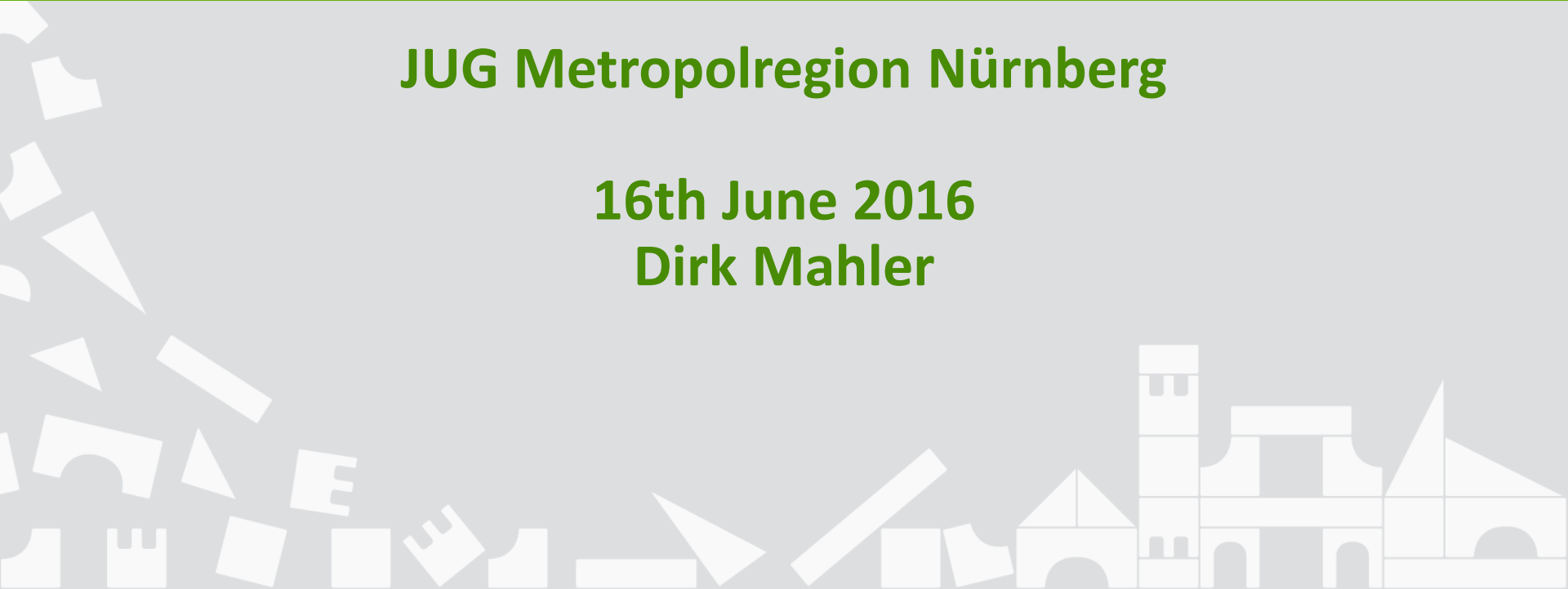


# **jQAAssistant**

## **Verify Your Design And Architecture**

**JUG Metropolregion Nürnberg**

**16th June 2016**  
**Dirk Mahler**



# AGENDA

- jqAssistant
- Software As A Graph
- Structures, Rules And Code
- Verifying Rules
- Wrap Up
- Q&A

**jqAssistant - Verify Your Design And Architecture**

**jqAssistant**

# jQAAssistant

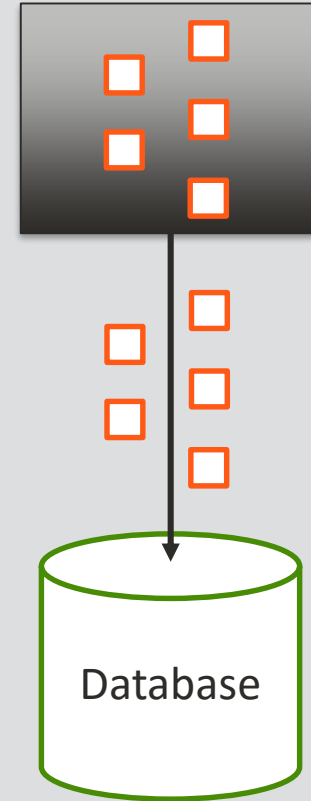
<http://jQAAssistant.org>

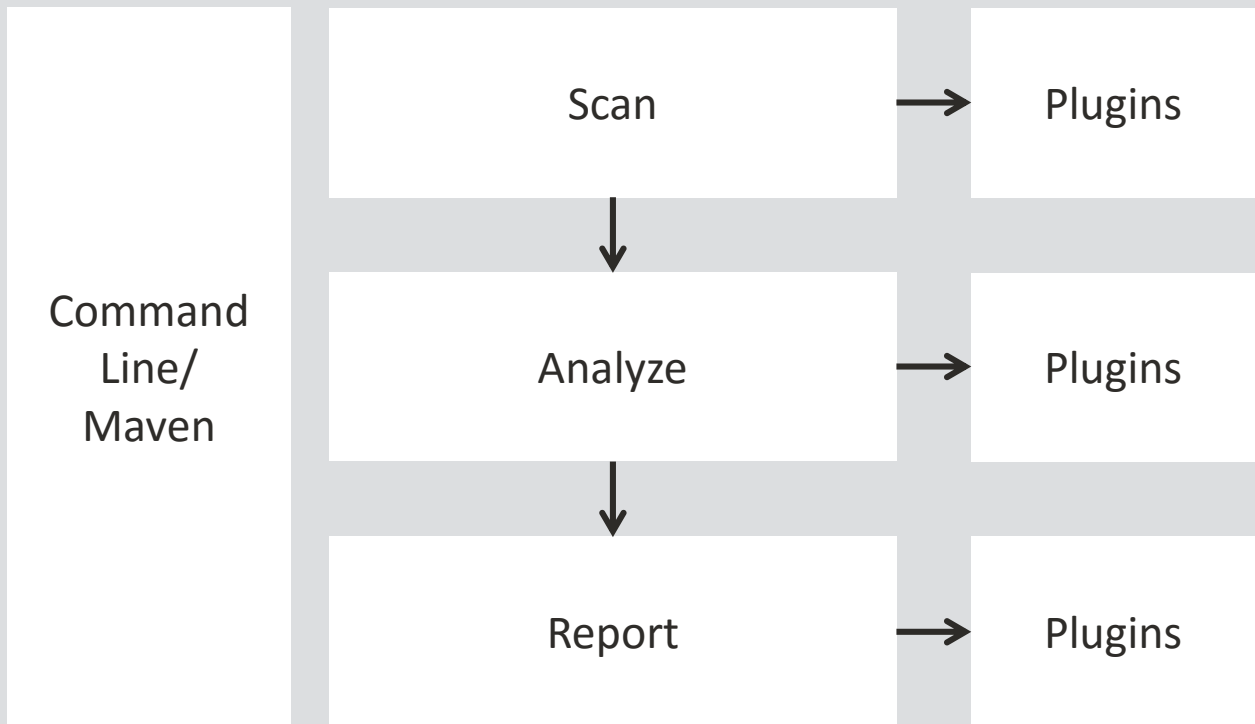
- Open Source: GPLv3
- Current release: 1.1.3
  - initiated: 03/2013
  - first stable release: 04/2015
- Neo4j Community Edition
  - <http://neo4j.org>
  - embedded, no installation necessary



