Development of an Ontology Tutorial for TurtleEditor - Implementation of the :turtle:Board Framework

Gligor Tasev, Ludmila Casautan, Louai Haddad

Institute of Computer Science, University of Bonn
s6gltase@uni-bonn.de, s6lucasa@uni-bonn.de, s6lohadd@uni-bonn.de

Abstract. The number of tutorials for learning some of the most used RDF serializations is increasing every year. Many of the tutorials, however, are specifically made for editors which require installation before the actual completion of the tutorial. The purpose of this lab project is to introduce a new way of learning one of the most important RDF serialization - Turtle. The project was developed based on the idea "learning by doing". It involves implementation of an online JavaScript tutorial framework for the Turtle editor. The learner/user of this framework is guided through different tutorials step-by-step. The framework helps the user by giving examples, hints and displaying user's mistakes.

1 Introduction

The representation of data using graphs is in expansion in the last years because of the huge amount of data which is around us. When the Resource Description Framework (RDF) was introduced, the representation of meta data was brought to a whole new level. RDF graphs can be represented using different serializations, however Turtle is one of the most important RDF serializations because of its structure. It is very easy to learn and very human-readable which makes it one of the most used serializations on the market. Turtle is also used as a basis for building different libraries and software packages by Semantic Web developers. The need of learning Turtle comes up as a follow-up from the usage of Turtle in the industry and scientific institutions. Although Turtle is very easy to grasp, mastering it involves certain effort. There are many Turtle tutorials available and many tools that support the Turtle format (.ttl). Majority of these tutorials though are constructed for a specific editor which requires local installation in advance. Here comes the motivation behind this problem - the implementation of a specific online tutorial framework which incorporates an editor in itself and requires zero installation costs. The development of the so called :turtle:Board framework (as we used to name our framework) involves not only step-by-step guidance through Turtle tutorials, but it also introduces a concept of communication between the learner and the framework by giving the learner certain feedback. The user is able to follow a tutorial, be guided through it, get help and see his/her mistakes, and what is the most important thing - learn from those mistakes by doing more examples.

2 Background

The concept of e-learning is one of the most used pedagogic methods nowadays. People are more and more spending time on their machines sitting, rather than doing physically some things, like reading a physical book. The same goes for learning programming, scripting or markup languages. In the past, people who wanted to learn some language, they would take a book and learn some of the most important concepts. Those core changes in the learning methods have constructed the need of introducing a tutorial framework. As we mentioned before, the needs of new learners go even further. The users' satisfaction shows that if the process of learning something new involves some preparation work (such as pre-installation of a desktop application/editor), then the users will not be so eager to learn. The learners want something that involves all of the needed concepts in one thing, at one place, so that they can concentrate on the specific learning intentions. Our solution is based exactly on these principles. The framework that we developed encourages the learner to learn by doing and by constantly informing him/her what is needed to get to the right solution.

3 Related Work

Currently, one the most used editors for learning Turtle is Protege, a free, open-source ontology editor. Using Protege, the learners can create classes and properties. However, it is just an editor. The learners have to do the tutorials by themselves and there are no examples. We can say that Protege is good for developing certain ontology, but the process of doing that is conducted offline. The editor also cannot be shared. Each learner has to install the editor first locally, in order to write certain data graphs. Protege also has poor didactic methods in terms of helping the user to learn and see what are his/her knowledge gaps. The same goes for the XTurtle editor. Speaking about tutorials though, we can say that there is a huge number of tutorials for Turtle. However, most of them involve just the theoretic side of learning the basic concepts of Turtle. The practical side is often excluded in these online tutorials because there is no editor on most of these website platforms. On the other hand, the number of validators and translators is surprisingly big. The solution of the problem lies exactly here: to connect certain editor and tutorial in an online framework package.

4 Proposed Approach

Before coming up with the strategy for implementation of our framework, we decided that firstly we are going to explore some of the existing tutorials for Turtle in order to get an idea what are the main learning concepts needed to be included in a tutorial, no matter if it is done online or offline. Secondly, we examined the Turtle editor's source code in order to see what is the logic behind this editor. The editor is written in JavaScript using different RDF libraries. This was important because we wanted to see how this editor works, so that we can incorporate some of our new functionality into this editor. Last, but not least we reviewed some JavaScript frameworks for getting a quick overview into what JavaScript can offer in terms of possibilities regarding software development.

