

Analysis of software systems using jQAAssistant and Neo4j

**Java User Group Hannover
June 2015
Dirk Mahler**

AGENDA

- About Structures, Rules And Code
- Software As A Graph
- Demo#1
- jQAssistant
- Demo#2

Analysis of software systems using jQAssistant and Neo4j

About Structures, Rules And Code

- Sketch of an architecture

- Sketch of an architecture

My Big Fat Demo Application

- Sketch of an architecture
 - Business modules

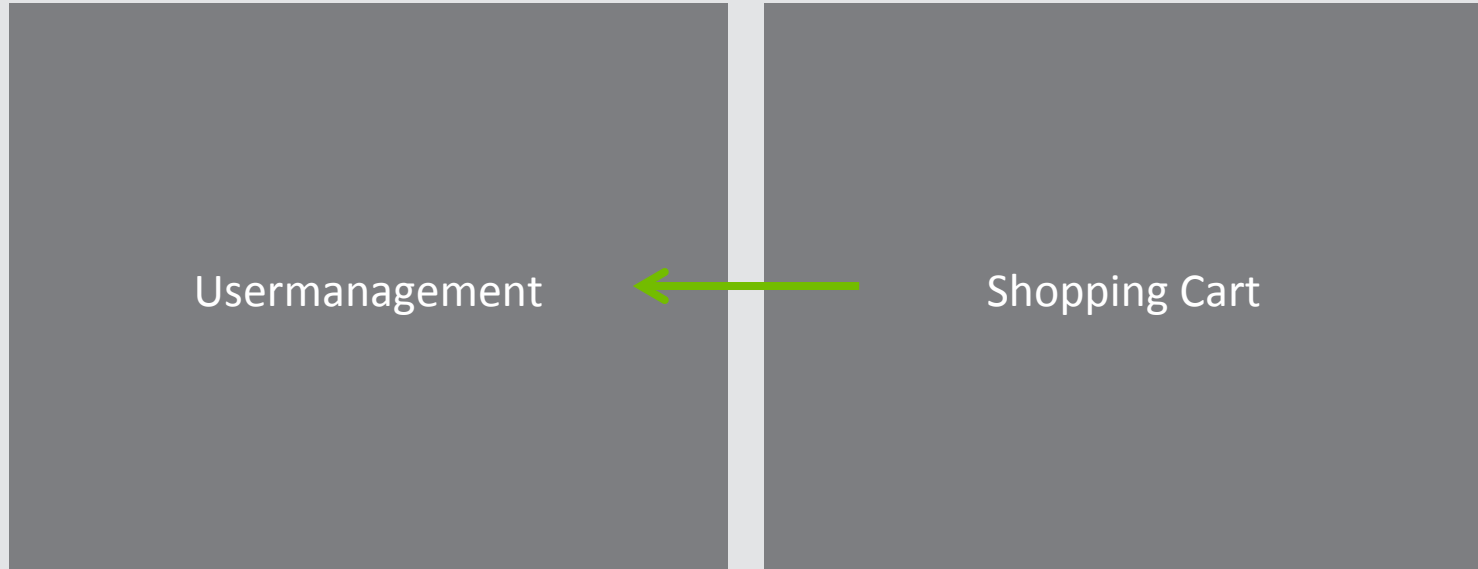


Usermanagement

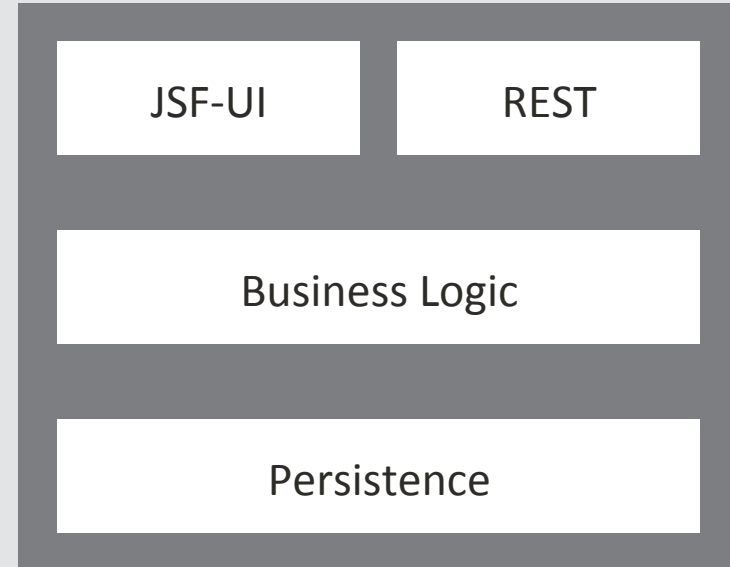
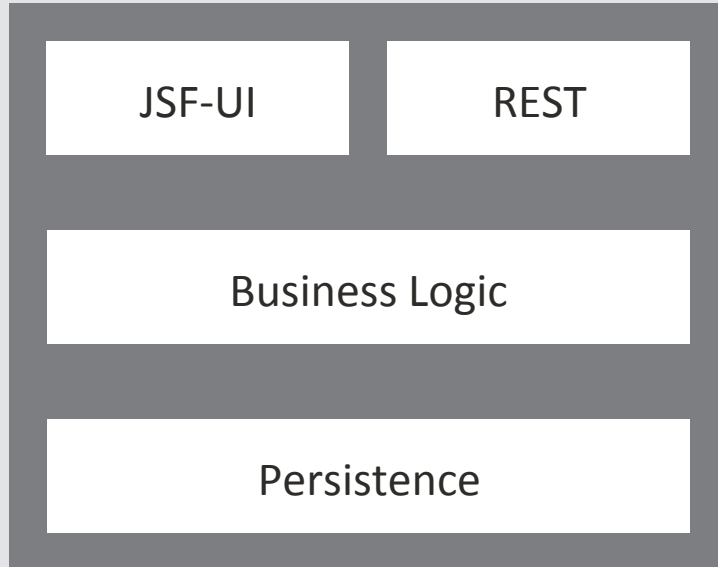


Shopping Cart

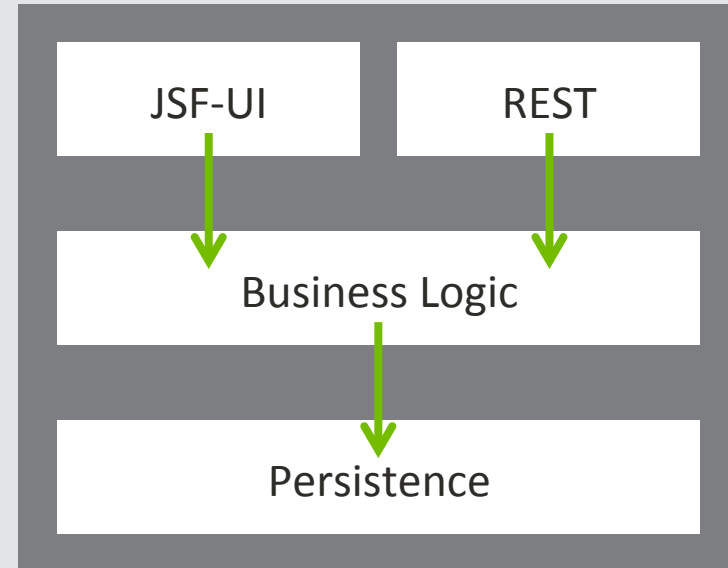
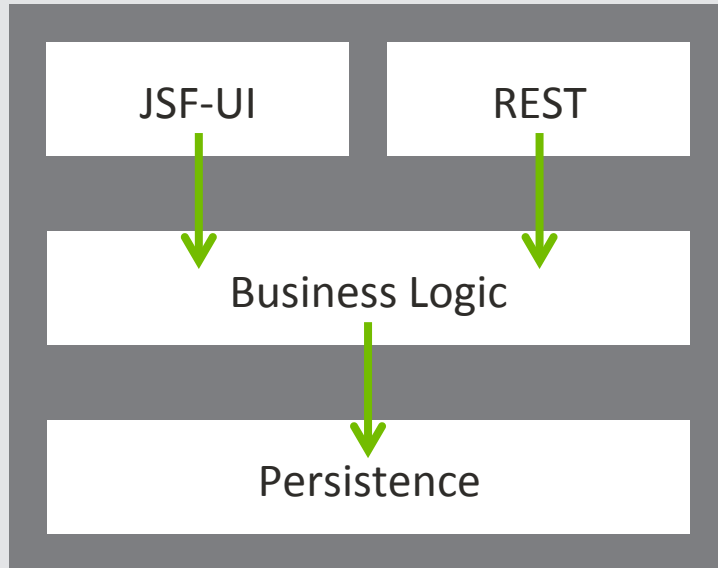
- Sketch of an architecture
 - Allowed dependencies between business modules



