Software Analysis Using jQAssistant and Neo4j

Java User Group Darmstadt
11/2015

Dirk Mahler

AGENDA

- jQAssistant
- Software As A Graph
- About Structures, Rules and Code
- Verifying Rules With The Graph Model
- Wrap Up
- Q&A

Software Analysis Using jQAssistant And Neo4j

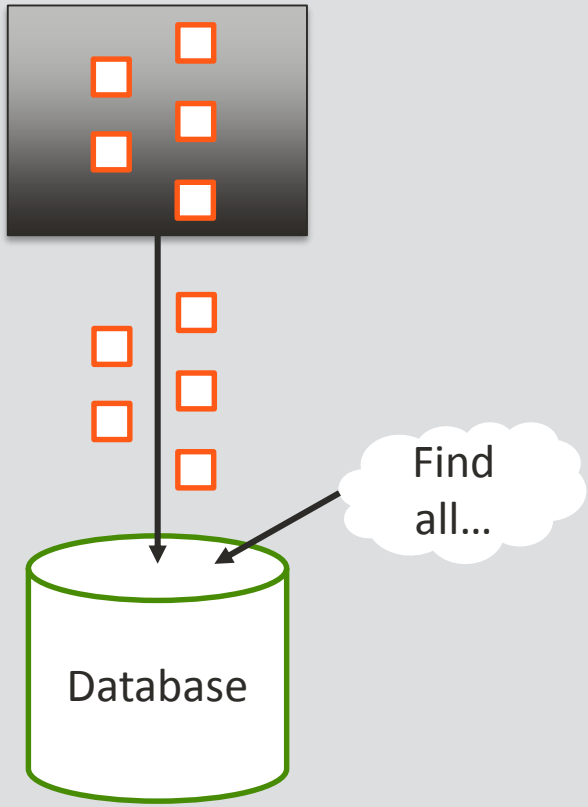
jQAssistant

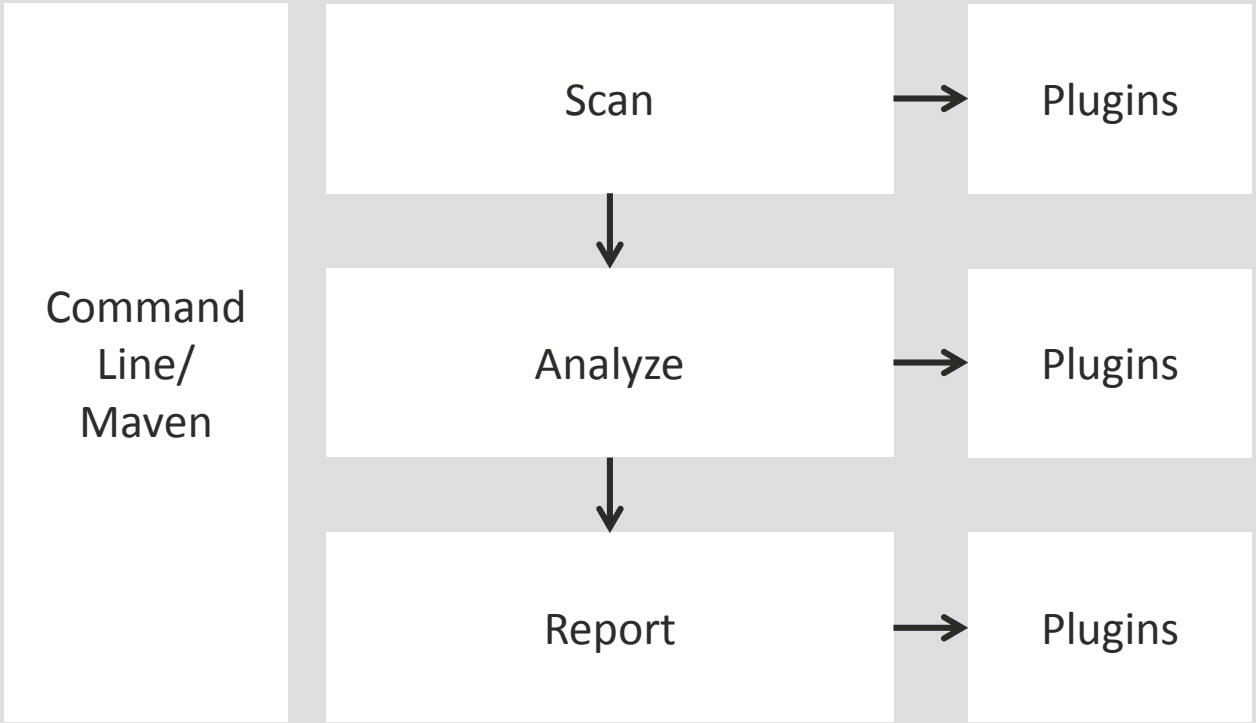
jQAssistant

http://jQAssistant.org

- Open Source: GPLv3
- Current release: 1.0.0 (1.1.0-RC2)
  - initiated: 03/2013
  - first stable release: 04/2015
- Neo4j Community Edition
  - http://neo4j.org
  - embedded, no installation necessary

- The Idea
  - Scan software structures
  - Store in a database
  - Execute queries
    - Explore
    - Add high level concepts
    - Find constraint violations
  - Create Reports




Command Line/ Maven
Scan
Analyze
Report
Plugins
Plugins
Plugins

- Available scanner plugins

ZIP

JAR, WAR, EAR

GZ

*.class

*.properties

*.yaml

MANIFEST.MF

/META-INF/services/*

web.xml

application.xml

beans.xml

pom.xml

surefire-reports.xml

*.xsd

Liquibase

CheckStyle

FindBugs

JaCoCo

Git

- Plugin API is public

M2 Repository

RDBMS Schema

Getting Started – Command Line

- `unzip jqassistant.distribution-1.0.0.zip`
- `cd jqassistant.distribution-1.0.0/bin`

- `jqassistant.sh scan -f model.jar`
- `jqassistant.sh scan -f acme.war`
- `jqassistant.sh scan -f acme.ear`
- `jqassistant.sh scan -u http://somewhere.com/acme.ear`
- `jqassistant.sh scan -u maven:repository::http://host/releases`
- `jqassistant.sh scan -u rdbms:schema::jdbc:oracle:thin:user/secret@host:1521:sid`

- `jqassistant.sh server`
  → http://localhost:7474

Getting Started – Maven Project

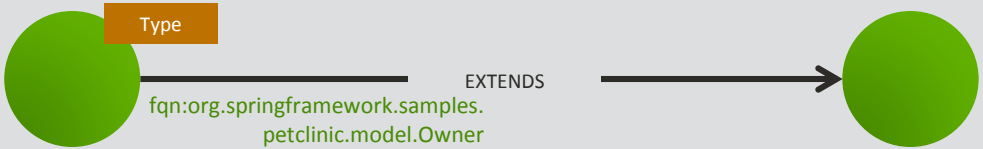
```
<build>
  <plugins>
    <plugin>
      <groupId>com.buschmais.jqassistant.scm</groupId>
      <artifactId>jqassistant-maven-plugin</artifactId>
      <version>1.0.0</version>
     </plugin>
  </plugins>
</build>
```

