5.– 8. September 2011
in Nürnberg

Herbstcampus

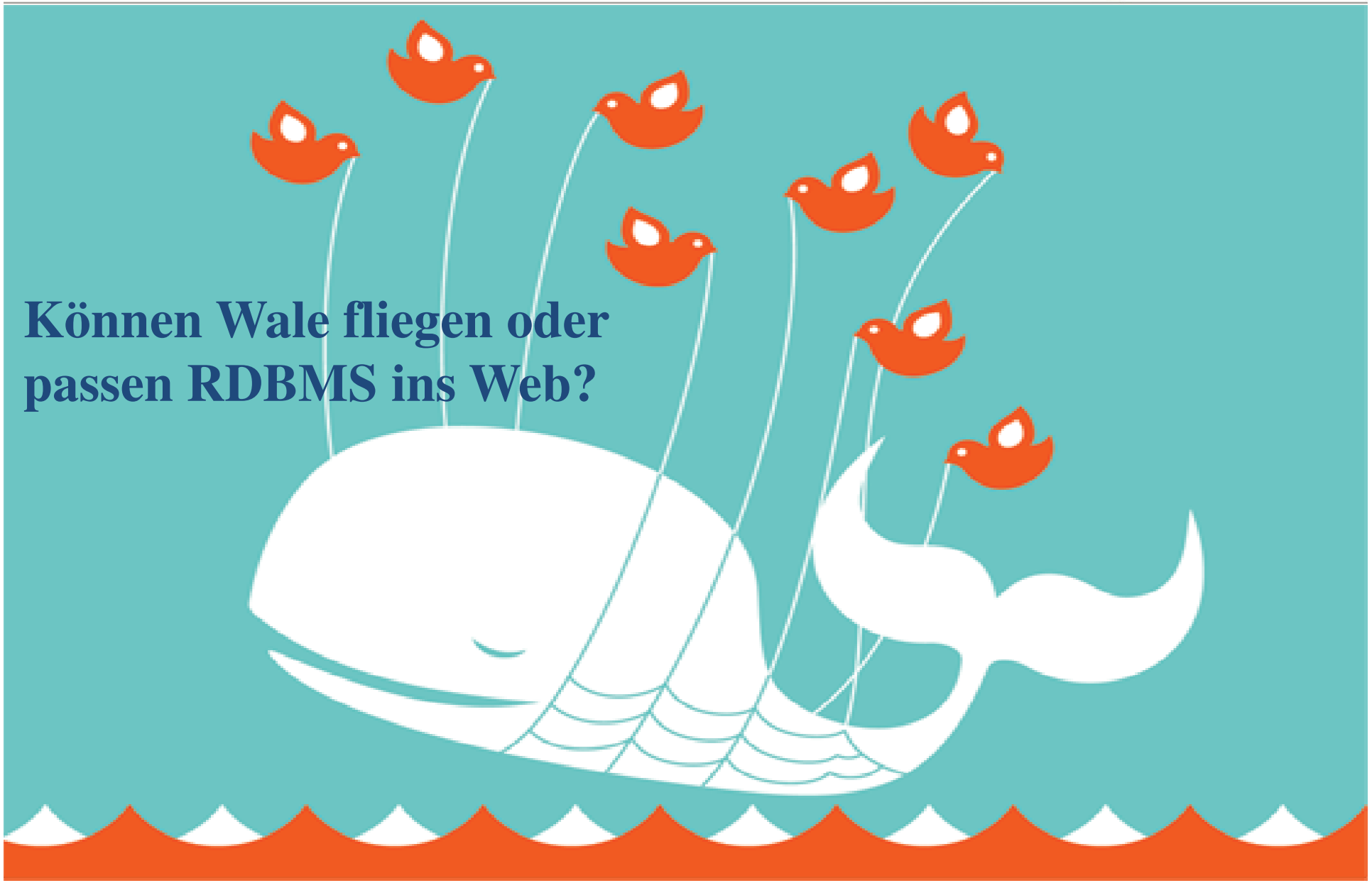
Wissenstransfer
par excellence

RESTFull Datenbanken

Wie kommen Daten mit Apache OpenJPA JEST einfach ins Web?

Frank Pientka

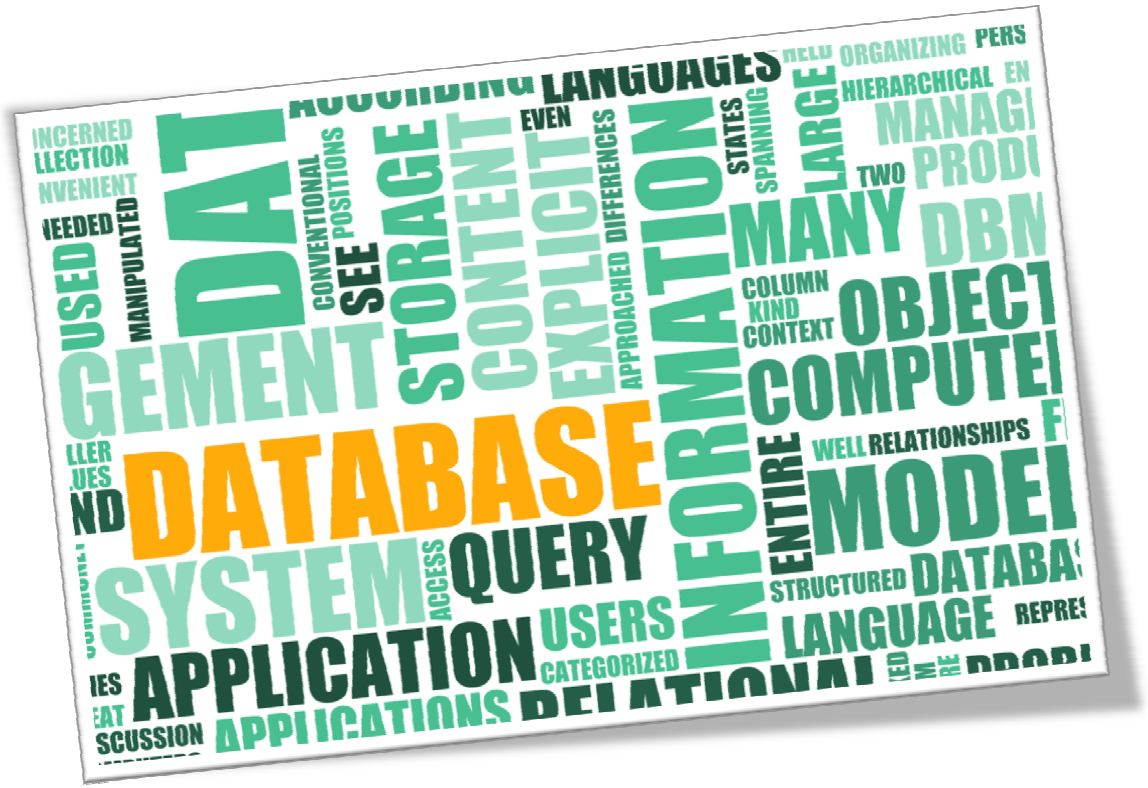
Materna GmbH, Dortmund


Können Wale fliegen oder
passen RDBMS ins Web?

Agenda

- REST und der Rest
  - Datenbanken und REST?
  - W3C trifft ISO/ANSI
  - Das REST-Options-Dreieck
  - Das Ressourcen-Dreieck
  - JPA-Zustände
- OpenJPA Vorstellung
- OpenJPA REST = JEST
- Beispiel
- Bewertung
- Fazit

