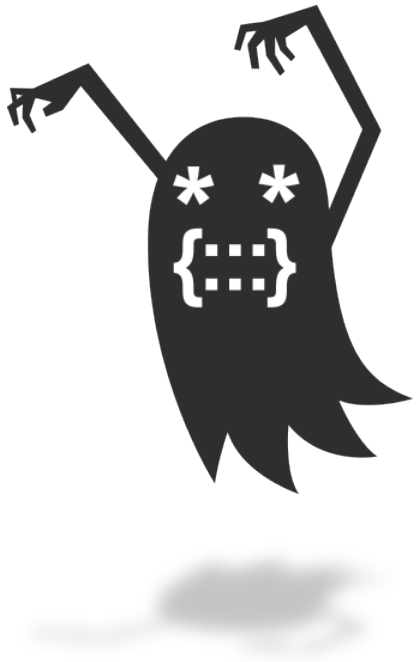
Agenda

- What is Apache Zest ?
- Zest Libraries & App Building Strategies
- **Sample App – Live Coding Session**
- Zest Community, Real World Apps & Outlook

„The only constant in the Universe is change“

Albert Einstein

Sample App – Live Coding Session



$ git clone https://github.com/eskatos/zest-apache-con.git

Agenda

- What is Apache Zest ?
- Key Components, Building Blocks & Ecosystem
- Sample App – Live Coding Session
- **Zest Community, Real World Apps & Outlook**

„To find yourself, think for yourself“

Socrates

Apache Zest™ Community

:: Community

The current Zest developer community is rather small, but regroups passionate individuals who are all convinced of Zest's merit and potential. Over the years there has been **28** code contributors in total. There is always lots to do and therefore the Zest community is looking forward to welcome new developers.

The topics to contribute just right now are :

- Discuss further developments on dev@zest.apache.org
- Review and test sample applications
- Review and contribute documentation
- Code contribution areas : core, exts, libs, tools
- We have some low hanging fruits like browser-based visualization of application assemblies (tools), a feature toggling library, browser-based dashboard for CircuitBreaker, EntityStores (exts, yes ! Writing an ES is pretty easy) and much more …

… so do not wait, JOIN us right now !!

Who is using Apache Zest™ ?

smarpay
simply clever mobile payments

Productive

http://smarpay.ch is a swiss company that offers mobile payment services in the vending business. Apache Zest is used to offer backoffice services like settlement and clearing for about 3k vending machines and about 350k mobile users.

P*

Not yet released

http://dieparkuhr.de is a german Startup that offers services around the car parking business. Apache Zest is used for the entire backoffice with a large number of services like User / Partner Management, Product Catalog, Reservation and Scheduling System, Payment Services.

Sf streamflow

Productive

Streamflow is a case management system for Swedish government departments.

Bali Automation

Not yet released

IoT data analytics and visualization.

...

further we know that Apache Zest is used is in a number of Fortune 500 companies as well as in a number of small companies. Zest is used in Trading, in security related applications in the Oil and Energy Sector and even in Game development.

Apache Zest™ Outlook

:: Current Version is **2.1**

- First release of the Qi4j codebase under the ASF umbrella. Still uses org.qi4j.* for backward compability.

:: Outlook for 3.0 (in 2016)

- Package change to org.apache.zest.*
- Internals of Zest will be ported to Java 8
- GeoSpatial Queries
- Enhancements on the Entity Storage
- Explicit Timeseries support
- Moving Javascript library from rhino to nashorn
- Configuration System enhancement (e.g. remote configuration)

Q & A

Thank you !

See you at https://zest.apache.org !