O/R/X et al. mapping

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau
Why we need O/R/X mapping ...

The Bermuda Triangle of data processing

In fact, there is yet other islands, e.g., JSON.
Different data paradigms
(Different technological spaces)
Different type systems
(Different schema languages)
Object model for 101companies system

```java
public class Company {
    private String name;
    private List<Department> depts = new LinkedList<Department>();
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public List<Department> getDepts() { return depts; }
}

public class Department { ... }

public class Employee { ... }
```
Aside: EBNF for 101companies system

company :
  'company' STRING '{' department* '}' EOF;

department :
  'department' STRING '{'
    ('manager' employee)
    ('employee' employee)*
    department*
  '}'';

employee :
  STRING '{'
    'address' STRING
    'salary' FLOAT
  '}'';

STRING : ''' (~'''')* '''';
FLOAT : ('0'..'9')+ ('.' ('0'..'9')+)?;
XSD for 101companies system

```xml
<xsd:element name="company">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" ref="department"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="department"> ... </xsd:element>

<xsd:complexType name="employee"> ... </xsd:complexType>
```

XML too

Grammar-like

Object model-like
Relational schema for 101companies system

CREATE TABLE company (  
id INTEGER PRIMARY KEY,  
name VARCHAR(100) UNIQUE NOT NULL  
)
CREATE TABLE department ( ... )
CREATE TABLE employee ( ... )

Observe one detail: companies do not refer to departments (but vice versa).
What’s O/R/X et al. mapping?

- An approach to data access
- Given at development time:
  - An object model $OM$
  - A corresponding R or X or et al. schema $S$
  - $OM$ may be derived from $S$ or vice versa
- Runtime support:
  - Represent $S$ data as OM objects and vice versa
What’s not O/R/X et al. mapping?

- Represent R/X et al. data with *concrete data type* on O side
  - e.g., JSON in Python dictionaries
- Represent R/X et al. data with *abstract data type* on O side
  - e.g., DOM (Document Object Model for XML)
- Process R/X et al. data by *parsing* in O world
  - e.g., StAX (Pull-based parsing of XML in Java)
- Process R/X et al. data by *events* in O world
  - e.g., SAX (Push-based parsing of XML in Java)
This lecture:
2 ways of mapping in and out O

- O/X mapping (also known as XML data binding)
- O/R mapping (related to persistence)
O/X mapping
(aka XML data binding)

XML data binding refers to a means of representing information in an XML document as a business object in computer memory. This allows applications to access the data in the XML from the object rather than using the DOM or SAX to retrieve the data from a direct representation of the XML itself.

Elevator pitch

Think of a business programmer who wants to focus on business rules and the object model for the application. Suddenly someone drops the X* word on her and she needs to send and receive messages in a XSD-ruled format. How can we make this person happy again?
Example of an XML language: **XBRL**--a language for the electronic communication of business and financial data

<table>
<thead>
<tr>
<th>CURRENT ASSETS</th>
<th>100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assets Held for Sale</td>
<td>100,000</td>
</tr>
<tr>
<td>Construction in Progress, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Inventories</td>
<td>100,000</td>
</tr>
<tr>
<td>Other Financial Assets, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Hedging Instruments, Current (Asset)</td>
<td>100,000</td>
</tr>
<tr>
<td>Current Tax Receivables</td>
<td>100,000</td>
</tr>
<tr>
<td>Trade and Other Receivables, Net,Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Prepayments, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Cash and Cash Equivalents</td>
<td>100,000</td>
</tr>
<tr>
<td>Other Assets, Current</td>
<td>100,000</td>
</tr>
<tr>
<td>Current Assets Total</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

