Visualization Agent of Vijjana, a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks

Biswajith Yarlagadda

Problem Report submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering

Approved by
Dr. Yenumula. V. Reddy, Ph.D., Chair
Dr. James D Mooney, Ph.D.
Dr. Sumitra Reddy, Ph.D.

Lane Department of Computer Science and Electrical Engineering Morgantown, West Virginia
2009

Keywords: Vijjana, Visualization, JAXB, Hibernate, Adobe Flex, Rest API
Abstract

Visualization Agent of Vijjana, a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks

Biswajith Yarlagadda

According to recent study it is estimated that amount of unique information added to the World Wide Web each year will be in the order of Exabyte’s. [1] Most of us rely on search engines to search for relevant information using keywords. One of the biggest limitations is many a time we will come across search results that are not relevant to our search keywords. Some of the solutions proposed to solve the problem of finding for relevant information on the World Wide Web is implementation of semantic web and extensive use of Social Bookmarking Websites. Making the entire web semantic is something that cannot be achieved in near future. Social bookmarking websites don’t follow the taxonomy model.

Integrating semantic web and social bookmarking, our team developed a Framework called Vijjana [2], a model for Collaborative, Self Organizing and Domain Centric Knowledge Networks.

In this report we explain how the Visualization agent (vA) in Vijjana is designed. vA is responsible for generating the visual representation of the knowledge network. We used the data from Open Directory Project [3] as the foundation to classify the URL into different domains. The Adobe Flex Front End [4] along with the Prefuse Flare toolkit [5] is used to create the visualization. The data required for the frond-end will be generated by the Java Middle layer with aid of O/R library Hibernate [6] and Java Architecture for XML Binding [7]. The communication between the front-end and Java middle layer and frond end happens through a
web service developed using the Rest API. [8] The web service was designed in such a way that various applications could be built based on the architecture designed.
Acknowledgments

I gratefully acknowledge my Committee Chair, Dr. Yenumula V. Reddy for his supervision, constant support, and proper guidance throughout the project. This project is originated from his ideas and evolved from the insights and suggestions of Vijjana team.

I would like to extend my thanks to Dr. James Mooney and Dr. Sumitra Reddy for serving on the committee and supporting the project with their suggestions and encouragement.

I would like to thank all my friends Raghu Doppalapudi, Gangadhar Jaliparthi, Rajesh Makineni, Luyi Wang for their valuable guidance and advises they have given me. I would also like to thank Vijay Shankar Yalamanchi, Raghu Chandra Prodduturi, RamRaj Kolli, Santosh Tekulapalli and Sai Venkatraman Pinnamaneni for always being supportive during various phases of my life.

I am deeply thankful to my beloved parents Mr. & Mrs. Venkateswara Rao Yarlagadda and my sister Tejaswini Sai Yarlagadda for their encouragement and support in all the things I do.
# Table of /Contents

Table of /Contents ......................................................................................................................... v
List of Figures ................................................................................................................................. viii
List of Tables .................................................................................................................................... ix
Chapter 1  Introduction ................................................................................................................... 1
  1.0 Background ............................................................................................................................. 1
  1.1 Problems with this Approach ................................................................................................. 1
  1.2 Solutions .................................................................................................................................. 2
    1.2.1 Semantic Web ...................................................................................................................... 2
    1.2.2 Social Bookmarking Websites .......................................................................................... 2
  1.3 What is Vijjana? ....................................................................................................................... 3
  1.4 Literature Survey ...................................................................................................................... 3
  2.0 Problem Statement ................................................................................................................... 4
Chapter 2: Vijjana Architecture ....................................................................................................... 6
  2.1 Introduction ............................................................................................................................. 6
  2.2 Vijjana Agents ......................................................................................................................... 7
    2.2.1 Taxonomy and Semantic Net of Knowledge (T and R) ..................................................... 7
    2.2.2 Discovery Agent (dA) ......................................................................................................... 7
    2.2.3 Organizing Agent (oA) ...................................................................................................... 7
    2.2.4 Consistency Agent (cA) .................................................................................................... 8
    2.2.5 Visualization Agent (vA) ................................................................................................... 8
    2.2.6 Search Agent (sA) ............................................................................................................ 8
    2.2.7 Rating Agent (rA) ............................................................................................................. 8
Chapter 3: Vijjana Database Design ................................................................................................. 9
  3.0 Database Design ....................................................................................................................... 9
Chapter 4: XML Generation using JAXB ....................................................................................... 11
  4.1 Why is XML needed? .............................................................................................................. 11
  4.2 XML ........................................................................................................................................ 11
    4.2.1 Sample XML file ............................................................................................................... 12
7.1 Visualization .................................................................................................................. 41
7.2 Why use Flex? [29] ........................................................................................................... 41
7.3 How does Flex Work? [31] ............................................................................................. 42
7.4 Action Script and MXML .............................................................................................. 43
   7.4.1 MXML ...................................................................................................................... 43
   7.4.2 Action Script ........................................................................................................... 44
7.5 Data Access in Flex ....................................................................................................... 45
   7.5.1 Data Access used in Vijjana’s Visualization Agent .................................................. 45
7.6 Prefuse Flare .................................................................................................................. 47