- `mvn install jqassistant:scan`
- `mvn jqassistant:server`

Software Analysis Using jQAssistant And Neo4j

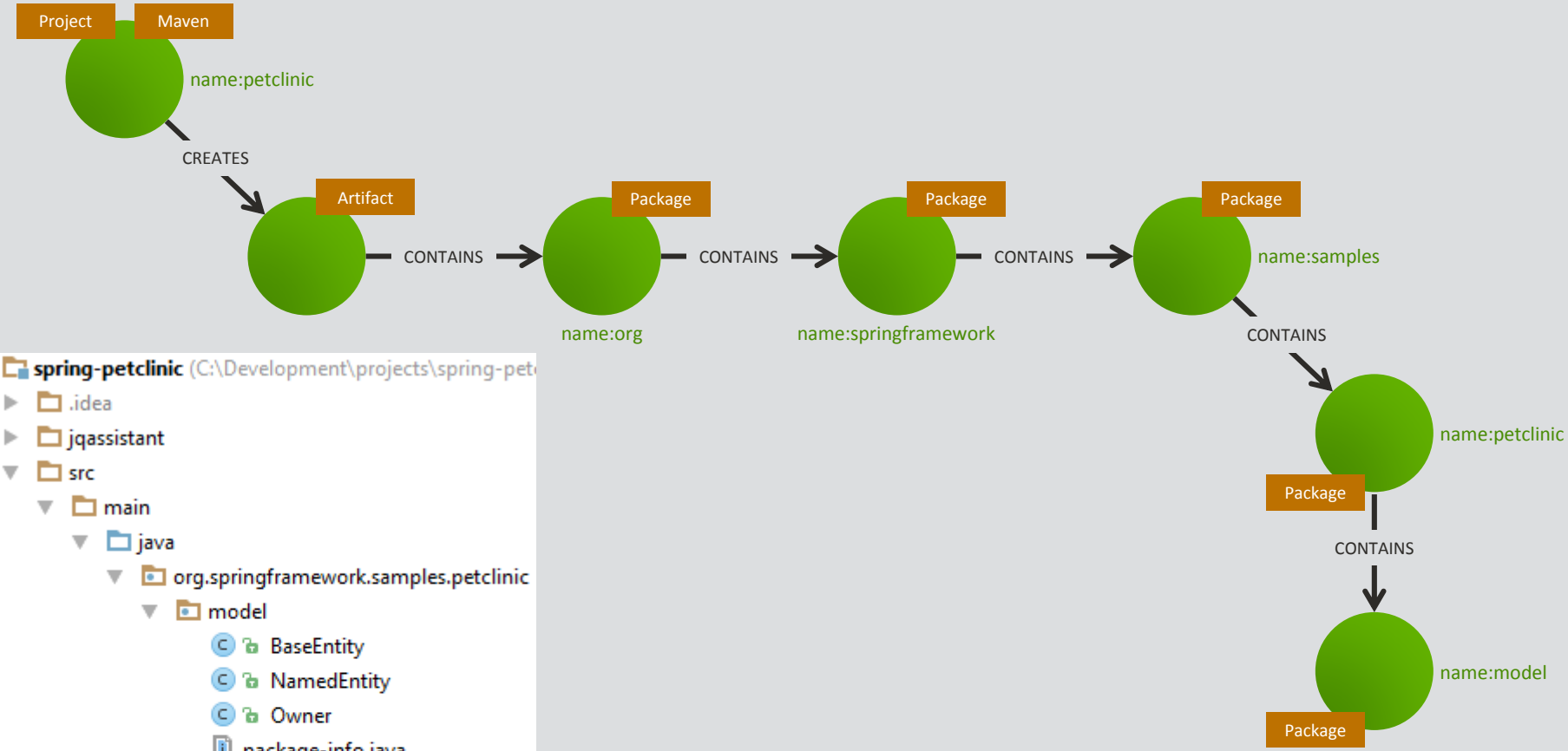
Software As A Graph

- All we need is…
  - Nodes
  - Labels
  - Properties
  - Relationships



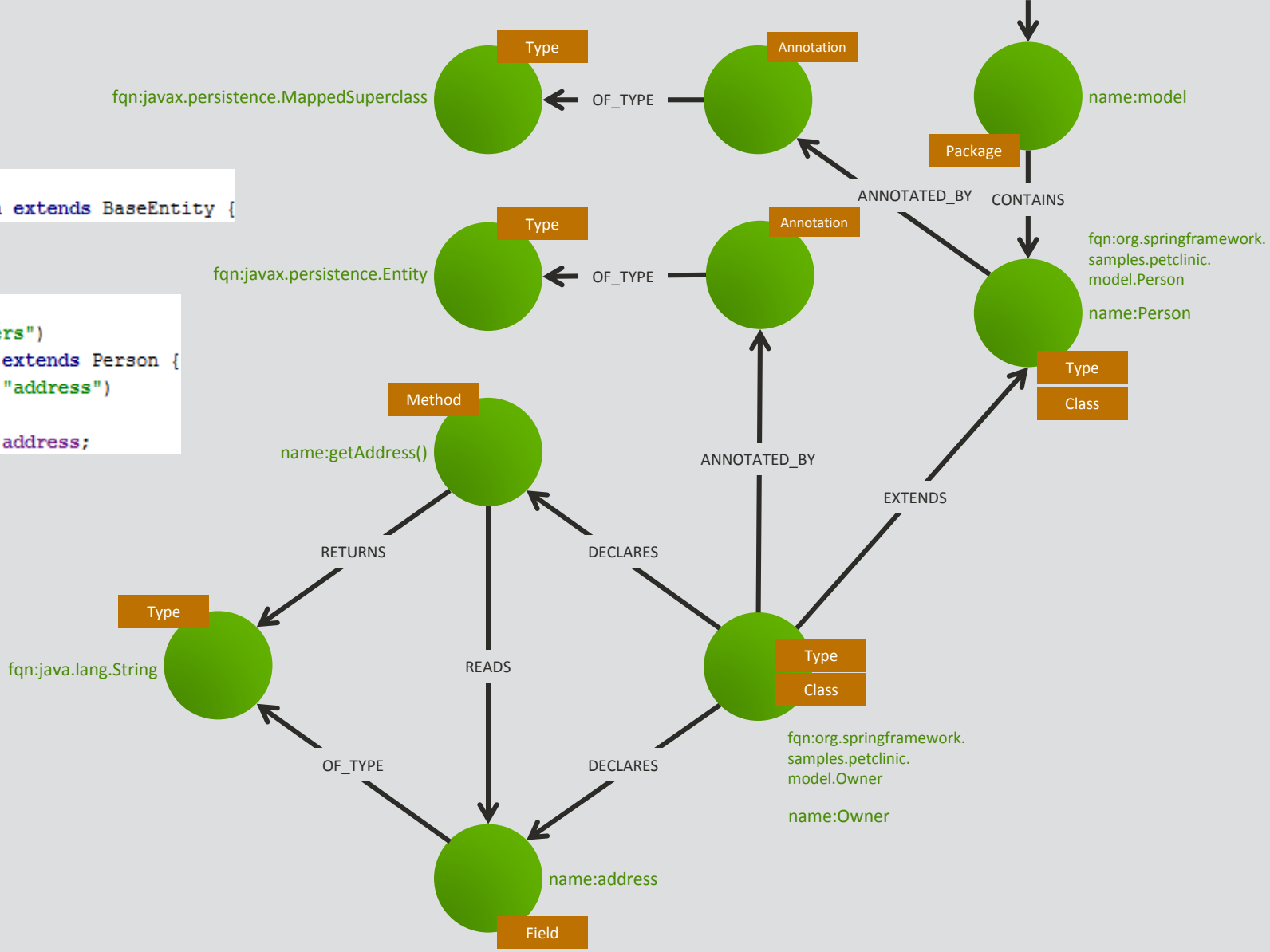
- Modeling is just…
  - Taking a pen
  - Drawing the structures on a whiteboard (i.e. the database)