- Sketch of an architecture
 - Technical layering

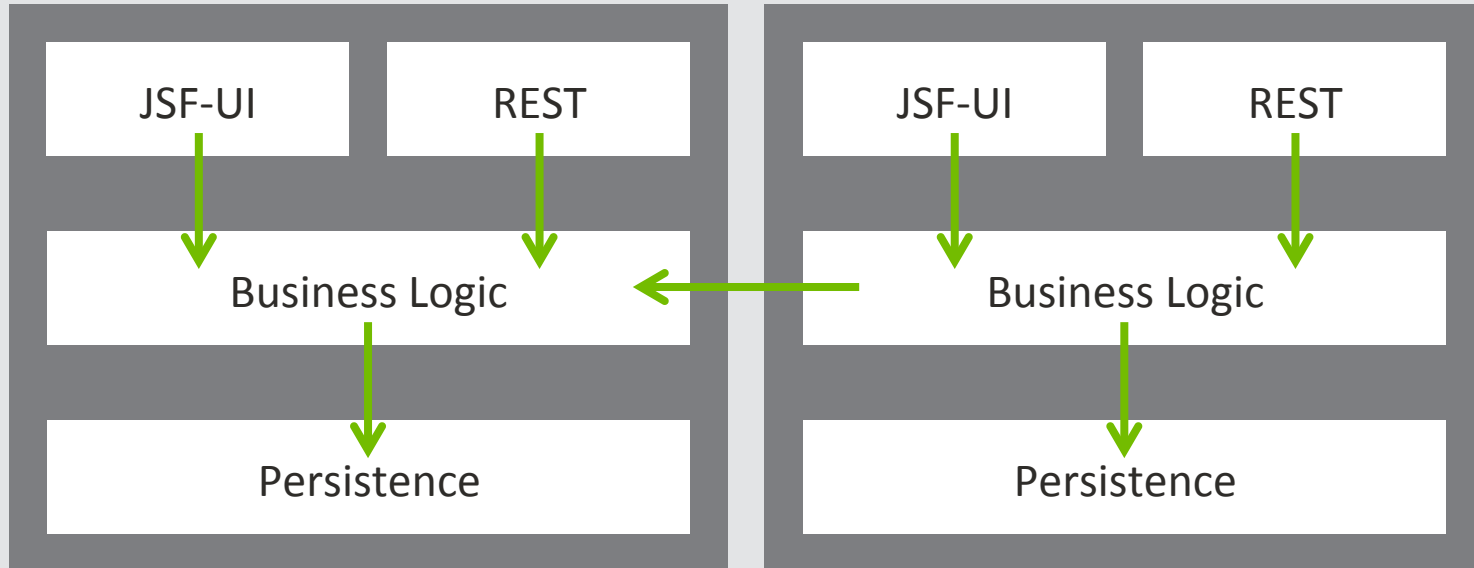


- Sketch of an architecture
 - Allowed dependencies between technical layers



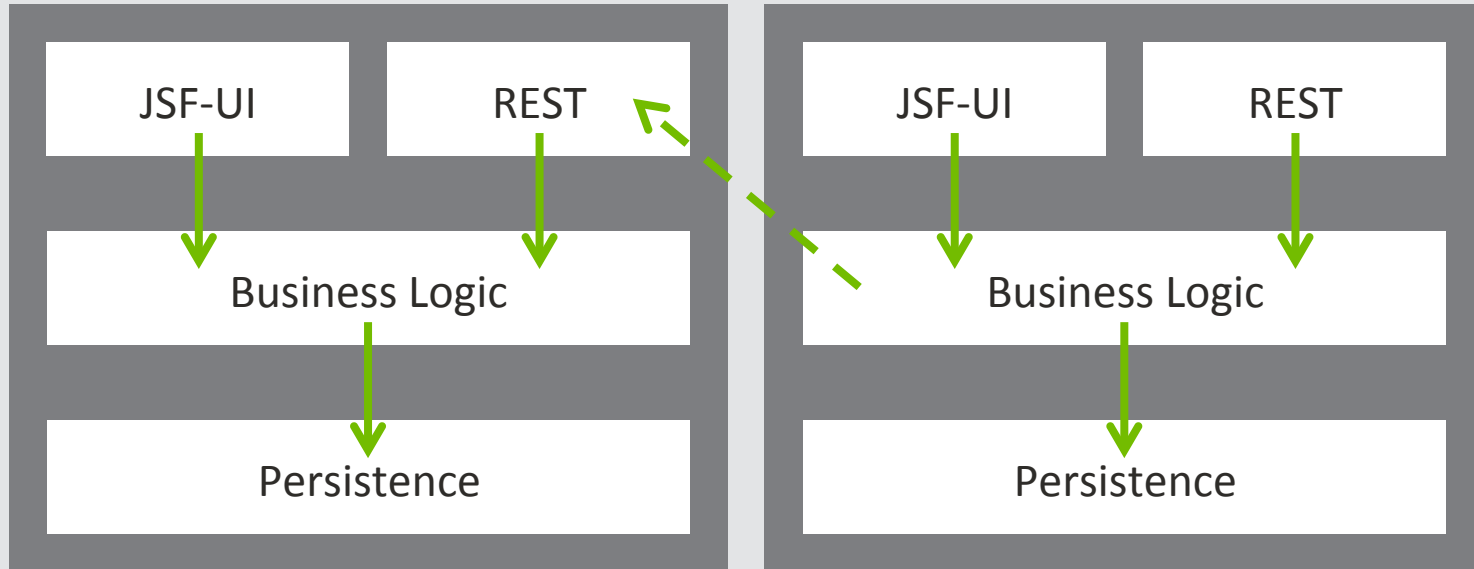
■ Sketch of an architecture

- Allowed dependencies between business modules and technical layers

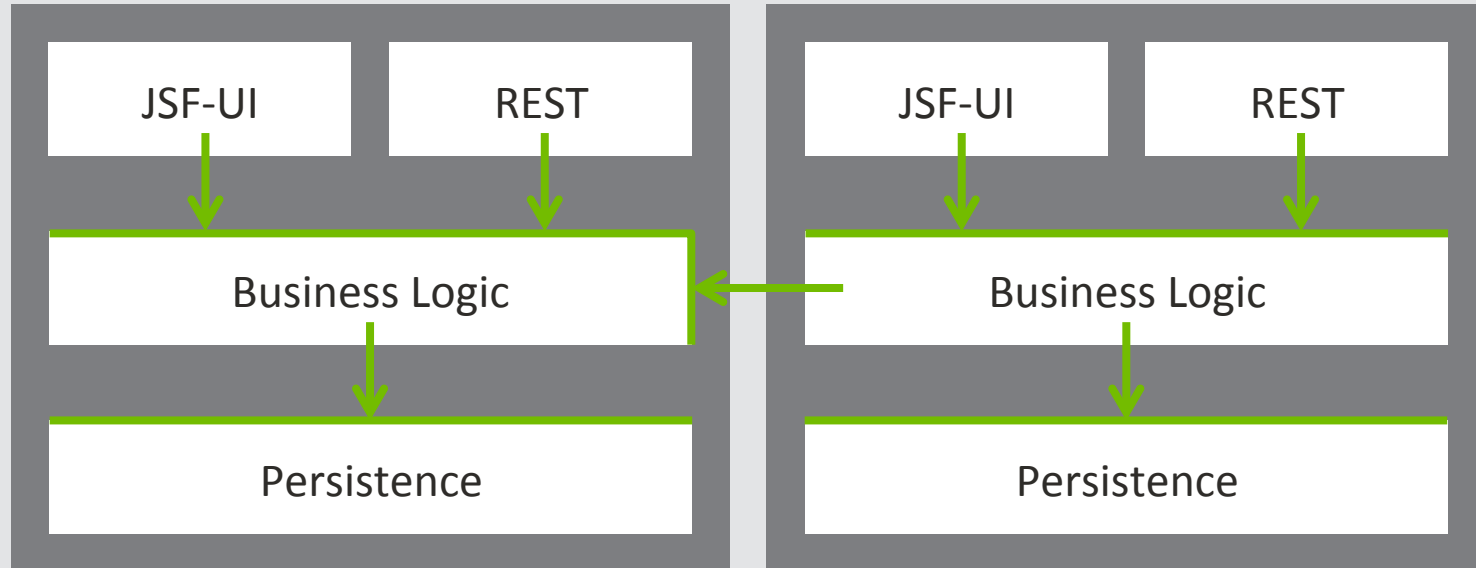


■ Sketch of an architecture

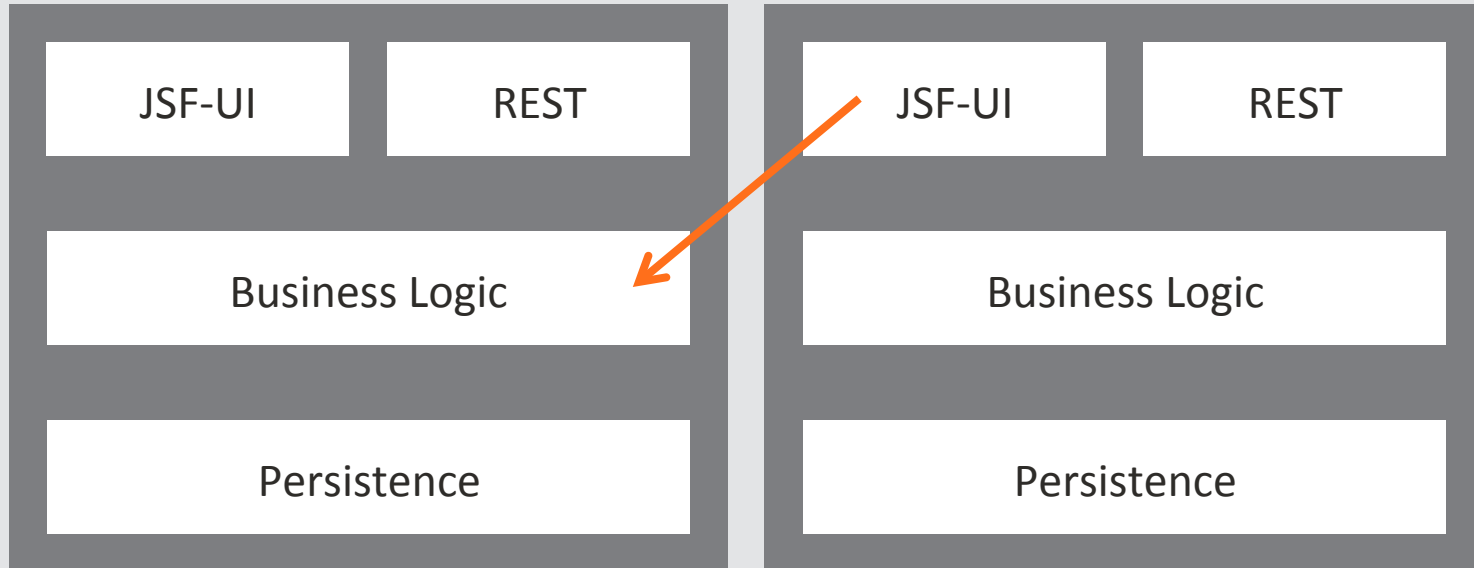
- Allowed dependencies between business modules and technical layers



