Preparation for Miniproject Part 2 - Examples of DSLs

This is a companion list of example DSLs that you can use to propose the topic of your project. The list of example topics below are annonated with the level of challenge, in order to guide your team in proposing a DSL with the right expectations:

⭐ : aiming for application (satisfactory) ⭐⭐ : aiming for competence (competent) ⭐⭐⭐ : aiming for excellence (excellent) ⭐⭐⭐⭐ : aiming for mastery (outstanding)

For further information on what is expected for each exercise, according to its level of achievement, please check the rubric.

Example 1: Support for markdown

Abstract: Markdown is a lightweight markup language with plain-text-formatting syntax. Markdown is often used for formatting readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor. The goal of this topic is to automatically translate markdown files to HTML documents. Examples can be found in this webpage.

Depending on how many features of the markdown language are covered the team could be aiming for different levels of achievement:

- ⭐ Basic markdown is covered (headings, sections, bold/italic/strikethrough/plain text)
- ⭐⭐ Additional features are considered: lists, quotes, tables, images.
- ⭐⭐⭐ A large extent of the language is covered, supporting features like footnotes, emoticons, code snippets tagged with their programming language (and their translation to HTML). Keywords in code snippets could be coloured in the output HTML using external libraries.
- ⭐⭐⭐⭐ A full variant of markdown (e.g. GitHub Flavored Markdown or Markdown Extra) is covered. Your team may have identified shortcomings of existing versions and may have proposed an improved version of the language (and of HTML generation).

A variant of this project is to generate other types of documents, e.g. LaTeX. The compiler needs to be implemented in Groovy, existing libraries cannot be reused to automate the translation.

Example 2: Generation of bibliographies in BibTex ⭐

Abstract: BibTeX is reference management software for formatting lists of references. The BibTeX tool is typically used together with the LaTeX document preparation system. In our DSL, we can define different types of documents (books, articles and miscellaneous) and we are going to generate an appropriate BibTex entry for each document, according to their type.

For example:

```
Book <LOTR:chapter1#10,chapter2#13,chapter3#0>
```

would be translated into

```
@book{
  title={LOTR}
}

@incollection{
  title={chapter1},
  booktitle={LOTR}
  pages={10}
}

@incollection{
  title={chapter2},
  booktitle={LOTR}
  pages={13}
}

@incollection{
  title={chapter3},
  booktitle={LOTR}
  pages={0}
}
```

This example follows from an open exercise in the worksheets of the second part of the module. Your group can reuse it and expand it with four or five types of publications.

Example 3: Creating databases from class diagrams ⭐⭐

Abstract: The goal of this DSL is to model class diagrams using PlantUML notation and to generate DDL scripts that can be used to create a relational databases. The DSL will cover classes, attributes and associations and will translate them into SQL DDL scripts. Classes will be translated to tables, attributes to columns and associations to foreign keys. Only one-to-one associations will be considered.

Example:

Source:

```
@startuml

class dummy {
  Sample table.
  ==
  #id int(10) -- A comment
  field1 int(10)
  .. Comment line, ignored ..
  field2 varchar(128)
}

@enduml
```

Target:

```
CREATE TABLE IF NOT EXISTS `dummy` (
  id               INT(10) COMMENT 'A comment',
  field1           INT(10),
  field2           VARCHAR(128),
  PRIMARY KEY (id));
```

This DSL is inspired in this utility.

Example 4: Generating ER diagrams from DDL scripts ⭐⭐

Abstract: The goal of this DSL is to model class diagrams using PlantUML notation and to generate DDL scripts that can be used to create a relational databases. The DSL will cover classes, attributes and associations and will translate them into SQL DDL scripts.

This DSL is inspired in Planter.

This is a typical example of low-code development platform with reverse engineering facilities. In this case for visualising code.

Example 5: A DSL to define functional tests ⭐⭐⭐

Abstract: Gherkin is a DSL for defining functional tests. It has a very flexible syntax that makes it look almost like free text. Basically developers, analysts and clients can sit around a table and define some scenarios. These scenarios will be then executable as tests, to verify whether the application meet the expectations. In this topic, the goal is to compile Gherkin specifications: Feature, Rule, Scenario, steps (Given, When, Then, And, But) and Background.

