

ORACLE®

Polyglot Persistence - From NoSQL to HTML5

Doug Clarke

Director of Product Management

Oracle Cloud Application Foundation



New York 2013

Tutorials: June 10-11
Conference: June 12-14



QCon

INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE

www.qconnewyork.com

About Me

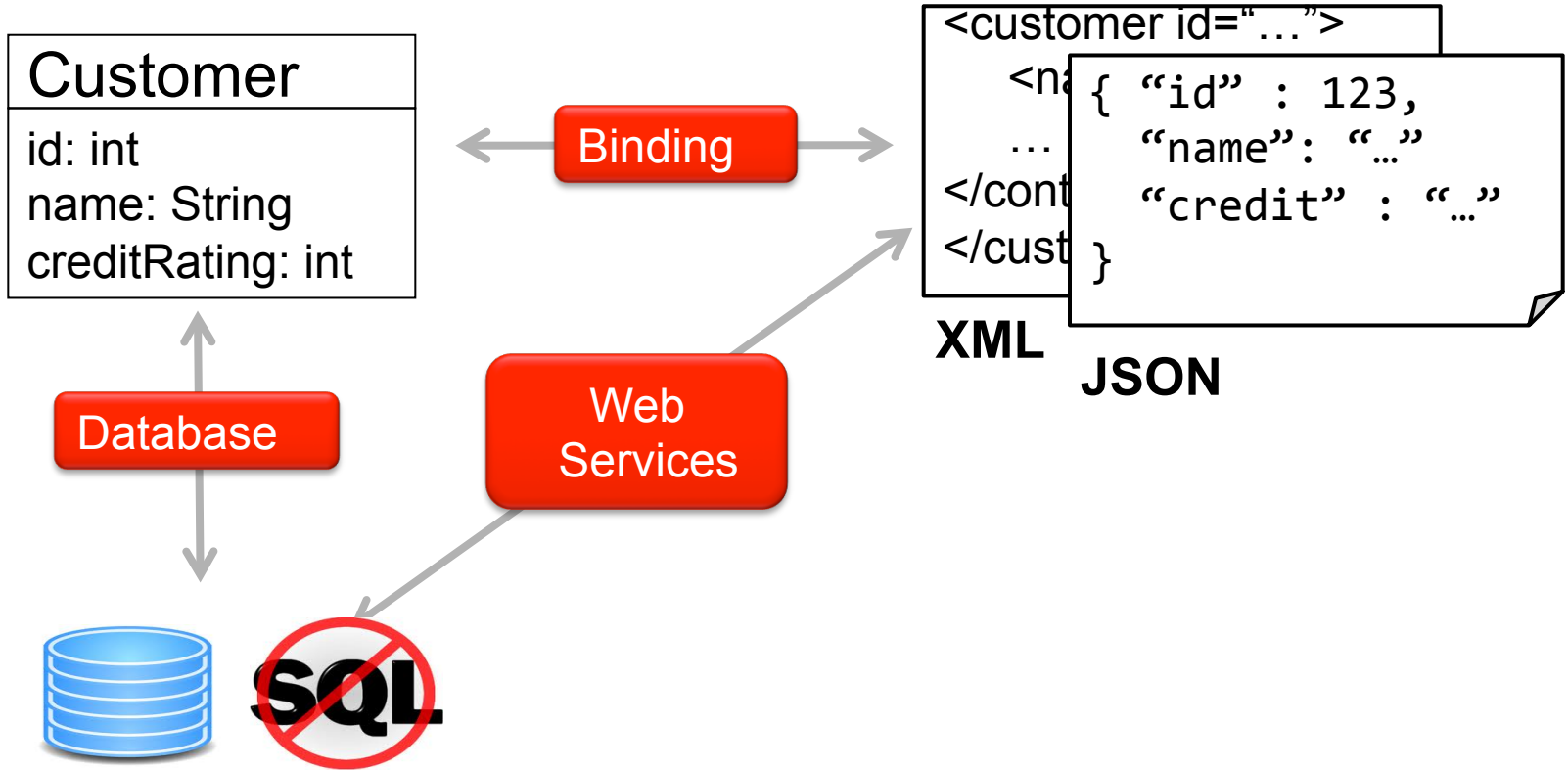
Doug Clarke – douglas.clarke@oracle.com

- EclipseLink Project co-lead
- Director of Product Management
- Still developing
 - Persistence, Scripting, Coherence