4.1 Problem Statement

As previously mentioned, there are many ontology tutorials. However, all of them are offline and can be executed only on an installed ontology editor. This way of learning is characterized with lack of step-by-step guidance, feedback and poor didactic methods. Learners also do not have the possibility to use hints or help during the learning process. The classification of mistakes and errors in both syntax and semantics terms are also poor which leads to lower benefits for the learner himself/herself.

4.2 Proposed Solution

Our solution consists of different parts which combined together build the online tutorial framework called :turtle:Board. Our framework is based on the implementation of a chosen tutorial by the user into the Turle Editor which was previously developed separately. The framework is unique as it puts together different didactic parts which are crucially important for learning Turtle.

The framework has a page called *Fundamentals* which is a page where the most important concepts of Turtle are being theoretically described, in the same way as it is done in the other tutorials which do not come together with an editor. Furthermore, the user can make a choice by selecting one of the three tutorials offered. The initial plan was to implement just one tutorial into this framework, but we decided as a team that it is better when there is a variety of difficulty knowledge levels. Depending of the previous knowledge of the user, he/she can choose to follow up the tutorial designed for beginners, for intermediate or for advanced learners. When certain tutorial has being selected, then the actual tutorial page opens up. The tutorial page incorporates the Turtle Editor, as well as the specific boxes for guidelines, examples, tasks, hints, answers, feedback comments and score table. All of the functionality of this framework is built as a part of the Turtle Editor which makes it really easy to work on the user's input. The advantage of our framework is that it has everything in one place. Moreover, it is an online framework, which means that there are no installation costs. :turtle:Board framework is the perfect framework for learning ontology engineering.

4.3 Implementation

Our framework consists of three different parts: front-end, back-end and data files. If we analyze the structure of our framework using MVC diagram then the front-end part is the Viewer part, the back-end part is the Controller, while the Model part is the communication between the front-end and the back-end, as well as the communication between the framework and the stored data files.

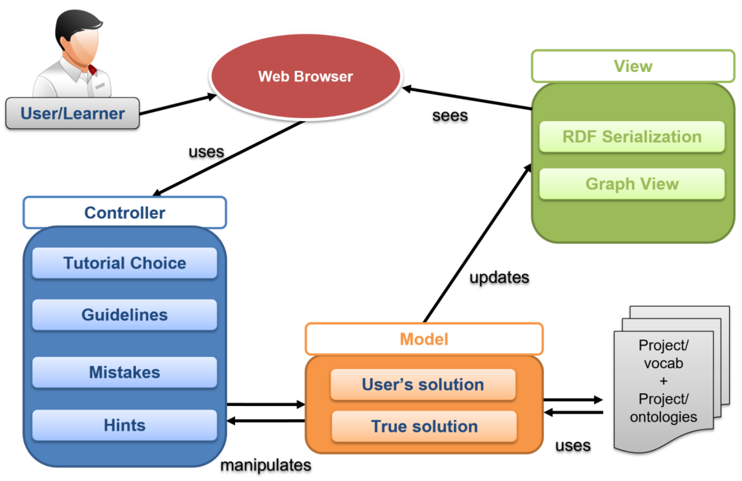

Fig. 1. Model-Viewer-Controller Diagram representing the :turtle:Board framework

The front-end part is basically the visual side of the framework. We used HTML and CSS, as well as PHP for dealing with the session variables which are important for keeping sessions' records. The back-end part is basically the brain of the framework and it was written in JavaScript. As we have already mentioned, our major functionality software solutions are part of the Turtle editor, because it was easier for us to work with the user's input inside the editor. Some of the major functionality include: clearing the editor; getting the prefixes; getting guidelines together with examples and tasks; getting hints; checking user's input; and displaying answers, comments and scores. In the early phases of the development of our project, we stored our data into text files. However, later on, after broad discussions with our mentor, we decided to change the format of the stored files. Therefore all of the stored files in our project are Turtle files. These files were validated by using an online free Turtle validator (4). To be more precise, the data files of our project include two sets of files: solutions and other data. The solutions of each tutorial stored as (.ttl) files are used in the comparison function for checking the correctness of the input. We have three solution files, one for each tutorial separately. The other data files are again a set of (.ttl) files, as we mentioned before. These files are used for all the displayed text for each tutorial level separately. In each of these files the steps of the tutorial are explained together with the other data elements (example explanation, the actual example, the task given to the user, the expected solution and the hint for the specific step). It is worth mentioning that the solution files are used for the second part of the checking function, while the expected solutions for each step are used for the ASK query in the first part of the checking function in which we check whether the triple exists at all or not.

