EclipseWorld 2007

November 6-8, 2007

EclipseLink MOXy

Doug Clarke

Eclipse Persistence Services Project co-Lead
Principal Product Manager, Oracle TopLink

A little about Me

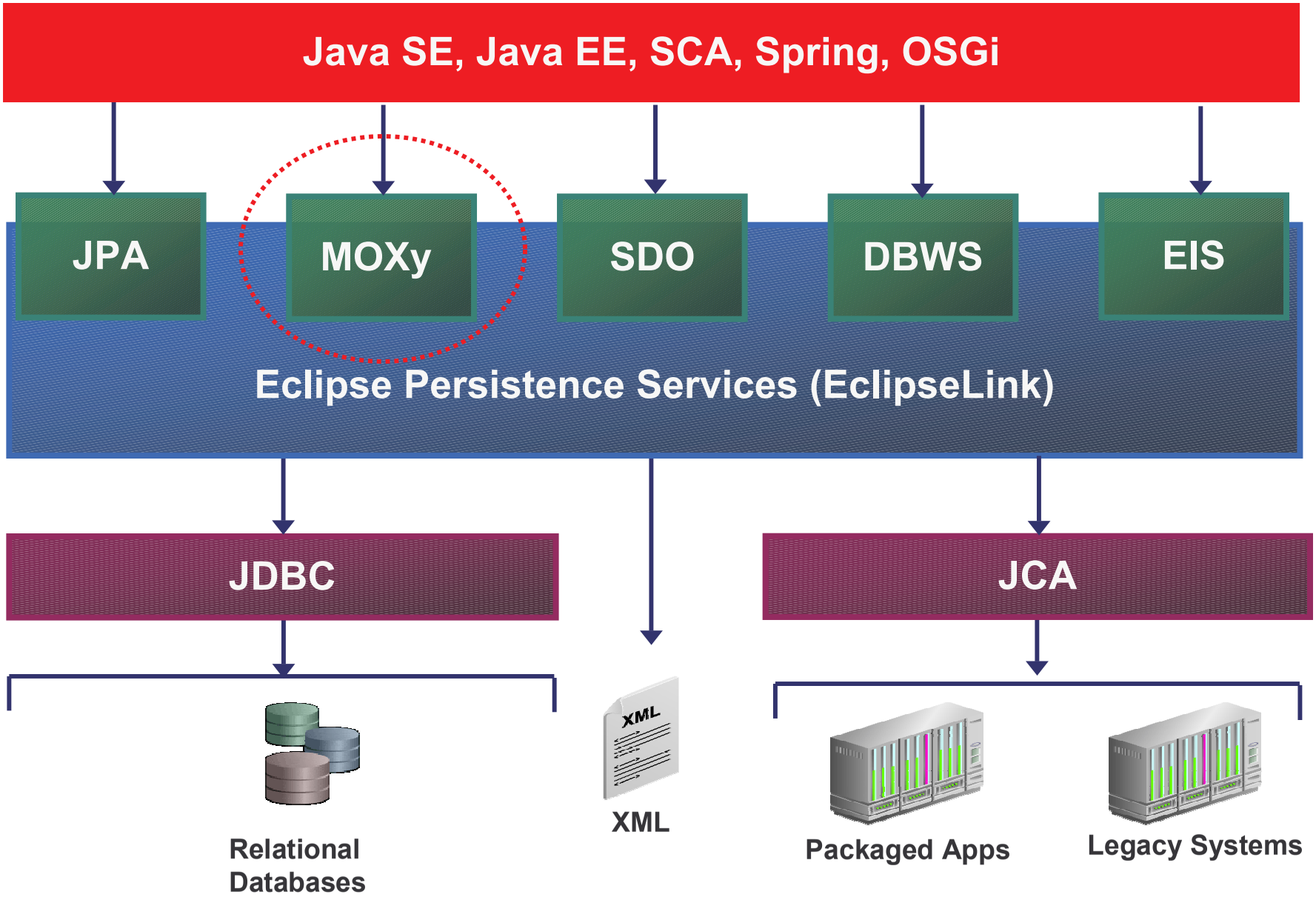
- Co-Lead Eclipse Persistence Services Project
- Principal Product Manager – Oracle TopLink
  - With product for 10 years
  - Product Developer
  - Consultant
  - Involved daily with development and customers
- Frequent speaker at conferences and JUGs primarily on persistence related topics

What you will learn

- What the Eclipse Persistence Services Project and MOXy is
- How MOXy can be used and its benefits
- Why you will want to use this project
- How you can get involved

Eclipse Persistence Services Project

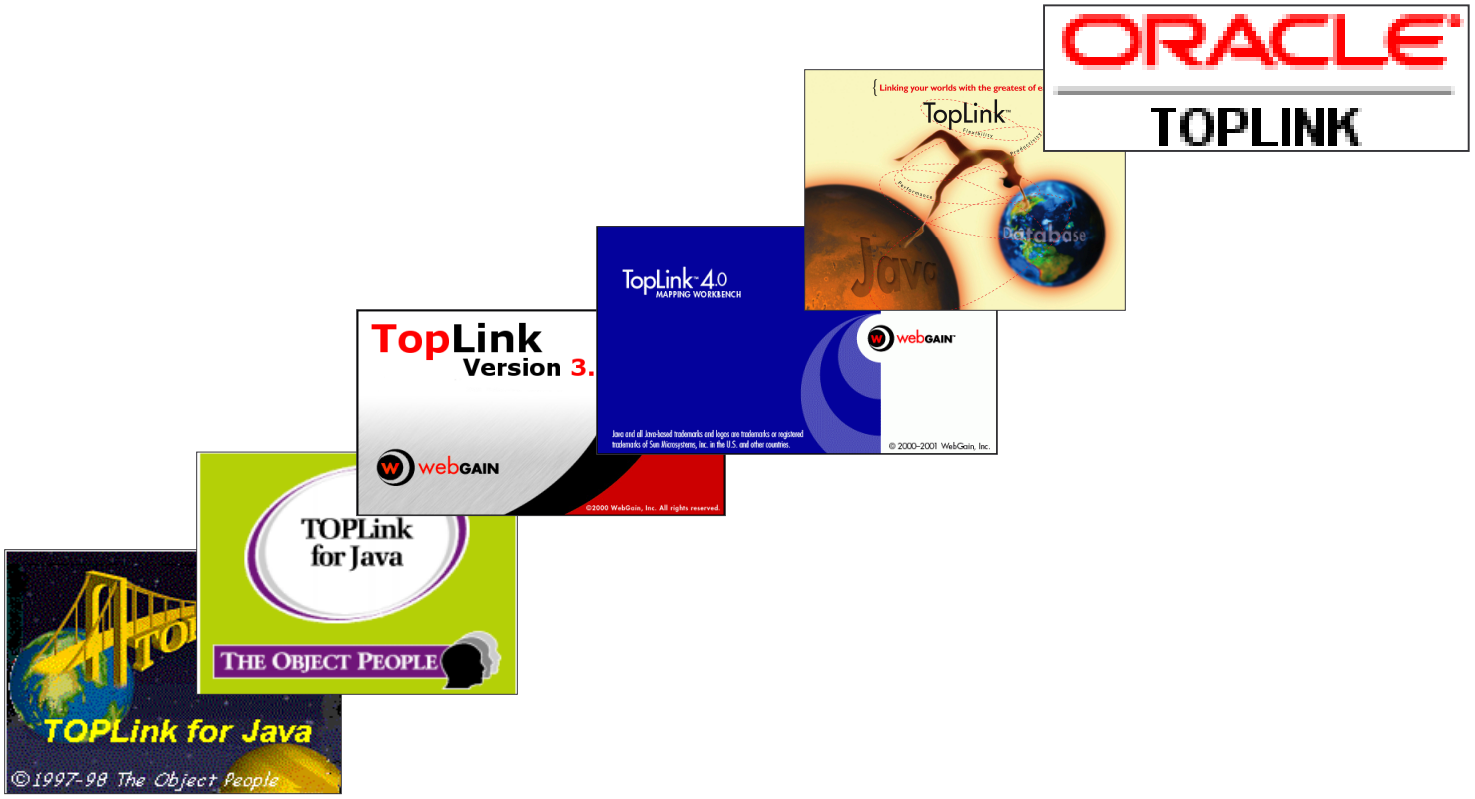
- Eclipse runtime project
  - Nicknamed "EclipseLink"
  - Currently Incubating in Technology Project
- Comprehensive
  - EclipseLink JPA: Object-Relational
  - EclipseLink MOXy: Object-XML
  - EclipseLink SDO: Service Data Objects
  - EclipseLink DBWS: Database Web Services
  - EclipseLink EIS: Non-Relational using JCA
- Defining blueprints for OSGi persistence services