ORACLE

Core Java Persistence Challenges

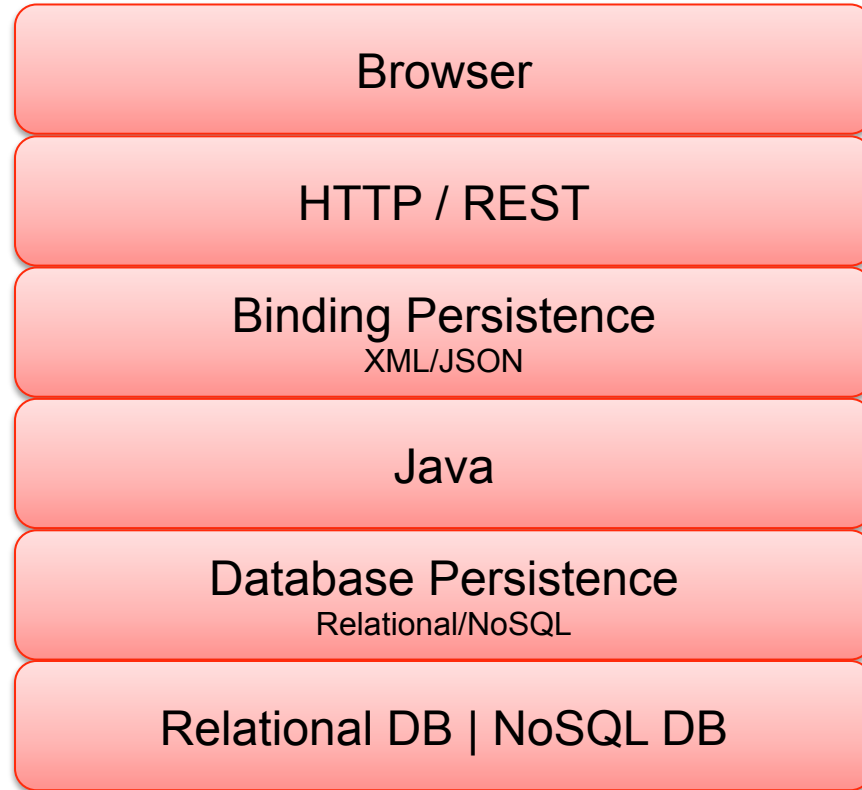


EclipseLink Project

www.eclipse.org/eclipselink

- Object-Relational: Java Persistence API (JPA)
 - JPA 1.0 part of EJB 3.0 standard (JSR 220)
 - JPA 2.0 standardized in JSR 317
 - EclipseLink is *JPA 2.0 & 2.1 Reference Implementation*
- Object-XML: Java Architecture for XML Binding (JAXB)
 - *JAXB 2.2 Certified Implementation*
- Object-XML: Service Data Objects
 - SDO 2.1.1 standardized in JSR 235
 - EclipseLink is *SDO 2.1.1 Reference Implementation*

Agenda—Database to Browser (and back again)



Agenda

Browser

HTTP / REST

Binding Persistence
XML/JSON

Java

Database Persistence
Relational/NoSQL

Relational DB | NoSQL DB

NoSQL Databases

- EclipseLink is best known for relational persistence but...
- NoSQL databases are increasingly popular
- No common definition (document, graph, columnar)
 - Differing feature sets
 - Some offer query language/API—some not
- No standards
- Every database offers a unique API
 - Cost in terms of learning each database
 - Zero portability across databases

EclipseLink NoSQL

- Support JPA-style access to NoSQL databases
 - Leverage non-relational database support for JCA (and JDBC when available)
- Define annotations and XML to identify NoSQL stored entities (e.g., @NoSQL)
- Support JPQL subset for each
 - Key principal: leverage what's available
- Initial support for MongoDB and Oracle NoSQL

Applicability of JPA to NoSQL

- Core JPA concepts apply to NoSQL:
 - Persistent Entities, Embeddables, ElementCollection, OneToOne, OneToMany, ManyToOne, Version, etc.
- Some concepts apply with some databases:
 - JPQL, NamedNativeQuery
- Pure relational concepts don't apply:
 - CollectionTable, Column, SecondaryTable, SequenceGenerator, TableGenerator, etc.