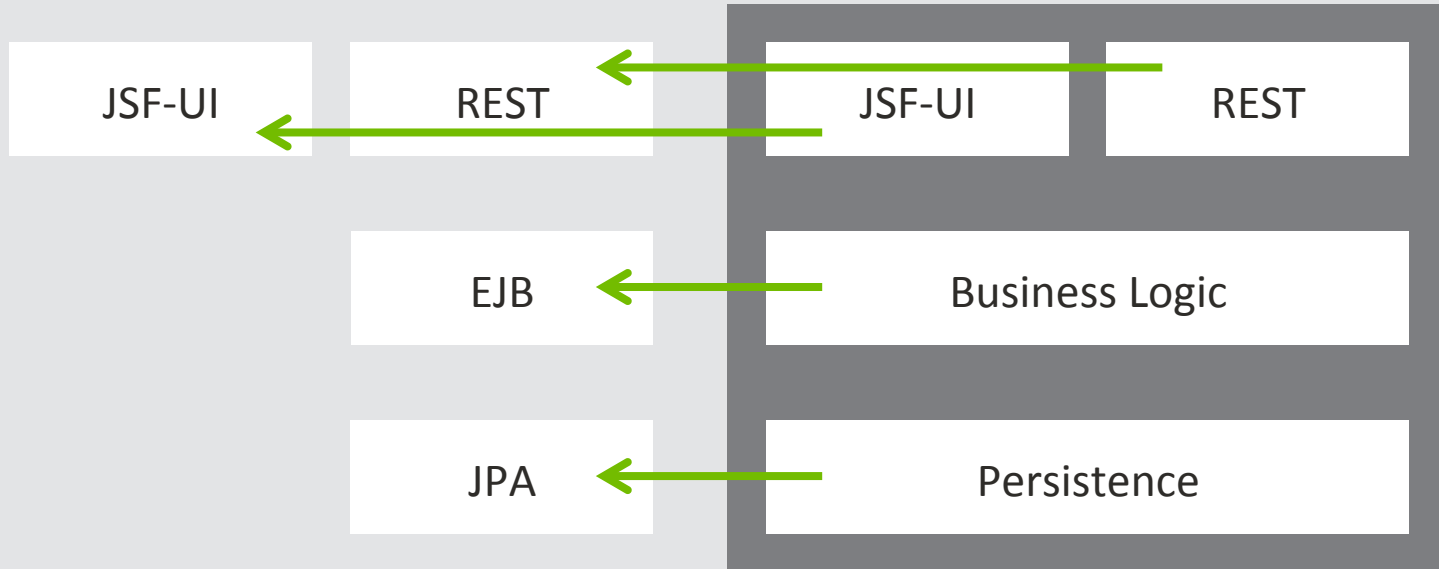
- Sketch of an architecture
 - Abstraction between layers (API vs. Implementation)



- Sketch of an architecture
 - Forbidden dependency



- Sketch of an architecture
 - Allowed external dependencies per layer



- 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“

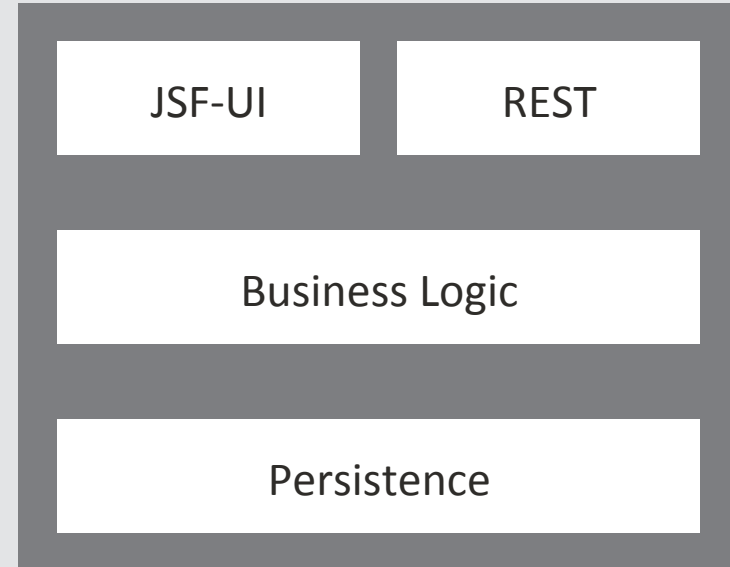
- Definition of business modules on „top level“

- 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

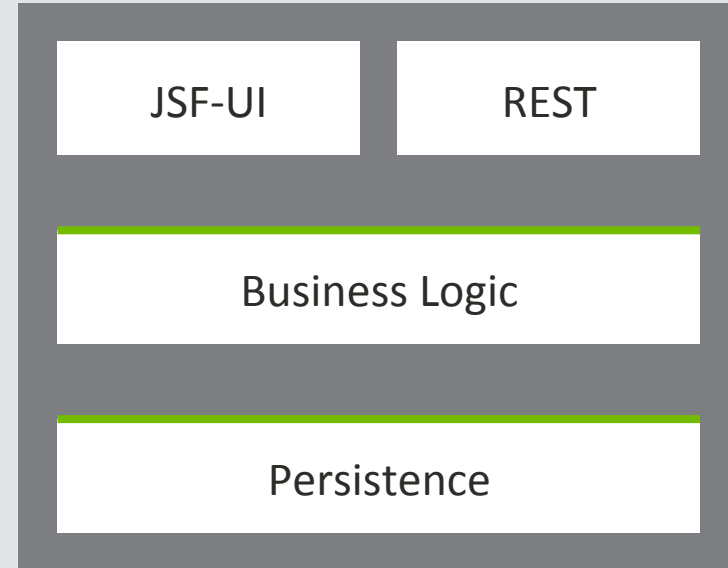


- Translation of architecture rules into a project structure

- 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



■ 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 „api.model“ packages.

- Test design

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

■ Abstraction

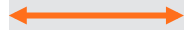
- Remote APIs must be interfaces declaring only primitives or immutables as parameter or return types.
- OSGi-Bundles must only export dedicated API packages.

Source Code

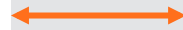
Java, Properties,
XML, YML

„Code Quality“

Complexity



Maintainability



Erosion

Source Code

Java, Properties,
XML, YML

People

Architect, Developer,
Test

Tooling

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 „api.model“ packages

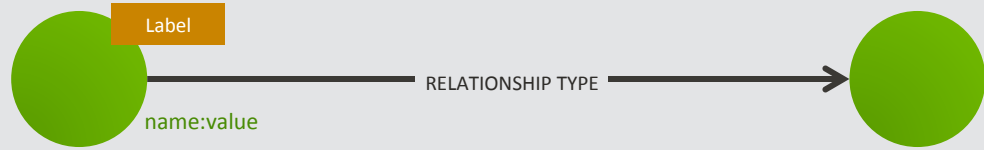
- Concepts and Constraints
 - How to make them visible?
 - Is there a language to describe them?

Analysis of software systems using jQAssistant and Neo4j

Software As A Graph

- All we need is...