The interesting part of this project was the parsing of the data files into the framework using JavaScript. We used different parsing techniques provided by certain RDF libraries which we had to modify to great extent in order to be able to read and work with the structures of the data files and their corresponding contents. All of the information parsed from the files was parsed and stored in arrays of different type depending on the type of the parsed information. Each of these arrays later were used in different functions for executing certain results.

The framework also involves smaller functions for clearing the editor, or reading the prefixes used in all of the tutorials. Furthermore, there is also computation of a session score which is just a sum of certain variables and counters that we decided to store. It is good to be mentioned that the rules behind these functions can be changed. In general, the changes in the content of our tutorials in any aspect can be easily done because of the definition of the structure of our data files. The testing of this framework was done by writing some unit tests and ontology samples which were used as measures for improvement of the core structure of the framework.

5 Use Cases

In our project we have several use cases, but in this report we will talk about the most important ones. The major actors in our framework are the user/learner and the developer. Each one of these actors has certain activities which are part of the functionality of this project. We will describe our use cases by specifying some of the actions that are done by the actors.

We will start with the user/learner. The user is actually any person that wants to learn the Turtle serialization or has an interest in RDF in general and wants to improve his/her Semantic Web knowledge. When the user opens the framework, he chooses which tabs he/she wants to open first. On the beginning, the user sees a page where the framework and its purpose are being described and introduced. Here, the user gets an idea of what is this framework about. Then he/she has the possibility to decide whether to start reading the fundamentals about Turtle or to start with the tutorials. After reading the Turtle fundamentals' page, he/she may start the desired level of the tutorial. If the user decides to go to the tutorial page immediately without reading the fundamentals, then on the tutorial page, the framework once again reminds him/her that it is recommended the fundamentals to be read before starting some of the tutorials. The user may open any of the tutorials (beginner, intermediate, advanced), based on his/her preferences. When a tutorial has been selected, then an info page opens up. On that page, the user has a chance quickly to get a glance of the most important things which he/she needs to know for completing the chosen tutorial. When the user clicks the start button, then the tutorial page opens up. On the tutorial page, the user sees the Turtle Editor opened which is integrated in this framework. The user has the possibility to follow the tutorial step by step or he/she may choose to see the final solution by executing all steps at once. The user gets a task that he/she needs to solve. Each task is one step. Every task is very well explained with a given example. The example that is given is very similar to the task that the user receives. While writing a triple in the Turtle Editor, the user gets hints, if he/she asks for them. These hints help the user when he/she has hard time to solve certain task. After writing down the triple, he/she clicks the check button. This button checks the user's input and if there are any mistakes, then the user gets a message in which is explained which part of the triple was wrong. After completing all the steps of the tutorial the user/learner may check the solution of the chosen tutorial by clicking the show final solution button. Finally, he/she gets a score for the completed tutorial. The score is based on certain parameters. Based on this score, the user can decide which tutorial he/she wants to do next. Then, the user chooses another tutorial. The same actions repeat for all three tutorials.

As mentioned above, the second important actor in our framework is the developer. After reviewing the Turtle Editor code, the existing JavaScript tutorial frameworks and existing Turtle ontology tutorials, the developer (in this project - the :turtle:Board team) has the ability to update certain functionality of this tutorial framework. The developer creates the levels of tutorials. Every level has one tutorial at this moment with the possibility more tutorials to be added when it is necessary or when it is desired. This is an advantage that comes up as a result of the very clear and readable structure of the data files used in the framework. These vocabulary files can be changed by the developer at any time. Therefore, new tutorials can be created very easily. These files are Turtle files. For every tutorial, the developer is the one who is creating a (.ttl) file with all the steps, all the tasks and examples, the hints, and the solution triples used for ASK queries. That is actually the process of the creation of tutorial. The developer can also edit the error messages that are prompted when a mistake is being made by the learner. Moreover, the developer can also edit the solution files which are also Turtle files. New triples can be added to these files as well, as