Why is this project important?

- First comprehensive open source persistence solution
  - Object-Relational, Object-XML, and much more
- Shared infrastructure
  - Easily share the same domain model with multiple persistence technologies
  - Leverage metadata for multiple services
- Important part of the Eclipse Ecosystem
- Based upon product with 12 years of commercial usage

History of EclipseLink



1996 → 2007

Challenge: XML Development

- With rapid adoption of SOA and Web Services, XML has become pervasive
- XML is an ideal data exchange format, but is difficult to develop with directly
  - Requires complex, cumbersome code
  - Couples application logic to specific XML structure
  - Difficult to maintain

Java Access of XML Data

- Direct JAXP – window on data
  - Direct use of an XML parser, uses DOM nodes and/or SAX/StAX events directly.
- Entities/Business Objects
  - Accessed as objects or components, transparent that the data is stored in XML
  - Need binding layer in middle tier to handle the object-XML mapping and conversion

Challenge: XML Development

Objective—obtain employee number

- JAXP

```
Node childNode = employeeElement.getFirstChild();
while(childNode != null) {
  if(childNode.getNodeName().equals("employee-number”)) {
      Node employeeNumberTextNode = childNode().getFirstChild();
      employeeNumber = new
      Integer(employeeNumberTextNode.getNodeValue()).intValue();
  }
  childNode.getNextSibling();
}
```

- Using XML binding

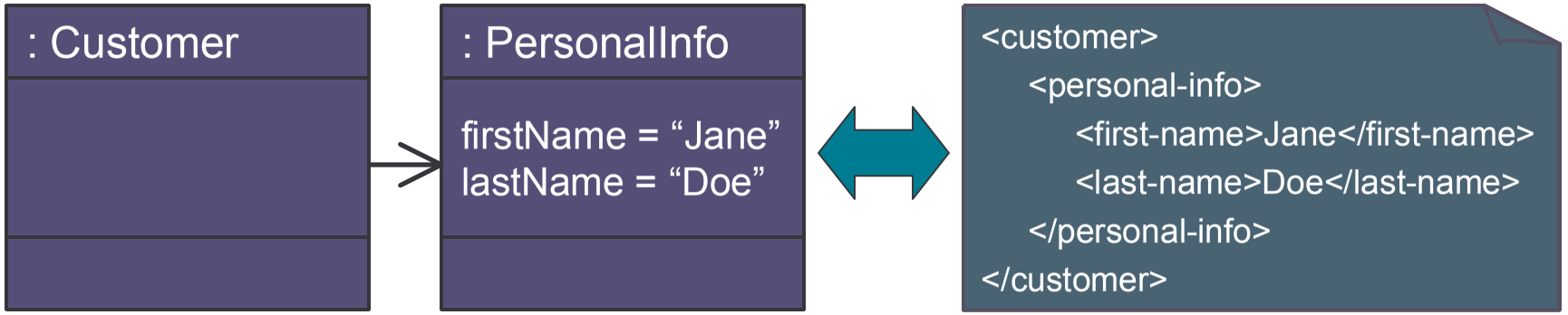
```
employee.getEmployeeNumber();
```


Data Binding Approaches

- Code Generation
- Declarative
  - Annotate Java Classes
  - Externalized Mapping Metadata

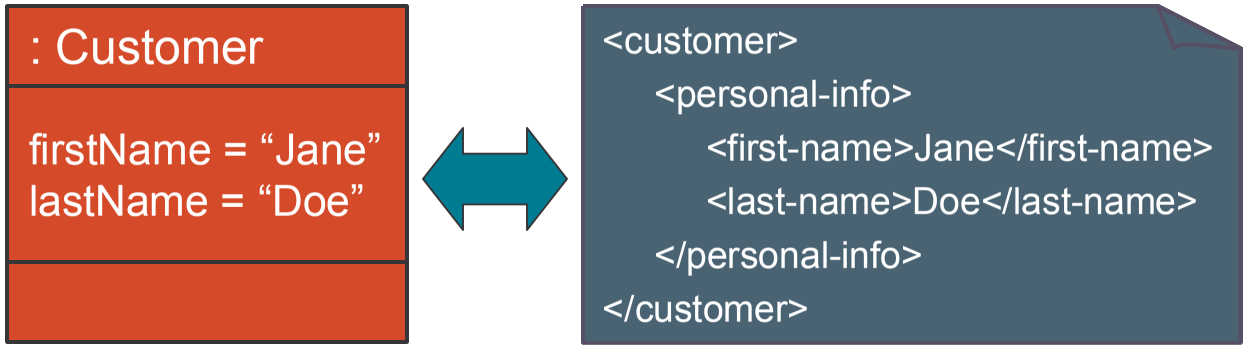
Code Generation

- Sun JAXB 1.0 Reference Implementation
  - Java Classes reflect schema structure
  - Generated classes not extensible/modifiable
  - All document contents marshalled & unmarshalled