Chapter 8: Implementation ..................................................................................................... 49
  8.0 Implementation ............................................................................................................. 49

Chapter 9 ............................................................................................................................... 51
  9.1 Future Work .................................................................................................................. 51
  9.2 Conclusions .................................................................................................................. 52

References .............................................................................................................................. 53
List of Figures

Figure 1: Communication between different vA components .................................................. 11
Figure 2: Communication between different vA components (expanded) ................................. 18
Figure 3: Client Server Communication .................................................................................. 19
Figure 4: Hibernate Architecture ......................................................................................... 22
Figure 5: Java Bean File ......................................................................................................... 23
Figure 6: Hibernate Mapping File ......................................................................................... 24
Figure 7: Hibernate Configuration File .................................................................................. 25
Figure 8: Application to Application Communication via Web Services .............................. 31
Figure 9: Restful Web Service .............................................................................................. 33
Figure 10: Flex Animation ..................................................................................................... 44
Figure 11: Prefuse Architecture ........................................................................................... 48
Figure 12: Implementation ...................................................................................................... 50
List of Tables

Table 1: Restful Web Service .......................................................... 33
Table 2: Example of Restful API ....................................................... 35
Chapter 1  Introduction

1.0 Background

The World Wide Web would not have been such a huge success without search engines. Every search engine employs automated computer robot called Spider which crawls through World Wide Web to find new web pages or documents and also find changes to existing documents/websites by revisiting sites that were already discovered on a regular basics. After this process is completed everything found by the spider will be updated to the index. Index of a search engine is like a giant book which has a copy of every web page the spider finds. If the information on about the web page changes, the index is updated to reflect the change. When a new web page is created it will not show up immediately on the search results until the spider find the page and the indexing process is completed. The other part of the search engine is software which is responsible for accepting the keywords from the user and then search for relevant web pages in the index and also assigns ranks to the web pages using ranking algorithms.

1.1 Problems with this Approach

- User should be intelligent enough to provide right keywords
- Constrained by keyword matching techniques employed by the search engine
- Do not allow the users to collaborate
1.2 Solutions

1.2.1 Semantic Web

"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”. [9]. The word semantic stands for related to the meaning of, the term Semantic Web means describing things in a way that computer applications can understand.

Problems with the Semantic Web

- A lot work research work is to done before the semantic web concept can be applied to the entire web.

1.2.2 Social Bookmarking Websites

Social Bookmarking sites store and organizes the web pages based on Meta data such as key words or tags. They also help also help in the easy discovery of related web pages.

Problems with this Social Book Marking

- Lack of controlled vocabulary for tagging web pages.
- No way of indicating hierarchical relationships between tags.
- With the growth in number of pages, search process becomes unmanageable again.
1.3 What is Vijjana?

Vijjana [2] is a solution proposed to solve the problem current World Wide Web system which combines concepts of semantic web and social bookmarking websites. Vijjana system can be described as a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks. Vijjana frame work helps the user to store bookmarks along with metadata. This meta-data is then used to establish relationships between the different bookmarks. Any bookmarks added to the Vijjana system is called a Jan. A collection of Jan can be stored in a centralized database that will act like a knowledge network. Vijjana frame work also helps the users access the knowledge network visually. The Vijjana system also provides the user tools to add, delete and edit the Jans in the Vijjana database.

1.4 Literature Survey

Stanford University is trying build a semantic e-mail addressing system called SEAMail [10] in which an e-mail sender can add recipients based on data about the recipients. This system uses information stored on the University databases, governmental databases to decide if a particular recipient can be associated a specific e-mail based on the e-mail’s purpose.