Querying NoSQL with JPA

- Two kinds of queries
 - JQPL—portable query language defined by the spec
 - Native query—lets you leverage database specific features
 - Dynamic or static `@NamedQuery`
- JPQL translated to underlying database query framework.

Example JPA Mapped Entity

```
@Entity
```

```
public class Order {  
    @Id  
    @GeneratedValue  
  
    private String id;  
    @Basic  
    private String description;  
    @OneToOne(cascade={CascadeType.REMOVE, CascadeType.PERSIST})  
    private Discount discount;  
    @ElementCollection  
    private List<OrderLine> orderLines = new ArrayList<OrderLine>();  
}
```

Example MongoDB Mapped Entity

```
@Entity
@NoSql(dataFormat=DataFormatType.MAPPED)
public class Order {
    @Id // Use generated OID (UUID) from Mongo.
    @GeneratedValue
    @Field(name="_id")
    private String id;
    @Basic
    private String description;
    @OneToOne(cascade={CascadeType.REMOVE, CascadeType.PERSIST})
    private Discount discount;
    @ElementCollection
    private List<OrderLine> orderLines = new ArrayList<OrderLine>();
}
```

MongoDB Query Examples

■ JPQL

```
Select o from Order o  
  where o.totalCost > 1000
```

```
Select o from Order o  
  join o.orderLines l where l.cost > :cost
```

■ Native Queries

```
query = em.createNativeQuery(  
    "db.ORDER.findOne({'_id\':" +  
    oid + "\'})", Order.class);  
Order order =  
    (Order) query.getSingleResult();
```

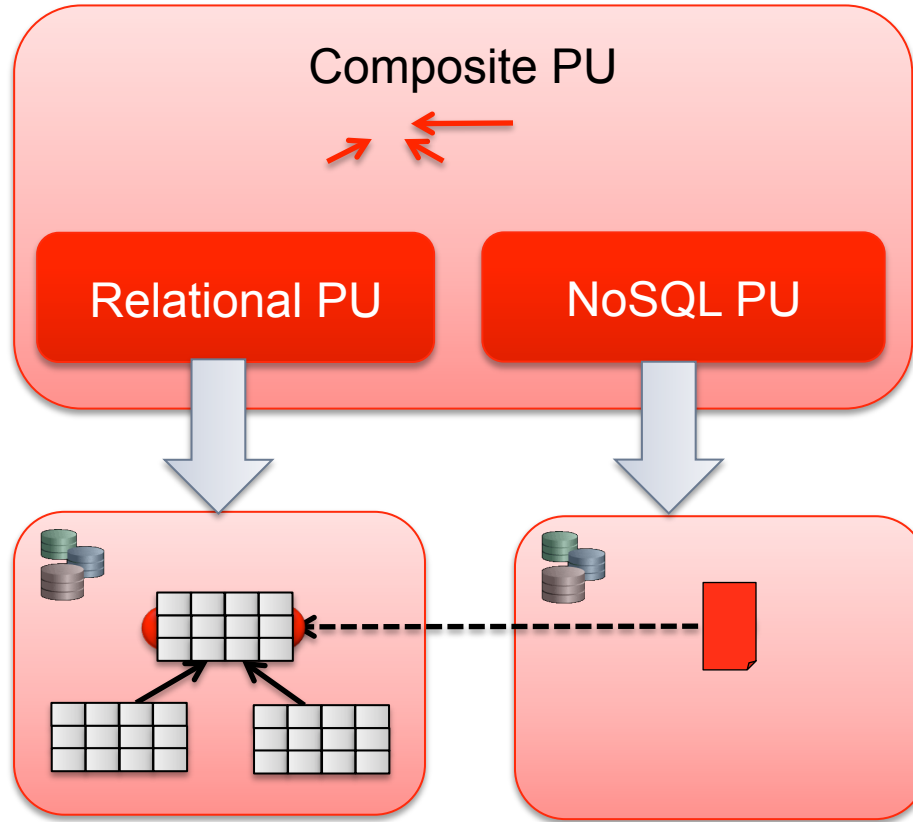
Demo EclipseLink NoSQL

“...we are gearing up for a shift to *polyglot persistence* - where any decent sized enterprise will have a variety of different data storage technologies for different kinds of data...we'll be first asking how we want to manipulate the data and only then figuring out what technology is the best bet for it..”