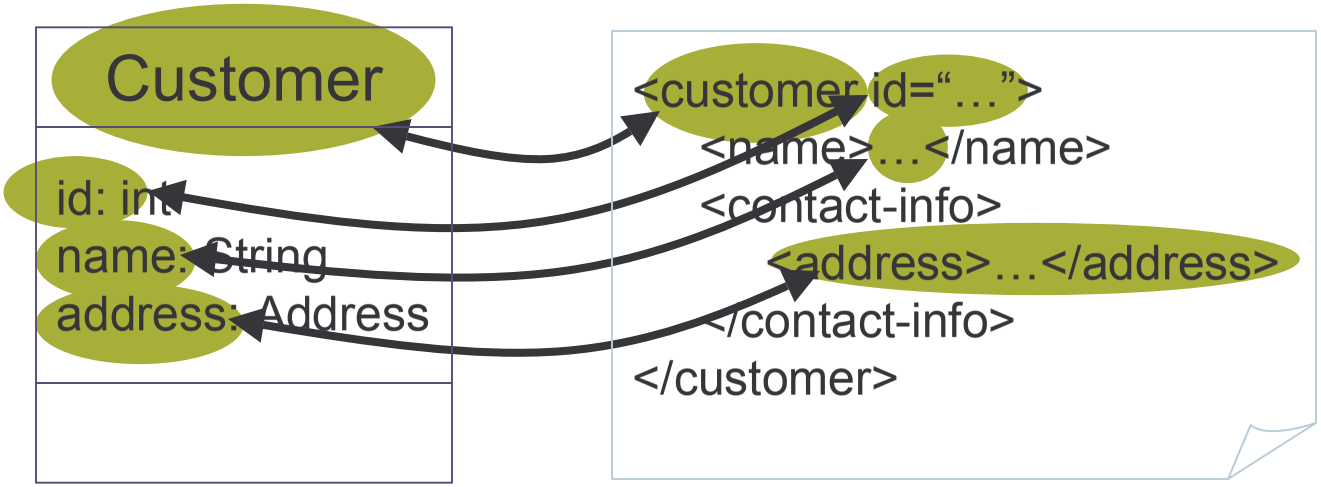
Declarative Binding

- MOXy
  - Arbitrary Classes mapped to any schema via mapping metadata



Mapping

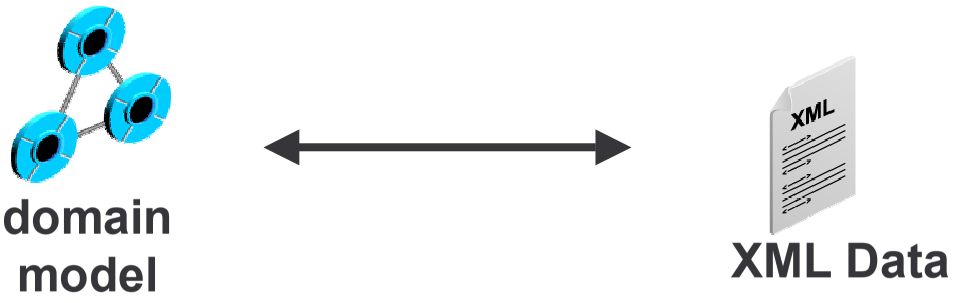
- The activity of ‘Mapping’ is the process of connecting objects/attributes to XML types/nodes.



EclipseLink MOXy

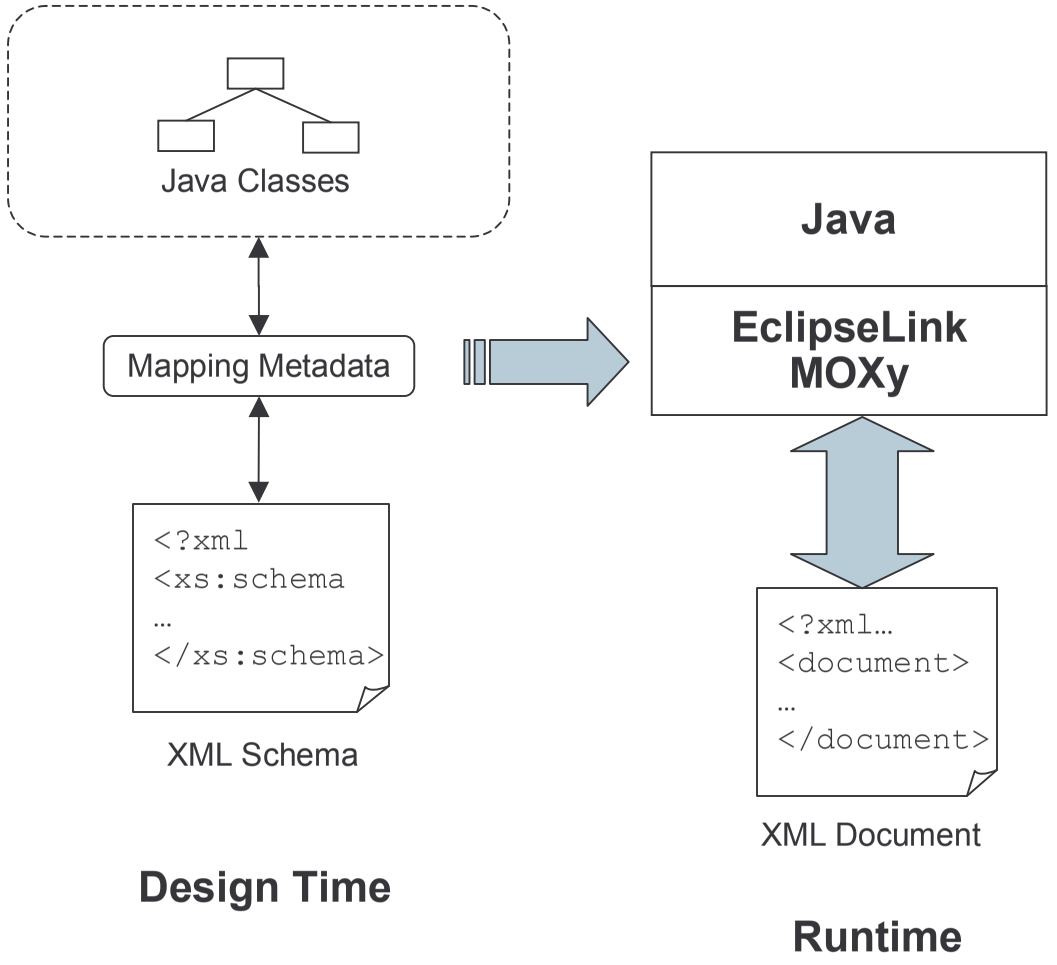
"**M**apping **O**bjects to **X**ML"

- Provides complete Object-XML mapping
  - Allows developers to work with XML as objects
  - Efficiently produce and consume XML
- Supports Object-XML standard - JAXB
  - Provides additional flexibility to allow complete control on how objects are mapped

