
## Introduction

This paper documents Word Cards, an interactive learning game created during the 2011 .NET Code Camp, held in Lappeenranta University of Technology. The paper is structured into four parts: it first explores the pedagogic problem behind the software and then proceeds to explain the solution, use case and its architecture. The last part summarizes the project in addition to proposing future improvements.

## Problem

According to U.N. about 69 million school-age children are not in school. Some of the factors which contribute to a child's access to education include location, gender and cost.

An important part of education is learning new words which in turn are a fundamental part of any language. Knowing the meanings of individual words not only enables us to communicate but also makes it possible to consume and produce written material. An integral part of this learning process is the ability to bind words with different meanings. This learning process should be made accessible through interactive experiences which engages and motivates us.

## Solution

Our solution addresses the 2nd Millennium Development Goal (achieving primary education for everyone) by providing a location-independent, gender-neutral and cost-effective learning tool that enables learners to associate words with their meanings, either through other languages, words or images.

Word Cards enables this learning process by providing a framework for dynamically loading association dictionaries that provide association mappings. The mappings can be presented as combinations of words and images, e.g. from image to image or from image to word.

## Use Case

Word Cards is mainly targeted for school-aged children, but it can also have a much broader potential user group. Association dictionaries can be tailored for special user groups such as doctors who want to learn for example the names of bones in Latin.

The game is initially started by selecting the source of the association dictionary. The source can either be a self-made and locally stored dictionary or a web service based dictionary. After the association dictionary has been loaded, the game can be started and a random description/image is presented to the user. For example an image of a cat is displayed and the user (in this case a child) tries to guess what word is associated with the image by typing the word (cat) into a textbox. The number of letters in the word is given as a hint. After the user has made a wrong guess, the correct word is shown so the user can learn from the mistakes. The number of correct answers and wrong guesses is also displayed.

## Architecture

Word Cards is implemented as .NET class libraries with initial graphical user interface (GUI) implementations in Windows Presentation Foundation (WPF), Windows Forms (Winforms) and Silverlight. Thus, also the core framework has .NET (CLR 4.0) and Silverlight (version 4) implementations. The framework (see Figure 1) provides a pluggable model for dynamically loading dictionaries by scanning assemblies for predefined interfaces, shown in Figure 2 and Figure 3.

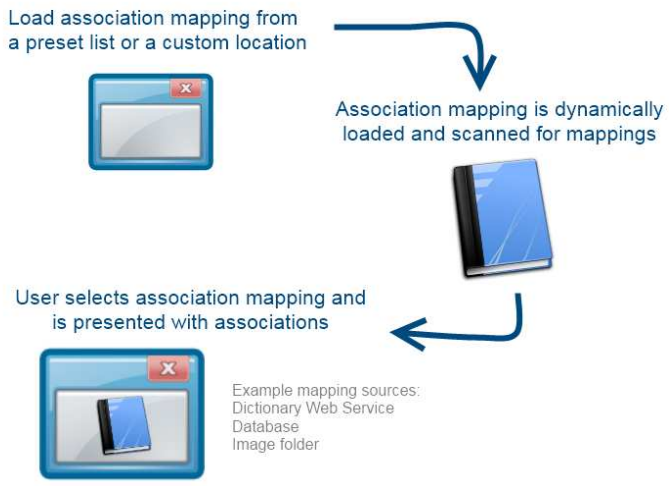


Figure 1. Core framework.

This allows developers to create rich dictionaries that can harness different association sources. For example, during the code camp, a REST web service based dictionary was created to fetch words and their descriptions in real-time from an online dictionary. Other association examples implemented during the code camp are the text-file based dictionary, where associations are loaded from a text-file, as well as image based dictionaries, where associations are loaded from image and the filename respectively.

**Public Member Functions**

| | |
|---|---|
| void | **BootStrap** () |
| IEnumerable < IQuestionAnswerAssociation > | **RandomMultipleAssociation** (int count) |
| IQuestionAnswerAssociation | **RandomSingleAssociation** () |

**Properties**

| | |
|---|---|
| string | **Description** [get] |
| string | **Name** [get] |

Figure 2. Interface for .NET 4.0 dictionary.

**Public Member Functions**

| | |
|---|---|
| IAsyncResult | **RandomMultipleAssociationBegin** (int count, AsyncCallback callback, object userstate) |
| IEnumerable < IQuestionAnswerAssociation > | **RandomMultipleAssociationEnd** (IAsyncResult asyncResult) |
| IAsyncResult | **RandomSingleAssociationBegin** (AsyncCallback callback, object userstate) |
| IQuestionAnswerAssociation | **RandomSingleAssociationEnd** (IAsyncResult asyncresult) |

**Properties**

| | |
|---|---|
| string | **Description** [get] |
| string | **Name** [get] |

Figure 3. Interface for Silverlight 4 dictionary.

In addition to .NET 4.0 base class libraries (BCL) and Silverlight 4 that were used in the base framework implementation, Windows Forms, WPF and Silverlight were used in the GUI creation. Finally, Autofac was used as a dependency injection container (DI container).

The pluggable model in the framework implementation allowed for flexible design of user interfaces. User interface designer does not need to have knowledge about underlying structure and can directly use association base classes. They provide the description and name of the association, as well as several different types of associations, both text and images.

## Conclusions

While only initial GUI implementations were done during the code camp, the base framework proved to be extensible and flexible as planned. More importantly, the solution demonstrated potential to assist the learning process of new words, thus providing a solution to the presented problem.

To take the project and learning process further, the GUI implementations should be finished. For example a countdown clock could be implemented to the user interface to add more urgency thus making the game more challenging. Overall user experience could be enhanced by adding difficulty levels according to the user groups. Word Cards could later also be integrated to Facebook to provide social context to it.

Furthermore, more dictionaries should be created to assist in learning different languages through the use of textual descriptions and imagery. User-friendly dictionary editors could be implemented to make the adding of new words easier.