[http://xbrl.org/](http://xbrl.org/)
<ifrs-gp:AssetsHeldSale contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:AssetsHeldSale>
<ifrs-gp:ConstructionProgressCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:ConstructionProgressCurrent>
<ifrs-gp:Inventories contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:Inventories>
<ifrs-gp:OtherFinancialAssetsCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:OtherFinancialAssetsCurrent>
<ifrs-gp:HedgingInstrumentsCurrentAsset contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:HedgingInstrumentsCurrentAsset>
<ifrs-gp:CurrentTaxReceivables contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:CurrentTaxReceivables>
<ifrs-gp:TradeOtherReceivablesNetCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:TradeOtherReceivablesNetCurrent>
<ifrs-gp:PrepaymentsCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:PrepaymentsCurrent>
<ifrs-gp:CashCashEquivalents contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:CashCashEquivalents>
<ifrs-gp:OtherAssetsCurrent contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">100000</ifrs-gp:OtherAssetsCurrent>
<ifrs-gp:AssetsCurrentTotal contextRef="Current_AsOf" unitRef="U-Euros" decimals="0">1000000</ifrs-gp:AssetsCurrentTotal>

http://xbrl.org/

© 2010-16, Software Languages Team Ralf Lämmel, rlaemmel@acm.org
101 in XML

Think of totaling and cutting salaries for all employees.

O/X mapping would allow us to represent XML in “business objects” and to implement functionality in an OO manner.
X-to-O type mapping

For instance, “xjc” for Technology: JAXB of the Java platform.
XML schemas (XSD)
XSD for 101

```xml
<xsd:element name="company">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" ref="department"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="department"> ... </xsd:element>

<xsd:complexType name="employee"> ... </xsd:complexType>
```
XSD – schema components

- Element declarations
- Complex type definitions
- Model-group definitions
- Simple type definitions
- Attribute declarations
- Attribute-group definitions
- Redefinitions
- Annotations

- Sets of XML trees rooted by a certain element name
- Recursive macros with subtyping
- Nonrecursive macros without subtyping
- Types of leafs in XML trees (both elements and attributes).
- Deprecated
- Comments and hints for schema processors
Sketch of the company schema

<xs:schema ...
   <xs:element name="company"> ... </xs:element>
   <xs:element name="department"> ... </xs:element>
   <xs:complexType name="employee"> ... </xs:complexType>
   <xs:element name="name"> ... </xs:element>
   <xs:element name="address"> ... </xs:element>
   <xs:element name="salary"> ... </xs:element>
</xs:schema>

Exercise: once you have seen the entire schema and completed this lecture, try to answer the following question: Why does it make (some) sense that both element declarations and complex-type definitions are put to work in the sample schema?
Model group compositors

- `<sequence>` juxtaposition in EBNF
- `<choice>` “|” in EBNF
- `<all>` “||” (permutation phrases)
- `minOccurs="0"` ?
- `minOccurs="1" maxOccurs="unbounded"` +
- `minOccurs="0" maxOccurs="unbounded"` *
An employee element has children for name, address, and salary.
In a variation of our preferred schema, a subunit (of a department) is either an employee or a department.
A company element has any number of department elements as its children.
Global vs. local

Declaration of a local element declaration

<xs:choice>
  <xs:element name="employee" type="employee" />
  <xs:element ref="department" />
</xs:choice>

Reference to a global element declaration
XSD simple types

• Comparable to primitive types in Java.
• Example:
  `<xs:element name="salary" type="xs:double"/>`
• There are predefined simple types in XSD.
• Attributes are of simple types.
• New simple types can be defined by:
  – Restriction
  – Union
  – List
XSD simple type system

Built-in Datatype Hierarchy

- anyType
- anySimpleType
  - all complex types

self derived types

- primitive types
  - duration
  - dateTime
  - time
  - date
  - gYearMonth
  - gYear
  - gMonthDay
  - gDay
  - gMonth
  - boolean
  - base64Binary
  - hexBinary
  - float
  - double
  - anyURI
  - QName
  - NOTATION
  - string
  - decimal
  - normalizedString
  - token
  - nonPositiveInteger
  - long
  - nonNegativeInteger
    - language
    - Name
    - NMtoken
    - negativeInteger
    - int
    - unsignedLong
    - positiveInteger
    - ID
    - IDREF
    - ENTITY
    - IDREFS
    - ENTITIES
    - short
    - unsignedInt
    - byte
    - unsignedShort
    - unsignedByte
A simple view on Object/XML mapping
(aka XML data binding)
O/X type mapping

Maps to

```xml
<element name="point">
    <complexType>
        <sequence>
            <element name="x" type="xs:int"/>
            <element name="y" type="xs:int"/>
        </sequence>
    </complexType>
</element>
```