Eine Ära geht zu Ende…

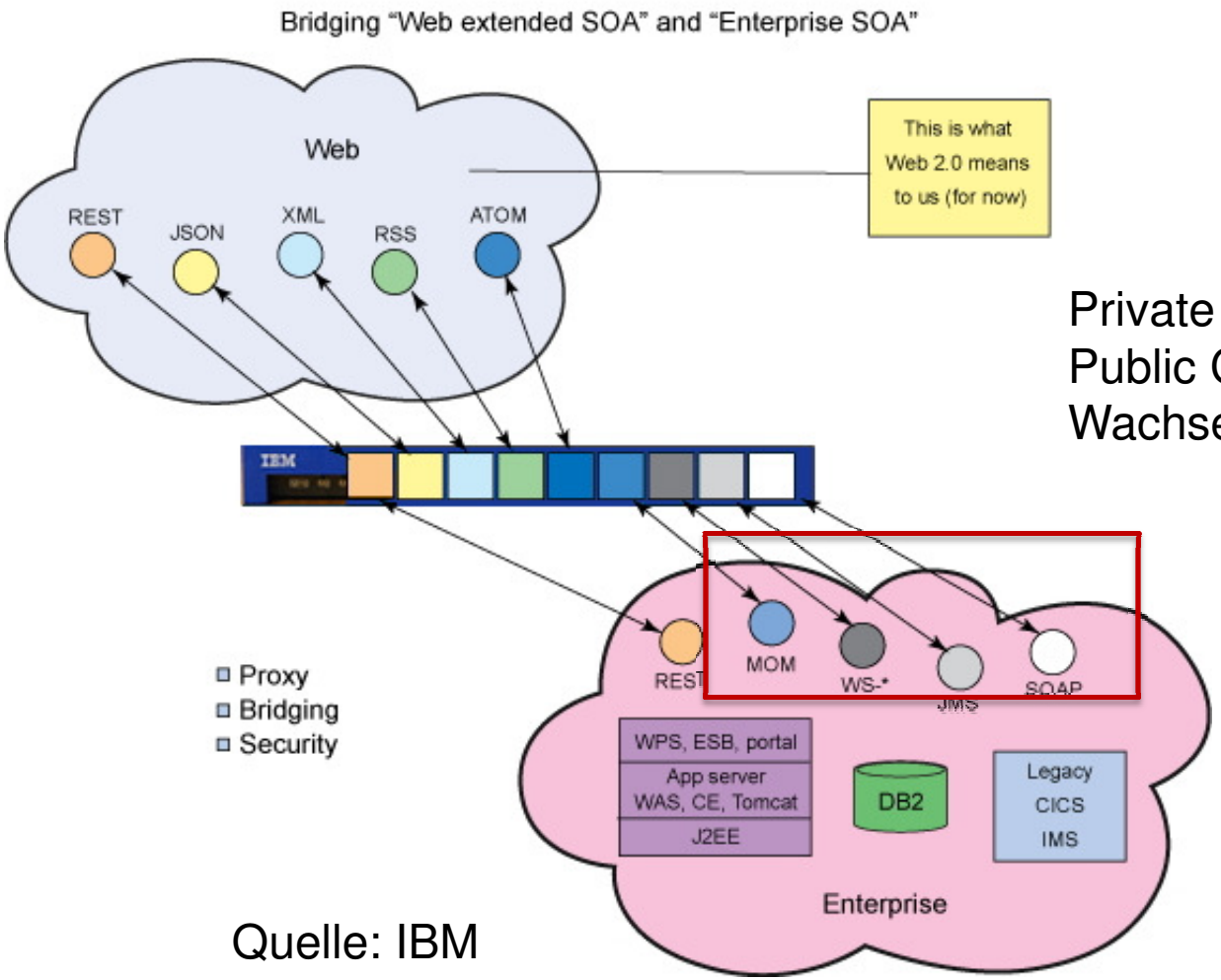


"The relational model of the 70s is not necessarily the answer"

(Stonebraker)

"The End of an Architectural Era", 2007, Michael Stonebraker et al.

Internet- und Unternehmensstandards: Wolke trifft Seife



Private und
Public Cloud
Wachsen zusammen

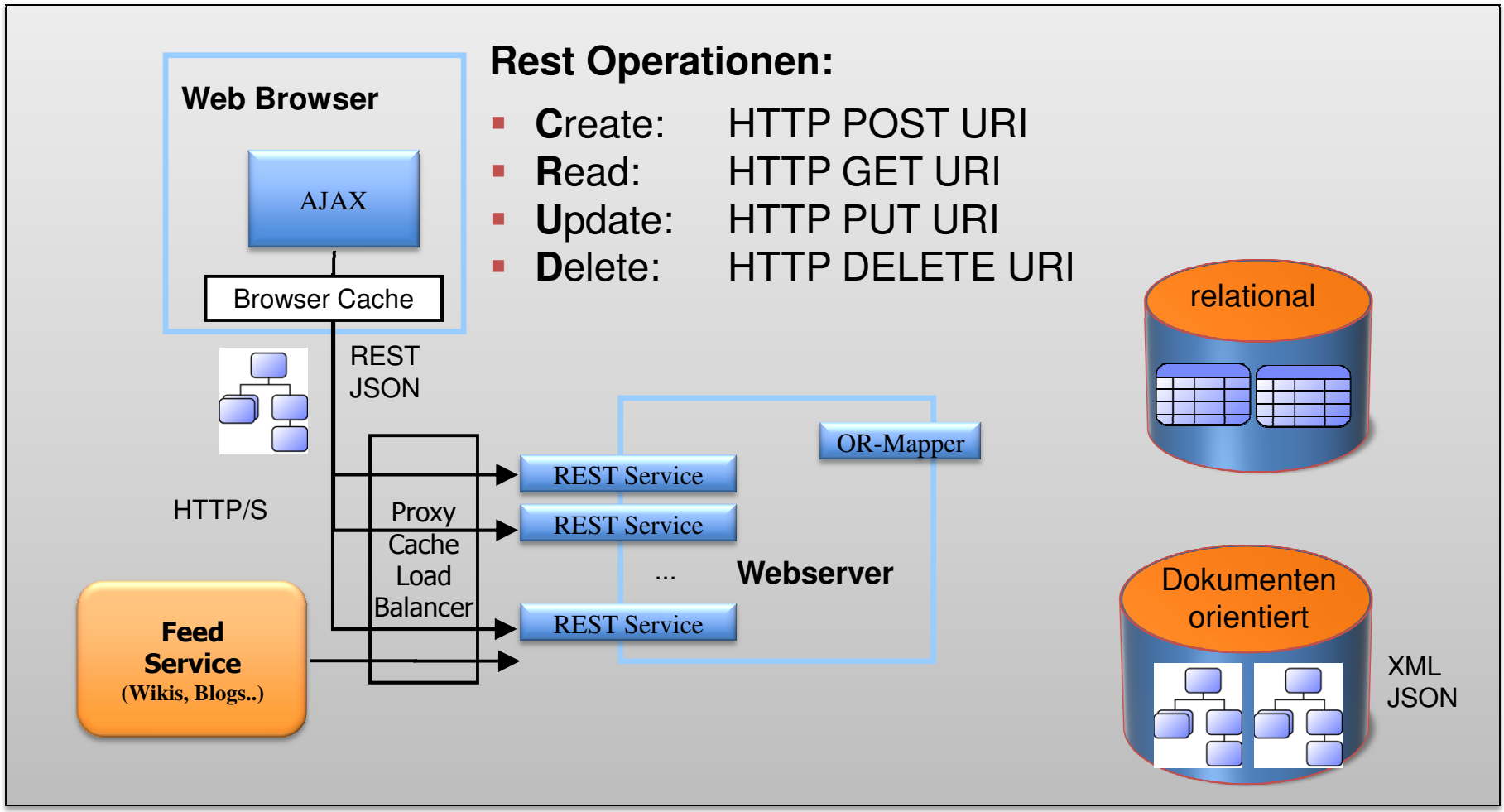
Quelle: IBM

REST – SQL: zwei Welten treffen aufeinander

REST
- W3C
- Zustandslos
- Ressourcenorientiert
- Einfache Syntax
- Standardbasiert
- Programmiersprachen unabhängig

SQL
- ISO/ANSI
- ACID
- Mengenbasiert
- Komplexe Syntax
- Standardbasiert, produktabhängig
- Client-Treiber-Installation nötig

Wie im Web 2.0 auf RDBMS zugreifen?



Datenbanken mit Java+REST/WS

- JSR-311: JAX-RS 1.1
  - Jersey
  - Restlet
  - RESTEasy
  - CXF, WINK
- Odata.org
- Datenbank
  - sqlREST 1.0
  - EclipseLink DBWS
  - Eclipse STP REST Support
  - OpenJPA JEST

Das Ressourcen-Dreieck

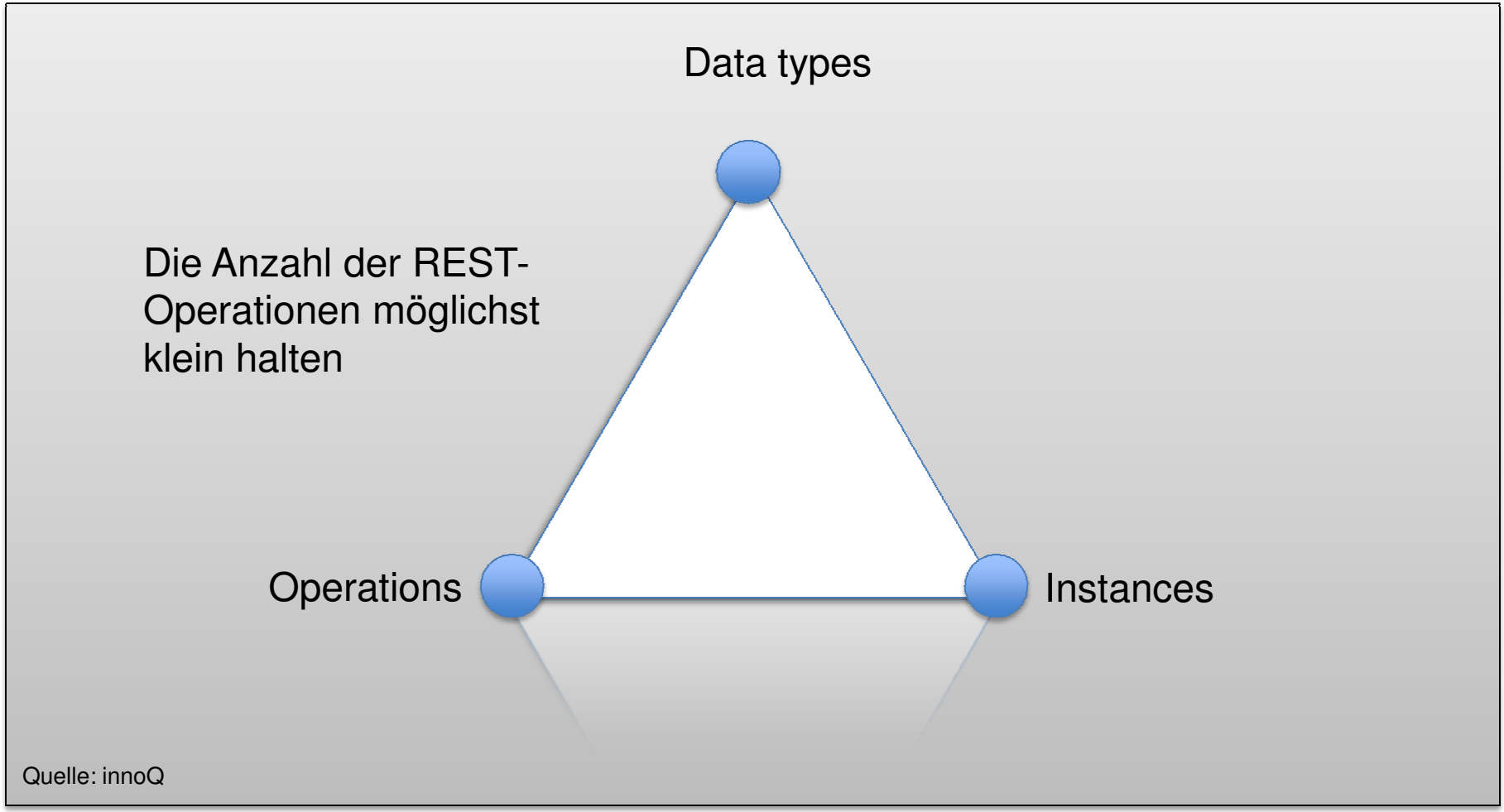
Nouns (Resources)
unconstrained
i.e., http://example.com/employees/12345

