Warm-up: Revisiting selected data technologies via *101companies*

Software Languages Team
University of Koblenz-Landau
Ralf Lämmel and Andrei Varanovich

Demo

101implementation:xquery

XML processing

with XQuery

Companies *in* ***XML***

```
<?xml version="1.0"
encoding="utf-8" ?>
<company ...>
  <name>meganalysis</name>

  <department>
    <name>Research</name>
    <manager>
      <name>Craig</name>
      <address>Redmond</address>
      <salary>123456</salary>
    </manager>
    <employee>
      <name>Erik</name>
```

Total salaries *with* ***XQuery***

```
declare default element namespace
  "http://www.softlang.org/company.xsd";

<result>
  {sum(//salary)}
</result>
```

Descendants axis

Cut salaries *with* **XQuery**

```
declare default element namespace
  "http://www.softlang.org/company.xsd";

copy $copy := .
modify
  for $salary in $copy//salary
  return replace value of node $salary
           with $salary div 2
return $copy
```

XQuery
with updates

Demo

101implementation:dom

In-memory XML processing

in Java

with DOM

Total salaries *with* ***DOM in Java***

```
public static double total(Document doc) {

  // The aggregation variable
  double total = 0;

    // Get the matching elements
    NodeList nodelist = doc.getElementsByTagName("salary");

    // Process the elements in the nodelist
    for (int i=0; i<nodelist.getLength(); i++) {
        // Get element
        Element elem = (Element)nodelist.item(i);
        total += Double.parseDouble(elem.getTextContent());
    }
  return total;
}
```

Descendants axis

Read text content

Cut salaries *with* ***DOM in Java***

```
public static void cut(Document doc) {

    // Get the matching elements
    NodeList nodelist = doc.getElementsByTagName("salary");

    // Process the elements in the nodelist
    for (int i=0; i<nodelist.getLength(); i++) {

        // Get element
        Element elem = (Element)nodelist.item(i);

        // Transform content of element
        double value = parseDouble(elem.getTextContent());
        elem.setTextContent(Double.toString(value / 2));

    }
}
```

Write text content

Demo

101implementation:csharpLinqToXml

In-memory XML processing

in C#

with LINQ to XML

Total salaries *with* ***LINQ to XML***

```
public static decimal Total(XDocument xml){

  return xml.Descendants("Salary").

          Sum(salary => decimal.Parse(salary.Value));

}
```

Anonymous function

Read text content

Cut salaries *with* ***LINQ to XML***

```
public static XDocument Cut(XDocument xml)
{
   foreach (var salary in xml.Descendants("Salary"))
      {
         salary.Value =
            (decimal.Parse(salary.Value)/2).ToString();
      }
      return xml;
}
```

Demo

101implementation:pyjson

Processing JSON-based data

in Python

Companies *in* ***JSON***

```
{
  "name" : "Acme",
  "departments" : [
    {
      "name" : "Research",
      "manager" : {
        "name" : "Fred",
        "salary" : 88888
      }
    },
    {
      "name" : "Development",
      "manager" : {
        "name" : "Marie",
        "salary" : 77777
```

Total salaries *with* ***Python***

```
import sys
import json

def total(object):
    global result
    if "salary" in object:
        result += object["salary"]
    else:
        if "manager" in object:
            total(object["manager"])
        if "departments" in object:
            for d in object["departments"]:
                total(d)
        if "employees" in object:
            for e in object["employees"]:
                total(e)

company = json.load(open(sys.argv[1], 'r'))
result = 0
total(company)
print result
```

Test for key

Read by key

Convert external JSON into dictionary

Cut salaries *with* ***Python***

```
import sys
import json

def cut(object):
    if "salary" in object:
        object["salary"] /= 2
    else:
        if "manager" in object:
            cut(object["manager"])
        if "departments" in object:
            for d in object["departments"]:
                cut(d)
        if "employees" in object:
            for e in object["employees"]:
                cut(e)

company = json.load(open(sys.argv[1], 'r'))
cut(company)
open(sys.argv[2], 'w').write(json.dumps(company))
```

Demo

101implementation:mySql

SQL-based implementation

tailored to MySQL

Company schema *in* ***SQL DDL***

```
CREATE TABLE IF NOT EXISTS company (
   id INTEGER AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS department (
   id INTEGER AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   cid INTEGER NOT NULL,
   did INTEGER,
   FOREIGN KEY (cid) REFERENCES company(id)
      ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (did) REFERENCES department(id)
      ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS employee (
   ...
);
```

Total salaries *with* ***SQL DML***

```
SELECT SUM(salary) FROM employee

 WHERE cid =

   (SELECT id FROM company

    WHERE name = "meganalysis");
```

Select a specific company

Cut salaries *with* ***SQL DML***

```
UPDATE employee
 SET salary = salary / 2
 WHERE cid =
  (SELECT id FROM company
   WHERE name = "meganalysis");
```