- Nodes
- Labels
- Properties
- Relationships

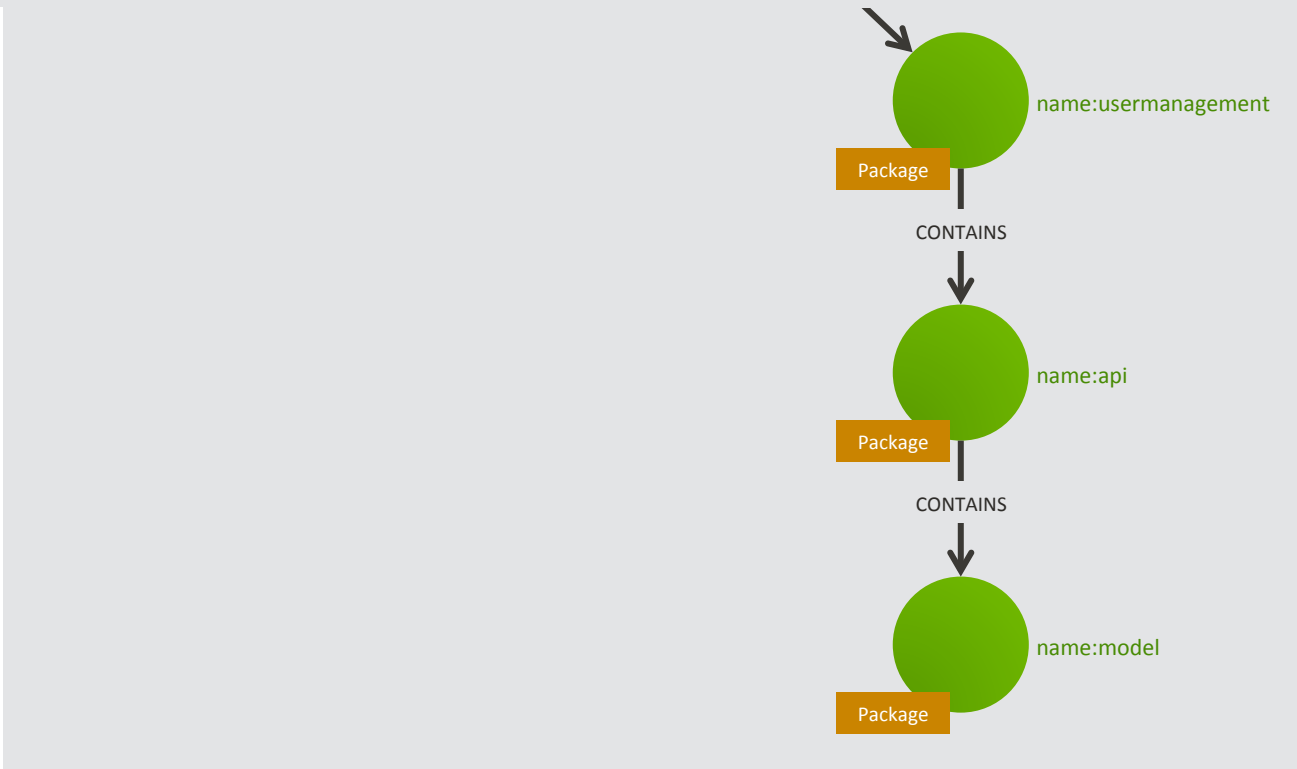
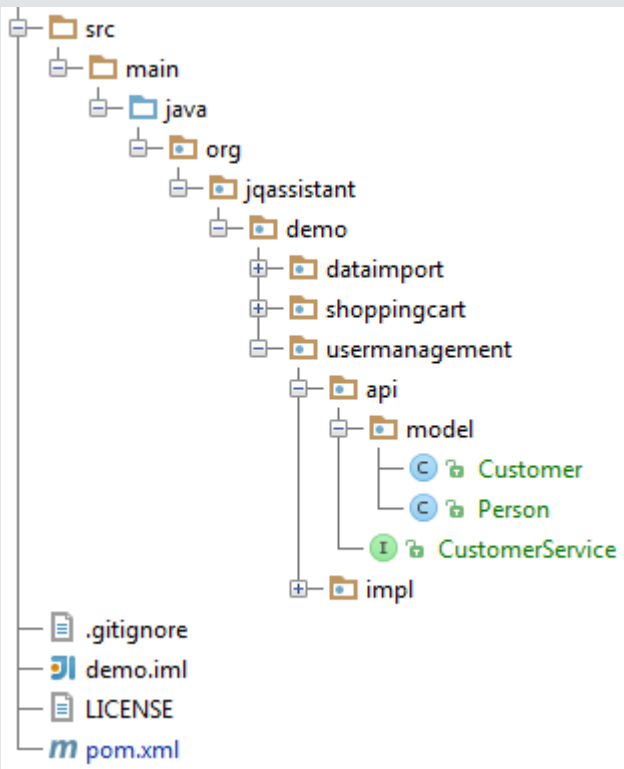
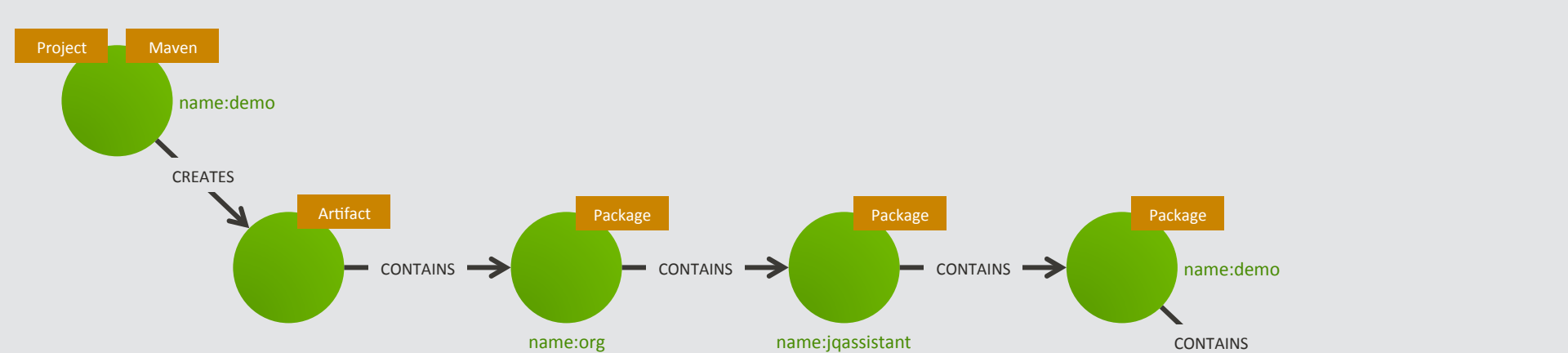


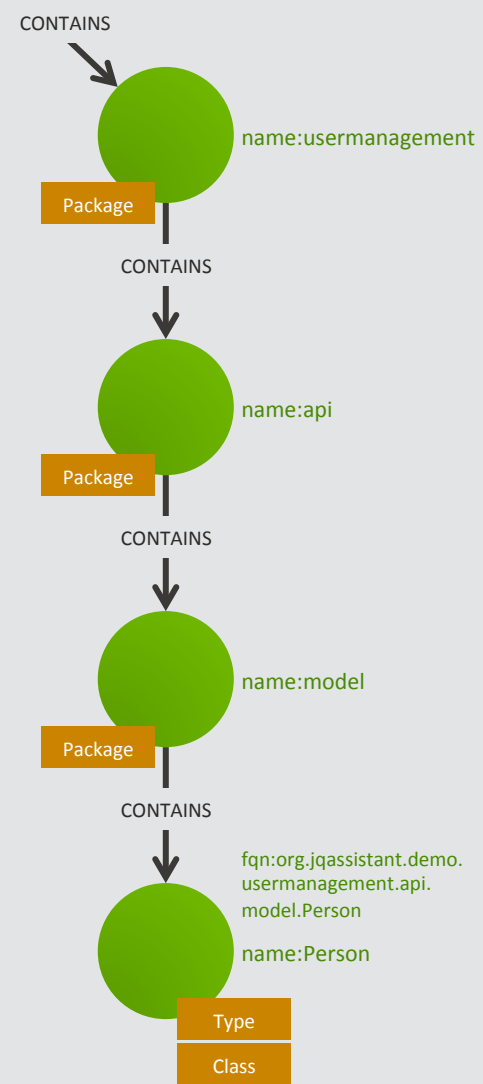
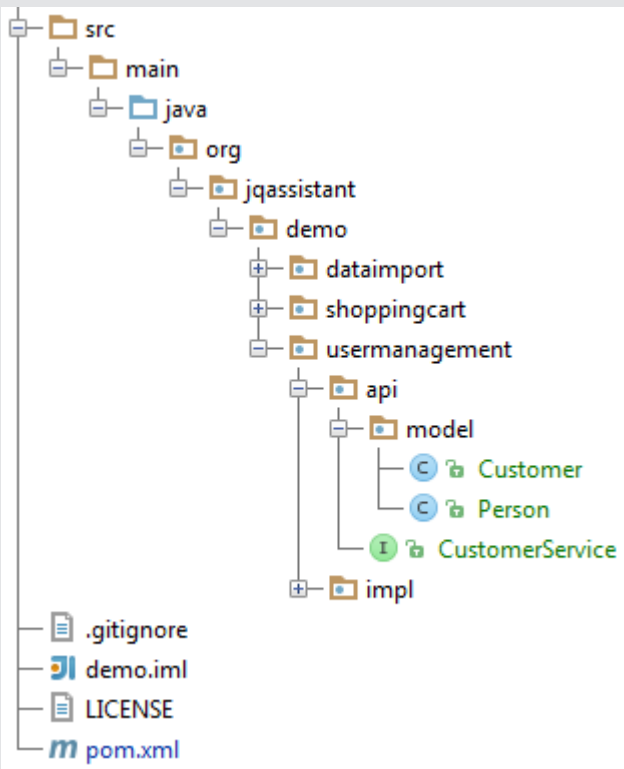
- Modeling is just...

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

- We don't need...

