Università degli Studi di Pisa

Laurea Specialistica in Informatica

Laurea Specialistica in Informatica e Networking

Advanced Programming

Final Term Paper

Start Date: 10/06/2013
Submission deadline: 1/07/2013 (send a single PDF file to attardi@di.unipi.it)

Rules:

The paper must be produced personally by the student, signed implicitly via his mail address.

You are allowed to discuss with others the general lines of the problems, provided that each student eventually formulates his own solution. Each student is expected to understand and to be able to explain his solution.

You are allowed to consult documentation from any source, provided that references are mentioned.

It is not considered acceptable:

- **to consult or setup an online forum, to request help of consultants in producing the paper**
- to develop code or pseudo-code with others
- to use code written by others
- to let others use someone's code
- to show or to examine the work of other students.

Violation of these rules will result in the cancellation of the exam and a report to the Presidente del Consiglio di Corso di Studio.

For the programming exercises you can choose a programming language among C++, C# and Java.

The paper must:

1. be in a single PDF file, formatted readably (**font size ≥ 10 pt** with suitable margins, single column), of **no more than 10 numbered pages**, including code: for each extra page one point will be subtracted from the score.
2. include the student name
3. provide the solution and the code for each exercise separately, referring to the code of other exercises if necessary. **Do not include in an exercise code only needed for a later exercise**.
4. cite references to literature or Web pages from where information was taken.

Introduction

In this project, you will develop a JET template generator. JET allows defining an abstract data model like this:

```
<model name="Category">
      <column name="id" type="int" />
      <column name="name" type="String" />
      <operation type="list" />
      <operation type="create" />
</model>
```

which can be turned to code by using transformations like this:

```
public <c:get select="/model/@name" /> create(
   <c:iterate select="/model/column"
              var="currColumn" delimiter=", " >
```

```
      <c:get select="$currColumn/@type" />
      <c:get select="$currColumn/@name" />
   </c:iterate>
) {
   <c:get select="/model/@name" /> obj =
      new <c:get select="/model/@name" />(
   <c:iterate select="/model/column" var="currColumn"
                delimiter=", ">
         <c:get select="$currColumn/@name" />
   </c:iterate>

   <c:get select="/model/@name" />List.add(obj);
   return obj;
}
```

Obtaining this:

```
public Category create(int id, String name) {
   Category obj = new Category(id, name);
   categoryList.add(obj);
   return obj ;
}
```

A JET template is a text documents with embedded JET transformations. Notice that this is different from XSLT which applies to XML documents.

The JET transformation elements may contain simplified XPath expressions made of either:

- constant names (e.g. `column`)
- attribute names (e.g. `@type`)
- variables (e.g. `$currColumn`) that are set through attribute `var` in a `c:iterate` tag to successive items through the iteration.

The `c:get` element produces as value the first item matching the select clause, typically an attribute name. The `c:iterate` applies its body to all elements matching the select and binds the variable named in the `var` attribute to each such element in turn.

Exercise 1

Design a set of classes to represent JET models and JET templates.

Exercise 2

Implement a recursive descent parser for JET models and templates using a streaming reader like XmlTextReader. Ensure that each parser method returns a parse tree for the input it analyzes.

Exercise 4

Write a JET generator, which, given a JET model and a JET template, will produce the result of the transformations applied to the model.

Exercise 5

Extend JET with a conditional construct so that the above example can be included in:

```
<c:if test="/model/operation/@type = 'create'" >
public <c:get select="/model/@name" /> create(
…
</c:if>
```

Exercise 6

Use JET to define classes for songs and music category. Then provide a JET template that creates an HTML page to list a set of songs grouped by category.

Exercise 7

Describe LINQ and in particular its use of lambda expressions. Are the lambda expressions in LINQ lexical closures? If so, show an example in which they are and on e in which they are not.