- We don‘t need…
  - Foreign keys
  - Tables and schemas
  - Deep knowledge in graph theory


Project
Maven
name:petclinic
CREATES
Artifact
CONTAINS
Package
name:org
CONTAINS
Package
name:springframework
CONTAINS
Package
name:samples
CONTAINS
name:petclinic
Package
CONTAINS
name:model
Package
spring-petclinic (C:\Development\projects\spring-pet
.idea
jqassistant
src
main
java
org.springframework.samples.petclinic
model
BaseEntity
NamedEntity
Owner
package-info.java
Person
Pet
PetType
Specialty
Vet
Vets
Visit

```java
@MappedSuperclass
public class Person extends BaseEntity {
```

```java
@Entity
@Table(name = "owners")
public class Owner extends Person {
    @Column(name = "address")
    @NotEmpty
    private String address;
```



- Explore an application using queries
  - Which class extends from another class?

- Let‘s convert this to ASCII art…
  - () as nodes
  - -[]-> as directed relationships

- Explore an application using queries
  - Which class extends from another class?

- Let‘s convert this to ASCII art…
  - () as nodes
  - -[]-> as directed relationships

```
()-[]->()
```

- Explore an application using queries
  - Which class extends from another class?

- Let‘s convert this to ASCII art…
  - () as nodes
  - -[]-> as directed relationships

```
(c1)-[]->(c2)
```

- Explore an application using queries
  - Which class extends from another class?

- Let‘s convert this to ASCII art…
  - () as nodes
  - -[]-> as directed relationships

```
(c1)-[:EXTENDS]->(c2)
```

- Explore an application using queries
  - Which class extends from another class?

- Let‘s convert this to ASCII art…
  - () as nodes
  - -[]-> as directed relationships

```
(c1:Class)-[:EXTENDS]->(c2:Type)
```

- Explore an application using queries
  - Which class extends from another class?

- Pattern matching is the core principle of Cypher

```
MATCH
  (c1:Class)-[:EXTENDS]->(c2:Type)
RETURN
  c1.fqn, c2.fqn
```

Software Analysis Using jQAssistant And Neo4j

Demo #1

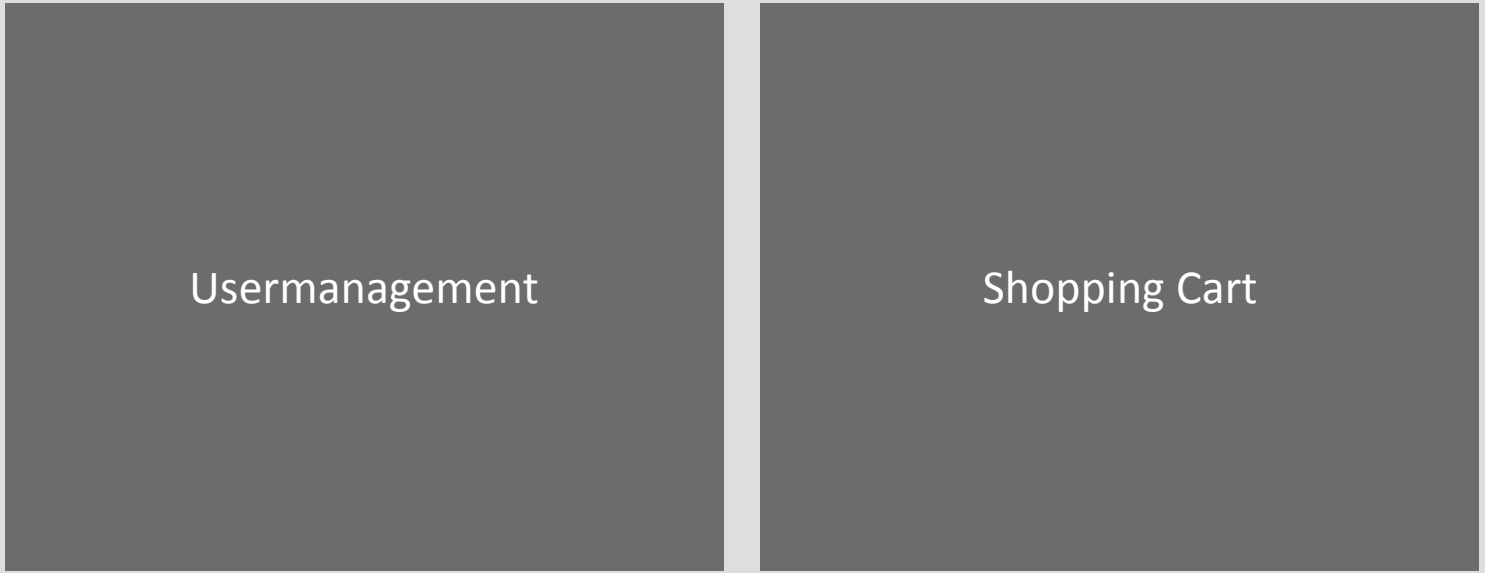
Analysis of software systems using jQAssistant and Neo4j