For example, given the following scenario

```
Scenario: Some cukes
  Given I have 48 cukes in my belly
```

the compiler generates test case in Java as follows:

```
package com.example;
import io.cucumber.java.en.Given;

public class StepDefinitions {
    @Given("I have {int} cukes in my belly")
    public void i_have_n_cukes_in_my_belly(int cukes) {
        System.out.format("Cukes: %n\n", cukes);
    }
}
```

For more information on this transformation, have a look at how Cucumber uses Gherkin to support behaviour-driven development.

By implementing more features of Gherkin (Outlines and Expressions), you would aim for a mastery level.

Example 6: Cloud DSL: A Language for Supporting Cloud Portability by Describing Cloud Entities 

Abstract: Different cloud platforms offer similar services with different characteristics, names, and functionalities. Therefore, describing cloud platform entities in such a way that they can be mapped to each other is critical to enable a smooth migration across platforms. In this paper, we present a DSL that uses a common cloud vocabulary for describing cloud entities covering a wide variety of cloud IaaS services. Through analysis of existing cloud DSLs, we advocate that our cloud DSL is more expressive for the purpose of describing different cloud IaaS services. In addition, when used along with TOSCA, our preliminary analysis sug- gests that our Cloud DSL significantly reduces the workload of creating cloud descriptions in a TOSCA specification.

Example 7: A Scalability Rule Language for Multi-cloud Environments (SRL) 

Abstract: SRL enables the specification of rules that support complex adaptation scenarios of cross-cloud applications. In particular, SRL provides mechanisms for specifying cross-cloud behaviour patterns, metric aggregations, and the scaling actions to be enacted in order to change the provisioning and deployment of an application. SRL provides mechanisms for (a) specifying event patterns, (b) specifying scaling actions, and (c) associating these scaling actions with the corresponding event patterns. Such rules will be compiled to scaling plans using the AWS Auto Scaling API.

Example 8: A WordPress plugin configuration DSL 

Abstract: A simple DSL (Domain-Specific Language) to generate scaffolding code for WordPress Plugins. The DSL will contain

- Plugin Name
- General Options to populare the file's headers (author, author URI, description,...)
- Whether your plugin has an admin view, a public view or both
- Menu information
- Settings information

From a Wordpress page description the model compiler generates a ready-to-use plugin that can be moved to a WordPress installation, where it can be activated. The core plugin information is used to initialize the plugin code according to the WordPress Boilerplate Project.

List of DSLs developed by researchers/professionals 

The following are examples of outstanding DSLs, with which you can aim for a top-notch mark:

- A DSL to define schemas for MongoDB databases (Xtext example)
- CAMEL-DSL: Cloud Application Modelling and Execution Language (CAMEL)
- SRL: A Scalability Rule Language for Multi-cloud Environments

- An HTTP routing language (Xtext example)
- A simple scripting language (Xtext example)
- A DSL for WordPress (Xtext example)
- Vorto: a DSL for digital twins (Xtext example)
- The Spatiotemporal Epidemiological Modeler (STEM) Project (Xtext example)
- A Domain Specific Language to generate videos based on video libraries and some math (Xtext example)
- PlantUML (Xtext example)
- Parsley: a DSL for generating UIs (Xtext example)
- Mita: a DSL for IoT (Xtext example)
- Xtext DSLs built by the community (Xtext examples)
- Cloud DSL
- A DSL for deployment and testing on the cloud
- CloudML : A DSL for model-based realization of applications in the cloud
- Smart contracts
- DSL for testing software using behaviour-driven development (SpecFlow) (see also: Gherkin, and Spock)
- RoboticsDSL zoo
- Modelling user interfaces
- Financial domain-specific languages
- Some examples from wikipedia
- Some examples from Strumenta
- Awesome open source DSLs

Please see the rubric for further information.