Martin Fowler

ThoughtWorks

Composite Persistence Unit





Demo Polyglot Persistence

<http://git.eclipse.org/gitroot/eclipselink/examples.git/jpa/polyglot>

Agenda

Browser

HTTP / REST

Binding Persistence
XML/JSON

Java

Database Persistence
Relational/NoSQL

Relational DB | NoSQL DB

Binding Persistence


XML and JSON Binding

- EclipseLink implements
 - JAXB for Java/XML binding—convert Java to/from XML
 - Java/JSON binding—convert Java to/from JSON
- Currently no Java/JSON binding standard
 - Java API for JSON Processing (JSR 535) is parsing, not binding
- EclipseLink interprets JAXB XML bindings for JSON
 - Content-type selectable by setting property on Marshaller/Unmarshaller


XML and JSON from JAXB Mappings

@XmlRootElement

```
public class Customer {  
    private long id;  
    private String firstName;  
    private String lastName;  
    private Address address;  
    @XmlElementWrapper(name="phone-numbers")  
    @XmlElement(name="phone-number")  
    private Set<PhoneNumber> phoneNumbers;  
}
```



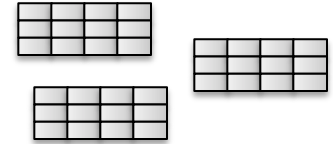
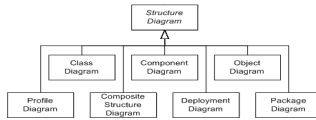
```
{  
  "phone-numbers": [{  
    "id": 2,  
    "num": "512-555-9999",  
    "type": "mobile"  
  }],  
  "address": {  
    "city": "New York",  
    "id": 1,  
    "street": "Central Park  
East"  
  },  
  "firstName": "Woody",  
  "id": 1,  
  "lastName": "Allen"  
}
```



```
<?xml version="1.0"  
encoding="UTF-8"?>  
<customer>  
  <phone-numbers>  
    <phone-number>  
      <id>2</id>  
      <num>512-555-1234</num>  
      <type>home</type>  
    </phone-number>  
  </phone-numbers>  
  <address>  
    <city>New York</city>  
    <id>1</id>  
    <street>Central Park East</street>  
  </address>  
  <firstName>Bill</firstName>  
  <id>1</id>  
  <lastName>Allen</lastName>  
</customer>
```

Challenges – Binding JPA Entities to XML/JSON

```
<customer>
  <phone-numbers>
    <phone-number>
      <id>1</id>
      ...
      <type>mobile</type>
    </phone-number>
  </phone-numbers>
</customer>
```



- Bidirectional/Cyclical Relationships
- Composite Keys/Embedded Key Classes
- Byte Code Weaving

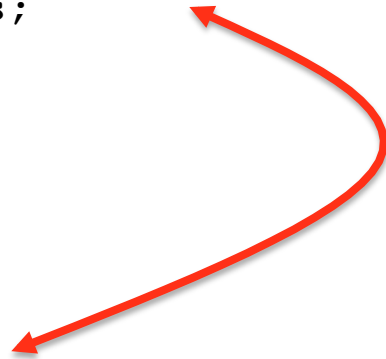
Bidirectional Relationship

@Entity

```
public class Customer{  
    ...  
    @OneToMany (mappedBy="owner")  
    private List<Phone> phones;  
}
```