## ■ The Idea

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





## ■ Available scanner plugins

ZIP

JAR, WAR, EAR

GZ

Liquibase

\*.class

\*.properties

\*.yaml

CheckStyle

FindBugs

MANIFEST.MF

/META-INF/services/\*

JaCoCo

Git

web.xml

application.xml

beans.xml

pom.xml

surefire-reports.xml

\*.xsd

M2 Repository

RDBMS Schema

## ■ Plugin API is public

## ■ Getting Started – Command Line

- `unzip jqassistant.distribution-1.x.x.zip`
- `cd jqassistant.distribution-1.x.x/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.x.x</version>
    </plugin>
  </plugins>
</build>
```

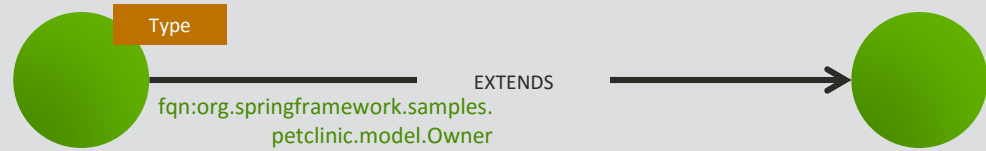
- mvn install
- mvn jqassistant:scan
- mvn jqassistant:server  
→ <http://localhost:7474>

**jqAssistant - Verify Your Design And Architecture**

# **Software As A Graph**

- All we need is...

- Nodes
- Labels
- Properties
- Relationships

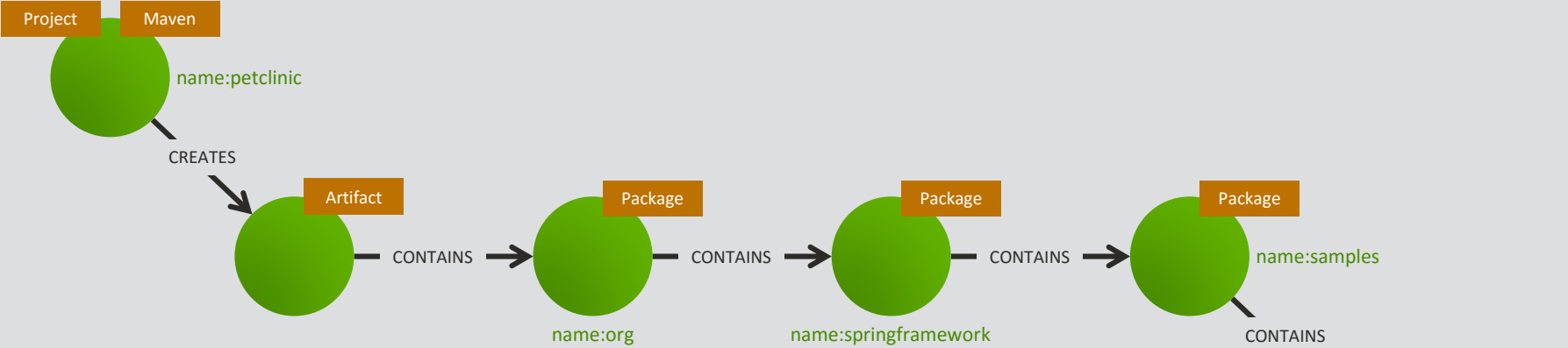


- Modeling is just...

- Taking a pen
- Drawing the structures on a whiteboard (i.e. the database)

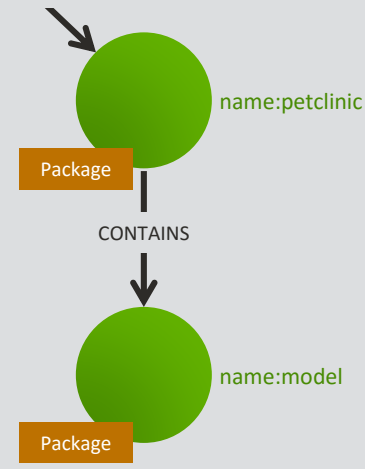
- We don't need...

- Foreign keys
- Tables and schemas
- Deep knowledge in graph theory



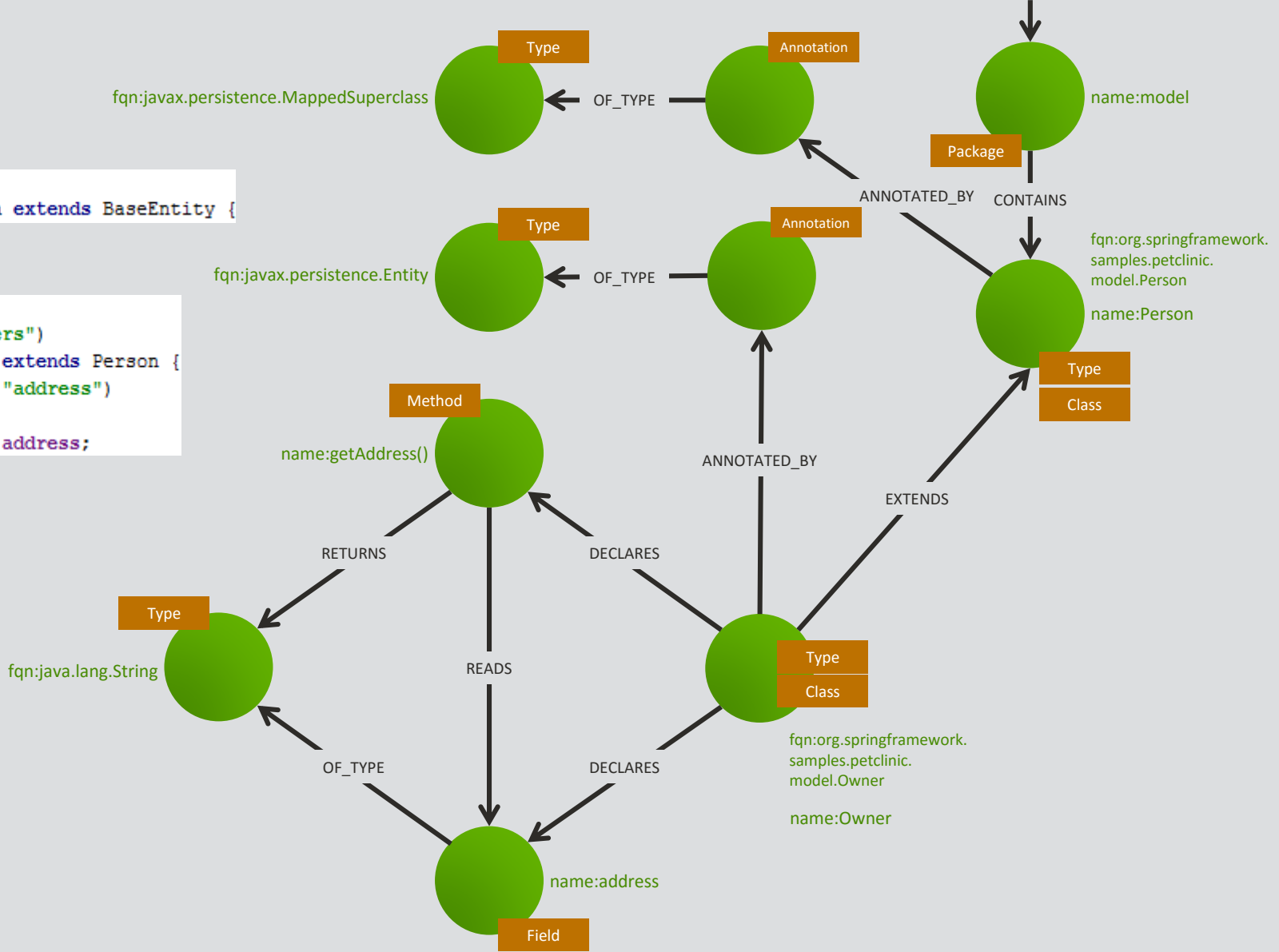
**spring-petclinic** (C:\Development\projects\spring-petclinic)

- ▶ .idea
- ▶ jqassistant
- ▼ src
  - ▼ main
    - ▼ java
      - ▼ org.springframework.samples.petclinic
        - ▼ model
          - BaseEntity
          - NamedEntity
          - Owner
          - package-info.java
          - Person
          - Pet
          - PetType
          - Specialty
          - Vet
          - Vets
          - Visit

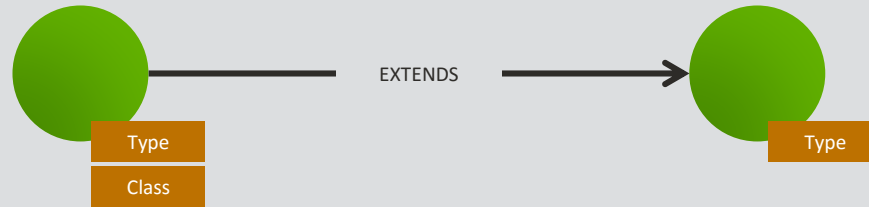