About Structures, Rules And Code

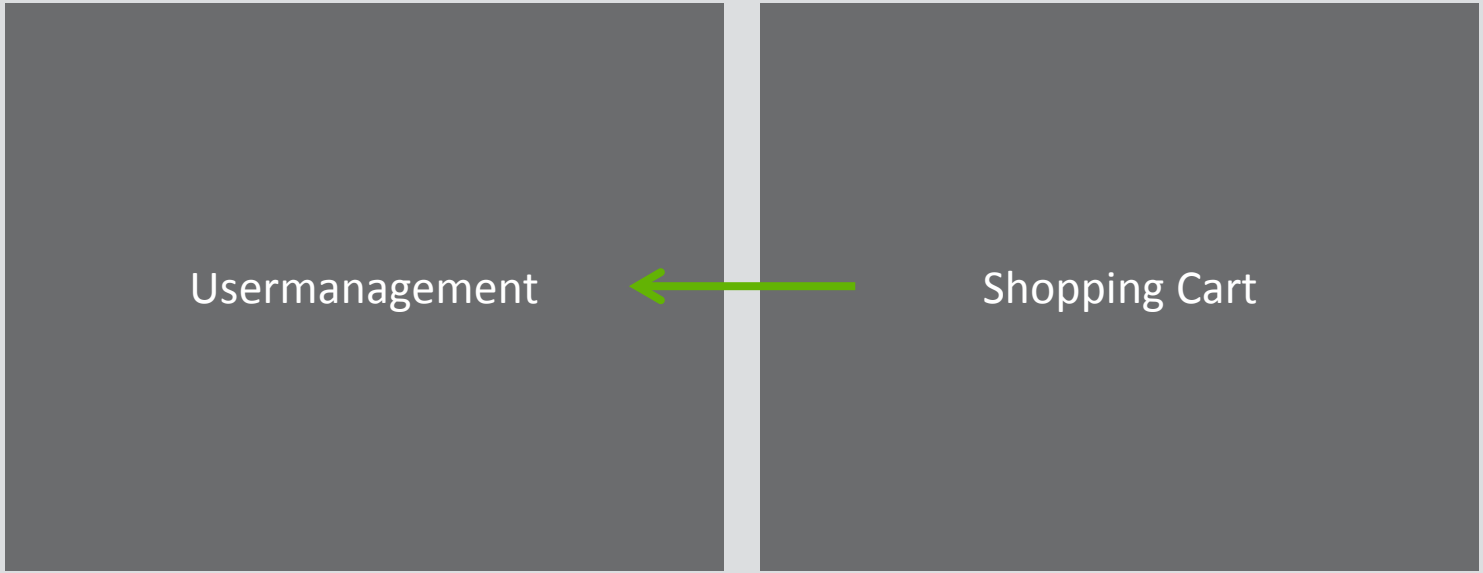
- Sketch of an architecture

- Sketch of an architecture

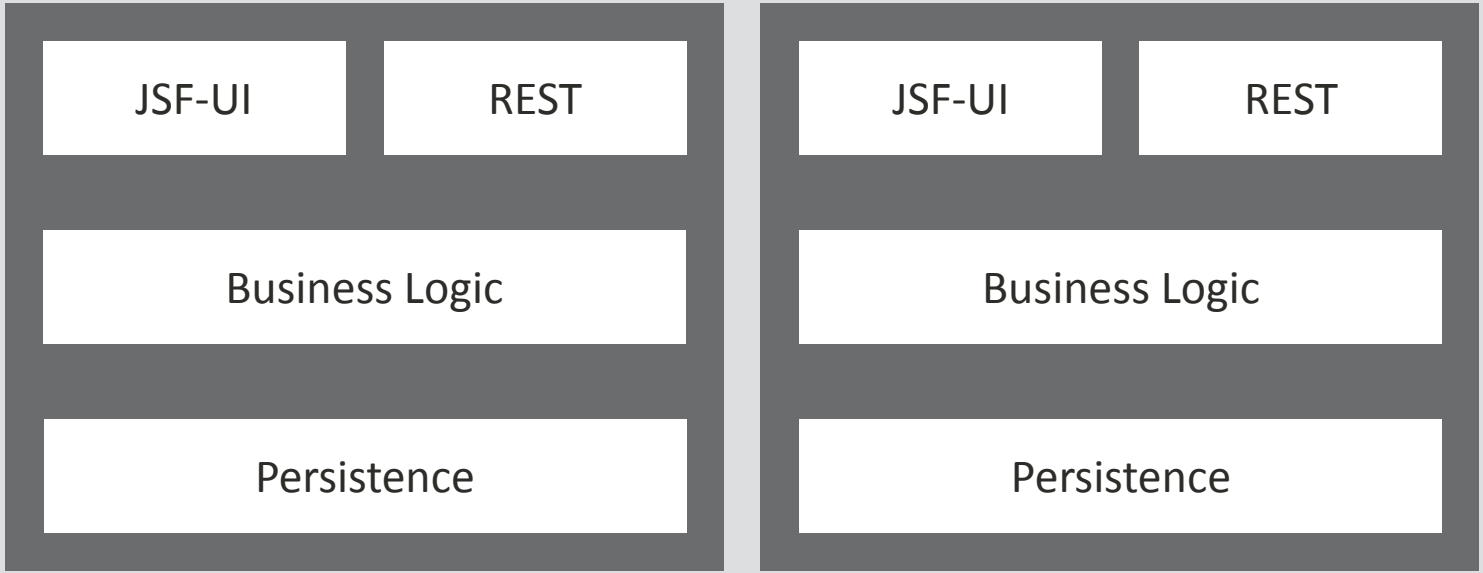
- Sketch of an architecture
  - Business modules



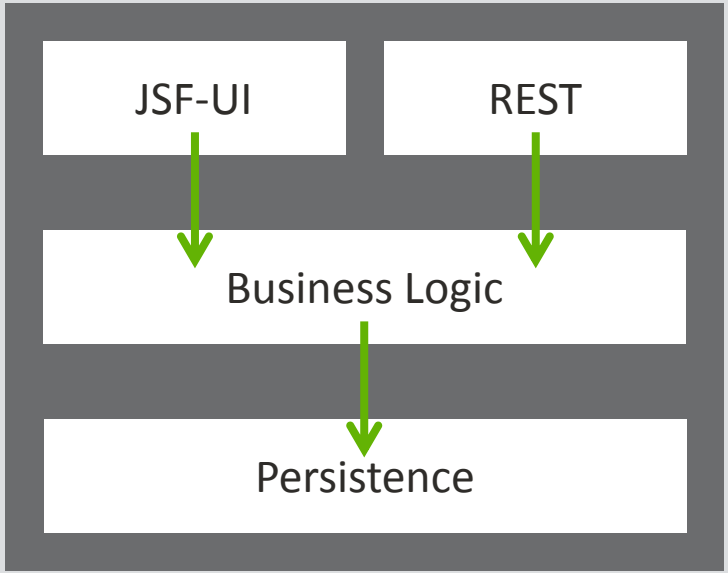
- Sketch of an architecture
  - Allowed dependencies between business modules

