Code Generator for Web Service

Name: Weiwei Hou
ID: U5202546
Supervisor: Dr. Shayne Flint
Outline

• Introduction
  – MDE
  – tasks
    • Database configuration scripts
    • Data service scripts

• Objectives

• Preparation
  – SQL
  – Executable UML
  – Jersey RESTful web framework

• Implementation

• Testing

• Conclusion
Introduction

• Model Driven Engineering (MDE)
  – Traditional MDE (increase level of abstract, need to know business content)
  – Domain Specific Modeling (only works on single products)
• Aspect Oriented Thinking (AOT)
• MediChart
  – an eHealth application supporting end-to-end medication management
  – Class Model Diagram
• AOT Software
  – Code generator for web service
    • Database configuration scripts
    • Data service scripts
Objectives

– Improved understanding of the web services
– Improved understanding of modeling and code generation
– Improved understanding of software engineering practice within a research context
– Improved technical communication skills
AOT

- AOT architecture
Preparation

• SQL statement
  – CREATE table, SELECT, DELETE, INSERT and UPDATE

• Executable UML Class Action
  – Query instances within class model diagram
    • Create new instance of a class
    • Delete instance of a class
    • Select one|many instances from a class
    • Query instance across an association

• Jersey
  – A pure java RESTful web framework
    • Parameters (pathparam, formparam)
    • HTML method (GET, PUT, POST, DELETE)
Design Architecture

• Code generator for web service
  – Data Layer -> Database configuration scripts
  – Business Layer -> web service scripts include the query statements
  – Model Layer -> representation classes scripts
Implementation

• Database configuration scripts
  – Classes->tables
  – Automatically add primary key
  – attributes->table attributes
  – Associations->table relationships
    • Many to many -> add extra table, use PK of both tables as FK of new table
    • One to many -> A has many B, A’s PK is B’s FK
    • Zero to one -> A has zero to one B, A’s PK is B’s FK
    • One to one -> A has one and only one B, either A or B’s PK can be another’s FK
## Design Query

<table>
<thead>
<tr>
<th>Query Name</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select</strong></td>
<td>`selectOne</td>
</tr>
<tr>
<td><strong>Create Object</strong></td>
<td><code>createObjectInstance</code>&lt;br&gt;<code>forClass &lt;classModel:/class:&gt;</code>&lt;br&gt;<code>parameter &lt;webService:/parameter:&gt;</code></td>
</tr>
<tr>
<td><strong>Update Object</strong></td>
<td><code>updateObjectInstance</code>&lt;br&gt;<code>forClass &lt;classModel:/class:&gt;</code>&lt;br&gt;<code>parameter &lt;webService:/parameter:&gt;</code>&lt;br&gt;<code>where &lt;whereQuery&gt;</code></td>
</tr>
<tr>
<td><strong>Delete Object</strong></td>
<td><code>updateObjectInstance</code>&lt;br&gt;<code>fromClass &lt;classModel:/class:&gt;</code>&lt;br&gt;<code>where* &lt;whereQuery&gt;</code></td>
</tr>
<tr>
<td><strong>Where</strong></td>
<td><code>where</code>&lt;br&gt;<code>parameter &lt;webService:/parameter:&gt;</code>&lt;br&gt;<code>select* &lt;selectQuery&gt;</code></td>
</tr>
</tbody>
</table>
Web Service Class

```java
WebServiceClass webService1
  pathName address
  webServiceMethod getAddressInfor
    pathName infor
    parameter [city]
    toQuery [q1]
    representationType [XML]

Query q1
  SelectOne s1
    AttributeOfAnInstance [address/type]
    Where [w1]
  Where w1
    Parameter [WebService1/city]

@Path("/address")
public class webService1
{
  @GET
  @Path("/infor")
  @procudure(XML)
  Public address getAddressInfor(
    @pathParam city String city)
  {
    sqlQuery="select type from address
      where city=?";............
    Return aAddress;
  }
}
Test

- Test database scripts
  - Three classes and two associations

```sql
CREATE TABLE Person(
P_Id int NOT NULL,
PRIMARY KEY (P_Id))
;
CREATE TABLE Address(
P_Id int NOT NULL,
PRIMARY KEY (P_Id),
type enum('Residence', 'Postal', 'FamilyHome', 'Billing',
'Business', 'BusinessPostal'),
city varchar(255))
;
CREATE TABLE ContactNumber(
P_Id int NOT NULL,
PRIMARY KEY (P_Id),
type enum('BusinessEmail', 'PrivateEmail', 'BusinessFax',
'HomeFax', 'BusinessPhone', 'HomePhone', 'MobilePhone'),
number varchar(255),
F_Id_Person int NOT NULL,
CONSTRAINT FOREIGN KEY (F_Id_Person) REFERENCES Person(P_Id)
on DELETE CASCADE on UPDATE CASCADE)
;
CREATE TABLE PersonAddress(
P_Id int NOT NULL,
PRIMARY KEY (P_Id),
F_Id_Person int NOT NULL,
CONSTRAINT FOREIGN KEY (F_Id_Person) REFERENCES Person(P_Id)
on DELETE CASCADE on UPDATE CASCADE,
F_Id_Address int NOT NULL,
CONSTRAINT FOREIGN KEY (F_Id_Address) REFERENCES Address(P_Id)
on DELETE CASCADE on UPDATE CASCADE)
;```
Test Web Service

- Test GET
  - Run URL in web browser
- Test POST
  - Write HTML page for submitting form parameter
- Test PUT, DELETE
  - JUnit Test

```java
public class AddressServiceTest {
    private String url = "http://localhost:8080/jerseywtest/rest/address";

    @Test
    public void testPut() {
        Client client = Client.create();
        WebResource webResource = client.resource(url + "addAddress/Canberra");
        ClientResponse response = webResource.put(ClientResponse.class);
        System.out.println("Response for put request: " + response.getStatus());
    }

    @Test
    public void testDelete() {
        Client client = Client.create();
        WebResource webResource = client.resource(url + "deleteAddress/Canberra");
        ClientResponse response = webResource.delete(ClientResponse.class);
        System.out.println("Response for delete request: " + response.getStatus());
    }

    @Test
    public void testGet() {
        Client client = Client.create();
        WebResource webResource = client.resource(url + "getAddress/Canberra");
        ClientResponse response = webResource.get(ClientResponse.class);
        System.out.println("Response for get request: " + response.getStatus());
    }
}
```
Conclusion

• MDE
  – Traditional MDE, DSM, AOT

• Code generator for web service
  – Database configuration scripts
  – Web service scripts

• Future work
  – Improve the primary key setting of each table