```
@MappedSuperclass
public class Person extends BaseEntity {
```

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

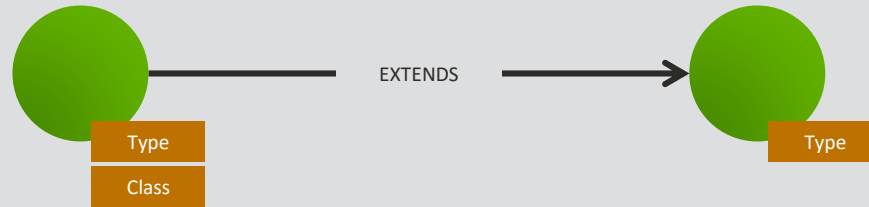


- Explore an application using queries
  - Which class extends from another class?



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

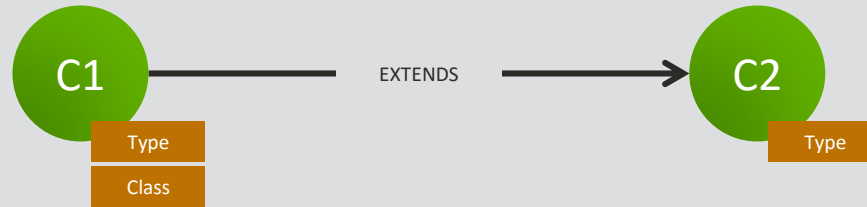
- Explore an application using queries
  - Which class extends from another class?



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

( ) - [ ] -> ( )

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

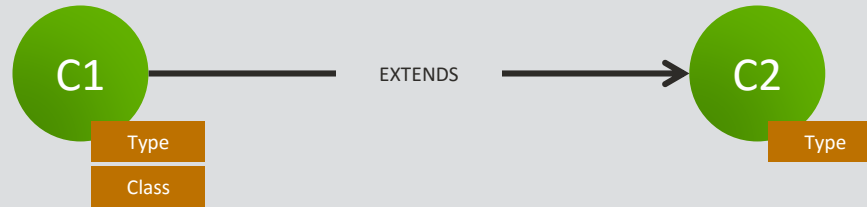


- Let's convert this to ASCII art...
  - () as nodes
  - -[]-> as directed relationships

(c1) - [] -> (c2)



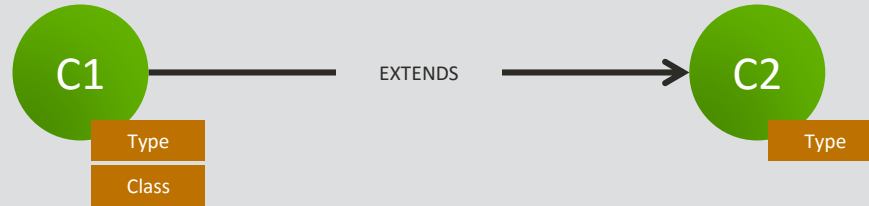
- Explore an application using queries
  - Which class extends from another class?