- Foreign keys
- Tables and schemas
- Knowledge in graph theory

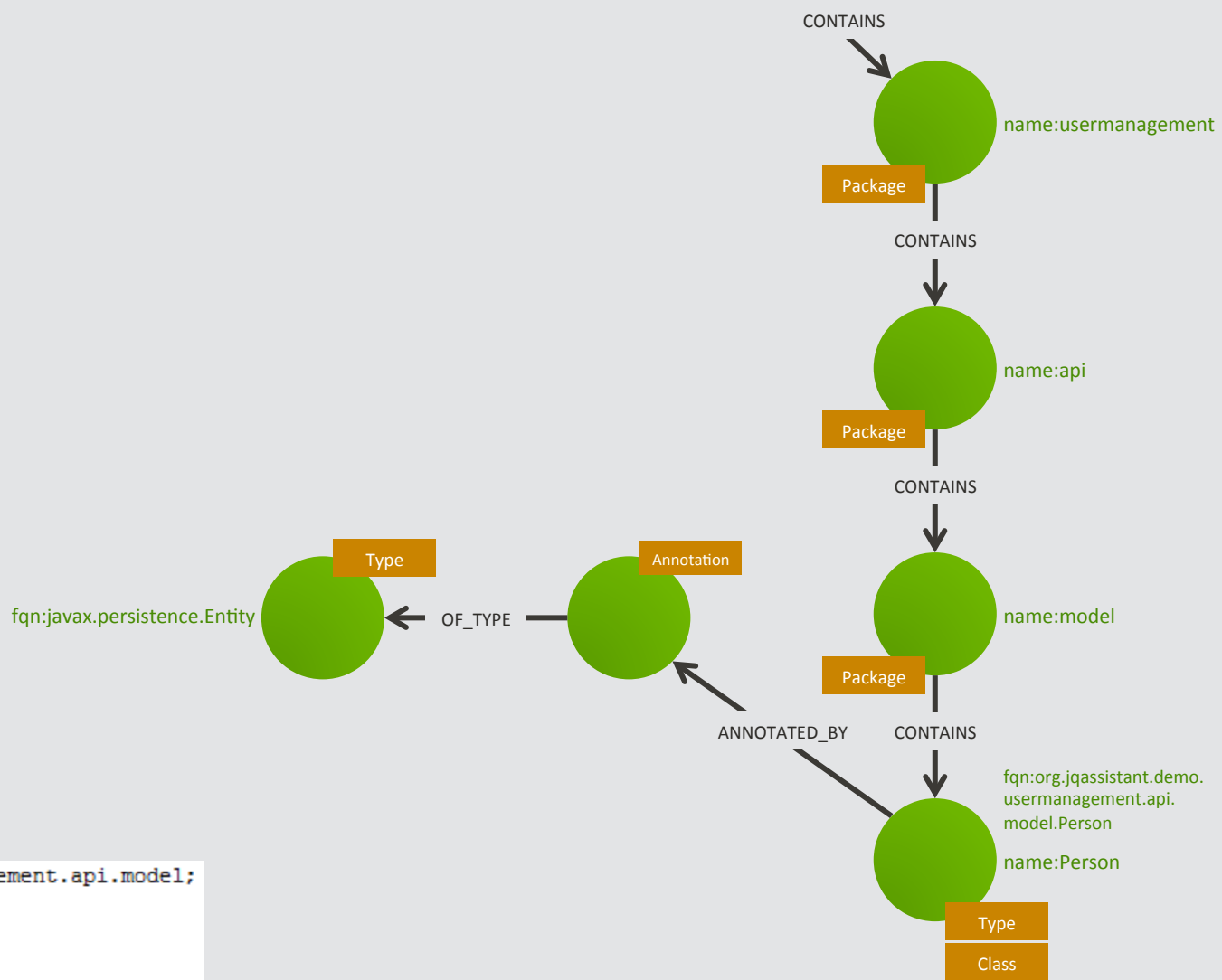


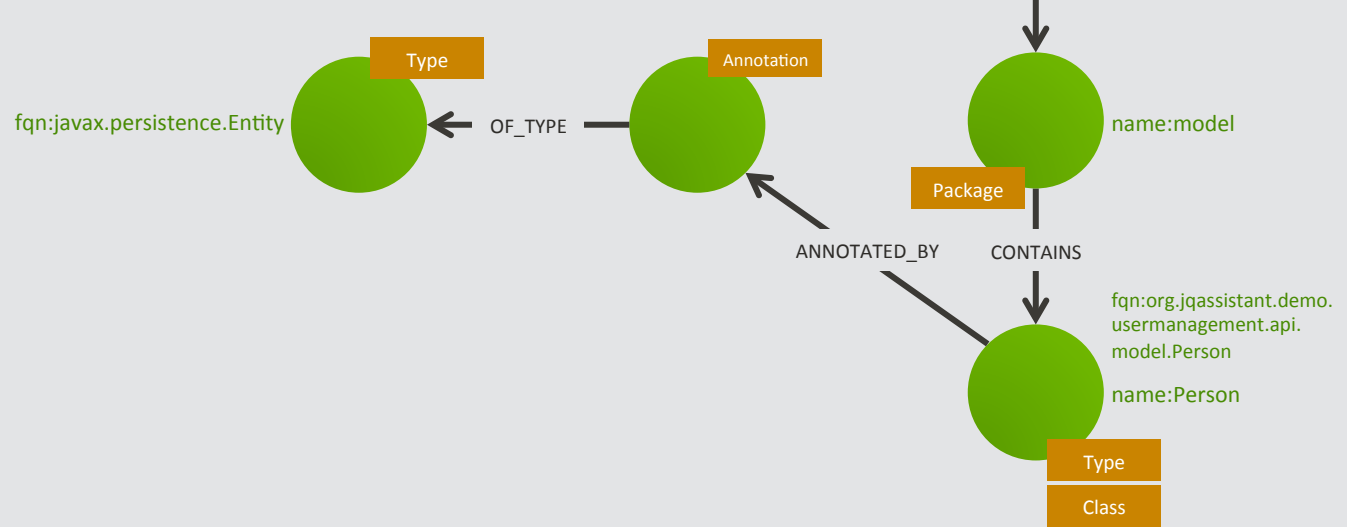


```
package org.jqassistant.demo.usermanagement.api.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Person {
```





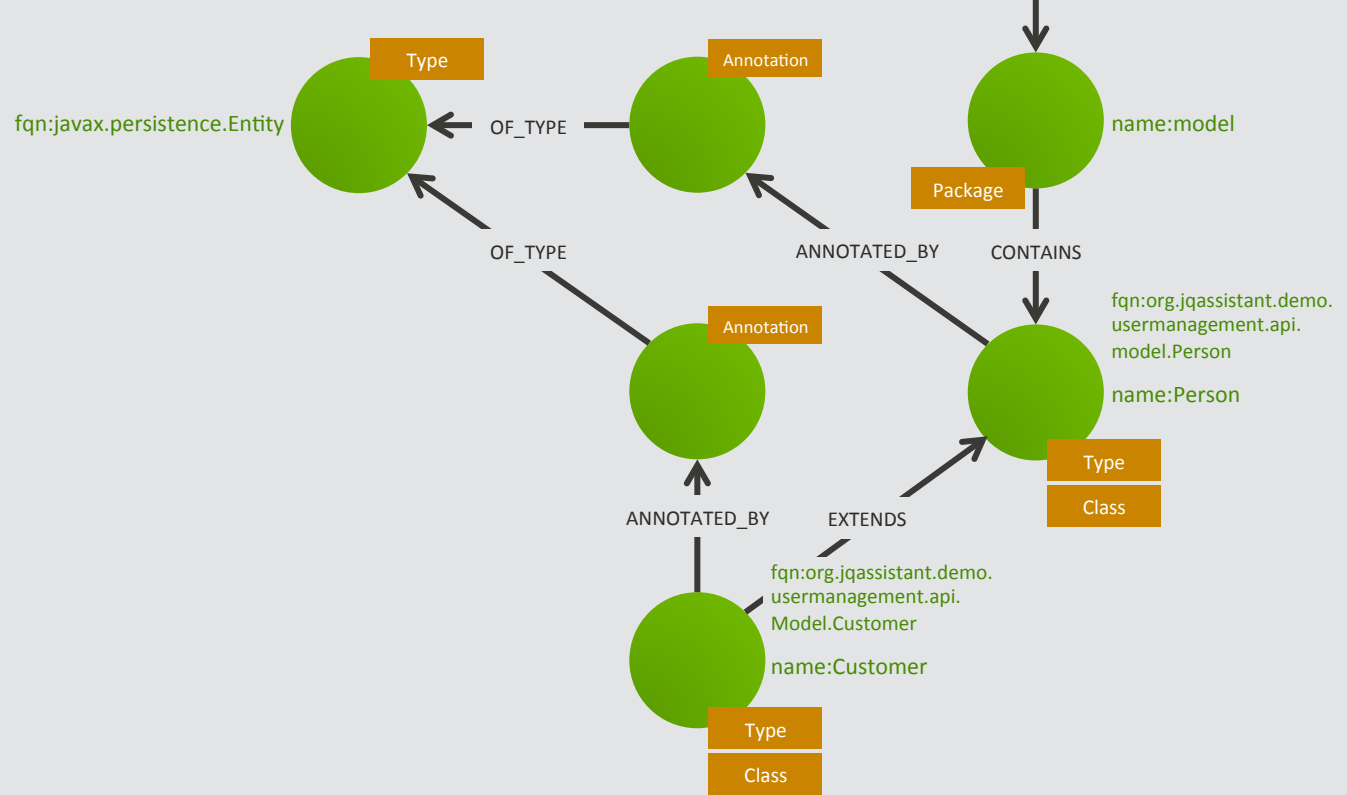
```

package org.jqassistant.demo.usermanagement.api.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Person {

```



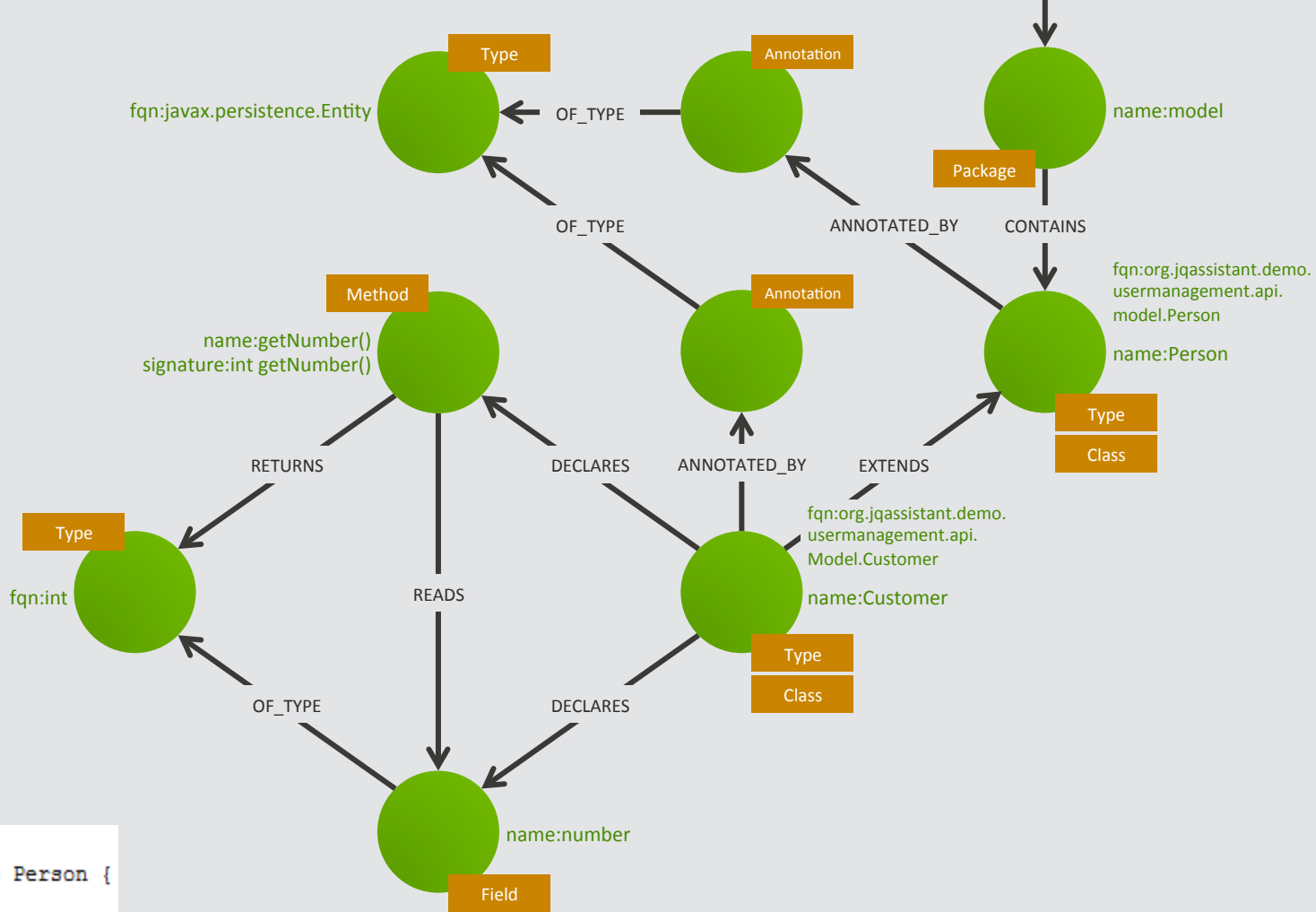
```

@Entity
public class Customer extends Person {

    private int number;

    public int getNumber() {
        return number;
    }
}