Maps to

```java
public class Point {
    public int x;
    public int y;
}
```
Object model for 101companies system

public class Company {
    private String name;
    private List<Department> depts = new LinkedList<Department>();
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public List<Department> getDepts() { return depts; }
}

public class Department { ... }

public class Employee { ... }

XSD for 101companies system

```xml
<xs:element name="company">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element maxOccurs="unbounded"
        minOccurs="0"
        ref="department"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="department"> ... </xs:element>

<xs:complexType name="employee"> ... </xs:complexType>
```

XML too
Grammar-like
Object model-like
http://101companies.org/wiki/
Contribution:jaxbComposition

For sanity’s sake, let’s look only at
Total.java and Cut.java.
Object/XML mapping

Directions for Object/XML mapping
- Generate classes from XML schemas.
- Generate XML schemas from classes.
- Describe mapping only without generation.

Motivations for Object/XML mapping
- Support valid XML output.
- Hide XML in OO programming.
- Use XML-based object serialization.
Some bits of the O/X impedance mismatch
How to map

“s = a:x (b:y+ c:z)+ ”?

```
<xs:element name="s">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="a" type="x"/>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="b" type="y" maxOccurs="unbounded"/>
        <xs:element name="c" type="z"/>
      </xs:sequence>
    </xs:sequence>
  </xs:complexType>
</xs:element>```

© 2010-16, Software Languages Team Ralf Lämmel, rlaemmel@acm.org
How to map

“s = a:x (b:y+ c:z)+ ”?

```java
class s {
    x a;
    y[] b;
    z[] c;
}
```

- Grouping of b’s and c’s is lost.
- This may be Ok for read access.
- This is not Ok for round-tripping.
- Occurrence constraints not enforced:
  - Mandatory a, b, c
XML data binding is difficult because XML are parented trees whereas ...
Objects are freewheeling graphs.
It's also difficult because XML trees are node-labeled whereas ...
Object graphs are edge-labeled.
Technology: JAXB
Option chosen by JAXB

• POJO with fields and getters/setters.
• Annotations define XSD-related properties.
• “*” and “+” are mapped to generics.
• Uses of heterogeneous (weakly typed) containers:
  – Nested composites
  – Mixed context
• An element factory is provided.
POJOs

public class Company {
    protected String name;
    protected List<Department> department;

    public String getName() { return name; }
    public void setName(String value) { this.name = value; }

    public List<Department> getDepartment() {
        if (department == null) {
            department = new ArrayList<Department>();
        }
        return this.department;
    }
}

What’s going on here?
public class Cut {
    public static void cut(Company c) {
        for (Department d : c.getDepartment())
            cut(d);
    }
    public static void cut(Department d) {
        cut(d.getManager());
        for (Department s : d.getDepartment())
            cut(s);
        for (Employee e : d.getEmployee())
            cut(e);
    }
    public static void cut(Employee e) {
        e.setSalary(e.getSalary() / 2);
    }
}
public static Company readCompany(File input) throws JAXBException {
    JAXBContext jAXBContext = JAXBContext.newInstance("org.softlang.company");
    Unmarshaller unMarshaller = jAXBContext.createUnmarshaller();
    return (Company)unMarshaller.unmarshal(input);
}
Annotations

@XmlElementType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "name",
    "department"
})
@XmlRootElement(name = "company")
public class Company {

    @XmlElement(required = true)
    protected String name;

    protected List<Department> department;

    ...

}
Liberal mapping

```xml
<xs:element name="subunit">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element ref="department"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

```java
public class Subunit {
    protected Employee employee;
    protected Department department;
    public Employee getEmployee() { return employee; }
    public void setEmployee(Employee value) { this.employee = value; }
    public Department getDepartment() { return department; }
    public void setDepartment(Department value) { this.department = value; }
}
```

What is (too) liberal here?
XML data binding with **JAXB**

JAXB is an integral part of the Java SDK since Java 6.
Samples on 101

- Contribution: jaxbComposition
- Contribution: jaxbChoice
- Contribution: jaxbExtension
- Contribution: jaxbSubstitution
O/R mapping
(related to persistence)

Object-relational mapping (OR mapping) products integrate object programming language capabilities with relational databases managed by Oracle, DB2, Sybase, and other RDBMSs. Database objects appear as programming language objects in one or more existing object programming languages.