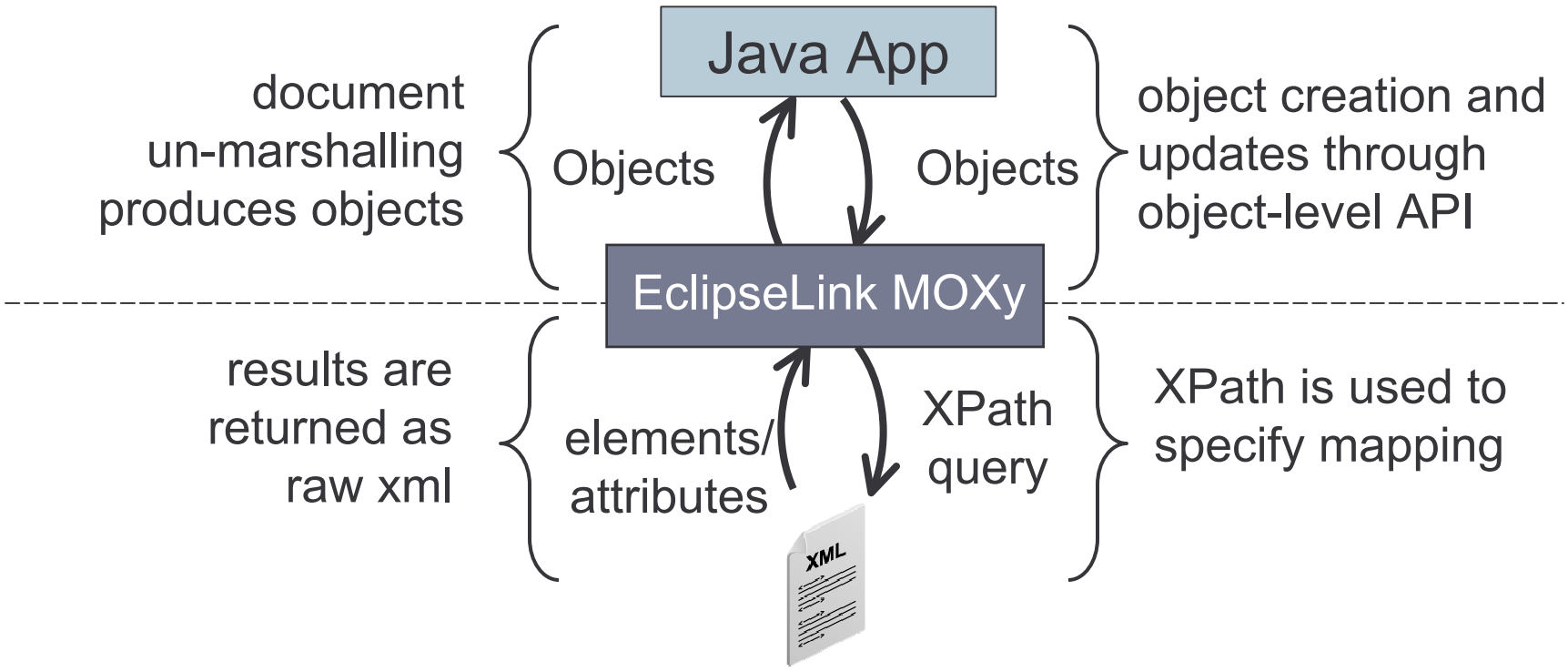


Where does MOXy fit in?

- MOXy runtime combines:
  - Java Classes
  - Mapping Metadata
  - XML



MOXy Binding Layer



EclipseLink MOXy Benefits

- Rich set of mappings providing complete control and flexibility to map objects to any XSD
  - Direct, composite object, composite collection, inheritance, positional, path, transformation ….
- Visual Mapping support using Workbench
- Partial Document Mapping
- Document Preservation
- Supports any JAXP compliant parser
  - SAX, DOM, StAX

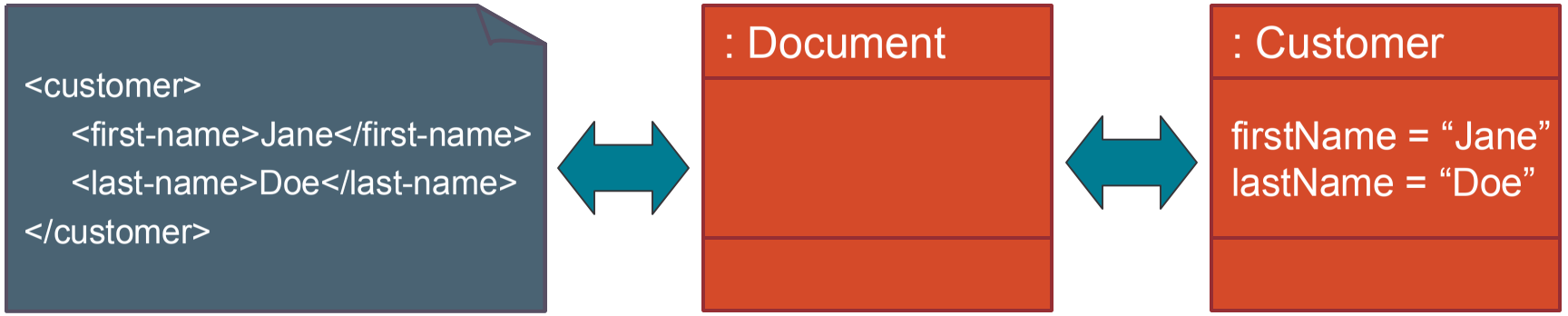
MOXy is Parser Independent

Write Once, Run Anywhere

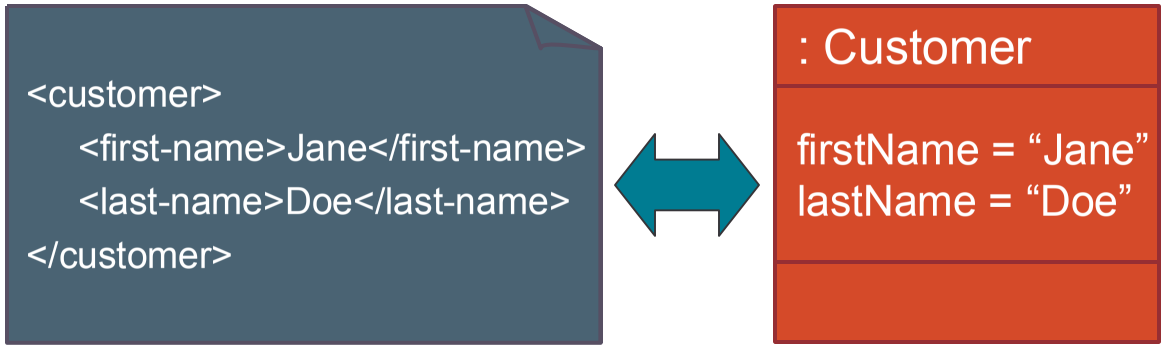
- Most XML binding layers are bound to a specific version of an XML parser.
- Most enterprise applications are run on application servers.
- Each application includes and depends on a specific version of an XML parser.
- If the binding layer and application server are dependent on different XML parsers, then it may be impossible to use them together.

DOM vs. Event Based

DOM Based – Requires an Intermediate Structure



Event Based – No Intermediate Structure Required



DOM Based Binding Solutions

Advantages

- Unmapped XML content can be preserved (such as comments).
- User can be given access to the underlying "DOM" structure.