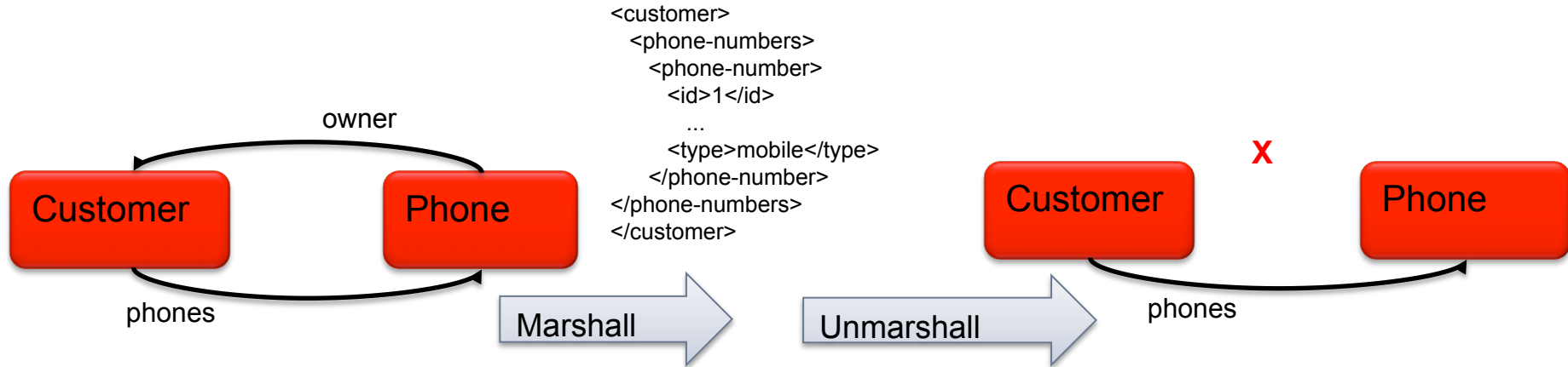
@Entity

```
public class Phone{  
    ...  
    @ManyToOne  
    private Customer owner;  
}
```



Bidirectional Relationships in JAXB

- JAXB specification does not support bidirectional relationships. One side must be marked **@XmlTransient**.
- But that loses the relationship!



EclipseLink XmlInverseReference

@Entity

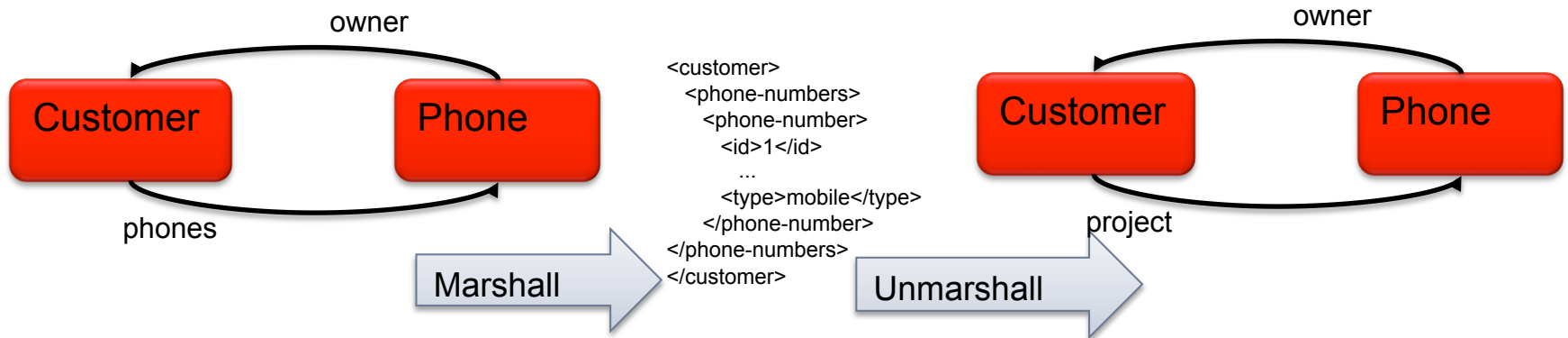
```
public class Customer{  
    ...  
    @OneToMany (mappedBy="owner")  
    private List<Phone> phones;  
}
```

@Entity

```
public class Phone{  
    ...  
    @ManyToOne  
    @XmlInverseReference (mappedBy="phones")  
    private Customer owner;  
}
```


EclipseLink XmlInverseReference

- EclipseLink restores relationships on unmarshall!





Demo

**JAXB/JPA Fidelity
JSON Binding**

<http://git.eclipse.org/gitroot/eclipselink/examples.git/jpa-moxy/jpa-moxy.simple>