Verbs
constrained
i.e., *GET*

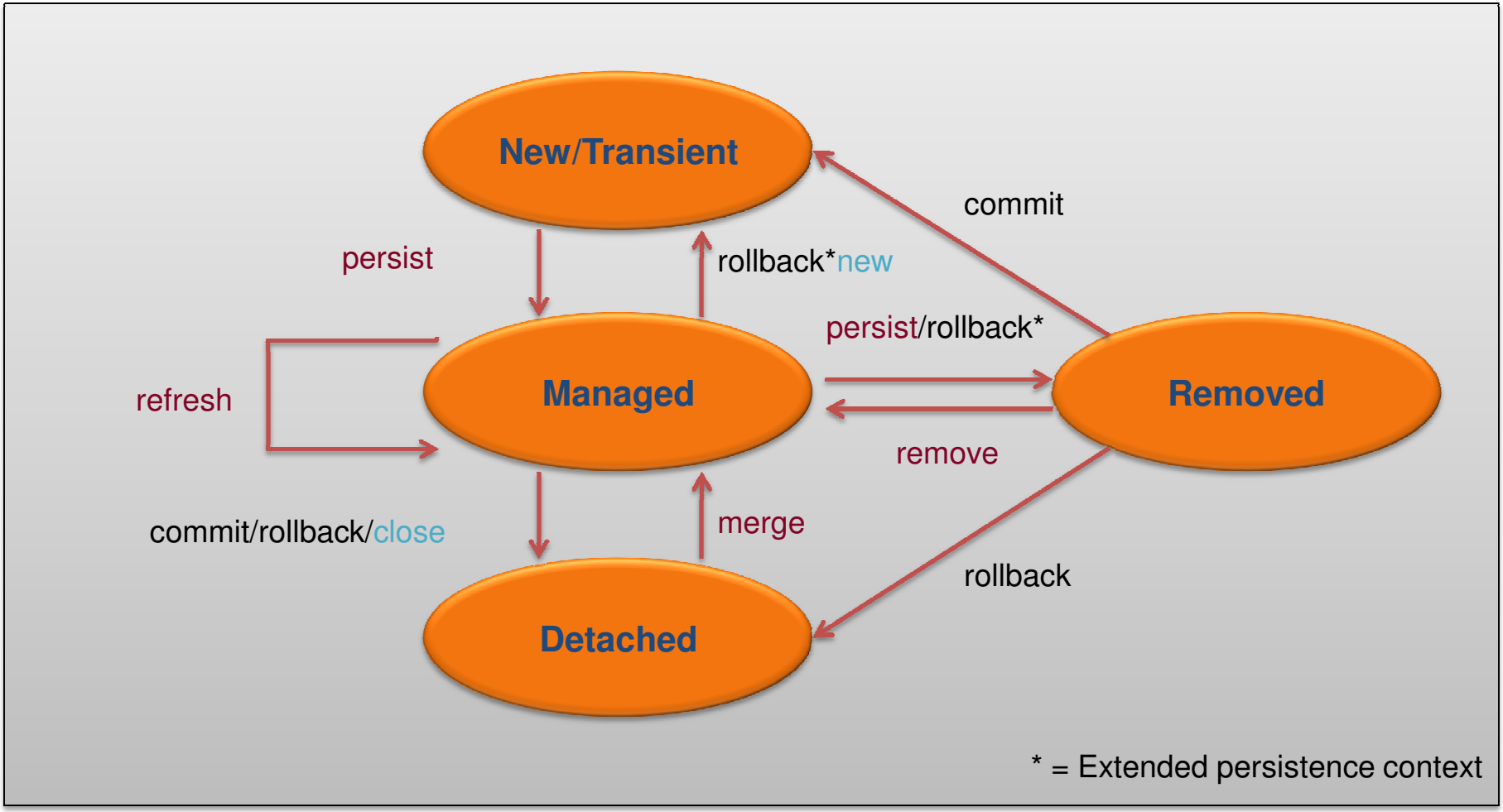
Representations
constrained
i.e., XML

Quelle: IBM

Das REST-Options-Dreieck



JPA-Zustandsübergänge



Agenda

- REST und der Rest
  - Datenbanken und REST?
  - W3C trifft ISO/ANSI
  - Das REST-Options-Dreieck
  - Das Ressourcen-Dreieck
  - JPA-Zustände
- OpenJPA Vorstellung
- OpenJPA REST = JEST
- Beispiel
- Bewertung
- Fazit

Apache OpenJPA

- 2006 BEA übergibt Kodo an Apache
- 2007 OpenJPA Apache-Top-Level-Projekt
- JSR-220 Java Persistence 1.0,JPA 1.0 TCK Releases
- Januar 2010 OpenJPA 1.0.4
- Mai 2008 OpenJPA 1.1.0
- Januar 2010 OpenJPA 1.2.2 → 1.2.3, 1.3

- JSR-317 Java Persistence 2.0, JPA 2.0 TCK Release
- August 2010 OpenJPA 2.0.1
- Februar 2011OpenJPA 2.1.0
- Juli 2011 OpenJPA 2.1.1 → 2.1.2, 2.2 (Zusatzfunktionen, JEST) (4,2 MB)
- Umfangreiches Handbuch (422 Seiten)

Wo verwendet?

ActiveMQ

Apache Camel

OpenEJB

bea

Apache ODE

WebSphere software

Apache Geronimo

EasyBeans

Spring Framework

OpenJPA Tools

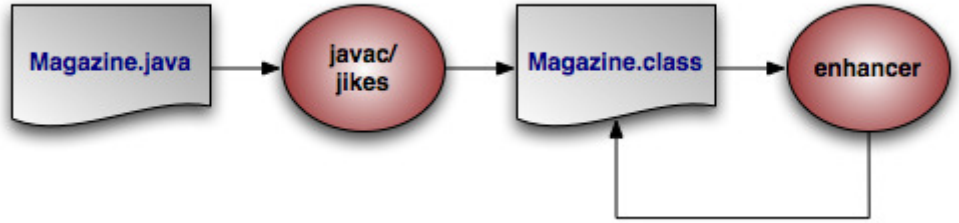
- Maven
- ANT
- OpenJPA Eclipse Tooling
- NetBeans
- jconsole –pluginpath

openjpa-tools-0.1.0-20101110.235301-21.jar

- EHCache 0.2 of L2 Plugin für OpenJPA

Apache OpenJPA Enhancer

Byte-Code Enhancing JPA Entities mit SERP



- Build Time (bis 2.0 aus Performance-Gründen empfohlen)
  - ANT: <taskdef name="openjpac" lassname="org.apache.openjpa.ant.PCEnhancerTask“>
  - Maven Plugin openjpa:enhance
- Deployment Time (muss vom JEE-Server unterstützt werden)
- Runtime (Ab JDK 6.0 dynamisch, default)
  - javaagent enhancer -javaagent:openjpa.jar
  - In persistence.xml deaktivieren
  - <property name="openjpa.DynamicEnhancementAgent" value="false"/>

OpenJPA Zusammenfassung

- Leichte Persistenzschicht für JSE, JEE, OSGi (6 MB)
- Hat sich in WebSphere, WebLogic, Geronimo u.a. bewährt
- Gute Primär-Dokumentation, jedoch kaum Sekundär-Dokumentation
- Beispiele vorhanden
- Werkzeuge, Monitoring, Cache vorhanden
- Byte-Code Enhancing am Besten zur Runtime
- JAVA 6! Wegen Annotationsverarbeitung