[Source: http://www.service-architecture.com/articles/object-relational-mapping]
Elevator pitch

How to persist objects in database tables?

How to map object models to relational schemas?

... or the other way around?
Exercises

Bing for this statement and read about it:

“O/R Mapping … is the Vietnam of Computer Science”!

Read about the “O/R impedance mismatch”!
Why is O/R mapping challenging?

**Objects vs. tables**

---

**Object graph of a company**

<table>
<thead>
<tr>
<th>:Company</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ACME Corporation</td>
</tr>
<tr>
<td>Departments</td>
<td>[…]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>:Department</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Research</td>
</tr>
<tr>
<td>Departments</td>
<td>[…]</td>
</tr>
<tr>
<td>Employees</td>
<td>[…]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>:Employee</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Craig</td>
</tr>
<tr>
<td>Address</td>
<td>Redmond</td>
</tr>
<tr>
<td>Salary</td>
<td>123456</td>
</tr>
<tr>
<td>Manager</td>
<td>WAHR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>:Employee</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Erik</td>
</tr>
<tr>
<td>Address</td>
<td>Utrecht</td>
</tr>
<tr>
<td>Salary</td>
<td>12345</td>
</tr>
<tr>
<td>Manager</td>
<td>FALSCH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>:Employee</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Ralf</td>
</tr>
<tr>
<td>Address</td>
<td>Koblenz</td>
</tr>
<tr>
<td>Salary</td>
<td>1234</td>
</tr>
<tr>
<td>Manager</td>
<td>FALSCH</td>
</tr>
</tbody>
</table>
Why is O/R mapping challenging?

**Tables vs. objects**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>ACME Corporation</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Company**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>Research</td>
</tr>
<tr>
<td>43</td>
<td>Development</td>
</tr>
<tr>
<td>44</td>
<td>Dev1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Department**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Department</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>Research</td>
<td>NULL</td>
<td>88</td>
</tr>
<tr>
<td>43</td>
<td>Development</td>
<td>NULL</td>
<td>88</td>
</tr>
<tr>
<td>44</td>
<td>Dev1</td>
<td>43</td>
<td>88</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Employee**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Address</th>
<th>Salary</th>
<th>Manager</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Craig</td>
<td>Redmond</td>
<td>123456</td>
<td>WAHR</td>
<td>42</td>
</tr>
<tr>
<td>2</td>
<td>Erik</td>
<td>Utrecht</td>
<td>12345</td>
<td>FALSCH</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>Ralf</td>
<td>Koblenz</td>
<td>1234</td>
<td>FALSCH</td>
<td>42</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Arrows are inverse when compared to objects.
Why is O/R mapping challenging?

<table>
<thead>
<tr>
<th>O</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collections refer to members</td>
<td>Rows use foreign key</td>
</tr>
<tr>
<td>no NOT NULL</td>
<td>NOT NULL</td>
</tr>
<tr>
<td>Non-primitive attributes</td>
<td>Normal form</td>
</tr>
<tr>
<td>In memory with GC</td>
<td>Huge data volume</td>
</tr>
<tr>
<td>No Transactions</td>
<td>Transactions</td>
</tr>
</tbody>
</table>
Poor men's O/R mapping
Employee POJOs

public class Employee {

    private int id;
    private String name;
    private String address;
    private double salary;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }
    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
}

int id = employee.getId();
String sql = "SELECT * FROM employee WHERE id = ?";
PreparedStatement stm =
   myConnection.getConn().preparedStatement(sql);
stm.setInt(1, id);
ResultSet result = stm.executeQuery();
result.next();
employee.setSalary(result.getDouble("salary"));
employee.setName(result.getString("name"));
employee.setAddress(result.getString("address"));
public class Employee {

    // Some additional / revised members are shown.