Disadvantages

- Slower and requires more memory
  - Underlying "DOM" structure must be built and traversed

Event Based Binding Solutions

Advantages

- Better performance since an intermediate structure need not be built.

Disadvantages

- Unmapped XML content cannot be preserved (such as comments).
- User cannot be given access to the underlying “DOM” structure

MOXy gives you Choices

- MOXy supports both SAX, DOM, and StAX parsers.
- Choose your parsing strategy based on your application needs.

Supported Development Approaches

- Bottom Up: compile schema to generate classes
  - JAXB 2.0—annotated POJOs (internal metadata)
  - POJOs with external metadata
- Meet in the middle
  - Annotate POJOs with mapping annotations
  - Combine POJOs with external metadata
- Top Down
  - Generate schema from annotated POJOs

MOXy's External Mapping Metadata

- Mapping information captured in XML and not in the objects.
- External metadata means this approach is NOT at all intrusive on either the object model or the XML schema.
- The object model can be mapped to multiple XML representations.

MOXy Internal Mapping Metadata

Mapping Information Using JAXB 2.0 Annotations

```
@XmlRootElement
public class Customer {

    @XmlAttribute(name="id")
    public int getId() {…}
    public void setId(int id) {…}

    @XmlElement(name="billing-address")
    public Address getBillingAddress() {…}
    public void setBillingAddress(Address address) {…}

}
```


Advantages of JAXB 2.0

JAXB 2.0 Standardized on POJOs

- No binding logic in the generated classes.
- Metadata specified using Java annotations.
- The only compile time dependencies are standard JAXB classes and interfaces.
- Classes generated by one vendors compiler can be used in another vendors runtime.
- JAXB 2.0 included in Java SE 6 (Mustang)

MOXy API has Standard API

JAXB 2.0 Standardized runtime API

```
// Instantiate the JAXB context.  The context path
// indicates which classes are involved in the XML binding
JAXBContext context =
    JAXBContext.newInstance(CONTEXT_PATH);

// Unmarshal the objects from XML
File file = new File("input.xml");
Unmarshaller unmarshaller = context.createUnmarshaller();
Customer customer = (Customer)
    unmarshaller.unmarshal(file);

// Marshal the objects to XML
Marshaller marshaller = context.createMarshaller();
marshaller.marshal(customer, System.out);
```


Mapping in MOXy

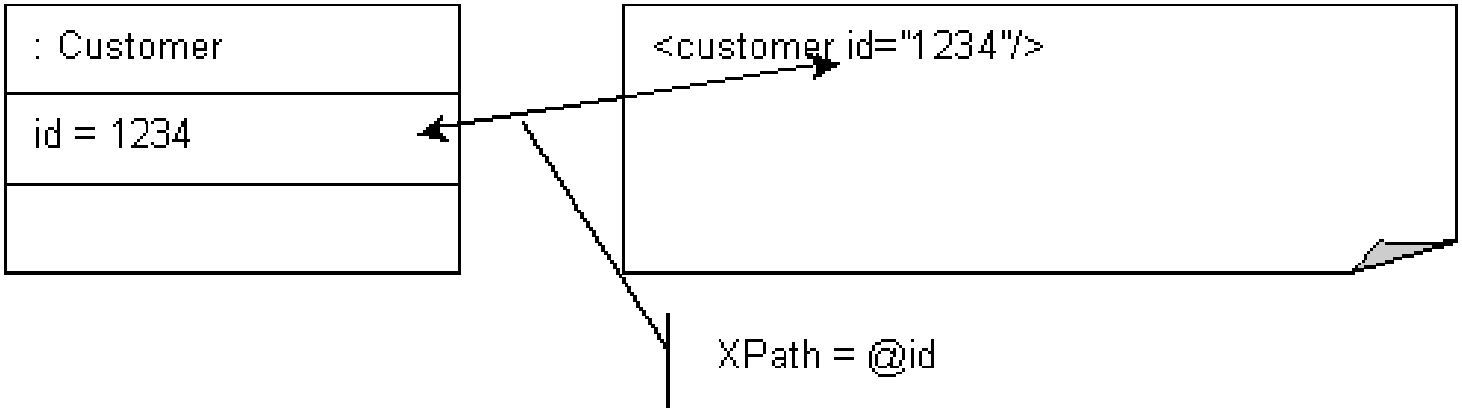
- Powerful mapping approach:
  - XPath based Mapping
  - Positional Mapping
- Extensive Mapping Types
  - Direct
  - Composite Object
  - Composite Collection
  - Direct Collection
  - Relationships
  - Transformation
  - Complex Type Inheritance

XPATH

- MOXy uses XPath expressions to identify XML content that is mapped:
  - XPath by Name
  - XPath by Path and Name
  - XPath by Position
  - Self XPath

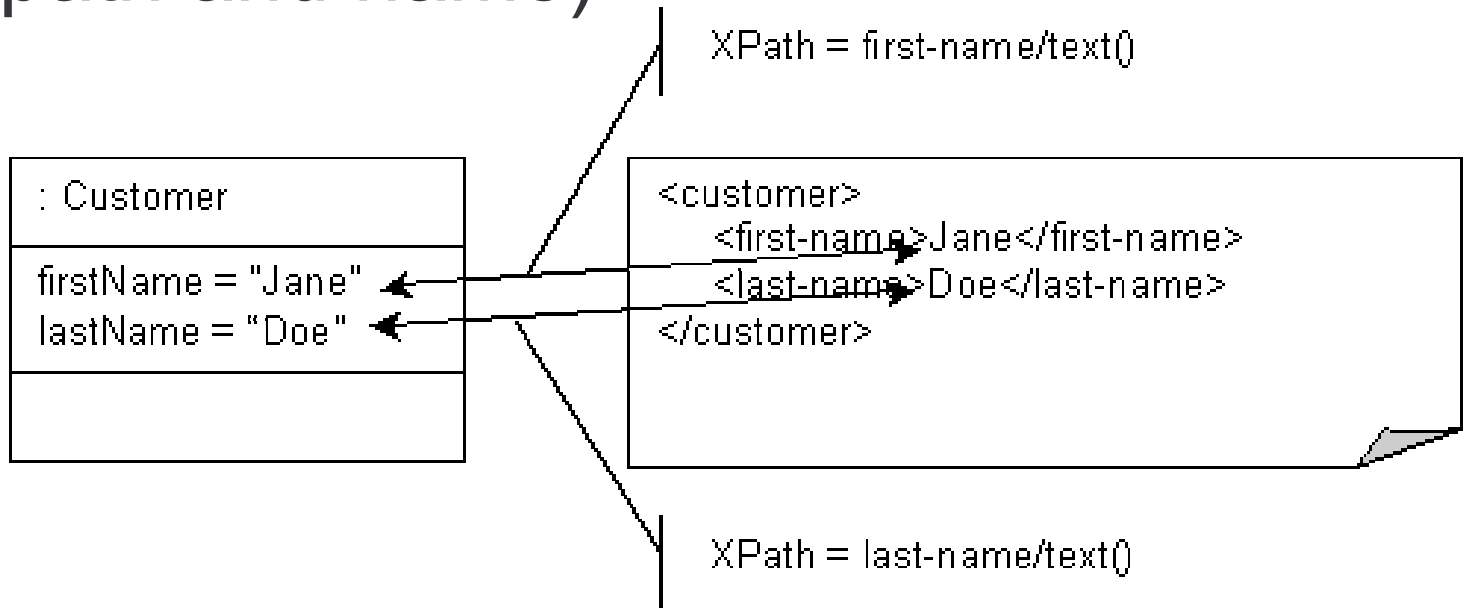
Direct Mapping: Attribute

- Mapping a Java field to an XML *attribute* is done with a DirectMapping and XPath (name).