Agenda

- REST und der Rest
  - Datenbanken und REST?
  - W3C trifft ISO/ANSI
  - Das REST-Options-Dreieck
  - Das Ressourcen-Dreieck
  - JPA-Zustände
- OpenJPA Vorstellung
- OpenJPA REST = JEST
- Beispiel
- Bewertung
- Fazit

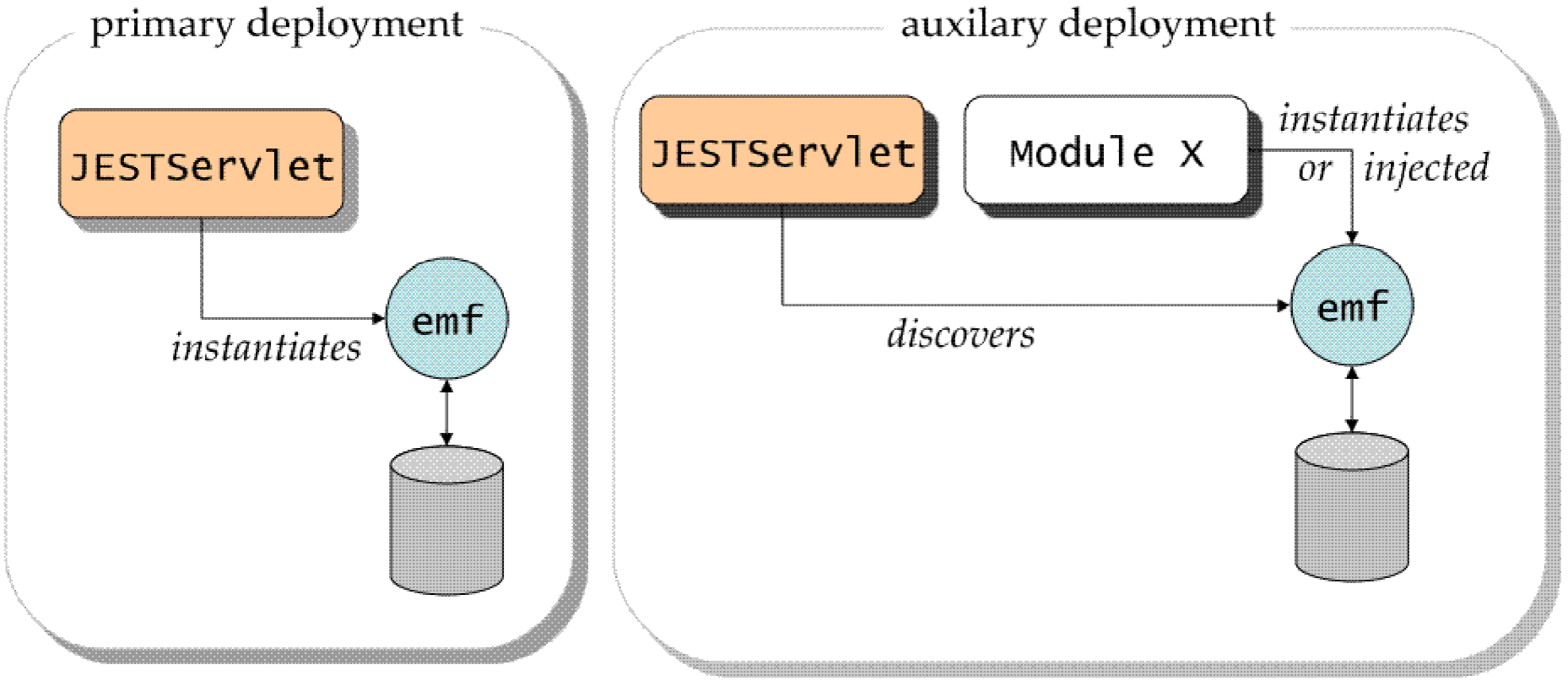
JEST: REST on OpenJPA

- REST (Representational State Transfer) ist
  - zustandslos
  - Ein Architekturstil, kein Protokoll, Standard
  - Programmiersprachen-, werkzeugneutral
  - Schemafrei
  - Ergebnisgraph
- JPA ist
  - transaktional, zustandsbehaftet
  - Schemabehaftet
  - Datenbankunabhängig
  - Flache Daten

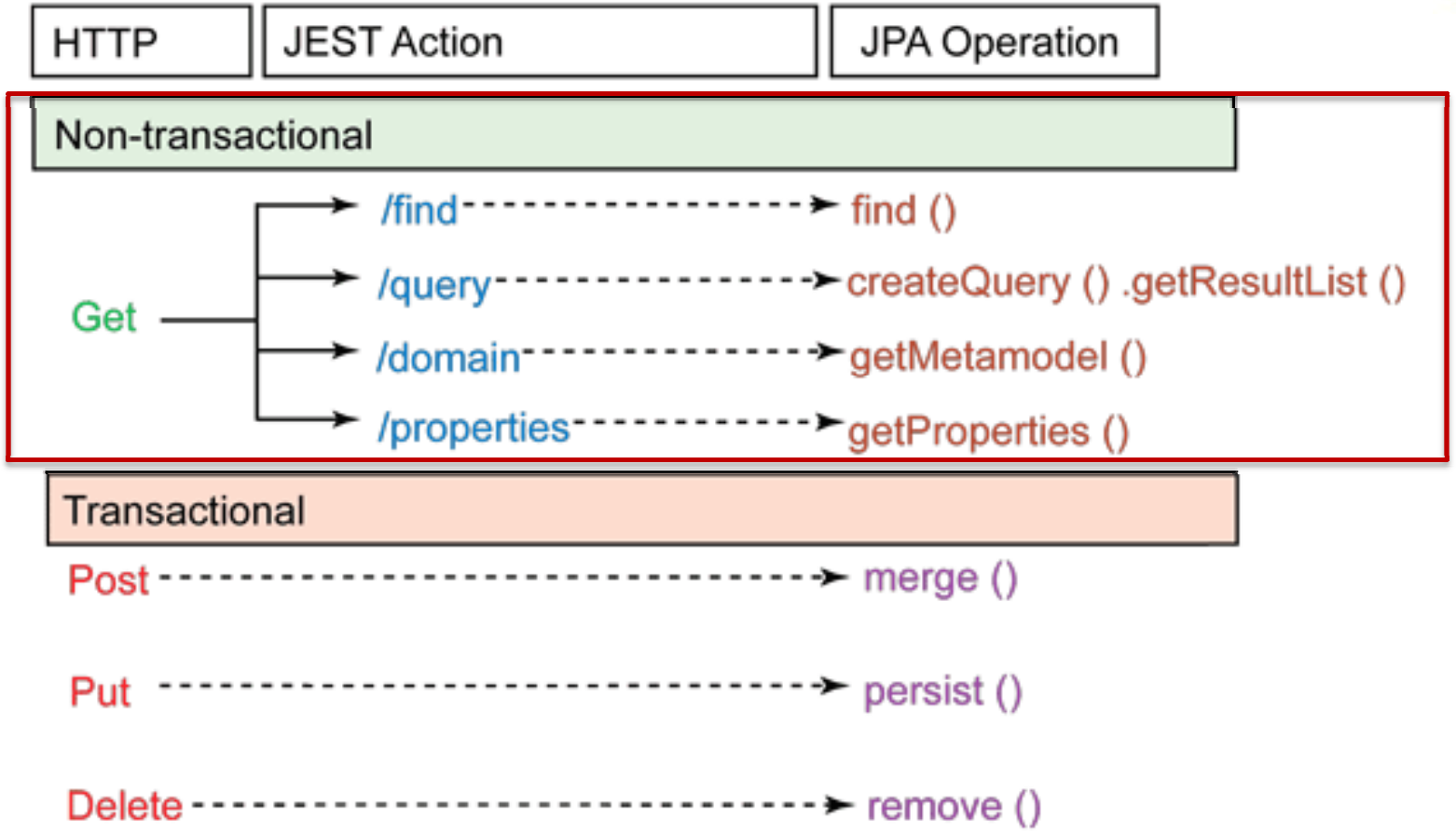
JEST: REST on OpenJPA

- Generische Erweiterung ohne zusätzliche Codeänderungen (Nicht-invasiv)
- Muss nur als JESTServlet mit deployt werden
- JEST unifies lesenden REST mit JPA
- JEST erweitert JSON/XML-Ausgaben (sprachneutral)
- Einfache und kostengünstige Möglichkeit, um programmiersprachenunabhängig auf Datenbankinhalte per JQL zuzugreifen
- Einfach Weboberfläche zur Erstellung der REST-URIs