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

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

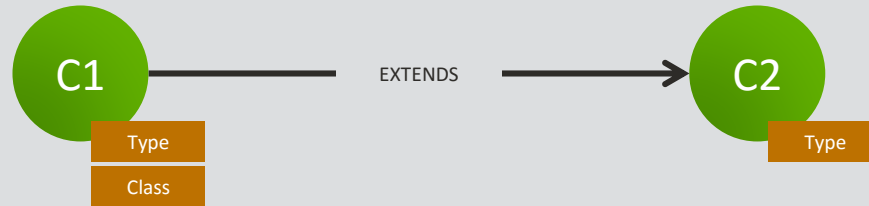
- Explore an application using queries
  - Which class extends from another class?



- 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`

**jqAssistant - Verify Your Design And Architecture**

# **Demo #1**

**Analysis of software systems using jQAssistant and Neo4j**

# **Structures, Rules And Code**

- Sketch of an architecture

- Sketch of an architecture

My Big Fat Demo Application

- Sketch of an architecture
  - Business modules



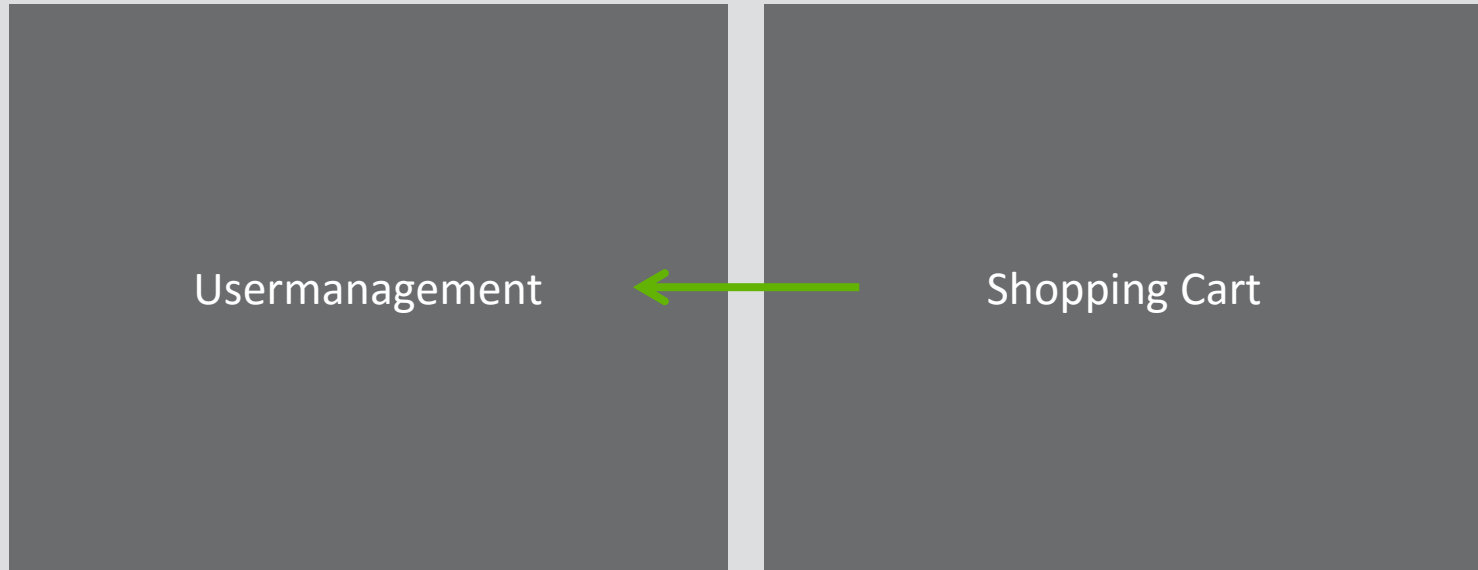
Usermanagement



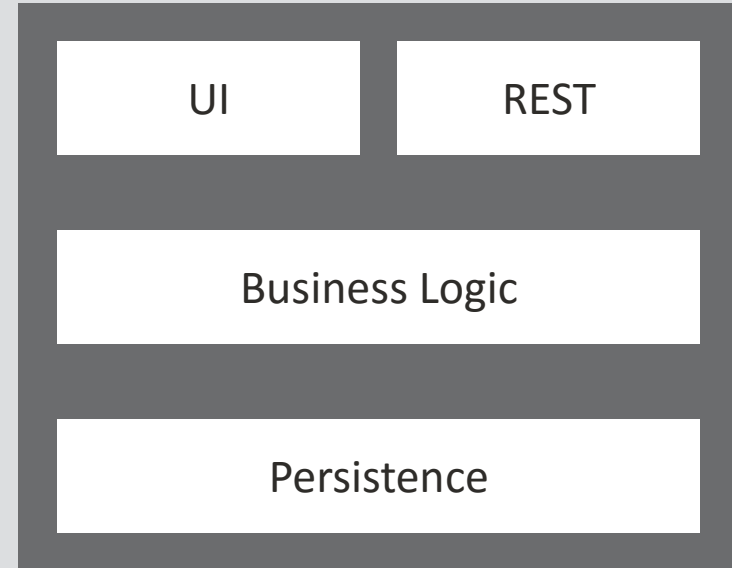
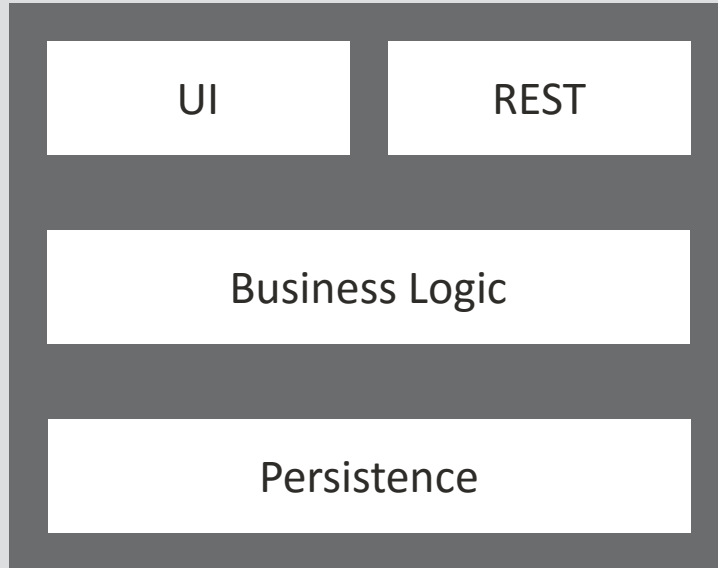
Shopping Cart



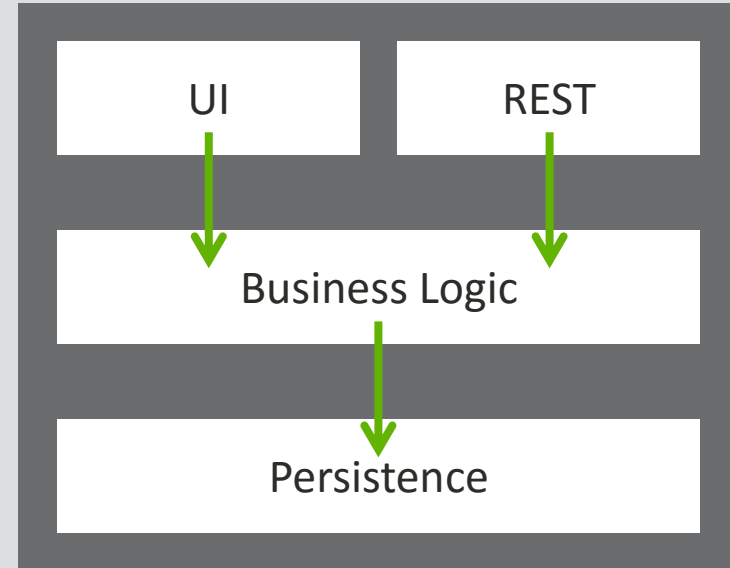