```



```

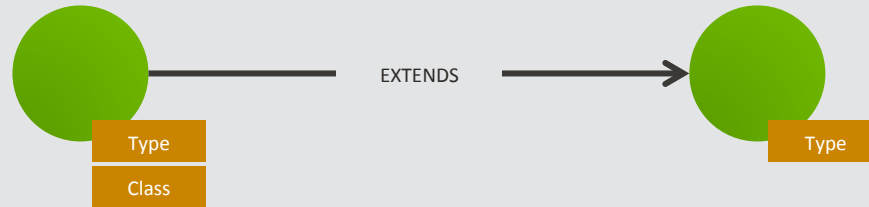
@Entity
public class Customer extends Person {

    private int number;

    public int getNumber() {
        return number;
    }
}

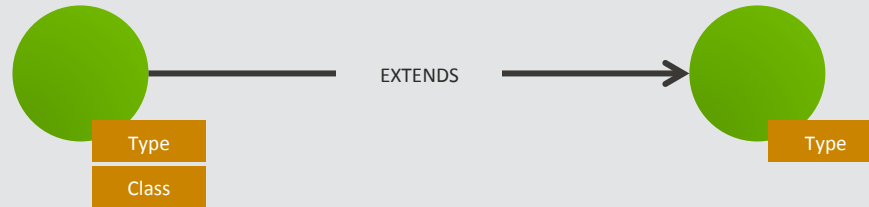
```

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



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

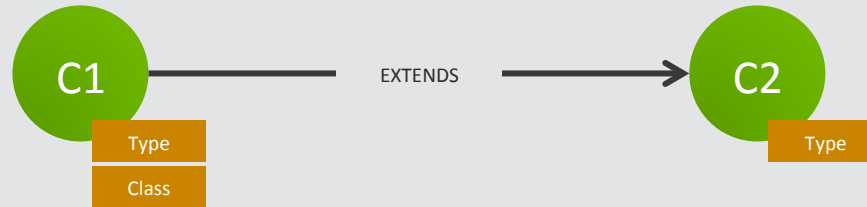
- Explore an application using queries
 - Which class extends from another class?



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

() - [] - > ()

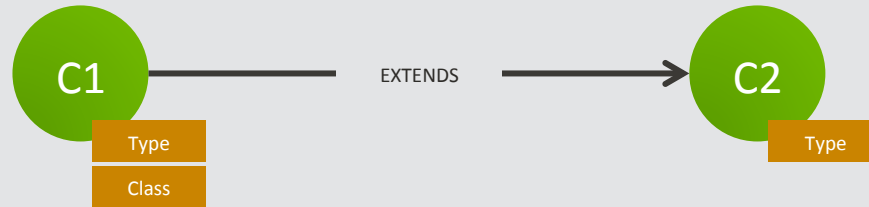
- Explore an application using queries
 - Which class extends from another class?



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

(c1) - [] -> (c2)

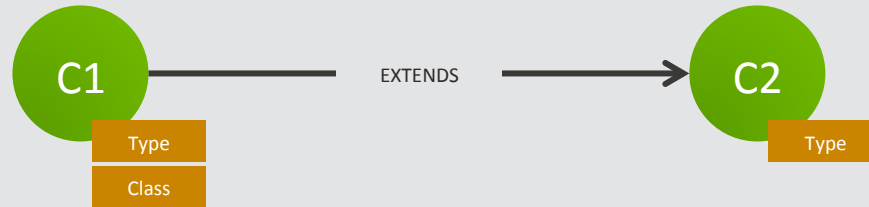
- Explore an application using queries
 - Which class extends from another class?



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

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

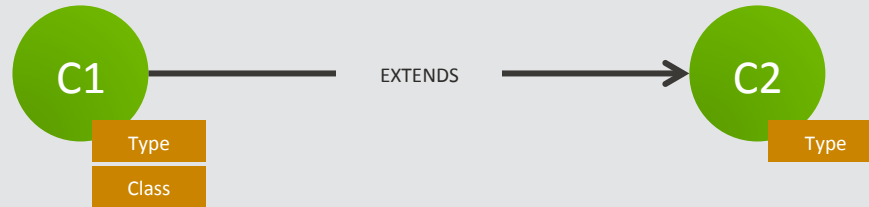
- Explore an application using queries
 - Which class extends from another class?



- 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`

- Which classes contain the highest number of methods?

- Which classes contain the highest number of methods?

MATCH

```
(class:Class) - [:DECLARES] -> (method:Method)
```

- Which classes contain the highest number of methods?

MATCH

```
(class:Class) - [:DECLARES] -> (method:Method)
```

RETURN

```
class.fqn, count(method) as Methods
```

- Which classes contain the highest number of methods?

MATCH

```
(class:Class)-[:DECLARES]->(method:Method)
```

RETURN

```
class.fqn, count(method) as Methods
```

ORDER BY

```
Methods DESC
```

- Which classes contain the highest number of methods?

MATCH

```
(class:Class)-[:DECLARES]->(method:Method)
```

RETURN

```
class.fqn, count(method) as Methods
```

ORDER BY

```
Methods DESC
```

LIMIT 20

- Which class has the deepest inheritance hierarchy?

- Which class has the deepest inheritance hierarchy?

MATCH

```
h=(class:Class)-[:EXTENDS*]->(super:Type)
```

- Which class has the deepest inheritance hierarchy?

MATCH

```
h=(class:Class)-[:EXTENDS*]->(super:Type)
```

RETURN

```
class.fqn, length(h) as Depth
```

- Which class has the deepest inheritance hierarchy?

MATCH

```
h=(class:Class)-[:EXTENDS*]->(super:Type)
```

RETURN

```
class.fqn, length(h) as Depth
```

ORDER BY

```
Depth desc
```

- Which class has the deepest inheritance hierarchy?

```
MATCH
```

```
  h=(class:Class)-[:EXTENDS*]->(super:Type)
```

```
RETURN
```

```
  class.fqn, length(h) as Depth
```

```
ORDER BY
```

```
  Depth desc
```

```
LIMIT 20
```

- Which classes are affected by IOExceptions?

```
MATCH
```

```
(e:Type) - [:DECLARES] -> (init:Constructor)
```

```
WHERE
```

```
e.fqn="java.io.IOException"
```

```
WITH
```

```
e,init
```

```
MATCH
```

```
(type:Type) - [:DECLARES] -> (method:Method),  
path=(method) - [:INVOKES*] -> (init)
```

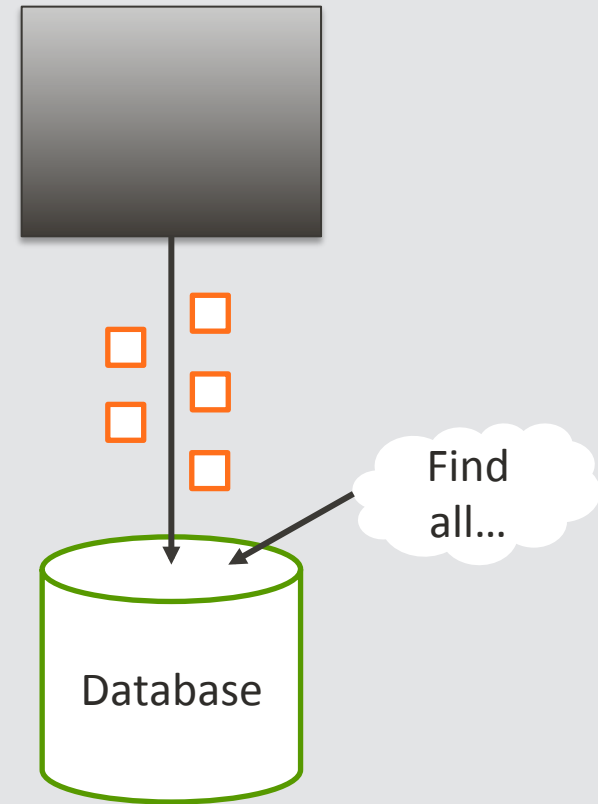
```
RETURN
```

```
e,type,path
```

```
LIMIT 10
```

■ Why?

- Holistic view on the „real world“
- Exploration
 - Effective searches beyond IDE capabilities
 - Impact Analysis
- Individual metrics and visualization
 - Complexity, Dependencies
- Validation
 - Coding rules, design & architecture constraints
- Reporting
 - Continuous
- It's fun!