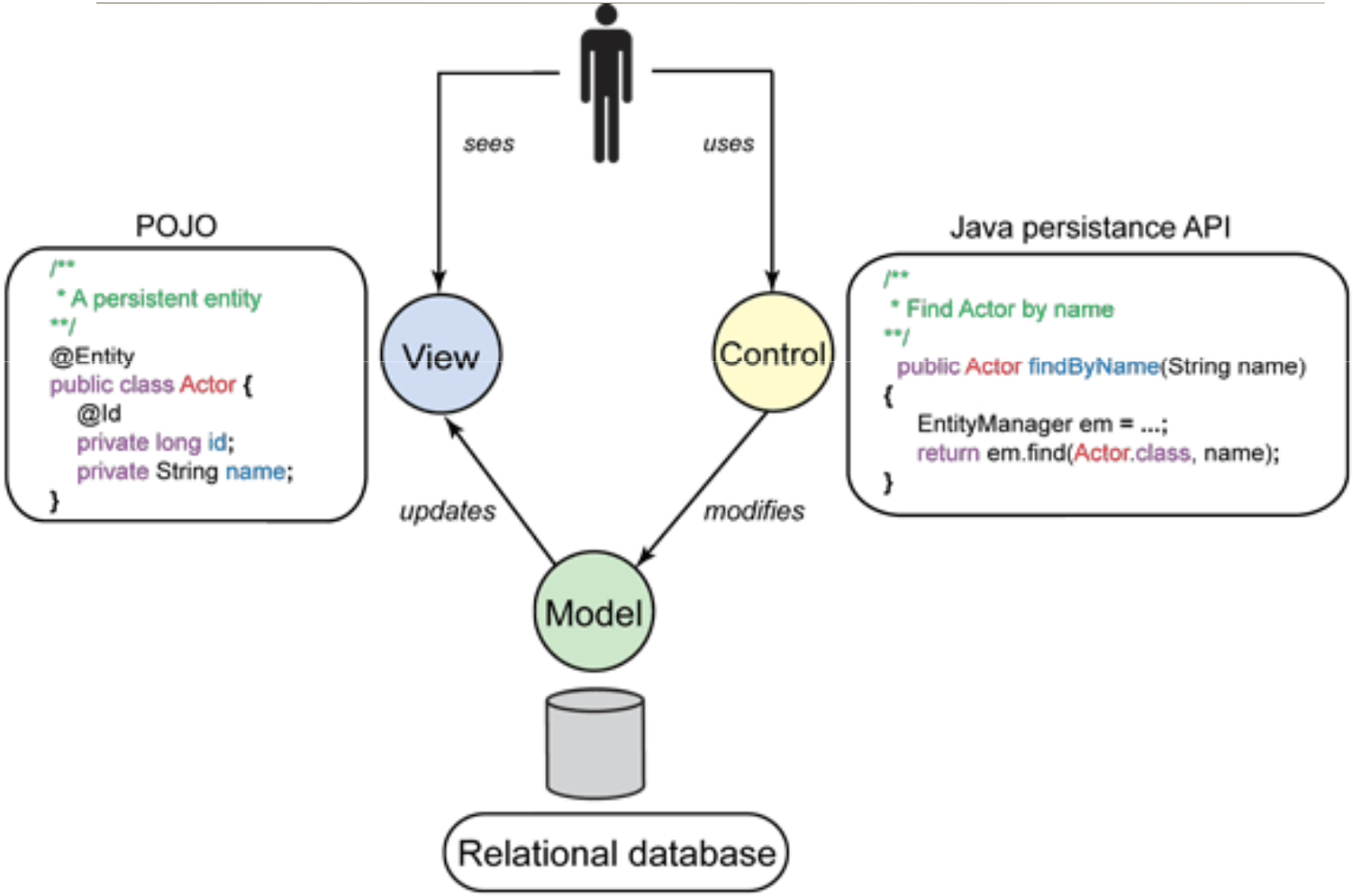
JEST-Nutzungsarten



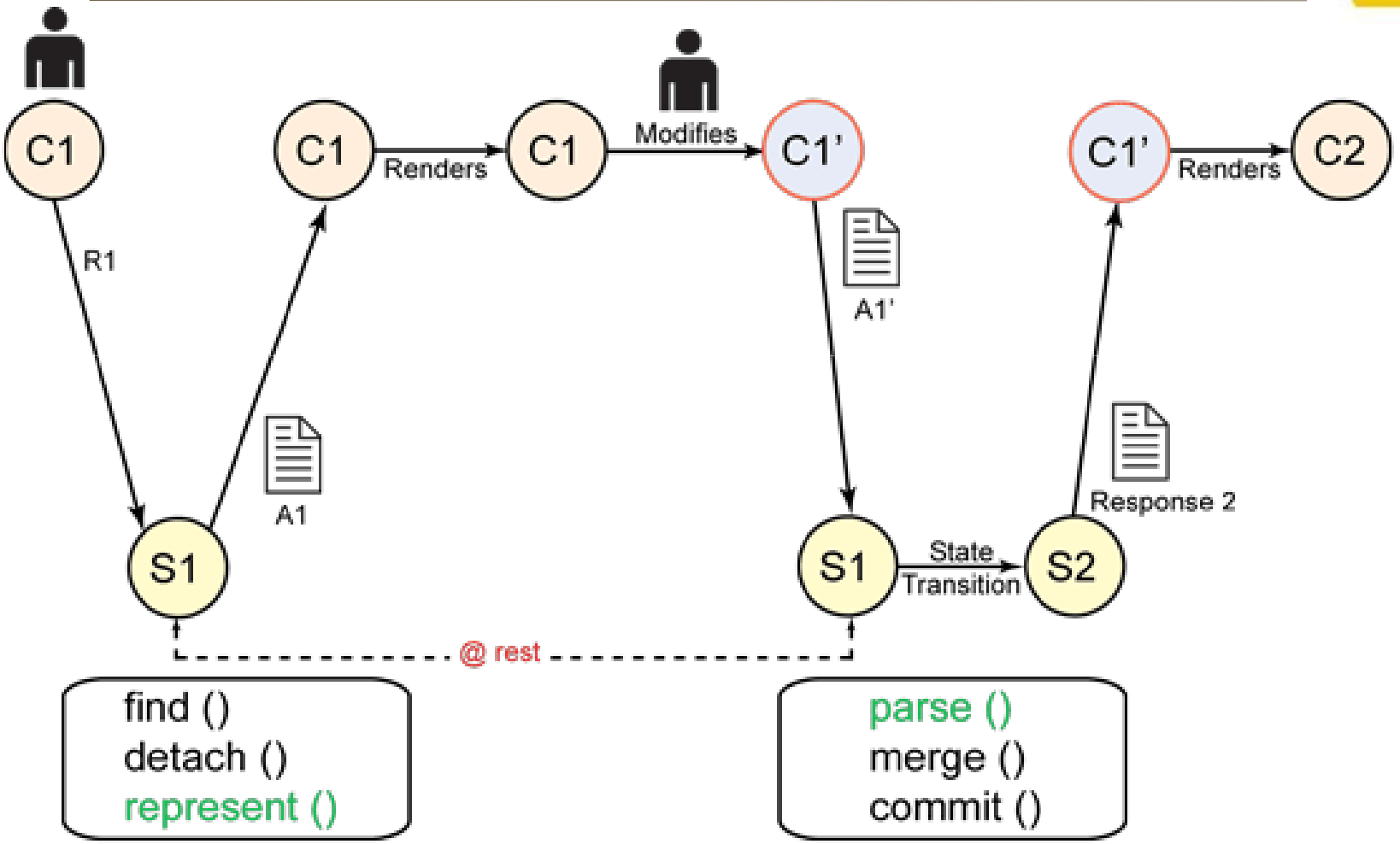
Mapping HTTP verbs to JPA persistence operations



JPA: Model-View-Controller für object persistence



Detached Transaktion um REST und JPA zu verbinden



JEST in action: Beispiel-Oberfläche

Info

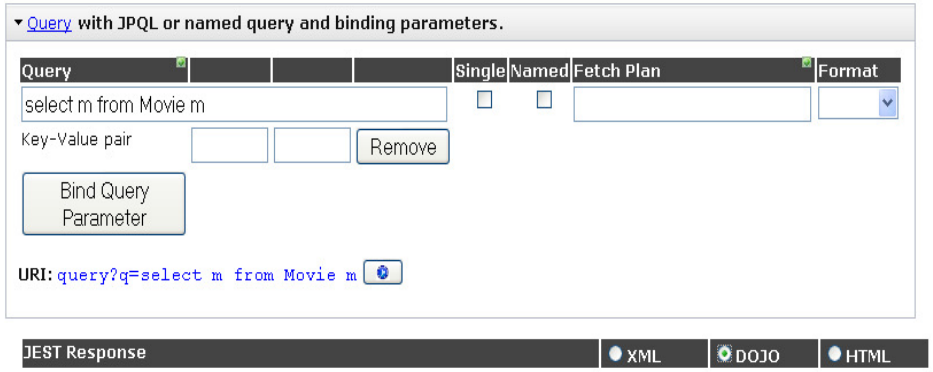
browse domain

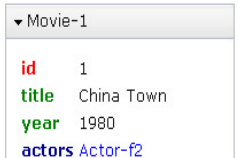
find instances

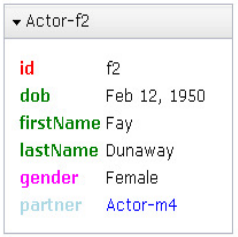
query objects

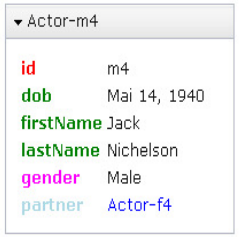
view properties

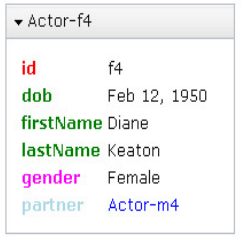
deploy

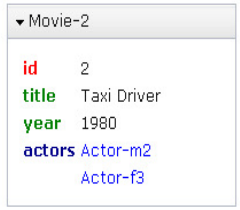


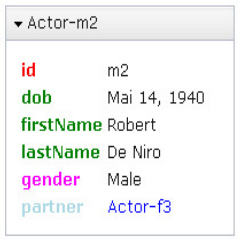


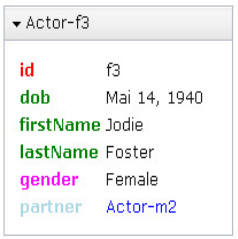


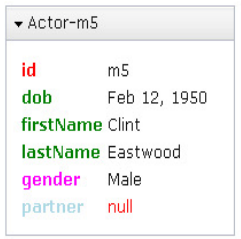












Wie JEST nutzen?