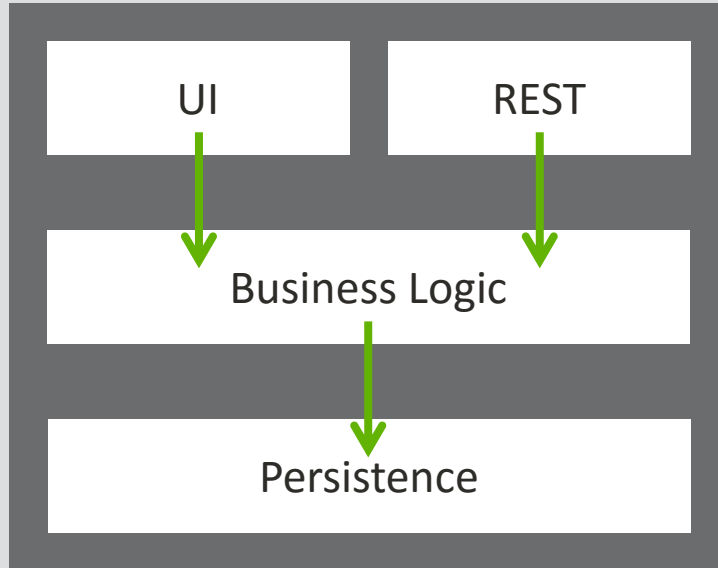
- Sketch of an architecture
  - Allowed dependencies between business modules



- Sketch of an architecture
  - Technical layering

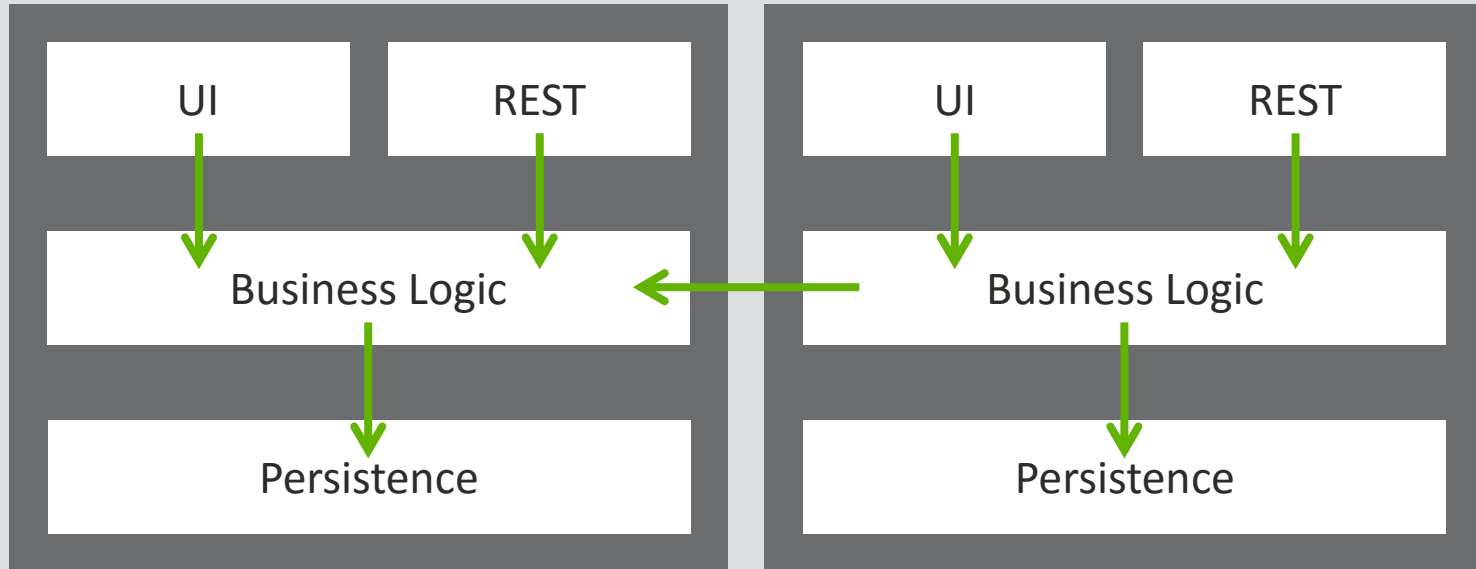


- Sketch of an architecture
  - Allowed dependencies between technical layers



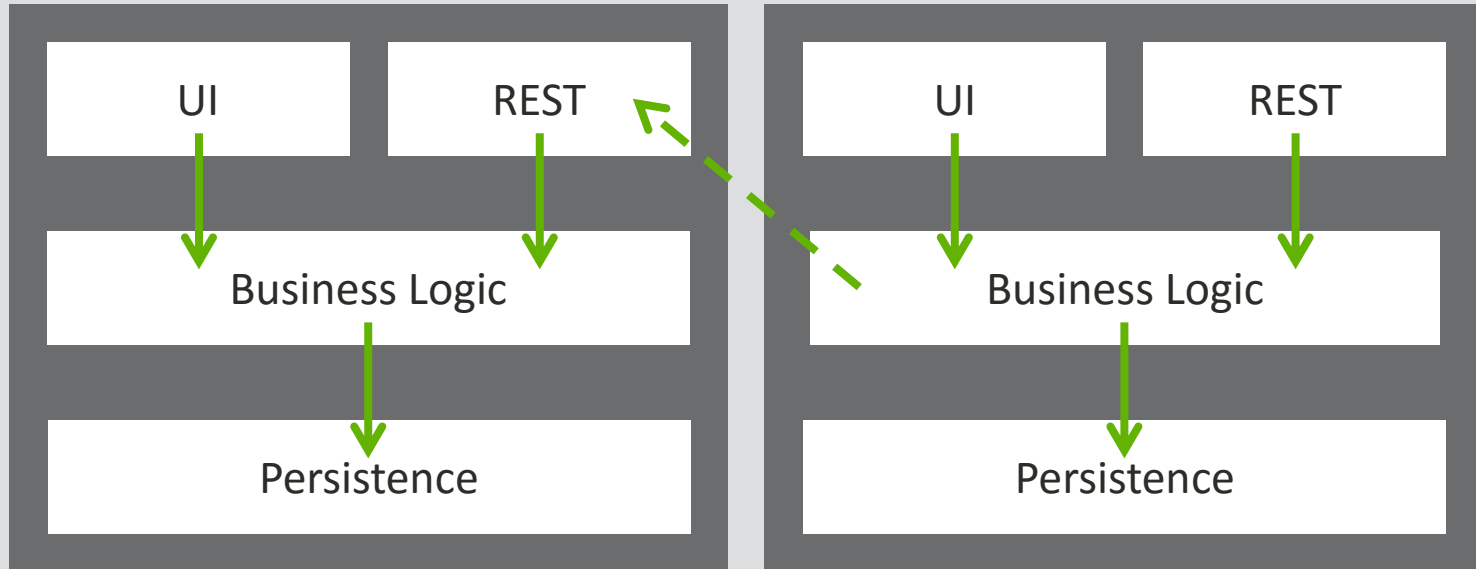
## ■ Sketch of an architecture

- Allowed dependencies between business modules and technical layers

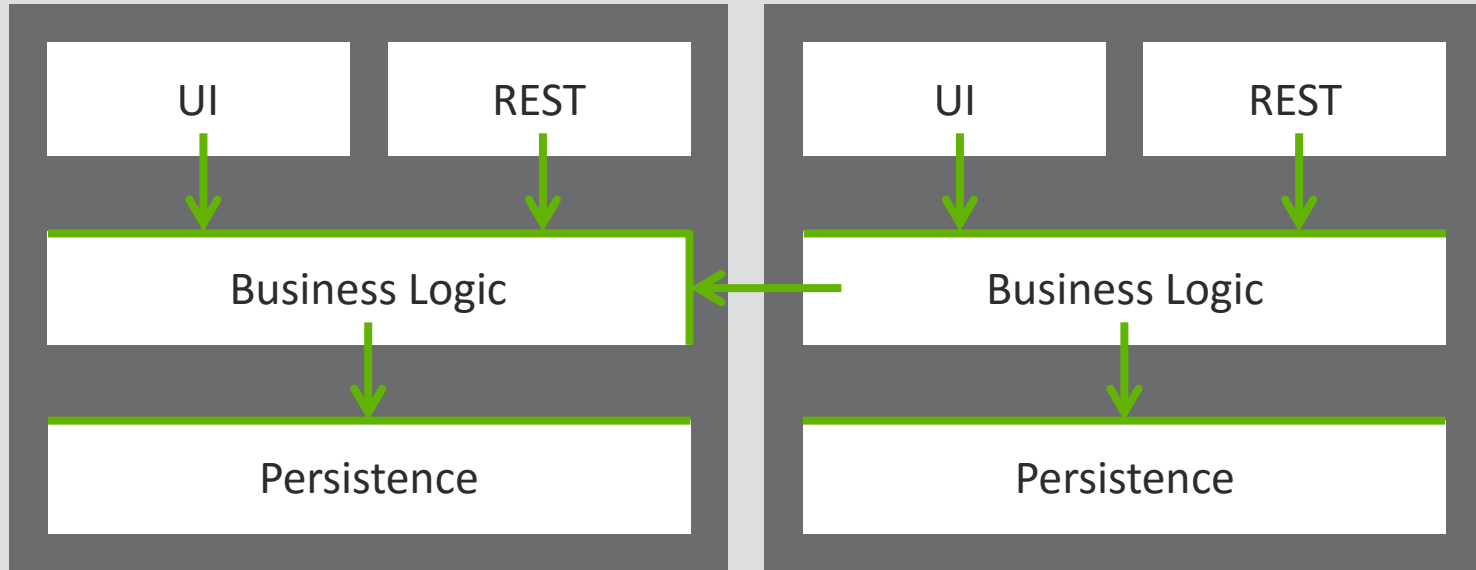


## ■ Sketch of an architecture

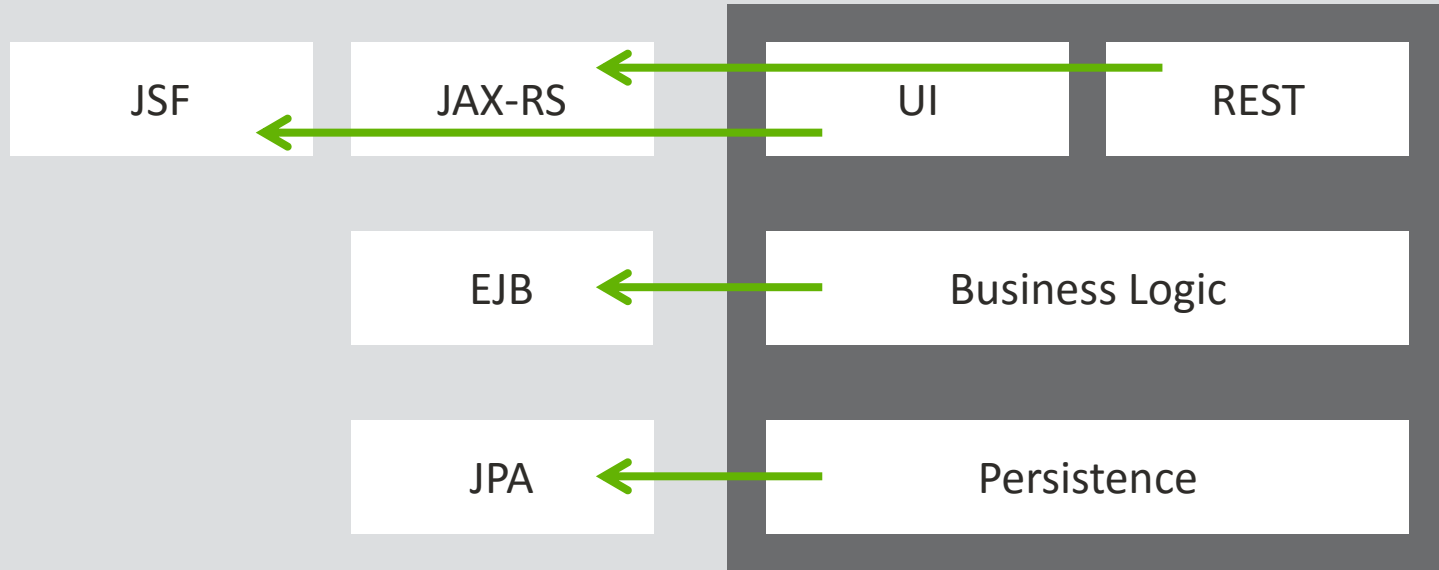
- Allowed dependencies between business modules and technical layers



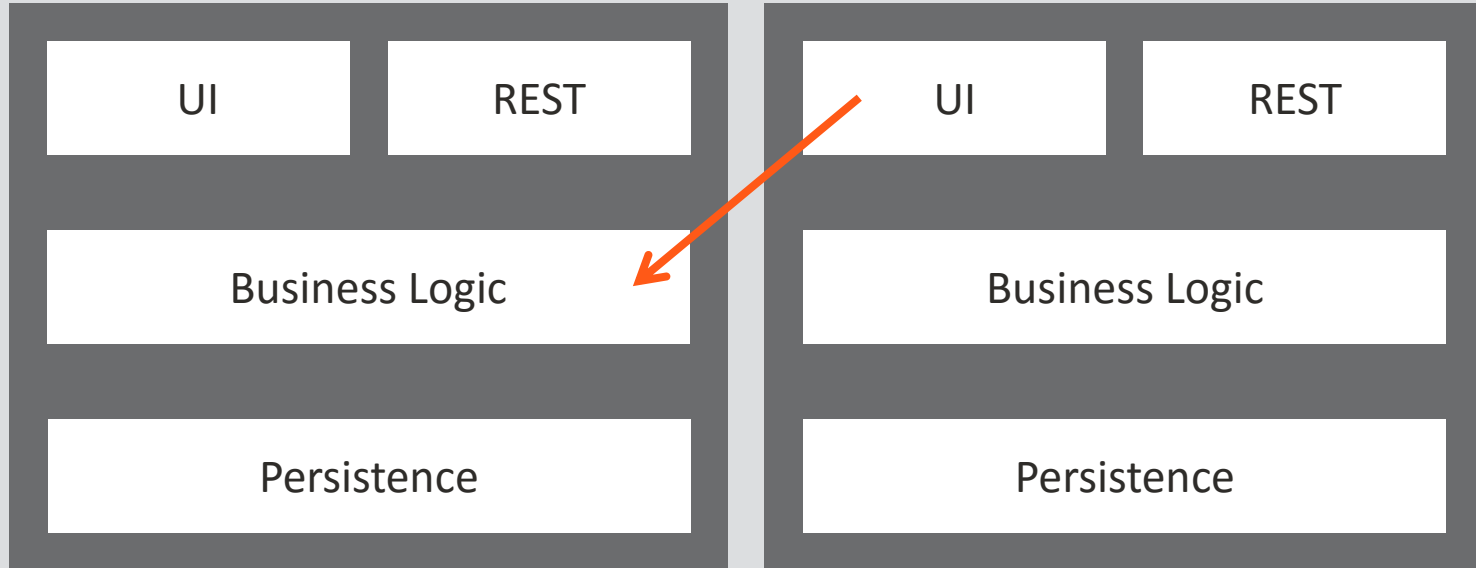
- Sketch of an architecture
  - Abstraction between layers (API vs. Implementation)



- Sketch of an architecture
  - Allowed external dependencies per layer