    private boolean changed;

    public void setName(String name) {
        this.name = name;
        changed = true;
    }

    public void setChanged(boolean changed) {
        this.changed = true;
    }

    public boolean isChanged() {
        return changed;
    }
}
Save employee

if (employee.isChanged()) {
    if (employee.getId() == 0) {
        String sqlInsert = "INSERT employee ...";
        ...
        stmInsert.executeUpdate();
        String sqlSelectId = "SELECT max(id) AS maxid FROM employee";
        ...
        ResultSet maxid = pstmtSelectId.executeQuery();
        maxid.next();
        employee.setId(maxid.getInt("maxid"));
    } else {
        String sqlUpdate = "UPDATE employee ...";
        ...
        stmUpdate.executeUpdate();
    }
    employee.setChanged(false);
}
Loading a company lazily

String sqlDepts = "SELECT id FROM department WHERE did IS NULL " + "AND cid = (SELECT id FROM company WHERE name = ?);";

PreparedStatement stmDepts =
    myConnection.getConn().prepareStatement(sqlDepts);
stmDepts.setString(1, company.getName());
ResultSet deptsR = stmDepts.executeQuery();
while (deptsR.next()) {
    Department dept = new Department(deptsR.getInt("id"));
    dept.setObjectFactory(this);
    dept.setLoaded(false);
    company.getDepts().add(dept);
}
company.setChanged(false);
company.getDepts().setUnchanged();
101implementation:jdbc2

DEMO
Java Persistence API
(JPA)
JPA: Simplifying persistence in Java

- Capabilities
  - Store objects in database tables, one object per row.
  - Restore objects in different programs / runs.
The JPA architecture

Application

Persistent Objects

JPA

JPA properties

Database
A persistent class

@Entity
public class Cat {
    @Id
    private String id;
    private String name;
    private char sex;
    private float weight;
    public String getId() { return id; }
    private void setId(String id) { this.id = id; }
    // … other getters and setters …
}
A database table

The CAT table in the database

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Number(10)</td>
<td>not null</td>
</tr>
<tr>
<td>name</td>
<td>Varchar</td>
<td>not null</td>
</tr>
<tr>
<td>sex</td>
<td>character(1)</td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>Number(10,5)</td>
<td></td>
</tr>
</tbody>
</table>

PRIMARY KEY (ID)
A JPA session
(in Java code)

```java
EntityManagerFactory fac = Persistence.createEntityManagerFactory(…);
EntityManager em = factory.createEntityManager();
em.getTransaction().begin();
Cat princess = new Cat();
princess.setName("Princess");
princess.setSex('F');
princess.setWeight(7.4f);
em.persist(princess);
em.getTransaction().commit();
em.close();
```

- **Set up session and begin transaction**
- **Regular OO code**
- **Make an object persistent and commit changes.**

© 2010-16, Software Languages Team Ralf Lämmel, rlaemmel@acm.org
How to retrieve persistent objects?
Use JPQL (Java Persistence query language).

```java
Query query = em.createQuery(
    "select c from Cat c where c.sex = :sex"");
query.setParameter("sex", 'F');
List<Cat> cats = (List<Cat>) query.getResultList();
for (Cat cat : cats) {
    out.println("Female Cat: " + cat.getName() );
}
```
So what’s O/R mapping?

Wikipedia’s definition

Ralf’s definition (relatively naïve version):
• Category 1:
  – Start from (idiomatic) classes.
  – Map object model to relational schema.
  – Deploy relational schema in database.
  – Encode CRUD in OO code.
  – Add transactions to OO code.
• Category 2:
  – Start from database (schema, instance, SPROC).
  – Derive object model to encapsulate data access.
  – Continue as above ...
• Category 1 + 2: classes and tables given, mapping wanted.
• Category 2’:
  – Like Category 2 but ...
  – ER/relational model-level mapping.
  – Coverage of distributed database and data integration.
https://101wiki.softlang.org/Contribution:jpa
The usual features
Object model / Mapping files
JPA configuration
Data dir to be used by SQLITE
Employee Mapping File

@Entity
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String name;
    private String address;
    private double salary;

    public long getId() {
        return id;
    }

    // ... more methods ...
    public void cut() {...}
}
Department Mapping File

```java
@Entity
public class Department {

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int id;

private String name;

@OneToOne(cascade=CascadeType.ALL)
private Employee manager;

@OneToMany(cascade=CascadeType.ALL)
private List<Department> subdepts = new LinkedList<Department>();

@OneToMany(cascade=CascadeType.ALL)
private List<Employee> employees = new LinkedList<Employee>();

...}
```