Google’s wonder wheel [11] is a tool that will help the user find information related to the specify keyword. The wonder wheel is a simple flash object which will allow the user to explore topic by clicking on braches available on the wheel and will allow the user to dig deeper into the subject of the main keyword. This is a similar concept to Visualization agent in Vijjana expect that the agent we have developed will only deal with data stored by the user in his personal network and information on our database not the entire web.
DeepaMehta [12] is a Semantic Desktop software platform which uses semantic web concepts to organize file directories on the desktops. To organize the information on the desktop first the A topic map is created. A topic map is similar to the taxonomy defined in the Vijjana Architecture. The Topic maps list all the topics and the association between these topics. Each topic is assigned an ID and each file will have be mapped to one of these topics. Each file on the desktop can be uniquely identified by an ID such as a number or a directory path to the file.

2.0 Problem Statement

One of the most important agents in the Vijjana Frame work is the Visualization agent. The visualization agent will allow the user to represent his knowledge network in the form of a graph. The agent also allows the user to transverse through the graph and finds the required data (URL in the case of Vijjana). The problem statement is to develop a Visualization agent for the Vijjana Frame work and this agent will create a visual representation of the knowledge network for each user.

In chapter 2 we have discussed about Vijjana Frame architecture and its various agents and what each agent does. Vijjana visualization agent has four important components namely

- Database system
- Java middle layer
- Web service
- Flex Front end.
Database has all the information that we need to create the visual representation of the knowledge network. More information about Vijjana database system is in chapter 3. Java middle layer is responsible for generating the XML file after it receives a request from the web service. More information about the XML file generation is available in chapters 4 and chapter 5. Web service in Vijjana will help the Flex front end communicate with the Java Middle layer. More information about the web service is available in Chapter 6. Flex front end is responsible for sending out HTTP requests and processing the incoming XML file to generate the visual. More information about the Flex front-end is available in chapter 7.
Chapter 2: Vijjana Architecture

2.1 Introduction

Vijjana is an agent based framework and we define Vijjana as follows-

Vijjana-\(X\) = \{\(J\), \(T\), \(R\), \(dA\), \(oA\), \(cA\), \(vA\), \(sA\), \(rA\)\}

Where

\(X\) = the domain name

\(J\) = the collection of JAN’s in the Vijjana-\(X\)

\(T\) = the Taxonomy used for classification of JAN’s

\(R\) = the domain specific relations

\(dA\) = the discovery agent which find relevant JAN’s

\(oA\) = the organizing agent which interlinks the JAN’s based on \(R\)

\(cA\) = the consistency/completeness agent

\(vA\) = the visualization agent

\(sA\) = the search agent

\(rA\) = the rating agent

More information about each of agents in the following sections.
2.2 Vijjana Agents

2.2.1 Taxonomy and Semantic Net of Knowledge (T and R)

According to Wikipedia Taxonomy “is the practice and science of classification”. One of the most important requirements of the Vijjana system is it should be able to classify all the Jans into different groups and also it should be able to identify the relationships between them to form a semantic net. In order to achieve this we have to develop a taxonomy that can be applied to all domains.

2.2.2 Discovery Agent (dA)

As of now the Discovery Agent of Vijjana is the Human who will discover new knowledge by performing normal search operation and then markup the Jans to the Vijjana server where it will be classified based on the meta information provided by the user and also meta information generated by the Vijjana framework.

2.2.3 Organizing Agent (oA)

Once the Jan is marked up (submitted) to the Vijjana system the organizing agents takes over and then classifies the Jan based on the meta – information provided by the user and also meta – information generated by the Vijjana system. Organization agent will help the system in identifying the relationship of current Jan can have with Jans which already exist on the database. It will also notify the user about the status of new Jan in the system.
2.2.4 Consistency Agent (cA)

The consistency agent is responsible for maintaining integrity of the data on the Vijjana database. It does that by visiting all the URLs once in a while and will make sure that they are still “alive”. The URL will be considered “alive” if the consistency agent receives a 200(OK) HTTP status code which tell it that the resource on that URL is still on the server. The consistency agent will receive 404(Not found) HTTP status code which means that resource that corresponds to URL does not exist on the server anymore.

2.2.5 Visualization Agent (vA)

The visualization agent is responsible for displaying the knowledge network of specific user in form of a graphic.

2.2.6 Search Agent (sA)

The search agent is responsible for providing the user with search functionalities. The current version of the Search agent searches the database using the keyword input that the user provides. All the URL’s that correspond to that keyword will be displayed to the user.

