Facade Layer for Apache JENA

1Rewat Lerlertvanich and 2Wiwat Vatanawood
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
Emails: rewat.l@student.chula.ac.th1, wiwat@chula.ac.th²

ABSTRACT
The challenges of an application development are not only to launch the new products fastest, but need to be aware of making things more flexible and more manageable in the future. We should realize that anything can be changed rapidly. The architecture and design are mostly concerned. Therefore, incoming with semantic web technology, the recommended standards and tools always contend. For example: OWLAPI, Apache Jena, Sesame, etc. The problems of developer and software engineer are learning curve to use those tools with lacking of knowledge, experience, case studies and the lack of the ideas for mapping on their existing knowledge. In this paper, we design and demonstrate the facade layer on the top of Apache Jena and provide the typical structures that will be useful for the developers to reuse the facade interfaces and components in their own applications. Our experiments show that the facade layer is the obvious choice that can be used to reduce the complexity and the lines of code of the implementation.

Keywords: Facade Layer; Semantic web; Apache Jena; Application frame work

1. INTRODUCTION
Recently, there has been wide interest in the semantic web technology. The concept of this web is needed to extend the web abilities by making the web contents to be more understandable by any computer—that is the idea of Tim Berners-Lee. Berners-Lee said that “If HTML and the Web made all the online documents look like one huge book, RDF, schema, and inference languages will make all the data in the world look like one huge database” [1]. So the development of semantic web is not different with the current web. It just increases more efficiency between computer and human by defining the definition of existing data on the web page [2]. Considering its nature, it is not easy to use and learn for the developers and researchers. Namely, they have to know many things such as the theory, concept and tools – but there are so many tools to use for development. It depends on developing environment and programming languages. Many developers start from the framework with their programming languages such as Apache Jena, OWL API and Sesame for the java developers, 4Suite 4RDF, pyparql and pySesame for the Python Developers, RDFSharp, DotSesame, SesameWindowsClient for the C# and .Net Developers and so on.

When tools and components had been considered for the development, then, it is the time to pull out the personal skills for the practical reason. One would intuitively expect for shifting the burden of learning by associating the development of semantic web, consistent with prior knowledge to remind the changing of the application strategy in the future and reduce the effort of development. In this paper, we give the preliminary result of using the facade design pattern to design the facade layer for Apache Jena to solve the problems that we mentioned above.

The remainder of this paper is structured as follows: Section 2 is concerned basic knowledge mentioned this issue and related work. Section 3 is the methodology and concept of the facade layer. Section 4 shows the comparative results between the traditional direct access to Apache Jena and the usage of the facade layer, in the end of this paper is ended with some concluding remarks and future work in Section 5.

2. RELATED WORK
This research focuses on the design and development of a facade layer for the semantic web by using the technique of reusing component to reduce the steps of invoking Apache Jena to develop semantic web. This research involves the following.

2.1. Software Design Pattern
Design pattern is a solution that could be applied in several environments in the object oriented programming. It shows the relationship between classes and objects; including the interaction between them. In 1994, the book “Design Patterns: Elements of Reusable Object-Oriented Software” was published by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides also known as the GoF (Gang of four). They proposed the design patterns classification shown in Figure 1.

![Figure 1: A design pattern classification [3]](image-url)
The proposed facade pattern is used for simplifying the structure of the complex objects by wrapping the whole complex objects and providing only a simpler and meaningful interface. In addition, it reduces the coupling between client classes and invoked classes. It also reduces the steps to call the functions. The structural template and behavioral template of facade pattern are shown in figure 2 and 3.

![Figure 2: The structural of the facade pattern [3]](image1)

![Figure 3: The behavioral of the facade pattern [3]](image2)

2.2. Semantic Web Technology

The semantic web concept is used to develop a web that based on the link data and share data together in the form of RDF Graph. The structure represents the relationship of data is called ontology. The relationship between the concept of the World Wide Web and the semantic web is shown in figure 4.

![Figure 4: A relationship between the WWW and the Semantic Web [4]](image3)

Those standards are conducted by World Wide Web Consortium (W3C). The semantic concept could be classified into two major components as ontology and languages. The first part is ontology which represents the knowledge and specifications that explains the concepts in a domain. Therefore in one specific domain, everyone can understand the meaning of those things in the same way. The ontology structure will show the fact of appearance of the domain and the knowledge that may be derived when using the logical inference from the existing fact, called Ontology Reasoning. The graph relationship is represented by RDF graph in form of Extensive Markup Language (XML) file that contains the definitions and metadata to explain the various types of data and type of information that are associated with. This means the ontology consists of vocabularies and taxonomy such as to identify the relationship among them. The second part is the language associated with semantic technology which consists of two languages. One is, Web Ontology Language (OWL) [6] and the other is a Semantic Web Rule Language (SWRL) [7]

2.3. Apache Jena

Apache Jena [5] is a framework for java programming language for developing semantic web in the form of java libraries. It helps web developers to manage the various semantic components of the semantic web and linked-data application to conform with the standards of the W3C.