- svn checkout http://svn.apache.org/repos/asf/openjpa/trunk openjpa /openjpa-jest/
- mvn compile
- ls -lh openjpa-project/target/site/downloads/
- Oder https://repository.apache.org/snapshots/org/apache/openjpa/apache-openjpa/2.2.0-SNAPSHOT/
- http://openjpa.apache.org/builds/latest/docs/javadoc/index.html

Deployment Descriptor für JESTServlet mit Persistence-Unitname in web.xml

```
<servlet-name>jest</servlet-name>
<servlet-class>
org.apache.openjpa.persistence.jest.JESTServlet
</servlet-class>
    <init-param>
        <param-name>persistence.unit</param-name>
        <param-value>jestdemo</param-value>
    </init-param>
</servlet>
persistence.xml
<persistence-unit name="jestdemo">
```

JEST URI Syntax

Formal syntax of a JEST URI :

URI := http://{host}[:port]/{context}/{action}[/qualifier]* [?argument][&argument]*

- protocol is always http
- host (www.example.com) and optional port number locates the JEST servlet
- context := JEST servlet context root
- action := find|query|domain|properties
- zero or more qualifier := qualifier-key[=qualifier-value] argument := [argument-key=]argument-value qualifier-key := any valid Java identifier qualifier-value := string
- zero or more argument-key := string argument-value := string

JEST URI Syntax: Actions find, properties

Action: find: Returns the result of a find() operation.

- qualifier-key qualifier-value Comment format xml or json default is xml
- plan name of one or more fetch plan(s). Each name separated by comma character.
- argument-key argument-value Comment **type** entity name Fully-qualified Java class name or alias of the entity primary key value can be used for simple identity without the id property name

Beispiele:

Person-Instanz mit primary key 1234 suchen und im JSON-Format anzeigen : http://localhost:8080/jest/find/format=json?type=Person&1234

Fetch Group onlyBasicFields kann weggelassen werden /find/plan=onlyBasicFields?type=Person&1234

Person-Instanz mit Namen suchen und im XML-Format anzeigen : e/find?type=Person&firstName=John&lastName=Doe

Action : properties: Returns the configuration properties in HTML format.

- Accepts no qualifier.
- Accepts no argument.

JEST URI Syntax: Actions query, domain

Action : query: Returns the result of a Query.getResultList() or Query.getSingleResult() operation.

- qualifier-key qualifier-value Comment format xml or json default is xml plan name of one or more fetch plan(s). Each name separated by comma character.

  single enforces single instance as query result e.g. /query/single?q=select p from Person p where p.name=:x&x=John named interprets the q argument

  value as a named query e.g. /query/named?q=PersonByName&x=John where PersonByName is named query with x its named parameter

- argument-key argument-value Comment **q** JPQL or Named Query e.g. /query/named?q=AllPerson or /query?q=select p from Person p e.g.

  /query?q=select p from Person p where p.firstName=:x&x=John bind parameter parameter value the values are converted to match the target type e.g.

  /query?q=select p from Person p where p.gender=:g&g=MALE

Action : domain: Returns the domain model in XML format.

- Accepts no qualifier.

  Accepts no argument.

Package org.apache.openjpa.persistence.jest

Package org.apache.openjpa.persistence.jest

Interface Summary	
Constants	Static String constants
JESTCommand	Interface for JEST commands.
JPAServletContext	An operating context provides a `persistence context` and utility functions within which all JEST commands execute.
JSON	A generic interface for a JSON encoded instance.
ObjectFormatter<T>	A parameterized interface defines the protocol for converting `managed` persistence instances or a persistent `domain model` into a form suitable for transport to

Class Summary	
AbstractCommand	The abstract base class for all commands available to JEST.
Classes	Computes closure of a collection of persistent objects
DomainCommand	Marshals a JPA meta model in the configured format to the response output stream.
ExceptionFormatter	Formats error stack trace
FindCommand	
IOR	String reference of a managed object.
JESTContext	An operational context combines a `persistence context` and a HTTP execution context expressed as a `request` and `response`.
JESTServlet	A specialized HTTP servlet to interpret HTTP requests as Java Persistent API commands on a running persistence unit.
JSONObject	A JSON instance for persistence.
JSONObject.Array	An array of objects.
JSONObject.KVMap	A map whose key or value can be JSON.
JSONObjectFormatter	Marshals a root instance and its persistent closure as JSON object.
MetamodelHelper	
MetamodelHelper.AttributeComparator	Compares attribute by their category and within the same category by name.
PropertiesCommand	Represents configuration properties in HTML.
PropertiesFormatter	Formats a key-value pair in a HTML Document.
PrototypeFactory<K,T>	A factory for a specific type of objects registered by a key.
QueryCommand	Executes query.
TokenReplacedStream	Reads from an input stream and writes to an output stream after replacing matched tokens by their counterpart.
TokenReplacedStream.Pattern	
XMLFormatter	Marshals a root instance and its persistent closure as an XML element.

Enum Summary	
JESTCommand.Format	Supported format monikers.
MetamodelHelper.AttributeCategory	Attribute Category makes a finer distinction over PersistentAttributeType declared in `Attribute.PersistentAttributeType` such as id, version, lob or enum.

Agenda

- REST und der Rest
  - Datenbanken und REST?
  - W3C trifft ISO/ANSI
  - Das REST-Options-Dreieck
  - Das Ressourcen-Dreieck
  - JPA-Zustände
- OpenJPA Vorstellung
- OpenJPA REST = JEST
- Beispiel
- Bewertung
- Fazit