Maps to Table: DEPARTMENT

Maps to columns: ID, NAME

@OneToOne maps to column: MANAGER_ID
(Foreign Key EMPLOYEE(ID)

@OneToMany maps to a TABLE: DEPARTMENT_EMPLOYEE with columns (Foreign Keys):
DEPARTMENT_ID, EMPLOYEES_ID
@Entity
public class Company {

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int id;

private String name;

@OneToMany(cascade=CascadeType.ALL)
private List<Department> depts = new LinkedList<Department>();

...
JPA configuration

**persistence.xml**

Name are used by EntityManagerFactory

Mapping files

Database settings

JPA settings

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="jpa" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>org.softlang.company.model.Company</class>
    <class>org.softlang.company.model.Department</class>
    <class>org.softlang.company.model.Employee</class>
    <properties>
      <property name="javax.persistence.jdbc.driver" value="org.sqlite.JDBC"/>
      <property name="javax.persistence.jdbc.url" value="jdbc:sqlite:data/jpa.db"/>
      <property name="eclipselink.logging.level" value="ALL"/>
      <property name="eclipselink.ddl-generation" value="create-tables"/>
    </properties>
  </persistence-unit>
</persistence>
```
https://101wiki.softlang.org/
Contribution:jpa

DEMO
Developer’s view on using JPA

The content on this slide is covered "in passing" in the lecture.
Developer’s view on using JPA

1. Link JPA libraries
2. Configure database
3. Create Entities (Object model)
4. Write CRUD code
Developer’s view on using JPA

1. Link JPA libraries
2. Configure database
3. Create Entities (Object model)
4. Write CRUD code
Link JPA Libraries

Required for JPA

- eclipselink-2.6.4.jar
- javax.persistence-2.1.1.jar
- ...

The simplest way: Maven Dependencies

https://mvnrepository.com/artifact/org.eclipse.persistence/eclipselink/2.6.4

Required for DB

specific DB Driver
e.g., SQLite

- sqlite-jdbc-3.16.1.jar

https://mvnrepository.com/artifact/org.eclipse.persistence/eclipselink/2.6.4

Tip: create a lib dir within your project and place all those jars over there. Then, make sure your project’s build path references the jars.
Developer’s view on using JPA

1. Link JPA libraries
2. Configure database
3. Create Entities (Object model)
4. Write CRUD code
JPA relies on a RDBMS

- Simple option
  - Use SQLite
    - sqlite-jdbc-3.16.1.jar
- More flexible option
  - Use any other SQL database via JDBC
Using SQLite

1. Add library

2. Create only folder

3. create entry in persistence.xml

```xml
<property name="javax.persistence.jdbc.driver" value="org.sqlite.JDBC"/>
<property name="javax.persistence.jdbc.url" value="jdbc:sqlite:.jpa.db"/>
<property name="eclipselink.ddl-generation" value="create-tables"/>
```

4. Use in Code

```java
EntityManagerFactory emFactory = javax.persistence.Persistence.createEntityManagerFactory("jpa");
EntityManager em = emFactory.createEntityManager();
```
Developer’s view on using JPA

1. Link JPA libraries
2. Configure database
3. Create Entities (Object model)
4. Write CRUD code
Create Entities

- Use POJOs
  - Use Annotations for Mapping (@Entity, @Id,...)
  - Provide a default constructor
  - Model an id field as primary key
  - Add Getter and Setter for all attributes
  - Optional add more methods
Developer’s view on using JPA

1. Link JPA libraries
2. Configure database
3. Create Entities (Object model)
4. Write CRUD code
Perhaps, don’t use JPA or any „explicit“ O/R mapping. Rather, leverage „implicit“ O/R mapping as integrated, for example, with popular web-application frameworks such as Django and Rails (but this also comes with unique challenges).
What we should also cover eventually …

Object/JSON mapping

„Implicit“ Object/Relational mapping

Acknowledgement: Jürgen Starek has significantly contributed to this content.
End of Lecture