2.2.7 Rating Agent (rA)

The Rating agent is responsible for rating the Jans in the database based on user ratings and other automated mechanism such as number of hits, relevance etc.
Chapter 3: Vijjana Database Design

3.0 Database Design

Vijjana system should allow the users to edit and visualize the knowledge network which is stored on Vijjana Database. One of the biggest challenges when developing the Vijjana system was coming up with the database design. The database was designed in such a way that information stored on the database helps the other agents such as the Visualization agent recognize/identify the association of user and his knowledge network, domains with their subdomains, web pages with their domains and also enable end to end user collaboration.

Vijjana database has three types of information

- Raw data
- User data
- Website data

Raw Data: All the information about every JAN any user submits to the Vijjana system will constitute the raw data. When a web page is marked up, the Vijjana system will generate keywords for that specific web resource. Using these keywords generated, the system will then decides the domain to which these web pages can be associated. Once this decision is made the web page is stored on the database under the domain assigned to it.

Since we don’t have a fully functional system with users all the raw data was taken from the Open Directory Project [3]. The Open directory project provided us with all the tables...
required to build up the taxonomy for various domains. It also provides us with data with several thousand URL’s which are already organized in according to a taxonomy.

One of the biggest advantages the Open Directory Project is that it helped us integrate the semantic web concepts in to our project.

User Data: User data includes login credentials, profile information and information associated with that specific user. The data also contains usage history of the user which is used by the system to understand the user interests.

Website Data: The Vijjana website data is used to provide efficient and effective services. Out of three types of information available the visualization agent uses raw data and user data parts to generate a visual graph for the user.
Chapter 4: XML Generation using JAXB

4.1 Why is XML needed?

Information about every user's knowledge network is stored on Vijjana database. The Prefuse Flare toolkit accepts information in some abstract forms only. Some of formats supported by the Flare framework [5] are Tab limited format, JSON and GraphML [13]. (“GraphML”, an XML format for representing knowledge networks with nodes and edges). Visualization agent uses XML files to transfer data from Java middle-layer to Front-end Adobe Flex [4]. XML [14] is W3 standard which will facilitate communication between different components of an application.

![Diagram of communication between different VA components](image)

Figure 1: Communication between different VA components

4.2 XML

Extensible Markup Language also called XML is a W3 standard that is a bunch of rules based on which XML documents can be either written or created. Following same set of
predefined rules will help all applications read and write XML files irrespective of the framework being used to develop the application and also helps in communicating with other applications irrespective of environment in which the other application is developed or deployed. XML files are simple text files which have data enclosed within tags that help us transport and store data.

Benefits of using XML

- Data can be stored in a simple text files.
- Data transport between different applications happens easily because all XML documents adhere to the same fixed set of rules.
- XML also simplifies the process of platform changes as most of the frameworks can read/write XML files.