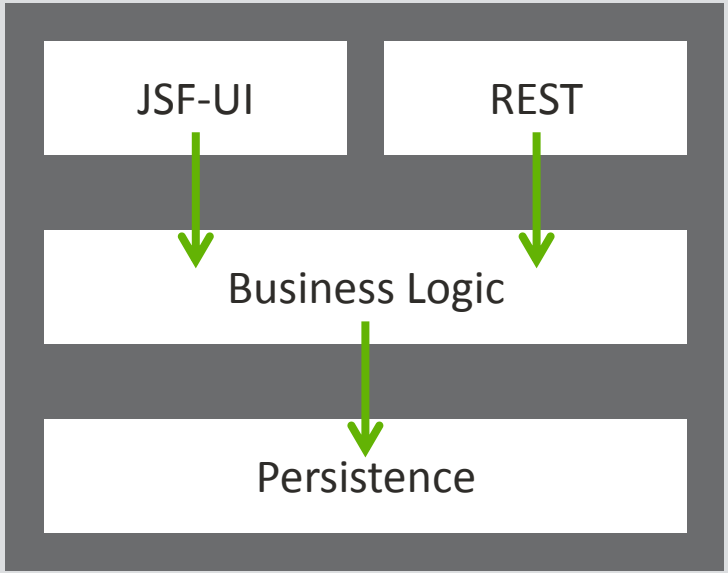


- Sketch of an architecture
  - Technical layering

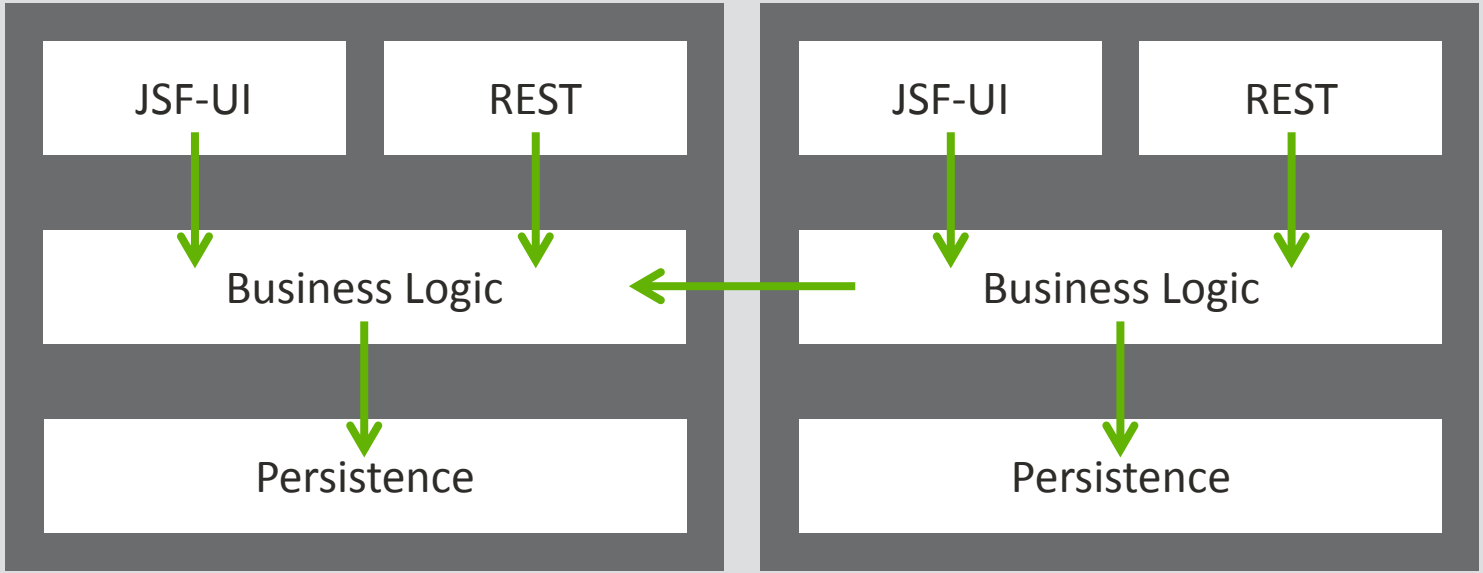


- Sketch of an architecture
  - Allowed dependencies between technical layers

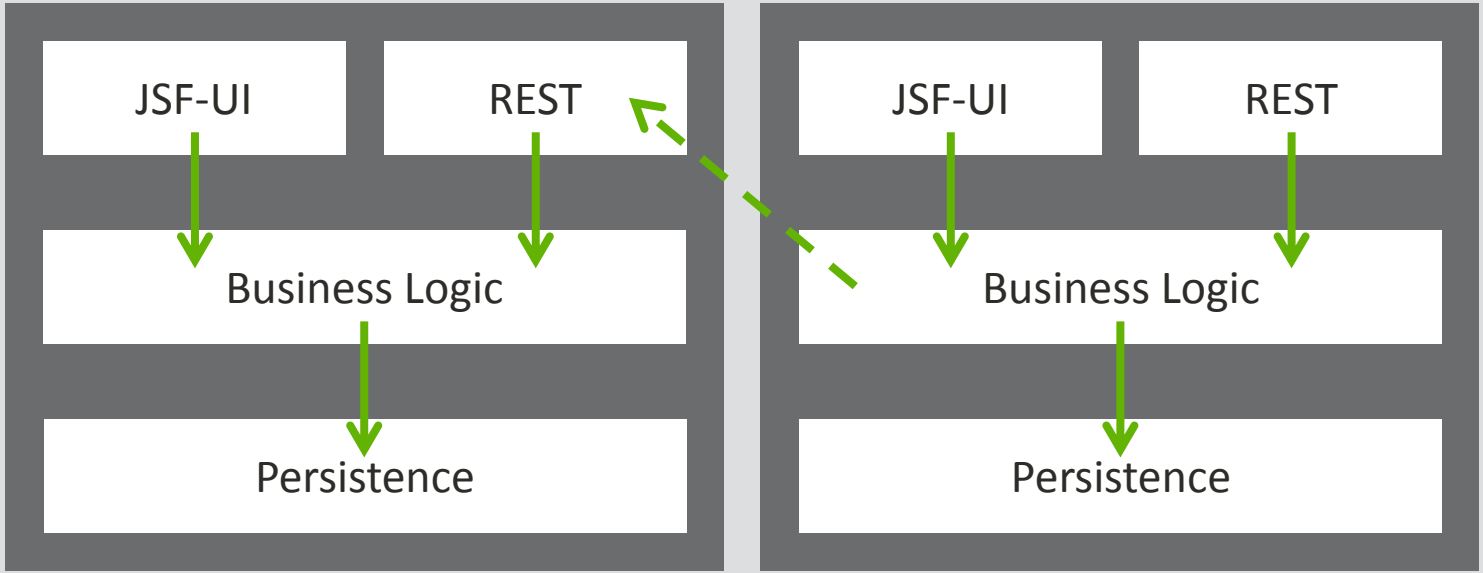




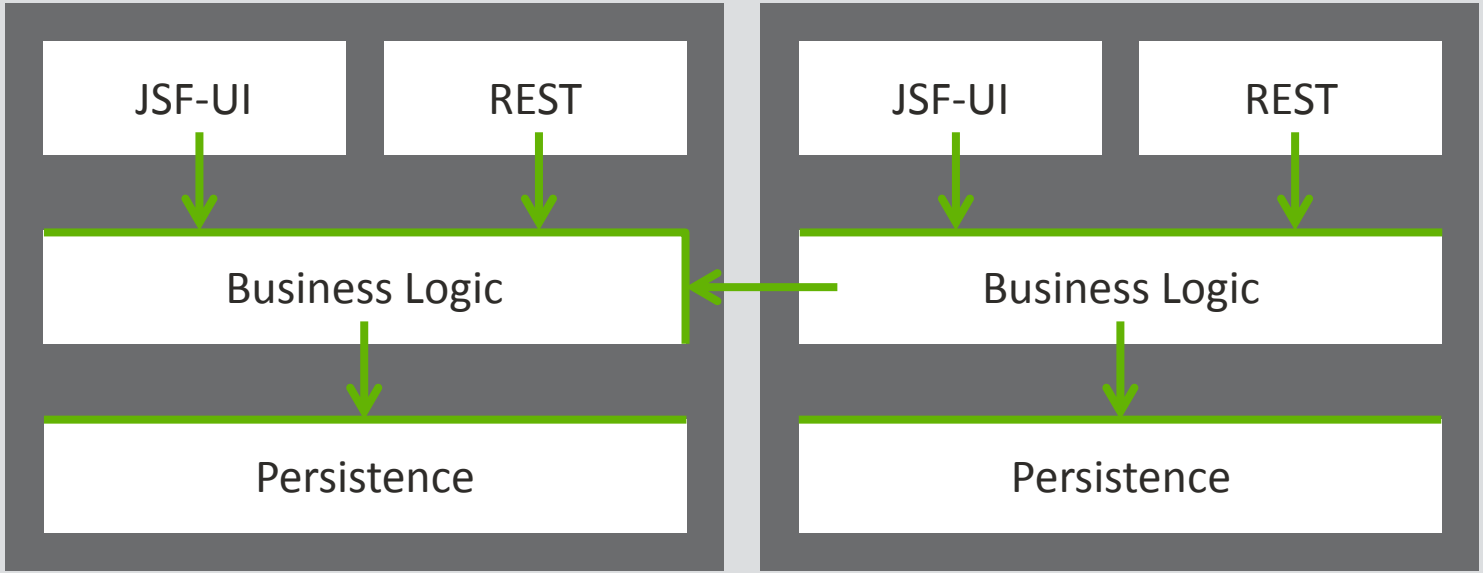
- Sketch of an architecture
  - Allowed dependencies between business modules and technical layers