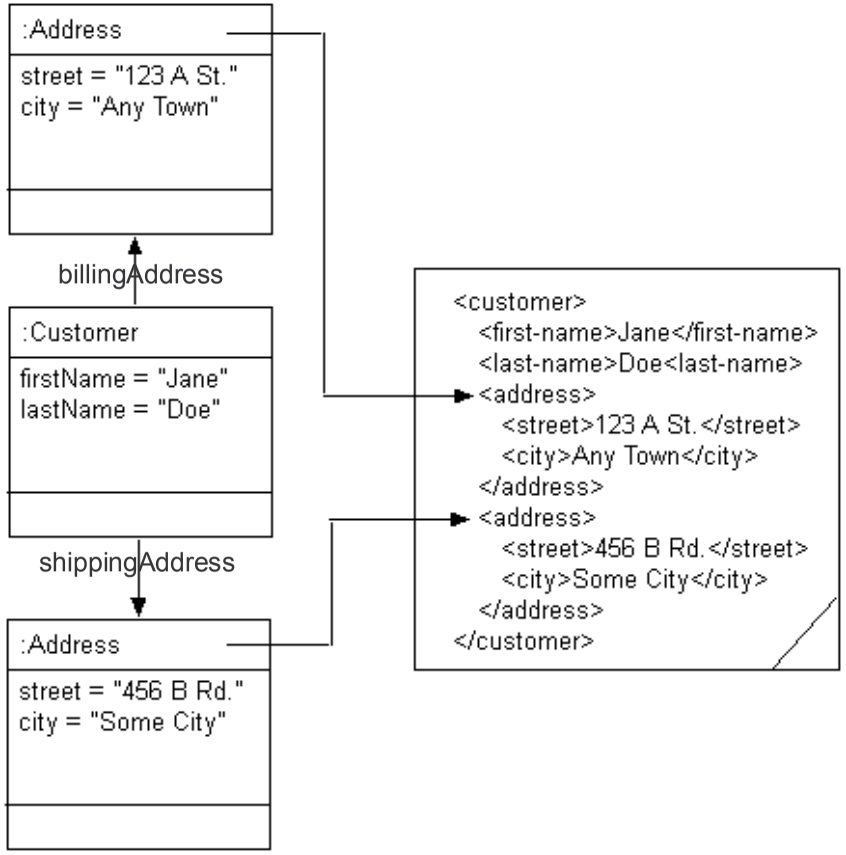
Direct Mapping: Elements

- Mapping a Java field to an XML *element* is done with a DirectMapping and XPath (path and name)



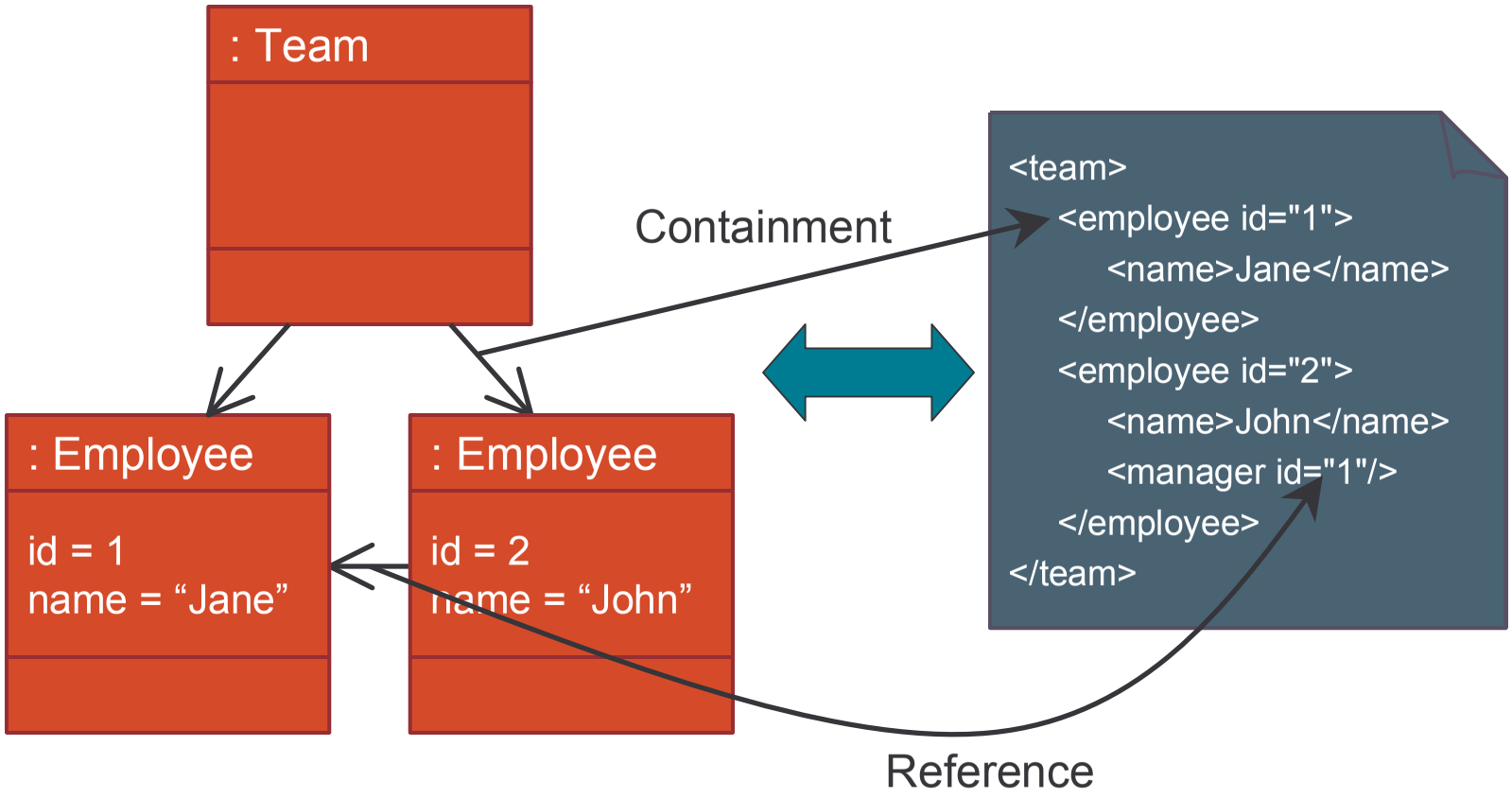
Elements by Position Example—Composite Object