4.2.1 Sample XML file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<user>
  <name>Sam</name>
  <major>CS</major>
  <GPA>3.45</GPA>
</user>
```

The sample code above is a small piece of XML code which has information about a student called Sam. Since machines with little to no intelligence read XML documents, humans have to make sure that the XML files are well-defined.
4.3 XSD

XML schema describes the structure of the XML file. The biggest advantage of XSD [15] over DTD is that XML schema document are also XML documents.

XML schema document contains

- All the elements that can appear in the XML file
- All the attributes that can appear in the XML file
- Define data types for all attributes and elements.
- Specify if a specific element can be null
- Specify the default values for a given XML element.
- Helps us define the relationship between different elements.
- XML documents usually don't have a schema, but if they have one they should adhere to the rules specified in the schema. So in that way they help not only humans but also automated systems in generating well-defined XML documents.

4.3.1 XML schema in Vijjana

- The Adobe Flex part of Visualization agent which is responsible for generating the visual representation of the knowledge network requires Java Middle layer to supply it with an XML file that uses the GRAPHML schema. The GraphML schema defines nodes and edges that graph should display. This schema makes sure that the XML file generated by the Java middle layer follows the GraphML schema that can be parsed by the Front-end Flex part of the Visualization agent.
4.3.2 Sample XSD code

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/Vijjana"
xmlns:tns="http://www.example.org/Vijjana" elementFormDefault="qualified">
  <element name="User" type="tns:User"></element>
  <complexType name="User">
    <sequence>
      <element name="Name" type="string"></element>
      <element name="Major" type="string"></element>
      <element name="GPA" type="int"></element>
    </sequence>
  </complexType>
</schema>

4.3.3 Sample XML Code that uses the GraphML Schema

<?xml version="1.0" encoding="UTF-8" ?>
<graphml>
  <graph edgedefault="undirected">
    <!-- data schema -->
    <key id="name" for="node" attr.name="name" attr.type="string" />
    <key id="dept" for="node" attr.name="dept" attr.type="string" />
  </graph>
</graphml>
- <!-- nodes -->

- <node id="1">
  <data key="name">Jo</data>
  <data key="dept">CS</data>
  </node>

- <node id="2">
  <data key="name">Ed</data>
  <data key="dept">EE</data>
  </node>

- <node id="3">
  <data key="name">Christiaan</data>
  <data key="dept">CHE</data>
  </node>

- <!-- edges -->

- <edge source="1" target="2" />
- <edge source="1" target="3" />

</graph>

</graphml>
4.4 Java Architecture XML Binding (JAXB)

XML and Java have become very important to web developers. Java programs access data in the XML files using tools called parsers. The two kinds of parser available are the SAX parser and the DOM parser [16]. SAX parser is event driven parser in which a user can specify a number of call back events that will deal with every event the parser will encounter while parsing an XML document. Some of the events that a SAX parser can encounter are

- XML Text nodes
- XML Element nodes
- XML Processing Instructions
- XML Comments

The DOM parser will create a tree of object that will represent the schema and data of given XML file. This tree will be stored in memory and any application that need to access or edit data has to navigate through the tree to get information.

This section will discuss the Java Architecture for XML Binding (JAXB) [7] approach. The first step involved in the parsing an XML file using JAXB is binding the XML schema of the XML document into Java classes that represent the document. So in order to use JAXB with XML files we should first create a schema for the XML file.

Once the binding is completed the following Java classes are created. All the classes that are created by JAXB library are referred to as Content Objects.

- ObjectFactory.java
- User.java
4.4.1 Why did we use JAXB instead of Traditional Parsers?

- **Java Data objects**: Allows us to create Java data objects based on the XML schema.
  
  SAX and DOM parser do not support creation of Java data objects.

- **Data Validation**: An application that uses JAXB application follows the definition of schema strictly, so use of JAXB will eliminate the possibility of invalid Java data object.

- **Speed**: SAX parser is well known for the speed at which it can process data from an XML file using the event driven mechanism. A JAXB application can match the speed of the SAX parser using the binding approach.

- **Extensibility**: The generated Java data classes can used with O/R libraries extended to perform other operation such as updating databases etc.

  The Java bean file which is User.java generated can be used to read the data from the XML file. The entire process described here is called unmarshalling an XML document.

  One of the main operations in the visualization process is the generating the XML file using the data from the database and then use this XML file with Prefuse Flare toolkit and Adobe Flex to generate the graph. The XML file to be generated should strictly adhere to the GraphML schema. JAXB has the ability to create the XML file from the available Java data objects and data in the database.

  - Design a sample XML schema for all the users. This XML schema will integrate the GraphML schema along with the elements and attributes that should show up on the graph.
• Generate the content objects for this schema.

• Using the content object along with Hibernate, the database is queried for information.

• Using the information, the content objects and JAXB, the XML file is generated.

The process of using the content objects and information from the database to generate the XML document is called Marshalling. In this section we have discussed how XML file is generated using JAXB and data from the database. In the next section we shall look into how data is extracted from the database.
Chapter 5 Database Connectivity and Data Extraction

5.0 Introduction

Every activity we perform on the World Wide Web which includes reading/writing e-mail, IM, watching a video on online video viewing websites such as YouTube involves data transfer from the server side to the client side. All data on the server side is stored in relational databases. The back end of every major commercial web application such as website like Amazon is a large-scale storage of ordered information. We cannot imagine the World Wide Web without databases.

Figure 3: Client Server Communication

Many programming languages such as Java, C# and VB have standardized how front-end component of an application interact with the databases, one of the most successful being Enterprise Java Bean which is the standard used by the Java Enterprise Edition.
In Java there are several ways in which the Front-End java application can interact with the Database.

- Enterprise Java Bean (EJB)
- Java Database Connection API (JDBC)
- Object Relational Mapping Libraries like Hibernate (ORM)

5.1 Database Persistence

All the above mentioned tools will help in Database persistence [17]. But what is database persistence in the first place. Data is said to be persist when data is stored on a permanent storage medium, which will protect it from being lost even when the system fails or when the computer is shut off. One of main objective of Data persistence is applications running independently should be able to read, write and edit information from the database.

5.3 Advantages of Hibernate

- Plain Old Java Objects: Java bean is also a Java object/class which follows certain conventions about method naming, constructor declaration and behavior making the process of creating and working with it very tedious. Plain Old Java Objects [18] are normal Java object they do not serve any special purpose nor implement any interface in the Java frame work. So POJOs will help in overcoming short coming of the Java bean classes.
  - Hibernate provides the user with the ability to store POJO’s in the database.
• Using Hibernate will make body of code easier to write and manage and also it will also provide the user with more security when compared to JDBC approach.

• EJB support two types of persistence [6]
  - Bean-managed persistence
  - Container-managed persistence

If we are using BMP, we have to provide the bean with appropriate JDBC logic to enable the bean to communicate with the database, in the case of CMP the container will be responsible for enabling the connection between the database and the Java bean.

5.3.1 Some problem with EJB’s

• Both CMP and BMP have a reputation to being slow

• They are not portable across applications.

5.4 Hibernate as a Persistence Solution

Hibernate overcomes most of the above mentioned problems. Hibernate will enable the user to construct POJO’s from a selection of table columns.[19] Hibernate promotes reuse of code as the same mapping file can be used over and over if the same database tables are reused in other applications. Hibernate is an open source library and it is available for developer free of cost.
5.5 Hibernate Architecture

![Hibernate Architecture Diagram]

Figure 4: Hibernate Architecture

5.6 How Does Hibernate Work [19]

The concept of how hibernate work will be discussed using a database table and how hibernate helps a user in communicating with this database table. The structure of the database table that will be used is as follows. Any JDK data types including primitive data types and Java collections can be used to map a java class to a database table.

create table students
(StudentID INT, LastName VARCHAR(40), FirstName VARCHAR(40), Major VARCHAR(40), GPA INT, PRIMARY KEY(StudentID) );


5.7 Hibernate in Action

The first step in creating of a Hibernate application is to create a Simple Java bean class that represents all the events that can occur when the Java application interacts with the database. The Java class will have setter and getter methods for all the fields in the database table that we are interacting with.