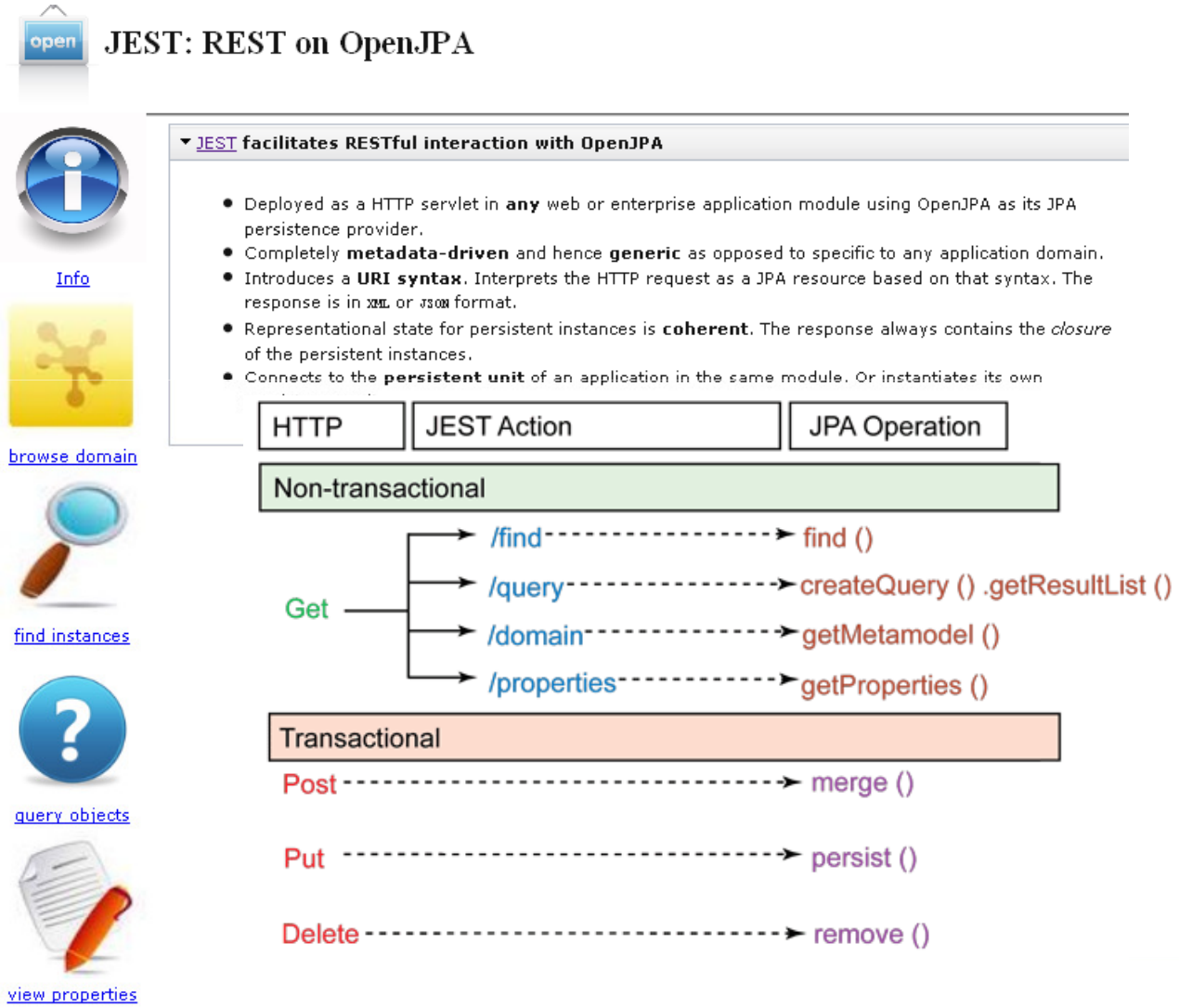
JEST-Beispiel erstellen

- svn co http://svn.apache.org/repos/asf/openjpa/trunk/openjpa-examples/jest
- META-INF/persistence.xml Datenbank-Verbindung konfigurieren
- build.properties, build.xml anpassen: openjpa-all-2.2.0-SNAPSHOT.jar
- SimpleApp: props.put("openjpa.EntityManagerFactoryPool", "true");

<pathelement location="${openjpa.dir}/lib/geronimo-jpa_2.0_spec-1.1.jar" />

- ant war
- demo.war deployen
- http://localhost:8080/demo/jtest aufrufen
- http://localhost:8080/demo/jest/properties/
- http://localhost:8080/demo/jest/query/named?q=AllPerson
- http://localhost:8080/demo/jest/find/format=json?type=Person&1234
- http://localhost:8080/demo/jest/find?type=Person&1234
- http://localhost:8080/demo/jest/find/plan=onlyBasicFields?type=Person&1234
- http://localhost:8080/demo/jest//query?q=select from Person p

JEST-Servlet-Anwendung



JEST Persistenz-Unit-Einstellungen

Info

browse domain

find instances

query objects

view properties

▾ View **configuration properties of the persistence unit.**

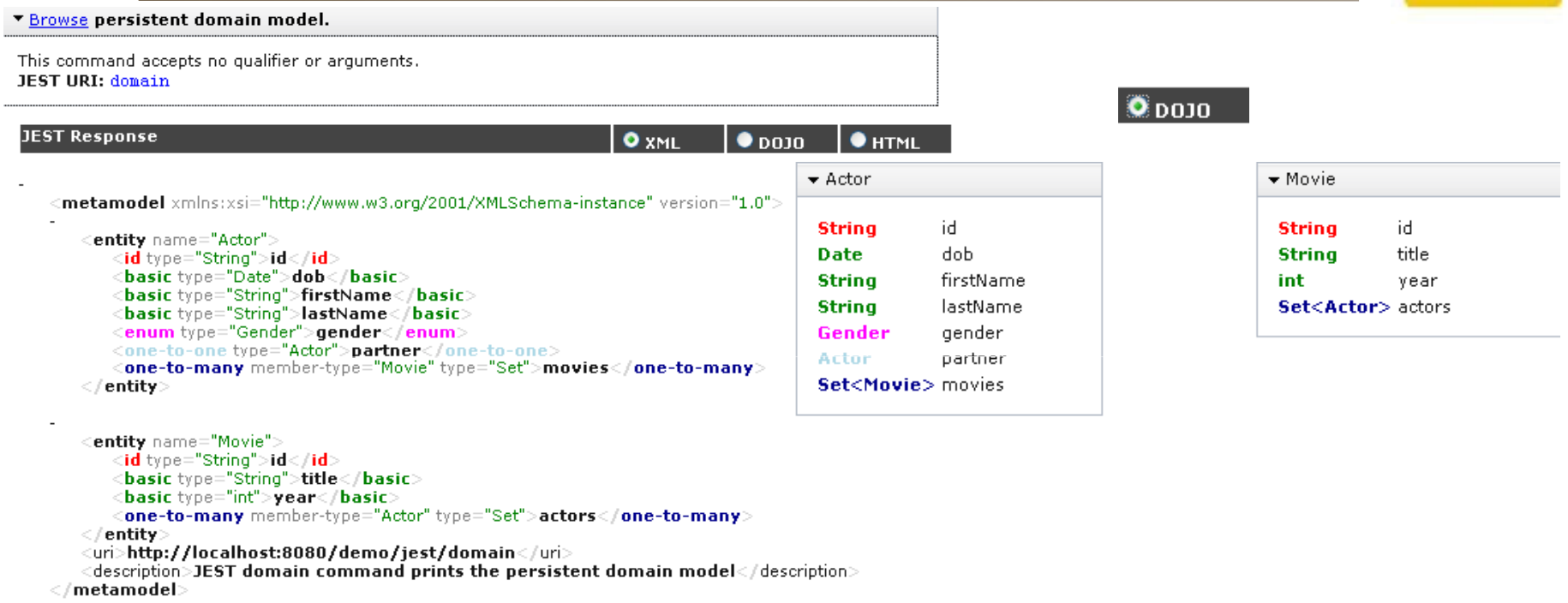
This command accepts no qualifier or arguments
URI: properties

http://localhost:8080/demo/jest/properties

JEST Response	XML	HTML

Configuration Properties	
javax.persistence.jdbc.driver	org.apache.derby.jdbc.EmbeddedDriver
javax.persistence.jdbc.password	******
javax.persistence.jdbc.url	jdbc:derby:jest;create=true
javax.persistence.jdbc.user	app
javax.persistence.lock.timeout	0
javax.persistence.query.timeout	0
javax.persistence.sharedCache.mode	UNSPECIFIED
javax.persistence.validation.group.pre-persist	javax.validation.groups.Default
javax.persistence.validation.group.pre-update	javax.validation.groups.Default
openjpa.AutoClear	DATASTORE
openjpa.AutoDetach	[CLOSE, ROLLBACK]
openjpa.BrokerFactory	jdbc
openjpa.BrokerImpl	default
openjpa.CacheDistributionPolicy	default
openjpa.CacheFinderQuery	true
openjpa.CachePreparedQuery	true
openjpa.Callbacks	default
openjpa.ClassResolver	default
openjpa.Compatibility	default
openjpa.ConnectionFactoryMode	false
openjpa.ConnectionFactoryProperties	PrintParameters=true

JEST-Domänen-Objekte anzeigen



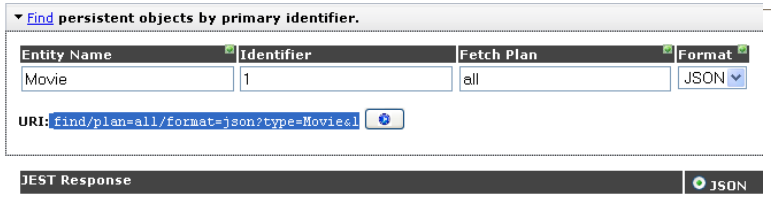
http://localhost:8080/demo/jest/domain

Ein Schauspieler kann in mehreren Filmen mitwirken

Eine Instanz mit REST abfragen und im JSON-Format anzeigen

http://localhost:8080/demo/jest/find/format=json?type=Movie&1

Eine Instanz und alle Beziehungen mit REST abfragen



http://localhost:8080/demo/jest/find/plan=all/format=json?type=Movie&1


```
[
{
    "$id": "Movie-1",
    "id": "1",
    "title": "China Town",
    "year": 1980,
    "actors": [
        {
            "$id": "Actor-m4",
            "id": "m4",
            "dob": "Tue May 14 00:00:00 CEST 1940",
            "firstName": "Jack",
            "lastName": "Nichelson",
            "gender": "Male",
            "partner": {
                "$id": "Actor-f4",
                "id": "f4",
                "dob": "Sun Feb 12 00:00:00 CET 1950",
                "firstName": "Diane",
                "lastName": "Keaton",
                "gender": "Female",
                "movies": [
                    {
                        "$id": "Movie-4",
                        "id": "4",
                        "title": "Godfather",
                        "year": 1980,
                        "actors": [
                            {
                                "$id": "Actor-m3",
                                "id": "m3",
                                "dob": "Sun Feb 12 00:00:00 CET 1950",
                                "firstName": "Al",
                                "lastName": "Pacino",
                                "gender": "Male",
                                "movies": [
                                    {
                                        "$ref": "Movie-4"
                                    }
                                ]
                            },
                            {
                                "$ref": "Actor-m4"
                            },
```