- An object may have multiple composite object mappings to the same reference class. Each composite object mapping must have a unique XPath, e.g.:
  - billingAddress is address[1]
  - shippingAddress is address[2]



Relationship Support

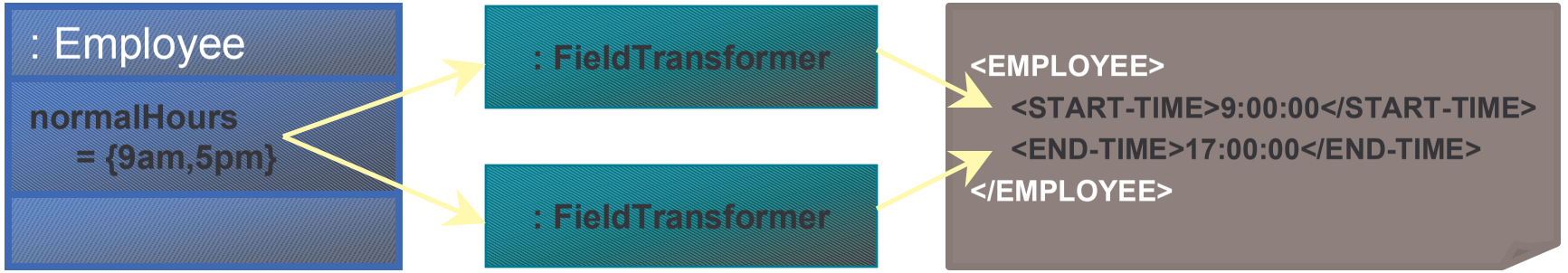
Containment and Reference (Key-Based)



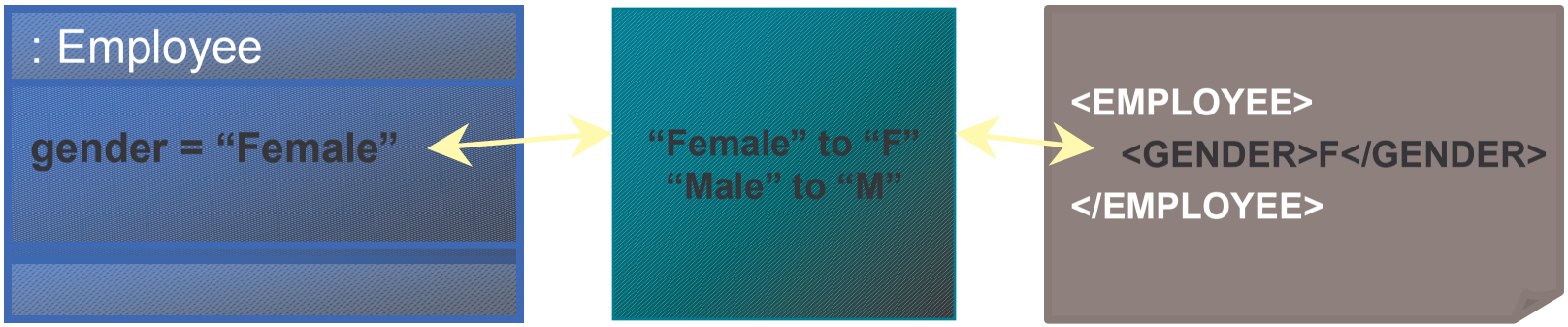
Transformation Mapping

Unmarshal (Read)

Marshal (Write)

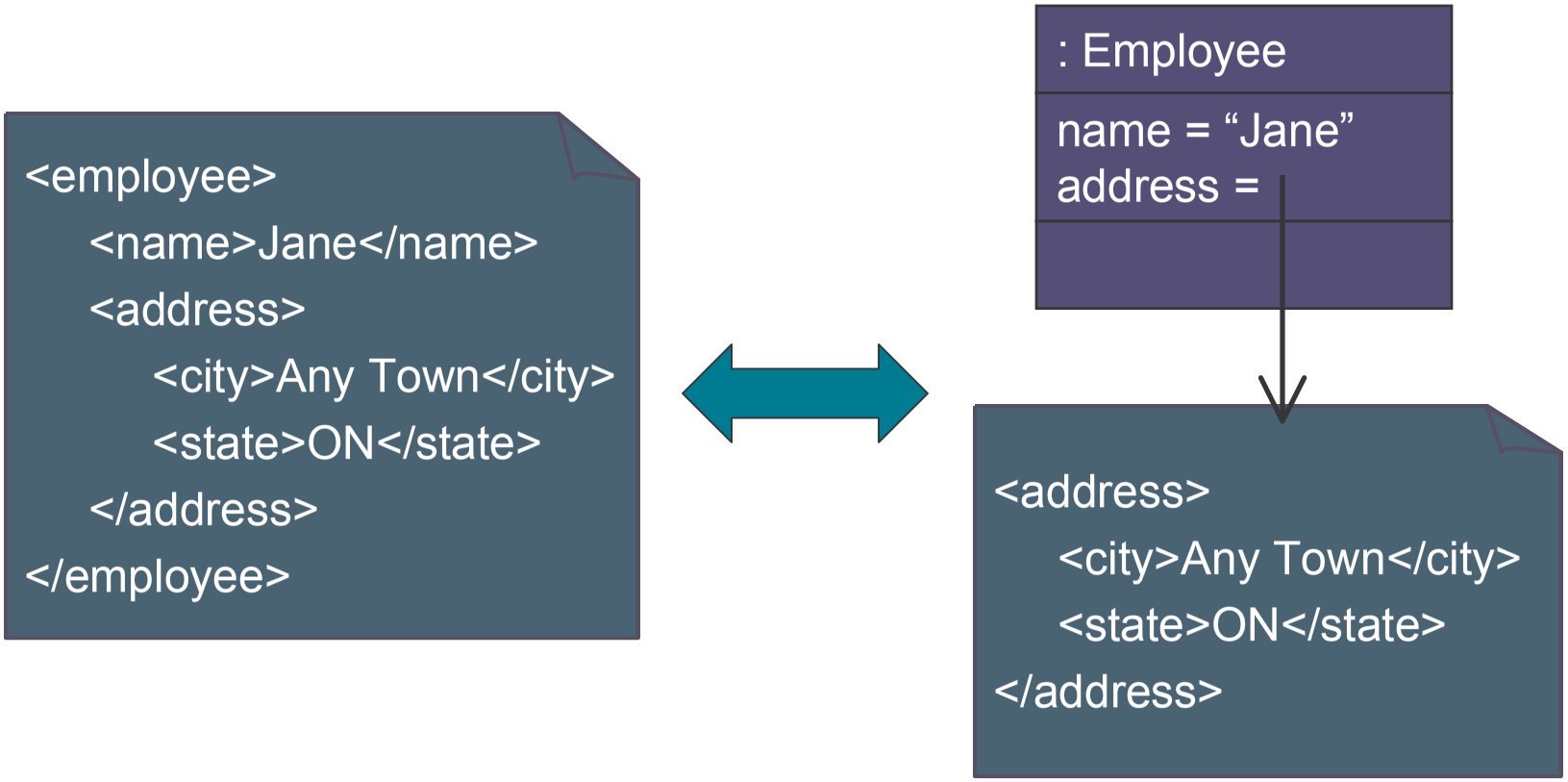


Object Type Converter



Partial XML Mapping

When All Else Fails, Leave it as XML



Combining Persistence Services

- External metadata based approach allows the same domain model to be mapped with multiple persistence services
  - Supports usage within Web Services/SOA/SCA
  - Domain model can be shared between persistence services (JPA, MOXy, EIS)
  - Transformations are bidirectional:
    - Unmarshall XML to objects and then persist
    - Marshall persistent objects to XML