```java
public class Student implements java.io.Serializable {
    private int studentId;
    private String lastName;
    private String firstName;
    private String major;
    private Integer gpa;
    public Student() {
    }
    public Student(int studentId) {
        this.studentId = studentId;
    }
    public Student(int studentId, String lastName, String firstName, String major, Integer gpa) {
        this.studentId = studentId;
        this.lastName = lastName;
        this.firstName = firstName;
        this.major = major;
        this.gpa = gpa;
    }
    public int getStudentId() {
        return this.studentId;
    }
    public void setStudentId(int studentId) {
        this.studentId = studentId;
    }
    public String getLastName() {
        return this.lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getFirstName() {
        return this.firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getMajor() {
        return this.major;
    }
    public void setMajor(String major) {
        this.major = major;
    }
    public Integer getGpa() {
        return this.gpa;
    }
    public void setGpa(Integer gpa) {
        this.gpa = gpa;
    }
}
```

Figure 5: Java Bean File
5.7.1 Mapping File

Hibernate uses the Mapping file to get information about how to read and write objects from the persistent class. The Hibernate mapping file gives the application information about the database table that maps to the object and the fields within the table to that map to variables in the Java persistence object.

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!--- Generated Nov 10, 2009 2:29:57 AM by Hibernate Tools 3.2.1 GA -->
<hibernate-mapping>
  <class name="org.vijjana.Students" table="students" catalog="vijjanaertest">
    <id name="studentId" type="int">
      <column name="StudentID" />
      <generator class="assigned" />
    </id>
    <property name="lastName" type="string">
      <column name="LastName" length="40" />
    </property>
    <property name="firstName" type="string">
      <column name="FirstName" length="40" />
    </property>
    <property name="major" type="string">
      <column name="Major" length="40" />
    </property>
    <property name="gpa" type="java.lang.Float">
      <column name="GPA" />
    </property>
  </class>
</hibernate-mapping>
```

Figure 6: Hibernate Mapping File

Between the start and end hibernate-mapping tags the class element helps us map a persistent class with a database table. In the Code Snippet 2, the students.java class in org.vijjana
package is mapped with the students table from the vijjanatest database. Each instance of this class is now a row in the database.