Analysis of software systems using jQAssistant and Neo4j

Demo #1

Analysis of software systems using jQAssistant and Neo4j

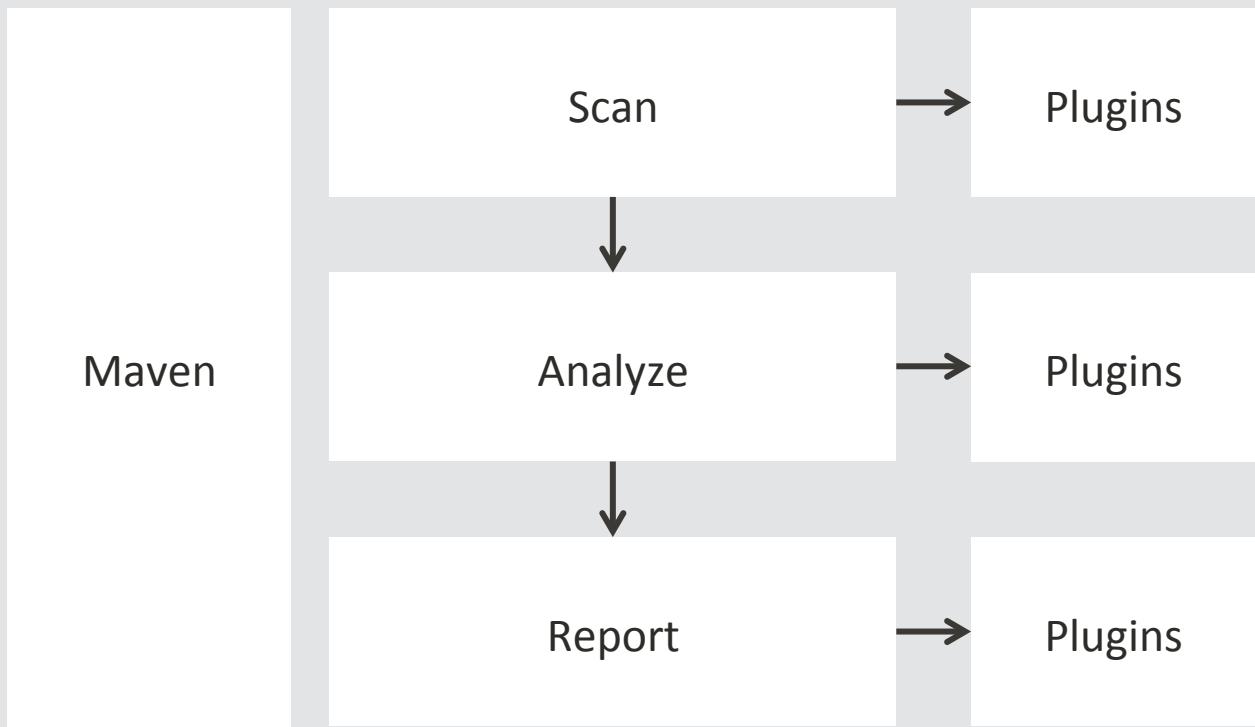
jQAssistant

jQAAssistant

<http://jQAAssistant.org>

- Open Source: GPLv3
- Current release: 1.0.0
 - initiated: 03/2013
 - first stable release: 04/2015
- Embedded Neo4j
 - Community Edition 2.x
 - no installation necessary





■ Getting Started – Command Line

- `unzip jqassistant.distribution-x.y.z.zip`
- `cd jqassistant.distribution-x.y.z/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

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

- mvn jqassistant:scan
- mvn jqassistant:server
- mvn jqassistant:available-rules
- mvn jqassistant:analyze

■ Scanner plugins

ZIP

JAR, WAR, EAR

GZ

*.class

*.properties

MANIFEST.MF

/META-INF/services/*

web.xml

application.xml

beans.xml

pom.xml

surefire-reports.xml

*.xsd

Liquibase

CheckStyle

FindBugs

JaCoCo

Git

*.java

■ Plugin API is public

M2 Repository

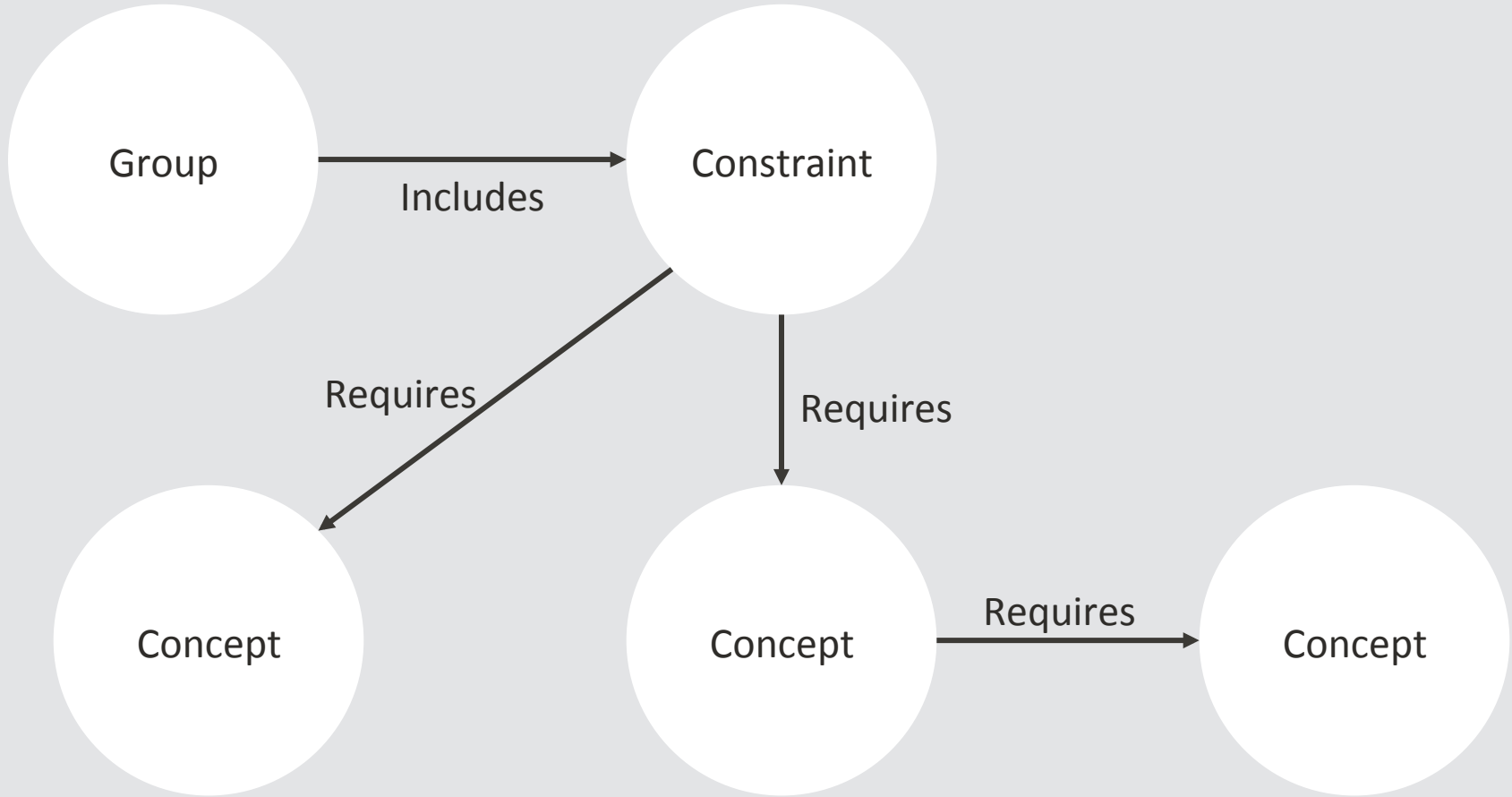
RDBMS Schema

- Analyze
 - Execution of rules

 - Concepts
 - Enrich data model

 - Constraints
 - Detect violations

 - Group
 - Allow different execution profiles



■ Concept

```
<jqa:jqassistant-rules xmlns:jqa="...">
```

```
<concept id="jpa2:Entity">  
  <description>Labels all types annotated with  
    @javax.persistence.Entity with JPA and ENTITY.  
</description>  
  <cypher><![CDATA[  
    MATCH  
      (t:Type)-[:ANNOTATED_BY]->()-[:OF_TYPE]->(a:Type)  
      WHERE a.fqn="javax.persistence.Entity"  
      SET t:Jpa:Entity  
      RETURN t AS Entity  
    ]]></cypher>  
</concept>
```

```
</jqa:jqassistant-rules>
```

■ Concept

```
<jqa:jqassistant-rules xmlns:jqa="...">
```

```
  <concept id="jpa2:Entity">
```

```
    <description>Labels all types annotated with  
      @javax.persistence.Entity with JPA and ENTITY.
```

```
  </description>
```

```
  <cypher><![CDATA[
```

```
    MATCH
```

```
      (t:Type)-[:ANNOTATED_BY]->()-[:OF_TYPE]->(a:Type)
```

```
      WHERE a.fqn="javax.persistence.Entity"
```

```
      SET t:Jpa:Entity
```

```
      RETURN t AS Entity
```

```
    ]]></cypher>
```

```
</concept>
```

```
</jqa:jqassistant-rules>
```