Since 2000, Jena is an open source project developed by researchers at HP Laboratories in the Bristol city in UK and later became popular in used widely. It was success to become part of the Apache Software Foundation in November of the year 2010. The Apache Jena architecture is shown in figure 5. There provided a Store API to communicate with the graph model in the RDF Store and it can also interact with a variety of databases by the specific database drivers. All Graph data will be loaded into memory for the query of the application and it can also communicate with the external reasoners such as pellet, Fact++, Hermit, etc.

![Figure 5: The Apache Jena architecture [5]](image4)

For using Apache Jena to develop an application in the real world is quite too complex. We show the sample steps for query the data in RDF Store in Figure 6. Since
the application is not only developed by only one person, but it appoints many people to join the developing. We are highly expected more on flexibility of function call to support the changing in the future. For example, we need to change data store ontology graph from TDB to MySql or Oracle. The question is, how we make this change in the shortest times and less effect to the application code. Because of when the data store changes, it will affect to the data store driver, the model and the query languages. The biggest problem occurs when the driver changes, the ontology graph and the model may have been affected directly. The impact will be reduced when the application is separated between application layer and data store layer and the source codes that associate with graph should not scattered or duplicated. So this is one of facade pattern behaviors. Another advantage for using the facade is to hide the complexity of the steps of function calls in the application. Meanwhile it will also reduce the effort and get more learnability.

Within the facade layer, we propose a set of facade classes by grouping into the components which provide the essential APIs for semantic web application development. Our facade API components contain the functions, e.g. execute query, import data to data store and export data from data store in several format such as json, xml, text and csv file format. The Structure API component contains the typical java structures such as HashVector and StringHashtable for storing the return data instead of the Apache Jena ResultSet. The purpose of Configuration API package is to carry the application variables and constants such as data store path, temp file path, the configuration value of JDBC drivers. All variables and constants will be read by JDOM Parser. The Ontology model API component contains the Apache Jena model which is binding to the Jena built-in rule engine or external rule engine such as pellet. There will have three inference levels - none, rdfs level and owl level. The development components are shown in figure 8. The shaded items are the components that we built and the others are the original components that are bundled such as Apache Jena and pellet.

3. METHODOLOGY
According to the development problems that we mentioned earlier, we propose a design of facade layer in between the traditional semantic web application and Apache Jena. The facade layer is designed to exploit the facade pattern in order to simplify the connection coupling between two layers. To increase more knowledge mapping for java developer with the semantic concept, we create the typical java structures class HashVector (the vector of hash table) to store the query result instead of Apache Jena ResultSet. The proposed conceptual model of facade layer for Apache Jena is shown in figure 7.
We conduct the experiments on counting the lines of code between the direct calling the method of Apache Jena and the calling the facade classes. The comparison, shown in table 1, indicates that it takes 91 lines of code to perform a “execQuery()” function, while 2-4 lines of code are written instead if we use facade classes for querying TDB results.

Table 1: The comparison lines of code between direct calling methods of Apache Jena and calling through facade layer.

<table>
<thead>
<tr>
<th>Facade function</th>
<th>Lines of code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TDB</td>
</tr>
<tr>
<td>execQuery()</td>
<td>91</td>
</tr>
<tr>
<td>getOntologyFile()</td>
<td>109</td>
</tr>
<tr>
<td>loadDataToDB()</td>
<td>27</td>
</tr>
</tbody>
</table>

Remark:
D: Direct calling method of Apache Jena
I: Calling through facade classes

5. CONCLUSION AND FUTURE WORK

The design patterns should be applied to cope with the changes of design in the future. The facade pattern is the useful one to reduce the complexity on the steps of the implementation, especially for enterprise application. In this paper, we propose the usage of the facade pattern to reduce the complexity of the Apache Jena and build the typical java structure for mapping knowledge of the java developer. It is also useful for reducing the effort and times of development. The facade components in the Facade layer in this paper cover the basic procedures to develop a typical semantic web. For the future work, we will elaborate the Facade layer to provide sufficiently domain specific features.

REFERENCES