Agenda

Browser

HTTP / REST

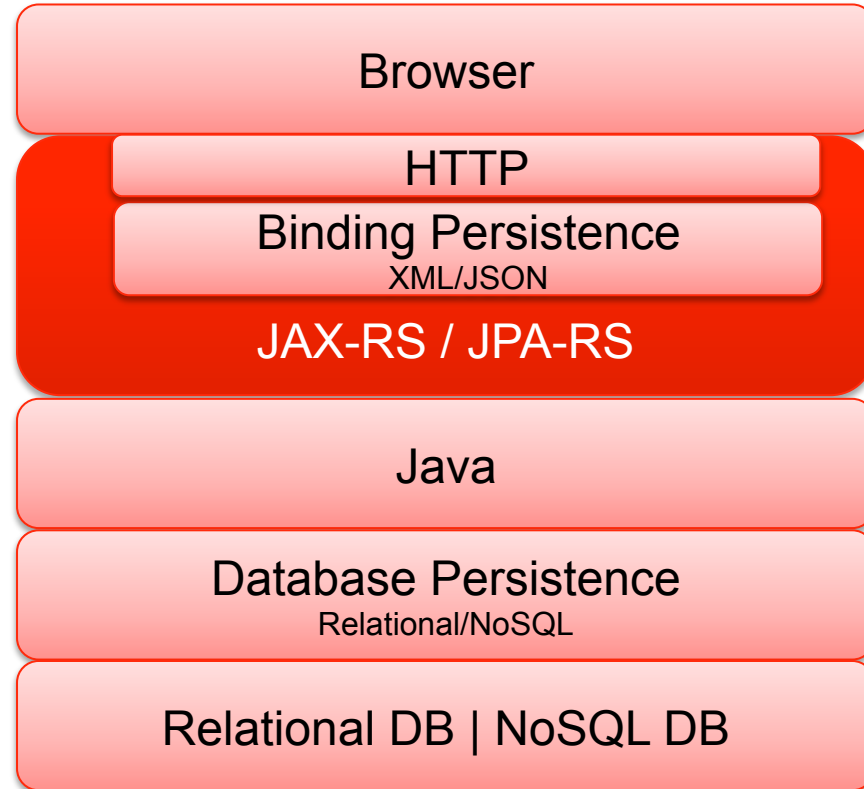
Binding Persistence
XML/JSON

Java

Database Persistence
Relational/NoSQL

Relational DB | NoSQL DB

JPA-RS



JPA-RS

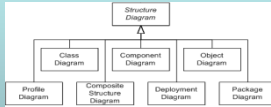
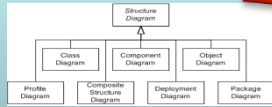
GET http://.../persistence/Accounting/Invoice/...

JAX-RS http://.../persistence/Accounting/Invoice/... mapped to **JPA-RS** service

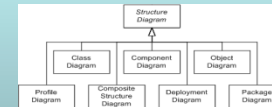
JPA-RS maps URI http://.../persistence/Accounting/Invoice/...

to **Accounting PU** and **Invoice** entity

Accounting PU



...



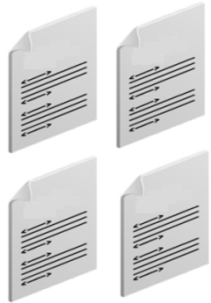
JPA



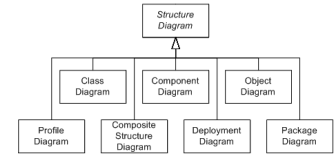
JPA-RS Features

- Access relational data through REST with JSON or XML
- Provides REST operations for entities in persistence unit (GET, PUT, POST, DELETE)
- Automatic generation of XML and JSON bindings
- Supports invocation of named queries via HTTP
- Server Caching—EclipseLink clustered cache
- Dynamic Persistence also supported
 - **Entities defined via metadata—no Java classes required**
 - Enables persistence services for HTML 5/JavaScript applications

Resource Model



Resource Model



JPA Entity Model

Resource Model & Links

```
{
  "firstName": "Frank",
  "gender": "Male",
  "id": 1,
  "lastName": "Smith",
  "responsibilities": [],
  "salary": 1,
  "version": 13,
  "address": {
    "_link": {
      "href": "http://localhost:7001/employee.web-js/persistence/employee/entity/Address/18",
      "method": "GET",
      "rel": "self"
    }
  }
},
```


Demo EclipseLink JPA-RS

<http://git.eclipse.org/gitroot/eclipselink/examples.git/jpa/employee>

Summary

- Java application needs are changing—and EclipseLink is evolving too!
- Support for NoSQL and Polyglot Persistence
- Support for building apps that go from ***Database to Browser***
 - JSON Binding
 - JAXB/JPA Fidelity
 - JPA-RS automating RESTful persistence service