JAXB 2.0 & EJB 3.0

Combining JAXB 2.0 and EJB 3.0 Annotations

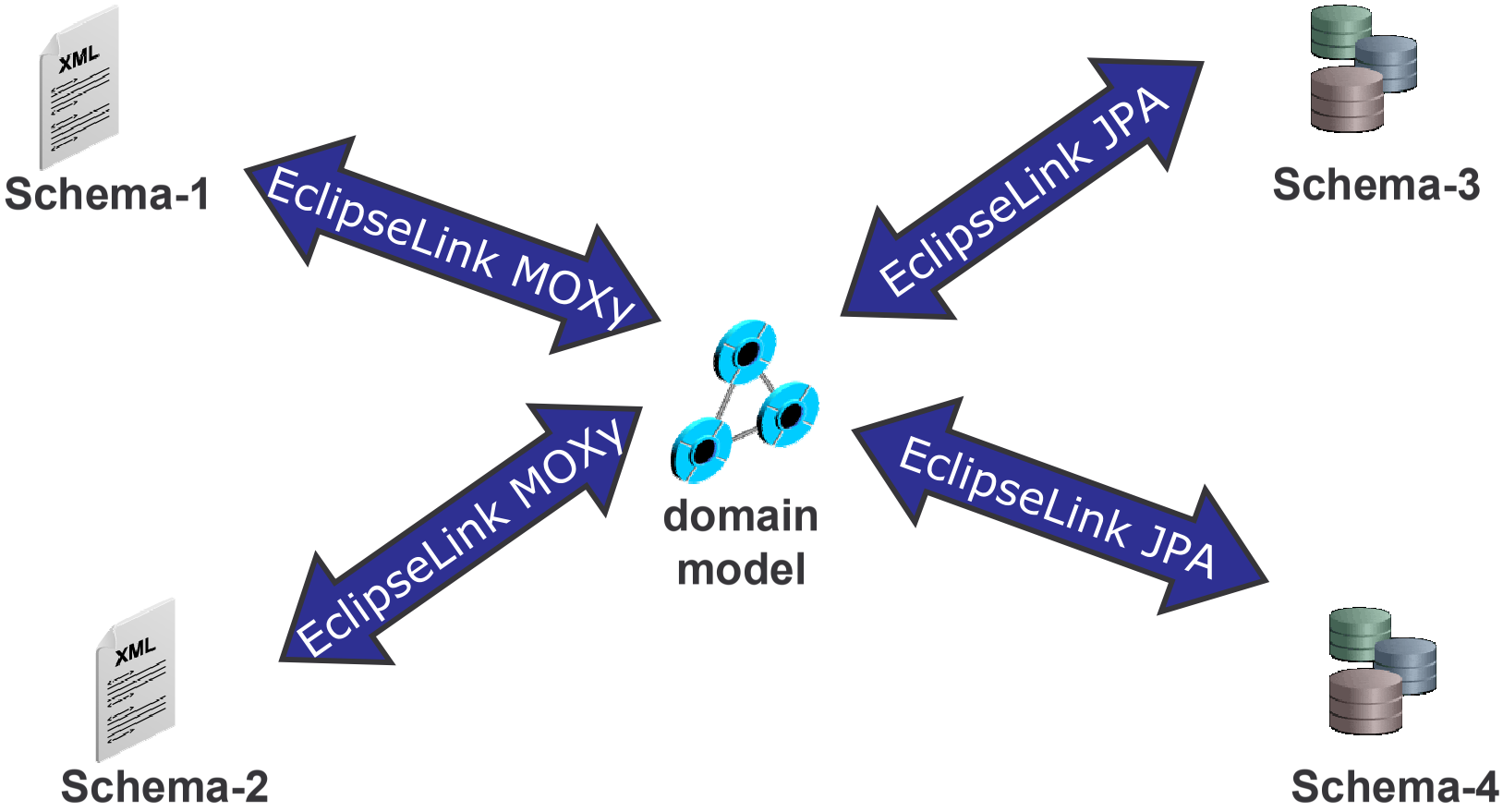
```
@XmlRootElement
@Entity
public class Customer {

    @XmlAttribute(name="id")
    @Id
    public int getId() {…}
    public void setId(int id) {…}

    @XmlElement(name="billing-address")
    @OneToOne
    @JoinColumn(name="ADDR_ID")
    public Address getBillingAddress() {…}
    public void setBillingAddress(Address address) {…}

}
```


Leveraging Common Domain Model



MOXy Tooling

- EclipseLink Workbench
  - Part of EclipseLink Utilities component
  - Standalone graphical mapping tool
  - Supports MOXy JAXB, ORM, and EIS
  - Design-time diagnostics
- Eclipse IDE
  - JAXB 2.0 mapping metadata is expressed in annotations. JDT provides some code assist

MOXy Summary

- Usability
- Flexibility
- Performance
- Full W3C XML Schema Support
- Standards Compliance
- Compatibility with Other Standards
- Compatibility with SOA

Road Map: Where's EclipseLink going?

- Delivery of initial 1.0 milestone 1: Nov 5, 2007
  - Build and testing processes
  - Initial contribution functional
- Specifications: JAXB 2.0, SDO 2.1
- OSGi packaging and usage examples
- Database Web Services (DBWS)
- Data Access Service (DAS) - SDO with JPA
- Simplified DataMap Access and Dynamic Persistence

EclipseLink Summary

- First comprehensive Open Source Persistence solution
  - EclipseLink JPA: Object-Relational
  - EclipseLink MOXy: Object-XML
  - EclipseLink SDO: Service Data Objects
  - EclipseLink DBWS: Database Web Services
  - EclipseLink EIS: Non-Relational using JCA
- Mature and full featured
- Get involved

Community: How can you get involved?

- Users
  - The 0.1-incubation milestone will be available soon
  - Try it out and provide feedback
  - File bug reports and feature requests
- Contributors
  - Contribute to roadmap discussions
  - Bug fixes
- Committers
  - Very interested in growing committer base

More Information

- www.eclipse.org/eclipselink
- Newsgroup: eclipse.technology.eclipselink
- Wiki: wiki.eclipse.org/index.php/EclipseLink
- Blogs
  - Committer Team blog: eclipselink.blogspot.com
  - My blog: java-persistence.blogspot.com