The next step in the mapping process is defining a unique identifier property that will map to the primary key of the database table. The mapping for the primary key column of the database is always defined between the id tags. The use of id tags tell Hibernate to use getStudentID() and setStudentID() methods to access the property. The <generator> element [20] is optional element under the <id> element and is used to way in which the primary key value is generated while saving a new record. In the example above generator value assigned was used in which the user will be required to provide a primary key value for each new record created.

The last step in the mapping process is to provide Hibernate with information about all other entity class properties. The property tag/element with tell Hibernate which setter and getter methods to use with a specify database column.

5.7.2 Configuration File

```xml
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.password">******</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/vijjanatest</property>
    <property name="hibernate.connection.username">******</property>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  </session-factory>
</hibernate-configuration>
```

Figure 7: Hibernate Configuration File
At this point we are done with creating the persistent class and its mapping file. Now we have to create the configuration file. The main objective of using Hibernate in any Java application is to help the application connect to the database and so the user should provide Hibernate with a configuration file which contains all information that it needs to connect the database.

The Hibernate session Factory is responsible for maintaining connection to a specific database. If the user has to use several databases the user is required to create several <session-factory> configurations in separate configuration files.

The configuration file has four important properties that will help Hibernate in connecting to the database. The driver used to communicate with the database, the JDBC URL which specifies the location of the database and the user name and password for authentication. It also specifies the java object to database table mapping file.

5.8 HQL: Hibernate Query Language

In the previous section we have discussed how Hibernate can be used to communicate with the database. One of most important advantage that databases offer over spread sheets and other data processing tools is they help the user retrieve required information. The tool that helps user find the required information is a Query. SQL is the standard language for creating queries, accessing and making changes to an existing database. Hibernate Query Language (HQL) [21] is a very useful hibernate tool which combines SQL and object-oriented approach of Java.
Hibernate offers the user with three different ways of retrieving objects from the database.

- Hibernate Query Language (HQL)
- Query by Criteria using Criteria API
- Native SQL queries

### 5.8.1 Advantages of using HQL [22]

- SQL queries can be represented in object oriented term such as objects and object properties. HQL supports both simple and complex queries.
- Native SQL queries usually return plain text data which cannot be operated on. In the case of HQL the output of a Query will be in form of object or collection of objects which can easily accessed and manipulated using Object oriented programs.
- HQL supports polymorphic queries. Polymorphic query is query where in all child objects of object that is to be returned as the query result are also included in the output.
- Syntax that HQL uses is similar to the syntax used by SQL.
- HQL allows the user to write database-type independent queries. HQL queries are converted to native SQL at run the depending the SQL dialect used in Hibernate configuration file.
5.8.2 HQL function classification

HQL functions can be classified into four different categories

**Projection:** Projection is a technique where the user can specify the objects and object properties that must be retrieved. Projection is equivalent to retrieving data from database table(s).

Examples:

```
from students;
```

This will return all instance of the Students class. This query is equivalent to the following SQL code

```
Select * from students;
```

**Restriction:** All that projection does is it will retrieve all the instance of object specified. Sometimes we do not need all the instances; this is where the restriction technique is useful. It will help us filter out based on conditions. Keyword Where is to specify a conditions within HQL or SQL query.

```
Select s from Students s where s.StudentID = 9001.
```

This query when executed will return the object instance where the object property (studentid) =9001.

This SQL equivalent of this HQL query is

```
Select * from students where StudentID=9001;
```

**Aggregation:** Aggregation helps in summarizing a group of values in a database table. Some of important aggregate functions are sum, avg and count.
Select max(s.gpa) from Students s;

This statement will return the maximum value of GPA (object property) in all the instances of Students object. The SQL equivalent of the above query is

Select max(GPA) from students;

In this section we have discussed how data from the Vijjana database can be extracted. This data will be used along with JAXB as described in Chapter 3 and XML file will be generated. This XML file will be sent to the Flex front end through the Web service developed using the Rest API. In the section we will discuss how the Web service in the Vijjana works
6.0 What is a web service?

One of very simple definition for a web service is it a set of standards that will help entities on the Internet talk to each other irrespective of the programming language and Operating system used on either of entities. A good example can be a web service that will help a Windows client talk with a UNIX server. Web services enable application to application communication.

Distributed systems which were widely used in 90's suffered from many drawbacks. Two of the major problems were [23]

- Interoperability: Since each software designer followed their own format for distant object communication, all others who want to access that service had to follow the same format which hinders interoperability. So if two applications were written in different programming languages they could not communicate with each other.
- Distributed system used non-standard ports for communication. So if two applications designed by different developers had to communicate, the firewall had to be reconfigured to allow communication on the port that is used by the application.