- Sketch of an architecture
  - Forbidden dependency





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

org.jqassistant.demo

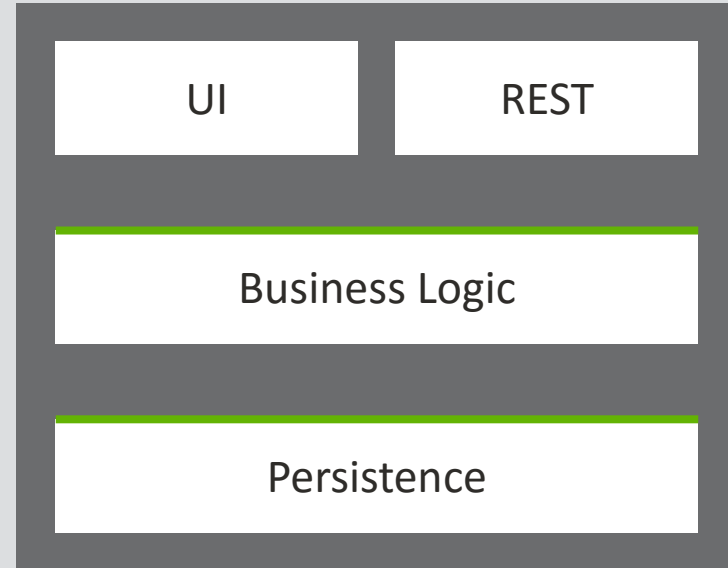
- Translation of architecture rules into a project structure
  - Java language element: Package

Usermanagement  
„org.jqassistant.demo.user“

Shopping Cart  
„org.jqassistant.demo.cart“

- 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

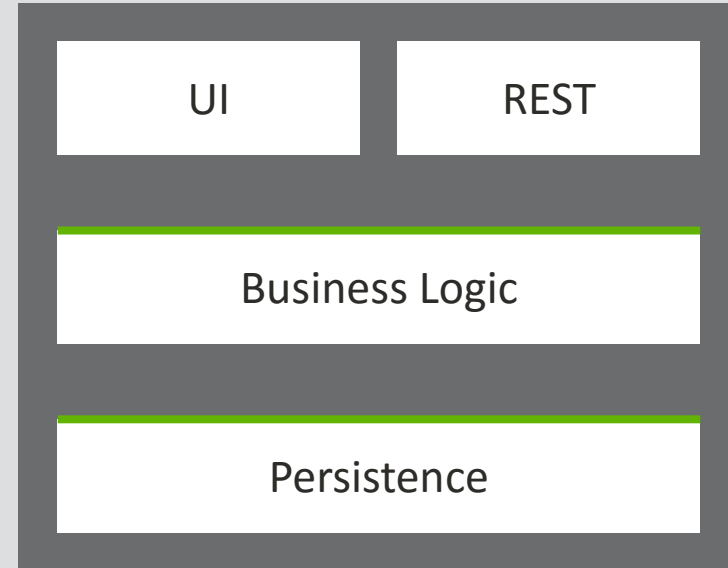


- Translation of architecture rules into a project structure

- 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**



- How to define allowed dependencies?

## ■ 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“.

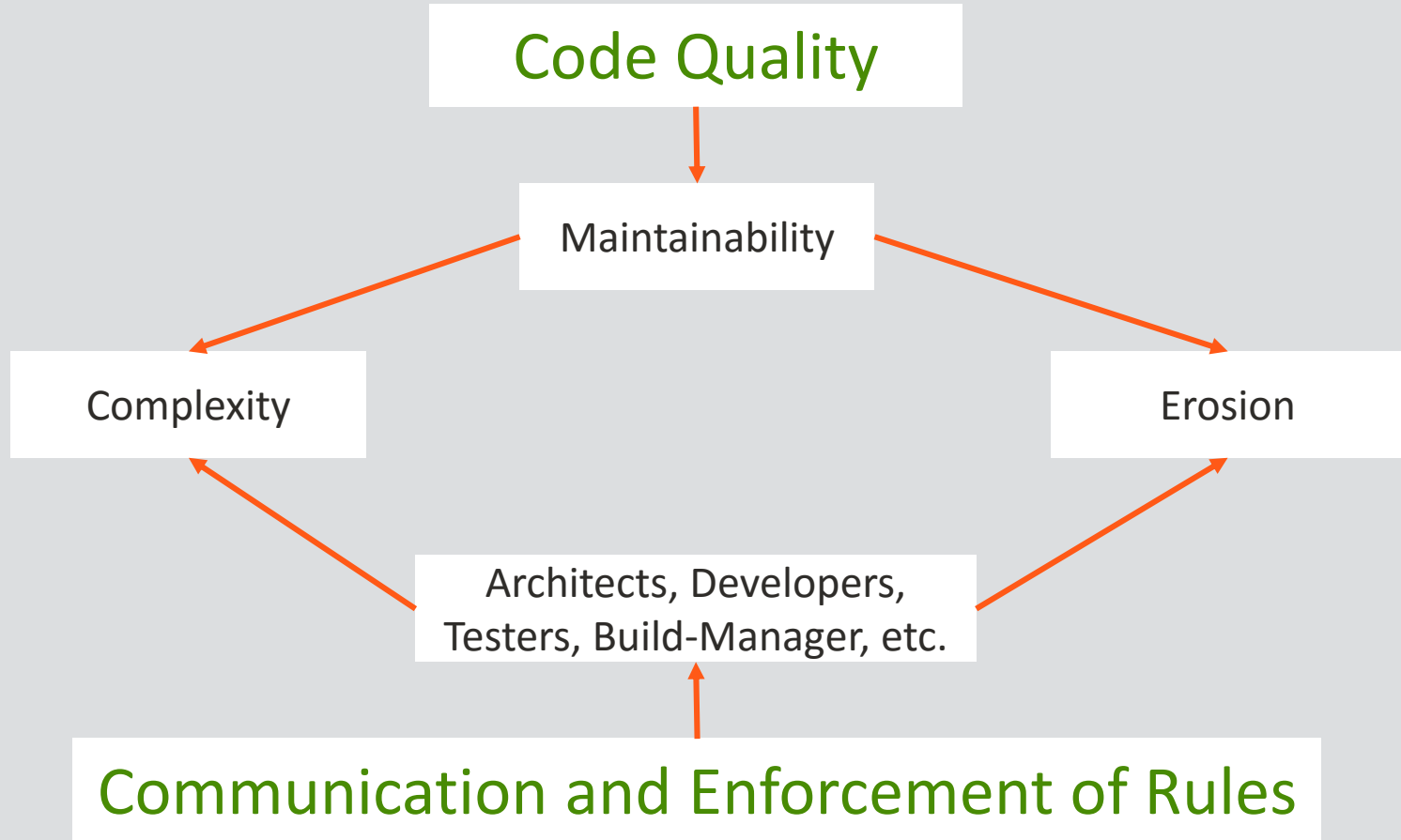
- Class location

- 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.





## People

Meetings, Pair-  