Eine Instanz mit REST abfragen und im XML-Format anzeigen

Find **persistent objects by primary identifier.**

Entity Name	Identifier	Fetch Plan	Format
Actor	m2		XML

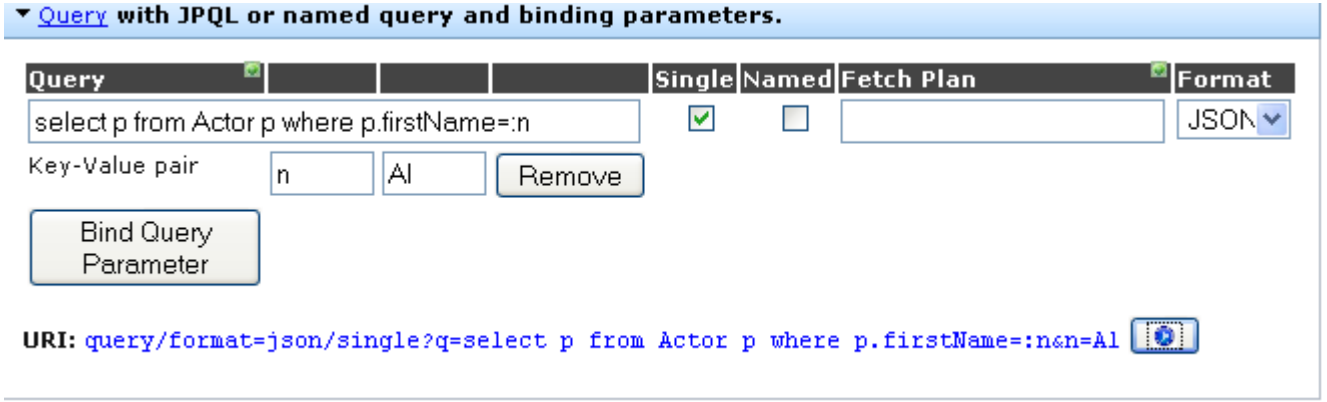
URI: find/format=xml?type=Actor&m2

JEST Response — XML | DOJO | HTML

```
-
<instances xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">
    <uri>http://localhost:8080/demo/jest/find/format=xml?type=Actor&
    m2&dojo.preventCache=1311326131460</uri>
    <description>JEST find command may return more than one result. Why? </description>
  -
    <instance id="Actor-m2">
        <id name="id" type="String">m2</id>
        <basic name="dob" type="Date">Mai 14, 1940</basic>
        <basic name="firstName" type="String">Robert</basic>
        <basic name="lastName" type="String">De Niro</basic>
        <enum name="gender" type="Gender">Male</enum>
      -
        <one-to-one name="partner" type="Actor">
            <ref id="Actor-f3"></ref>
        </one-to-one>
    </instance>
  -
    <instance id="Actor-f3">
        <id name="id" type="String">f3</id>
        <basic name="dob" type="Date">Mai 14, 1940</basic>
        <basic name="firstName" type="String">Jodie</basic>
        <basic name="lastName" type="String">Foster</basic>
        <enum name="gender" type="Gender">Female</enum>
    </instance>
</instances>
```

http://localhost:8080/demo/jest/find/format=xml?type=Actor&m2

Eine Instanz mit JPQL abfragen und im JSON-Format anzeigen

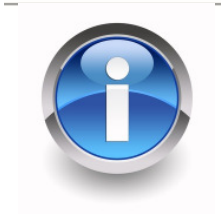



```
[
    {
        "$id": "Actor-m3",
        "id": "m3",
        "dob": "Sun Feb 12 00:00:00 CET 1950",
        "firstName": "Al",
        "lastName": "Pacino",
        "gender": "Male",
        "partner": null
    }
]
```

http://localhost:8080/demo/jest/query/single?q=select%20m%20from%20Movie%20m%20where%20m.title%20like%20:title&title=%25China%25

Agenda

- REST und der Rest
  - Datenbanken und REST?
  - W3C trifft ISO/ANSI
  - Das REST-Options-Dreieck
  - Das Ressourcen-Dreieck
  - JPA-Zustände
- OpenJPA Vorstellung
- OpenJPA REST = JEST
- Beispiel
- Bewertung
- Fazit



Lessons learned

- Openjpa. AnnotationProcessor benötigt Java 6!
- openjpa-all-2.2.0-SNAPSHOT.jar und das JESTServlet muss mit der Webanwendung deployt werden
- persistence unit mit
  openjpa.EntityManagerFactoryPool=true nicht in der persistence.xml konfigurieren
- **Google Libraries API** (content distribution network for the most popular, open-source JavaScript libraries.)
  - Die verwendete DOJO-Bibliotheken werden über das Netz geladen
- Datenbank mit http://localhost:8080/demo initial anlegen

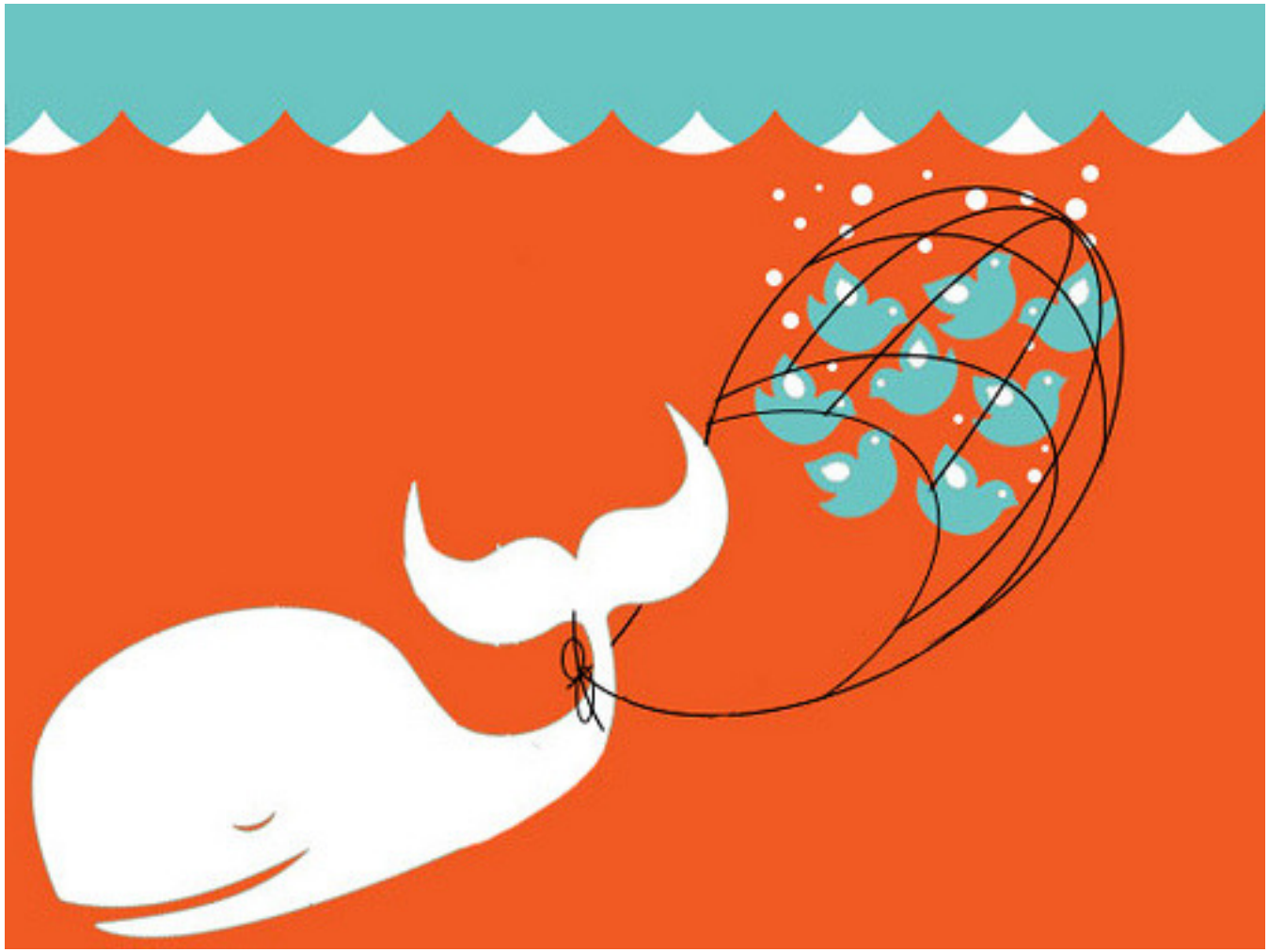
Fazit

- REST und SQL passen zusammen
- JPA und Zustandslosigkeit passen zusammen
- OpenJPA ist leicht gewichtigt
- Universeller Ansatz für REST- und CRUD-DB-Operationen
- JEST passt zum REST
- Bisher nur lesender Zugriff
- Noch keine finale Version

Quellen:

- Apache OpenJPA User's Guide
- http://openjpa.apache.org/jest-usage.html
- JEST: REST on OpenJPA, Pinaki Poddar, Februar 2011: http://www.ibm.com/developerworks/java/library/j-jest/index.html
- JPA Diagram Editor - http://download.eclipse.org/webtools/incubator/repository/jpaeditor/helios/
- Pinaki Poddar talks persistence: strategies, opinions, and his current interest, JEST (April 2011): http://www.ibm.com/developerworks/podcast/ag/gloverseries-drpinakipoddar.mp3

Wale können nicht fliegen, aber REST passt zu Datenbanken…JPA 4.2



5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Frank Pientka

Materna GmbH, Dortmund