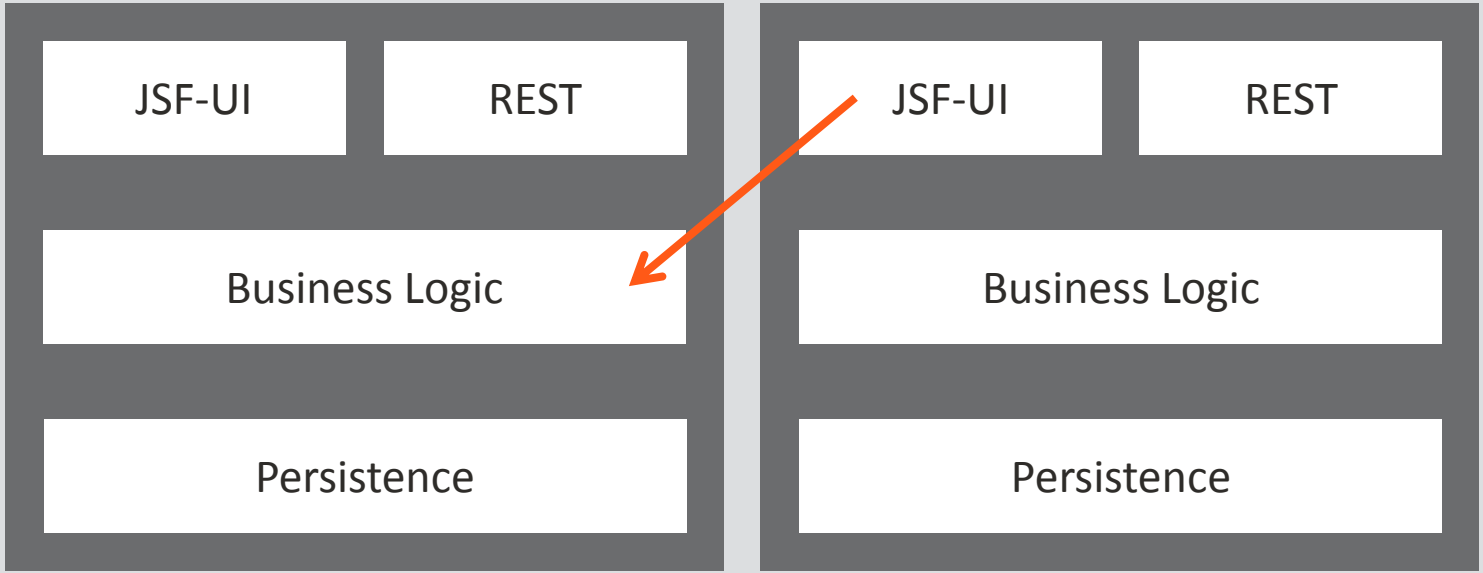
- Sketch of an architecture
  - Allowed dependencies between business modules and technical layers



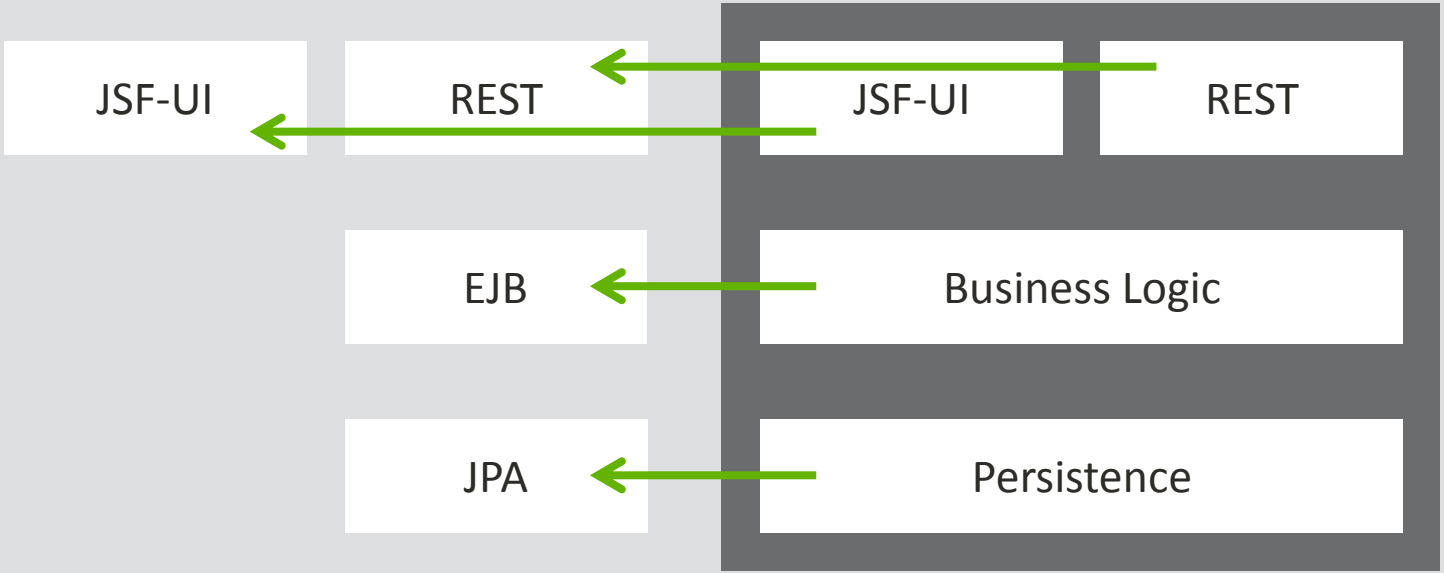
- Sketch of an architecture
  - Abstraction between layers (API vs. Implementation)



- Sketch of an architecture
  - Forbidden dependency



- Sketch of an architecture
  - Allowed external dependencies per layer

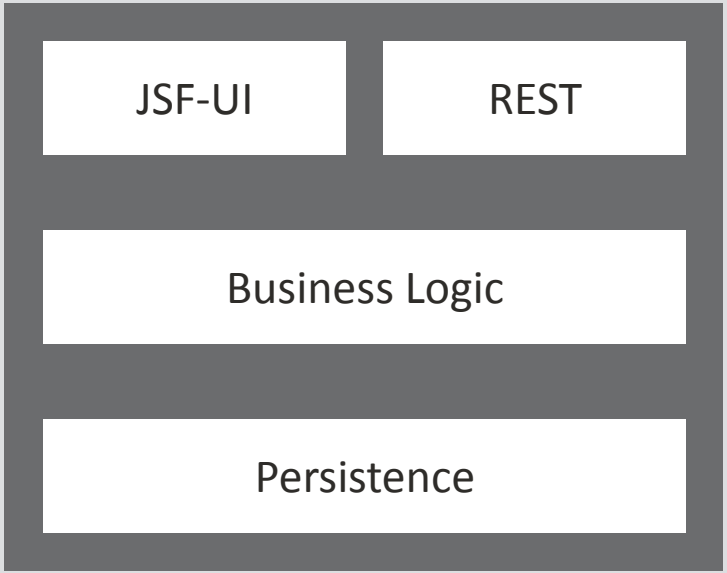


- Translation of architecture rules into a project structure
  - Java language element: Package