Programming, Reviews

## Documentation

UML, Wiki,  
readme.txt

## Tools

Compiler, Static Code  
Analysis, CI

## ■ Abstraction levels and validation supported by tools

Architecture

Module, Layer, Dependency

Design

Abstraction, Immutable, Factory

Java

Package, Class, Field, Method, Annotation

File System

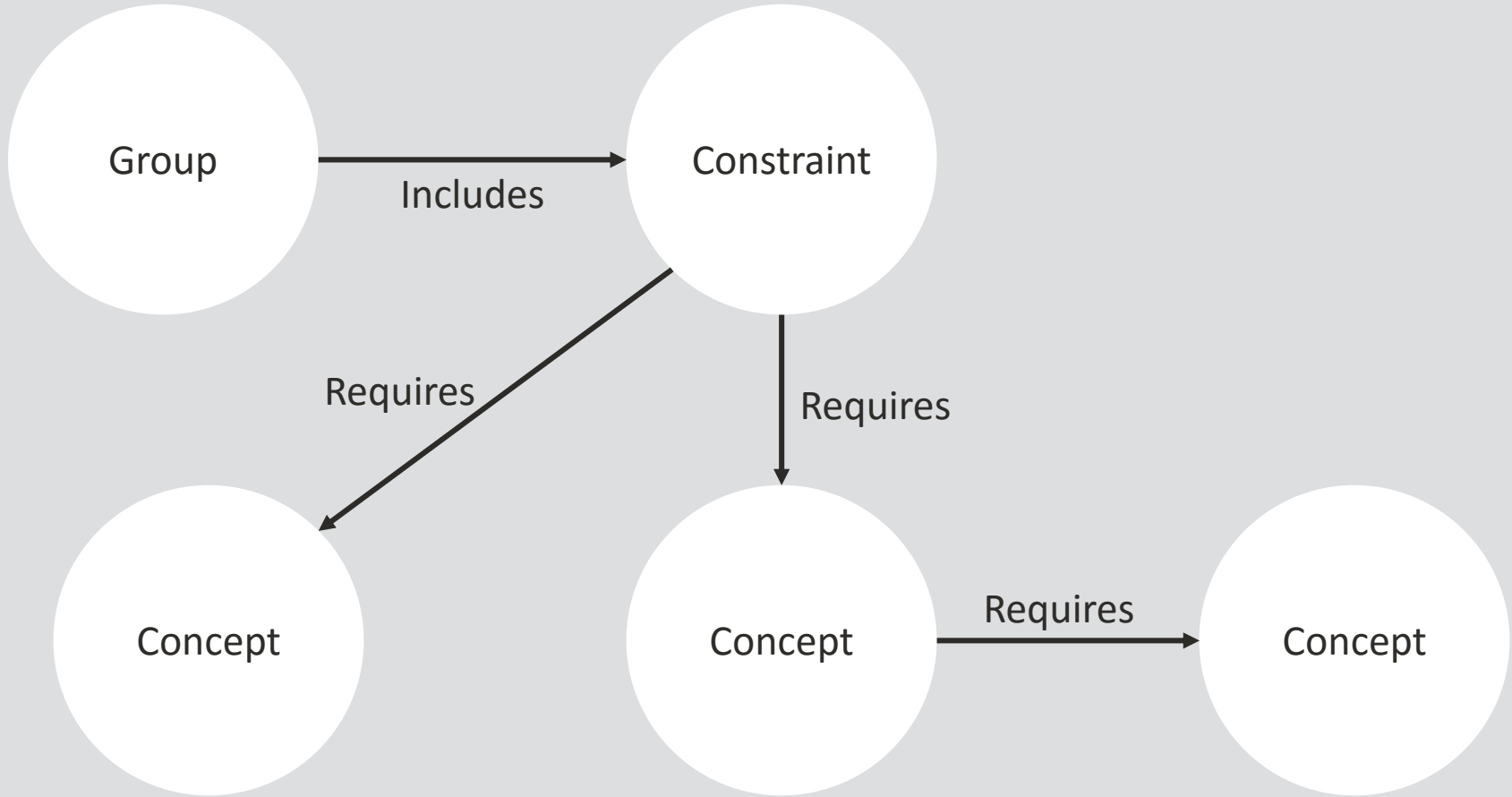
Folder, File

- Code elements represent higher level concepts
  - Maven Project ↔ Module
  - Java Package ↔ Layer
  - Annotated Class ↔ Entity
- Constraints apply to these concepts
  - JPA entities must be located in persistence layers
- Rules are usually project specific

**jqAssistant - Verify Your Design And Architecture**

# Verifying Rules

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



## ■ Concept

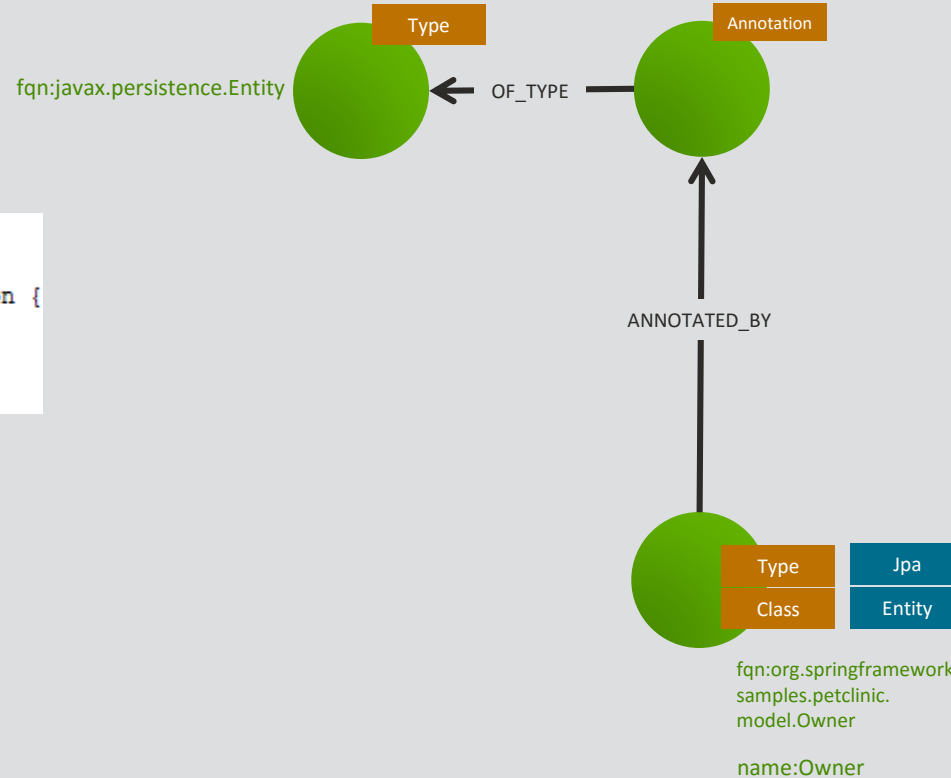
== JPA Entities