Demo

101implementation:hibernate

Object/Relational mapping

for Java and SQL/HQL

with Hibernate

Class-table mapping *for* ***Hibernate***

```
<hibernate-mapping>
   <class name="org.softlang.company.Company" table="COMPANY">
      <id name="id" column="ID">
         <generator class="native"/>
      </id>
      <property name="name"/>
      <set name="depts" cascade="all">
         <key column="COMP_ID"/>
         <one-to-many class="org.softlang.company.Department"/>
      </set>
   </class>
</hibernate-mapping>
```

field/column level

Hibernate-based POJOs

```
public class Company {
   private Long id;
   private String name;
   private Set<Department> depts;

   public Long getId() { return id; }
   private void setId(Long id) { this.id = id; }
   public String getName() { return name; }
   public void setName(String name) { this.name = name; }
   public Set<Department> getDepts() {
      if (depts==null)
         depts = new HashSet<Department>();
      return depts;
   }
   private void setDepts(Set<Department> depts) {
      this.depts = depts;
   }
}
```

Total salaries *with* ***Hibernate***

```
public class Total {
   public static double total(Company company) {
      double total = 0;
      for (Department dept : company.getDepts())
         total += total(dept);
      return total;
   }
   public static double total(Department dept) {
      double total = 0;
      for (Employee employee : dept.getEmployees())
         total += total(employee);
      for (Department subDepartment : dept.getSubdepts())
         total += total(subDepartment);
      return total;
   }
   public static double total(Employee employee) {
      return employee.getSalary();
   }
}
```

A family of static methods

Cut salaries *with* ***Hibernate***

```
public class Cut {
   public static void cut(Company company) {
      for (Department dept : company.getDepts())
         cut(dept);
   }
   public static void cut(Department dept) {
      for (Employee employee : dept.getEmployees())
         cut(employee);
      for (Department subDepartment : dept.getSubdepts())
         cut(subDepartment);
   }
   public static void cut(Employee employee) {
      employee.setSalary(employee.getSalary() / 2);
   }
}
```

Demo

101implementation:ef

Object/Relational mapping

with the .NET Entity Framework

Class-table mapping for ***EF***

```
<edmx:Edmx Version="2.0" xmlns:edmx="http://schemas.microsoft.com/ado/2008/10/edmx">
  <!-- EF Runtime content -->
  <edmx:Runtime>
    <!-- SSDL content -->
    <edmx:StorageModels>...</edmx:StorageModels>
    <!-- CSDL content -->
    <edmx:ConceptualModels>...</edmx:ConceptualModels>
    <!-- C-S mapping content -->
    <edmx:Mappings>
      <Mapping Space="C-S" xmlns="http://schemas.microsoft.com/ado/2008/09/mapping/cs">
        <EntityContainerMapping StorageEntityContainer="CompanyModelStoreContainer" CdmEntityContainer="CompanyDataContext">
          <EntitySetMapping Name="Companies">
            <EntityTypeMapping TypeName="CompanyModel.Company">
              <MappingFragment StoreEntitySet="Company">
                <ScalarProperty Name="company_id" ColumnName="company_id"/>
                <ScalarProperty Name="Name" ColumnName="Name"/>
              </MappingFragment>
            </EntityTypeMapping>
          </EntitySetMapping>
          <EntitySetMapping Name="Departments">...</EntitySetMapping>
          <EntitySetMapping Name="Employees">...</EntitySetMapping>
          <EntitySetMapping Name="People">...</EntitySetMapping>
        </EntityContainerMapping>
      </Mapping>
    </edmx:Mappings>
  </edmx:Runtime>
  <!-- EF Designer content (DO NOT EDIT MANUALLY BELOW HERE) -->
  <Designer xmlns="http://schemas.microsoft.com/ado/2008/10/edmx">...</Designer>
</edmx:Edmx>
```

A more complex mapping

EF-based POJOs

```
[EdmEntityTypeAttribute(NamespaceName="CompanyModel", Name="Company")]
[Serializable()]
[DataContractAttribute(IsReference=true)]
public partial class Company : EntityObject
{
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=false)]
    [DataMemberAttribute()]
    public global::System.String Name
    {
        get { ... }
        set { ... }
    }
    private global::System.String _Name;
    partial void OnNameChanging(global::System.String value);
    partial void OnNameChanged();

    [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
    [DataMemberAttribute()]
    public global::System.Guid company_id
    {
        get { ... }
        set { ... }
    }
}
```

Annotations to represent mapping

Total salaries *with* ***EF***

```
public static decimal Total(this CompanyDataContext company)
{
   var res = company.Employees.Sum(emp => emp.Salary);
      if (res != null) return (decimal) res;
   return 0M;
}
```

Cut salaries *with* ***EF***

```
public static void Cut(this CompanyDataContext company)
{
      foreach (var employee in company.Employees)
      {
         employee.Salary /= 2;
      }
   company.SaveChanges();
}
```

BTW: Consider 101companies' value for knowledge acquisition

- 101companies contribution *X* ...
  - ... uses language *L*,
  - ... uses technology *T*,
  - ... implements feature *F*,
  - ... demonstrates concept *C*.

Direct, declared knowledge

Indirect, inferrable knowledge

- Technology *T* helps with feature *F*.
- Technology T_1 always occurs together with technology T_2.
- Developer *D* has skills regarding language *L* and technology *T*.
- ...

Summary

You have seen ...

- different styles of XML processing,
- some basics of relational databases,
- some bits of object/relational modeling,
- and the systematic utilization of *101companies*.

Thanks for your interest.

Questions? Comments?