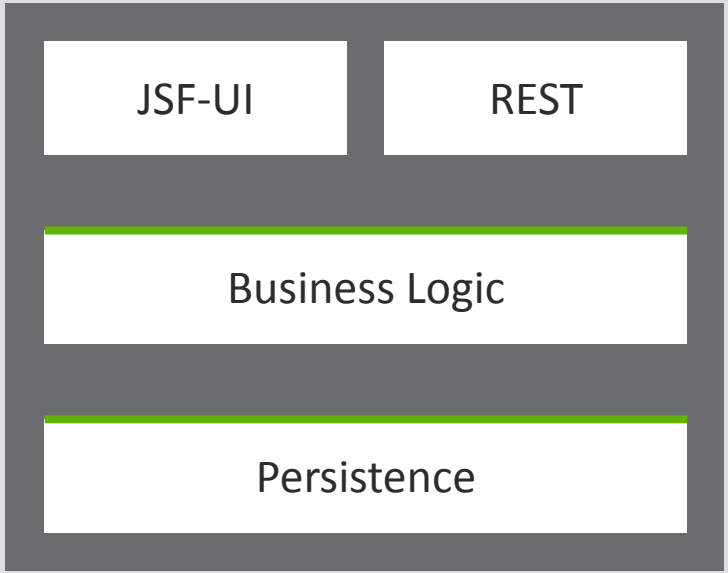
- Translation of architecture rules into a project structure
  - Java language element: Package

  - Definition of business modules on „top level“

- Translation of architecture rules into a project structure
  - Java language element: Package
  - Technical layers
    - …demo.cart.ui
    - …demo.cart.rest
    - …demo.cart.logic
    - …demo.cart.persistence



- Translation of architecture rules into a project structure
  - Java language element: Package
  - Technical layers
    - …demo.cart.ui
    - …demo.cart.rest
    - …demo.cart.logic.api
    - …demo.cart.logic.impl
    - …demo.cart.persistence.api
    - …demo.cart.persistence.impl



- Package Names
  - All packages in a Maven module must be prefixed with ${groupId}.${artifactId}
  - Example:
    groupId=org.jqassistant
    artifactId=demo

    => org.jqassistant.demo

- Class Names
  - Message Driven Beans must have a suffix „MDB“.

- Class location
  - JPA entities must be located in „model“ packages.

- Test design
  - Every test method must contain at least one assertion.
  - Each assertion must provide a human readable message.

- Abstraction
  - Remote APIs must be interfaces declaring only primitives or immutables as parameter or return types.
  - OSGi-Bundles must only export dedicated API packages.

Source Code

Java, Properties,
XML, YML

„Code Quality“

Source Code

Java, Properties,
XML, YML

People

Architect, Developer,
Test

Tools

Compiler, Static Code
Analysis, CI

Documentation

UML, Wiki,
readme.txt

Source Code – Abstraction levels

Architecture	Module, Layer, Dependency
Design	Abstraction, Immutable, Factory
Java	Package, Class, Field, Method, Annotation
File System	Folder, File

Source Code – Automated validation

Architecture	Module, Layer, Dependency
Design	Abstraction, Immutable, Factory
Java	Package, Class, Field, Method, Annotation
File System	Folder, File

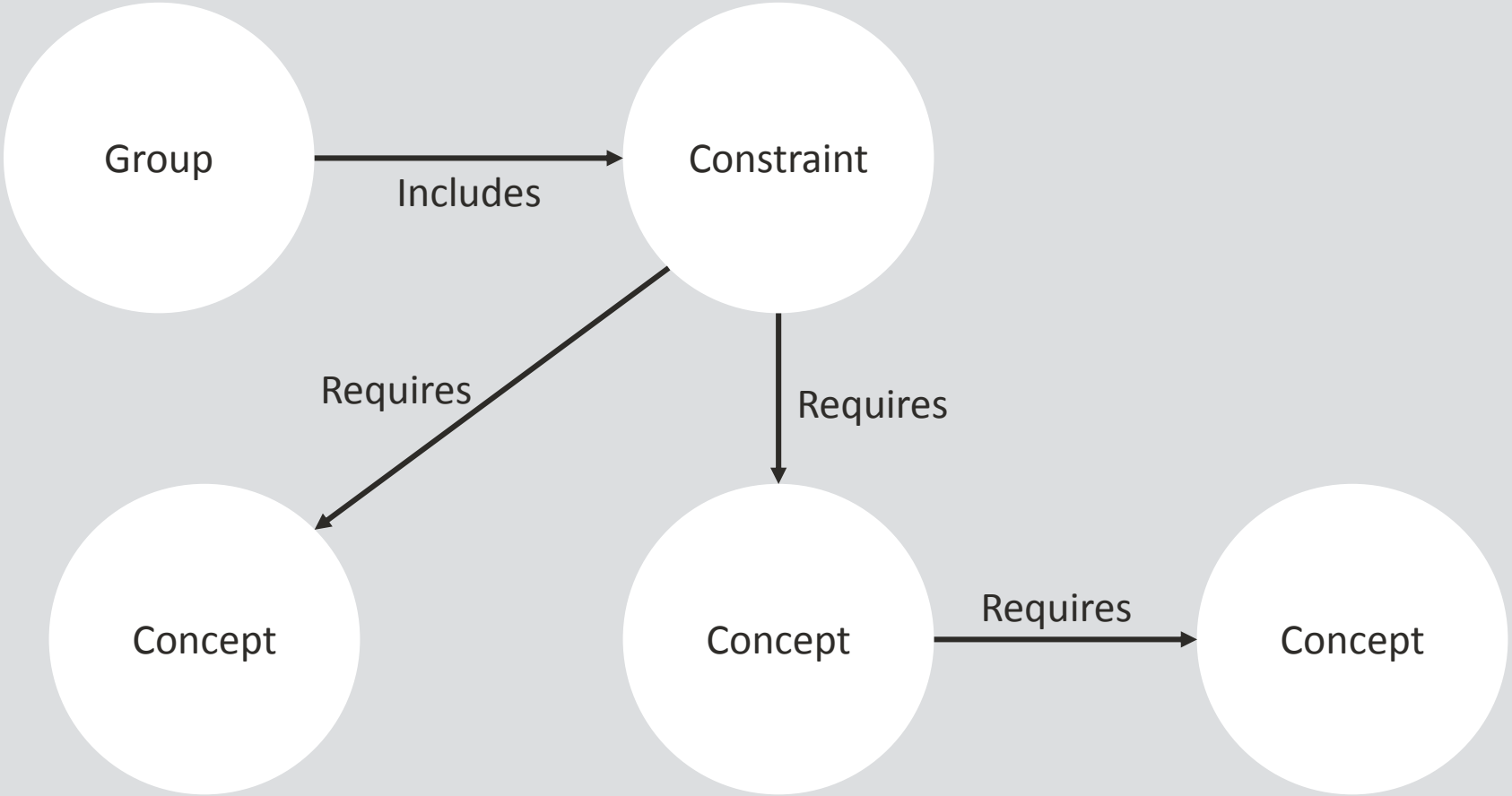
- Java elements represent higher level concepts
  - Package ⇔ Module
  - Annotated Class ⇔ Entity
- Constraints apply to these concepts
  - JPA entities must be located in „model“ packages

- Concepts and Constraints
  - How to make them visible?
  - Is there a language to describe them?