```
[[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:

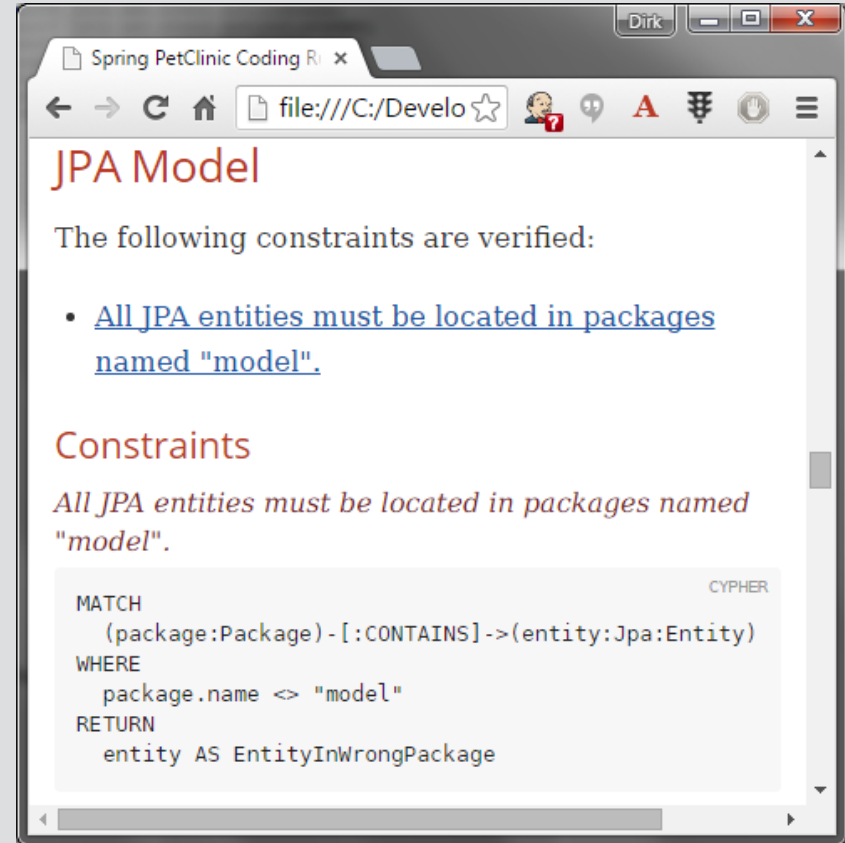
- <<model:JpaEntityInModelPackage>>

## ■ AsciiDoc

- Markup Language
- 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>

**jqAssistant - Verify Your Design And Architecture**

**Wrap Up**

# jQAAssistant

- Holistic and extensible view on code structure
- Flexible exploration of structures using queries
- Rules using concepts and constraints from architecture and design language
- Executable architecture and design specification

# Questions & Answers

Web: [jqassistant.org](http://jqassistant.org)

Twitter: [@jqassistant](https://twitter.com/jqassistant)

Google Group: [jqAssistant](https://groups.google.com/jqassistant)

Stackoverflow Tag: [jqAssistant](https://stackoverflow.com/questions/tagged/jqAssistant)