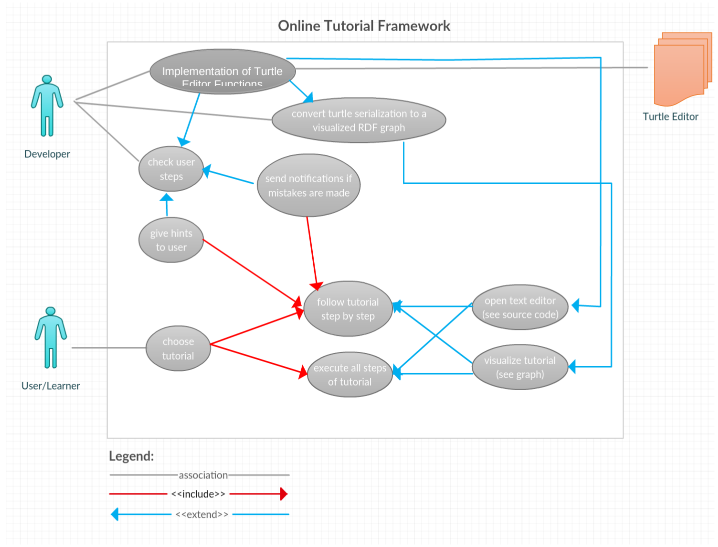
Online Tutorial Framework
Developer
User/Learner
Turtle Editor
Implementation of Turtle Editor Functions
convert turtle serialization to a visualized RDF graph
check user steps
send notifications if mistakes are made
give hints to user
choose tutorial
follow tutorial step by step
execute all steps of tutorial
open text editor (see source code)
visualize tutorial (see graph)
Legend:
association
<<include>>
<<extend>>

Fig. 2. Use Case Diagram

long as they are accompanied by another Turtle file for steps and hints, previously mentioned.The role of the developer also involves the updating part of the framework in terms of functionality. The developer creates all the functionality of the framework, and indirectly influences the guidance of the user through the tutorial.

6 Conclusions and Future Work

This report introduced a software solution for following an ontology tutorial using the Turtle Editor. We presented an online solution for learning ontology engineering and writing triples for different levels of learners (beginner, intermediate, advanced), with 0 installation costs and no payment or registration required for that. As young scientists and learners interested in the world of Semantic Web technologies, we developed the :turtle:Board framework based on our own engineering ideas guided by our mentor. This framework was specifically designed for the Turtle serialization which is one of the most human-readable syntax for RDF. After finishing the tutorial, the user of the :turtle:Board framework should be able to write, read, and understand any Turtle file.

There are various ontology tutorials on the Internet, but all of them are supposed to be done offline with an installed ontology editor. Because of this, we used JavaScript as a basis for achieving our goal which was the step-by-step guidance of the learner through an online Turtle tutorial. The method of learning by doing was our biggest motivation. The user of our framework is able to learn Turtle by writing triples, seeing examples, getting help when necessary, and correcting his/her mistakes.

Since everything can be represented in terms of RDF graph, the structure of our predefined ontology stored in our data files, can be easily used for any other description of certain tutorials. However, it is important to be mentioned that the amount of details in one tutorial depends strictly on the specified vocabularies. Therefore, as a result of the previous claim, our tutorial framework is limited to a specific set of vocabularies and specific type of semantics. Certainly, like any other project, we can always update and upgrade our work by coming up with more vocabularies for describing different properties and classes or different kind of examples, so that the user/learner will have more choices. Also giving a structured tutorial-oriented lessons before each tutorial will improve the ability of the user to learn faster and get better performance. Giving some detailed scores for the user progress will give him/her the opportunity to know his knowledge level for the particular language or syntax that he/she is learning. Part of this was already implemented in our project, however the scoring rules can always be changed. We can go beyond that and let the user create his/her own profile in which he/she will be able to store the steps, progress, results, etc.

To sum it up, we can conclude that :turtle:Board is a good starting software solution for building a big learning platform for any type of language that will incorporate different learning methods and syntax.

7 References

1. Horridge, M. *A Practical Guide To Building OWL Ontologies Using Proteg 4 and CO-ODE Tools.* The University Of Manchester. (March 24, 2011). http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4 v1 3.pdf.
2. Pollock, J.T. *Semantic Web for Dummies.* John Wiley and Sons, Inc. (2009). pp.170,304.
3. Prudhommeaux, E., Seaborne, A. (eds.). *Terse RDF Triple Language.* W3C Recommendation. (February 25, 2014). https://www.w3.org/TR/turtle/.
4. Turtle Validator. IDLab. Ghent University.(2014, 2015). http://ttl.summerofcode.be/.