Software Analysis Using jQAssistant And Neo4j

Verifying Rules With The Graph Model

- Analyze
  - Execution of rules
    - Defined in AsciiDoc or XML documents
  - Concepts
    - Enrich data model
  - Constraints
    - Detect violations
  - Group
    - Allow different execution profiles


Group
Includes
Constraint
Requires
Requires
Concept
Concept
Requires
Concept

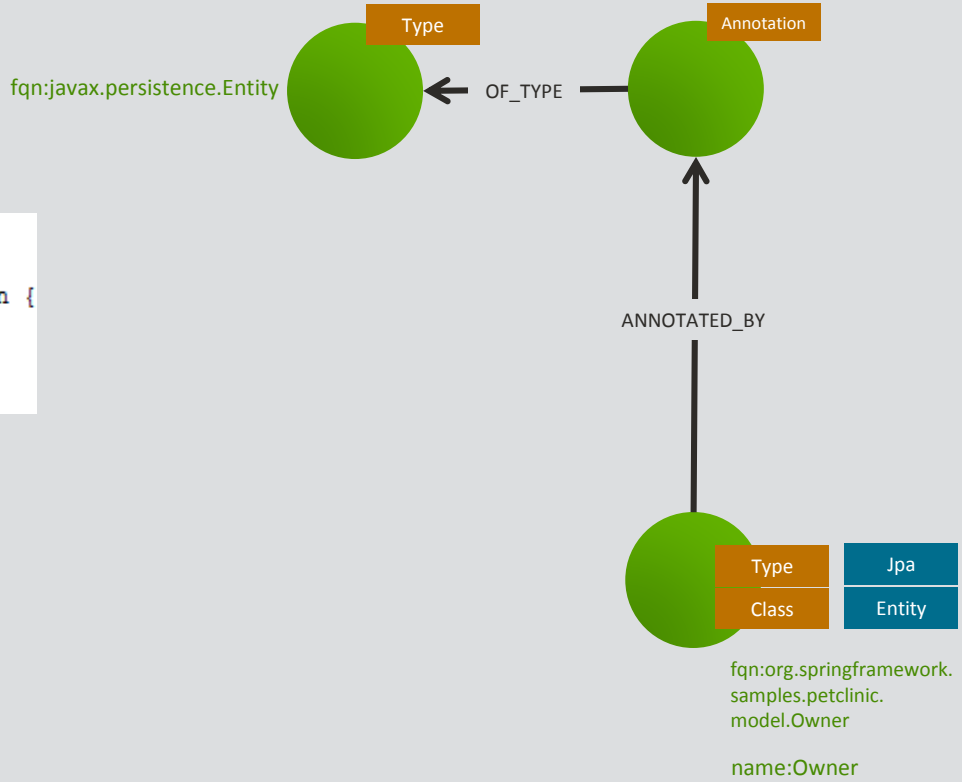
Concept

```
== JPA Entities

[[jpa2:Entity]]
.Labels all types annotated with @javax.persistence.Entity with
Jpa and Entity.
[source,cypher,role=concept]
----
MATCH
  (t:Type)-[:ANNOTATED_BY]->()-[:OF_TYPE]->(a:Type)
WHERE a.fqn ="javax.persistence.Entity"
  SET t:Jpa:Entity
RETURN t AS Entity
----
```

Concept

```
@Entity
@Table(name = "owners")
public class Owner extends Person {
    @Column(name = "address")
    @NotEmpty
    private String address;
```



- Constraint

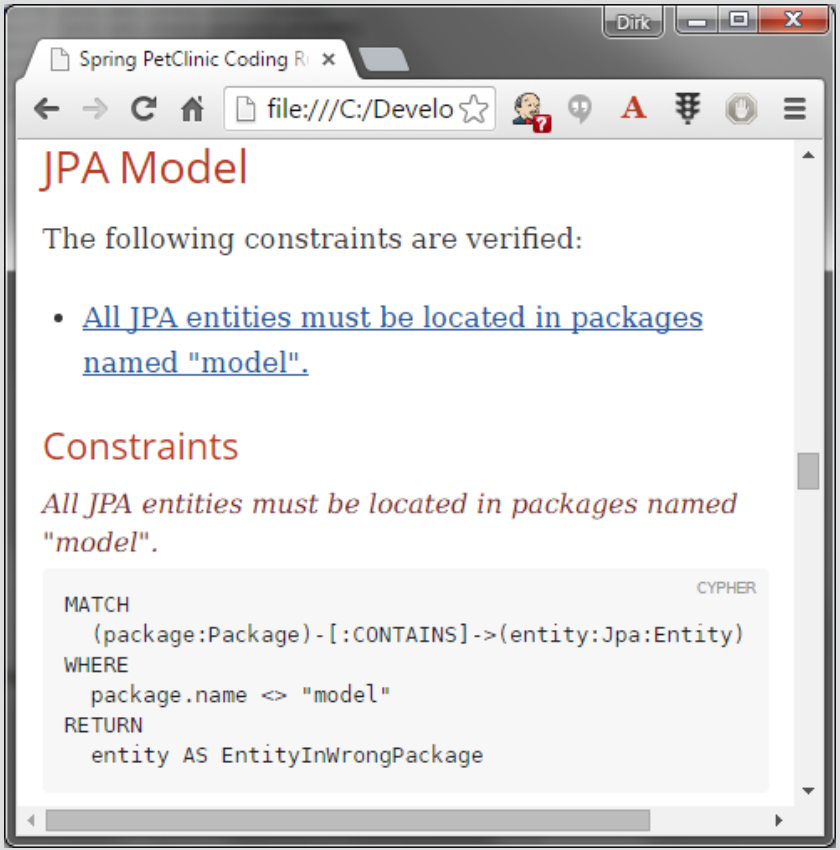
```
[[model:JpaEntityInModelPackage]]
.All JPA entities must be located in packages named "model".
[source,cypher,role=constraint,requiresConcepts="jpa2:Entity"]
----
MATCH
  (package:Package)-[:CONTAINS]->(entity:Jpa:Entity)
WHERE
  package.name <> "model"
RETURN
  entity AS EntityInWrongPackage
----
```

- Group

```
[[default]]
[role=group,includesConstraints="model:JpaEntityInModelPackage"]
== Naming Rules

The following naming rules apply:
- <<model:JpaEntityInModelPackage>>
```

- AsciiDoc
  - Wiki syntax
  - Rendering to
    - DocBook
    - HTML
    - PDF
- Executable specification
  - Coding rules
  - Design & Architecture



Software Analysis Using jQAssistant And Neo4j

Demo #2

http://github.com/buschmais/spring-petclinic

Software Analysis Using jQAssistant And Neo4j

Wrap Up

Benefits Of Using A Graph Database

- Easy modeling
  - natural language
  - low technical „noise“
- Flexible and extensible
  - Schema defined per node by its labels
  - Different aspects of software in the same database
  - Enables plugin architecture
  - Enrichment by queries (e.g. abstractions)
- Expressive queries
  - (Open-)Cypher

Thank You! – Questions?

Mail: info@jqassistant.org

Web: jqassistant.org

Twitter: @jqassistant