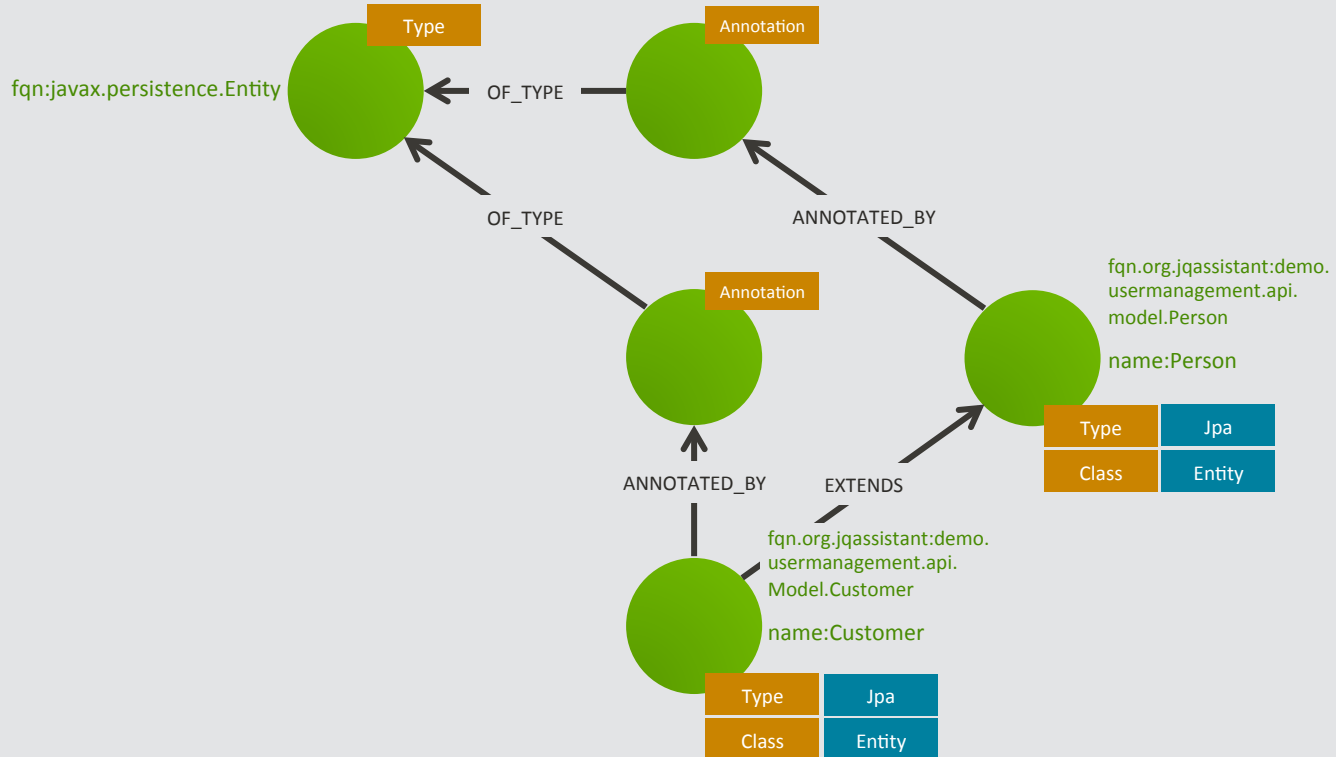
■ Concept

```
<jqa:jqassistant-rules xmlns:jqa="...">
```

```
<concept id="jpa2:Entity">  
  <description>Labels all types annotated with  
    @javax.persistence.Entity with JPA and ENTITY.  
</description>  
  <cypher><![CDATA[  
    MATCH  
      (t:Type)-[:ANNOTATED_BY]->()-[:OF_TYPE]->(a:Type)  
    WHERE a.fqn="javax.persistence.Entity"  
    SET t:Jpa:Entity  
    RETURN t AS Entity  
  ]]></cypher>  
</concept>
```

```
</jqa:jqassistant-rules>
```

■ Concept



■ Constraint

```
<jqa:jqassistant-rules xmlns:jqa="...">  
  <constraint id="JpaEntitiesInModelPackage">  
    <requiresConcept refId="jpa2:Entity"/>  
    <description>All JPA entities must be located in  
      packages named "model".  
    </description>  
    <cypher><![CDATA[  
      MATCH (p:Package)-[:CONTAINS]->(e:Jpa:Entity)  
      WHERE NOT(p.name = "model")  
      RETURN  
        e AS EntityInWrongPackage  
    ]]></cypher>  
  </constraint>  
</jqa:jqassistant-rules>
```

■ Constraint

```
<jqa:jqassistant-rules xmlns:jqa="...">
  <constraint id="JpaEntitiesInModelPackage">
    <requiresConcept refId="jpa2:Entity"/>
    <description>All JPA entities must be located in
      packages named "model".
    </description>
    <cypher><![CDATA[
      MATCH (p:Package)-[:CONTAINS]->(e:Jpa:Entity)
      WHERE NOT(p.name = "model")
      RETURN
        e AS EntityInWrongPackage
    ]]></cypher>
  </constraint>
</jqa:jqassistant-rules>
```

■ Constraint

```
<jqa:jqassistant-rules xmlns:jqa="...">  
  <constraint id="JpaEntitiesInModelPackage">  
    <requiresConcept refId="jpa2:Entity"/>  
    <description>All JPA entities must be located in  
      packages named "model".  
    </description>  
    <cypher><![CDATA[  
      MATCH (p:Package)-[:CONTAINS]->(e:Jpa:Entity)  
      WHERE NOT(p.name = "model")  
      RETURN  
        e AS EntityInWrongPackage  
    ]]></cypher>  
  </constraint>  
</jqa:jqassistant-rules>
```

■ Constraint

```
<jqa:jqassistant-rules xmlns:jqa="...">  
  <constraint id="JpaEntitiesInModelPackage">  
    <requiresConcept refId="jpa2:Entity"/>  
    <description>All JPA entities must be located in  
      packages named "model".  
    </description>  
    <cypher><![CDATA[  
      MATCH (p:Package)-[:CONTAINS]->(e:Jpa:Entity)  
      WHERE NOT(p.name = "model")  
      RETURN  
        e AS EntityInWrongPackage  
    ]]></cypher>  
  </constraint>  
</jqa:jqassistant-rules>
```


■ Group

```
<jqa:jqassistant-rules xmlns:jqa="...">
```

```
  <group id="default">
    <includeConstraint
      refId="JpaEntitiesInModelPackage"/>
    <includeConstraint
      refId="abstractness:ApiMustNotDependOnImplementation"/>
    <includeConstraint
      refId="EjbLocatedInImplementationPackage"/>
    <includeConstraint refId="TestClassNameHasTestSuffix"/>
    <includeConstraint refId="dependency:TypeCycles"/>
    <includeConstraint refId="dependency:ArtifactCycles"/>
  </group>
```

```
</jqa:jqassistant-rules>
```

■ Group

```
<jqa:jqassistant-rules xmlns:jqa="...">

  <group id="default">
    <includeConstraint
      refId="JpaEntitiesInModelPackage"/>
    <includeConstraint
      refId="abstractness:ApiMustNotDependOnImplementation"/>
    <includeConstraint
      refId="EjbLocatedInImplementationPackage"/>
    <includeConstraint refId="TestClassNameHasTestSuffix"/>
    <includeConstraint refId="dependency:TypeCycles"/>
    <includeConstraint refId="dependency:ArtifactCycles"/>
  </group>

</jqa:jqassistant-rules>
```

■ Rules as AsciiDoc

= Naming Rules

== JPA Entities

```
[[naming:EntitiesInModelPackages]]
.All JPA entities must be located in "api.model" packages.
[source,cypher,role=constraint,depends="jpa2:Entity"]
----
MATCH
  (p:Package)-[:CONTAINS]->(e:Jpa:Entity)
WHERE NOT
  p.fqn =~ '.*\\.api\\.model'
RETURN
  e.fqn as Entity
----
```

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

Naming Rules

Table of Contents

[JPA Entities](#)

[Packages](#)

JPA Entities

All JPA entities must be located in "api.model" packages.

```
MATCH
  (p:Package) -[:CONTAINS]->(e:Jpa:Entity)
WHERE NOT
  p.fqn =~ '.*\\.api\\.model'
RETURN
  e.fqn as Entity
```

CYPHER

Analysis of software systems using jQAssistant and Neo4j

Demo #2

- All packages in a Maven module must be prefixed with `${groupId}.${artifactId}`

MATCH

```
(module:Maven)-[:CREATES]->(artifact:Artifact)
```

WITH

```
artifact, artifact.group + "\\." + artifact.name + "\\  
\\..*" as prefixPattern
```

MATCH

```
(artifact)-[:CONTAINS]->(type:Type)
```

WHERE NOT

```
type.fqn =~ prefixPattern
```

RETURN

```
type as TypeInInvalidPackage
```

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

MATCH

(mdb:MessageDrivenBean)

WHERE NOT

mdb.name =~ ".*MDB"

RETURN

mdb as InvalidBean

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

MATCH

```
(p:Package)-[:CONTAINS]->(e:Jpa:Entity)
```

WHERE NOT

```
p.fqn =~ '.*\\.api\\.model'
```

RETURN

```
e.fqn as Entity
```


- Every test method must contain at least one assertion.

MATCH

```
(testType:Type) - [:DECLARES] -> (testMethod:Test:Method)
```

WHERE NOT

```
(testMethod) - [:INVOKES*..3] -> (:Method:Assert)
```

RETURN

```
testType AS DeclaringType,
```

```
testMethod AS Method
```

- Each assertion must provide a human readable message.

MATCH

```
(testType:Type)-[:DECLARES]->(testMethod:Method),  
(testMethod)-[i:INVOKES]->(assertMethod:Assert:Method)
```

WHERE NOT

```
assertMethod.signature =~  
'void assert.*\\(java.lang.String,.*\\)'
```

RETURN

```
i AS Invocation, // provides lineNumber  
testType AS DeclaringType,  
testMethod AS Method
```

- Remote APIs must be interfaces declaring only primitives or immutables as parameter or return types.

MATCH

```
(api:Package:API:Remote),  
(api)-[:CONTAINS*]->(interface:Interface),  
(interface)-[:DECLARES]->(method:Method),  
(method)-[:HAS|RETURNS]->(type:Type)
```

WHERE NOT

```
(type:Immutable OR type:Primitive)
```

RETURN

```
interface AS InvalidInterface, type AS InvalidType
```

- OSGi-Bundles must only export dedicated API packages.

MATCH

```
(bundle:Artifact:Bundle)-[:EXPORTS]->(package:Package)
```

WHERE NOT

```
(package:API)
```

RETURN

```
package AS InvalidExport
```

Thank You! – Questions?

Mail: info@jqassistant.org

Web: jqassistant.org

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