For a service to qualify to be a web service it should satisfy some conditions. [24]

- Provides the application with end points to send and receive information.
- Allowing the user to assess the system from any location even if they are not operating from the same server.
How did web services overcome the above mentioned problem?

Web services use XML files (messages) or HTTP requests to enable application to application communication.

![Application to Application Communication via Web Services](image)

- Request will contain all the required parameters and also the format in which it expects the response to be in.
- Response provide to the Web service will be in the Format requested by Application 2

**Figure 8: Application to Application Communication via Web Services**

### 6.1 Restful API

REST which stands for Representational state transfer [25] is one of the most recently developed type of Web Services. Restful web applications rely on named resources in the form of Uniform resource locators (URL’s) and Uniform Resource Identifier (URI's) instead of messages (XML files) as in SOAP (Simple Object Access Protocol). The Rest API uses HTTP requests and responses to send and receive information to/from applications.
The Rest API is platform independent and is available for all programming languages. One of the biggest advantages of using the Rest API is it uses Standard ports for information transfer. So if an organization is planning to use a web service developed using the REST API they need not reconfigure their firewall.

Rest API assumes that every resource is a noun which can be uniquely identified by a URI. Any request that can be associated with this resource (noun) should be treated like a verb. A good example for this is GET a document identified a unique URI. The use of URI's has helped in discovering services without even being published to the centralized repository.

Everything that is available on World Wide Web is resource and Rest API's [26] reliance on named resources rather than messages will help in retrieving a given representation of the resource. The resource representation which is also known as representation state supports the following operations.

- Retrieving Information
- Modifying Information
- Creating Information
- Deleting Information

Considering a URI to explain Restful Operations: http://vijjana.org/
Table 1: Restful Web Service

<table>
<thead>
<tr>
<th>Verb/Operation</th>
<th>Application Task</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Read/ Retrieve</td>
<td>If a get request is sent for the URL it retrieves the data of all the Users of the Vijjana System</td>
</tr>
<tr>
<td>POST</td>
<td>Create</td>
<td>Creates a User in the Vijjana system</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete</td>
<td>Deletes all the Users in the Vijjana system.</td>
</tr>
<tr>
<td>PUT</td>
<td>Update</td>
<td>Updates the Information in the Vijjana system</td>
</tr>
</tbody>
</table>

CRUD Operations - Rest API

![Diagram of CRUD operations]

- **GET URI**
  - Client
  - Resource
  - Sends Back information about the Specific Resource

- **POST URI**
  - Client
  - Resource
  - Create a new instance of the Resource

- **PUT URI**
  - Client
  - Resource
  - Updates a specific instance of the Resource

- **DELETE URI**
  - Client
  - Resource
  - Deletes the specific instance of the Resource

Figure 9: Restful Web Service
6.1.1 Some of Advantages Rest API offers

- Stateless: Rest API will help the user in creating Stateless Web Applications. A stateless web application requires the client to send a complete, independent request. When processing a complete and independent request the server need not retrieve any type of application context or state. It will also simplify design and development of the entire system. Since the intermediate servers need not store any state information, the problem with synchronizing session data with an external application is eliminated. The client will store the state information. It also helps in improving the overall response time of the system. One problem with this approach is the server can receive repetitive data sent in a series of requests, since the server does not maintain the connection information about a specific client.

- Cache: Cache improves network efficiency by eliminating multiple requests for the same data by storing the data on the client side.

- MIME: Rest API supports Multiple Internet Mail Extensions such as JSON, XML and XHTML.

- Support for a dictionary like URI structure.

6.2 Building RESTful API

This section will discuss how the Rest API can be used to create a restful web service.

Assumptions for this discussion are

- Application has to handle the course registration on campus.
- Courses can be added or dropped (deleted) from course list offering.
- Students can be added or dropped (deleted) from the course.
- User should be able to see all the courses listed and all the students in a specific course.

The important step in this process is to identify the named resource and the API becomes a series of URI patterns and using HTTP verbs with those URI to invoke the resource. The assumptions that were discussed earlier map nicely to CRUD, and HTTP supports CRUD via HTTP Post, Get, Put, Delete requests.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Action/Verb</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding Courses to the list</td>
<td>POST</td>
<td>courses.com/</td>
</tr>
<tr>
<td>Listing all the courses</td>
<td>GET</td>
<td>courses.com/</td>
</tr>
<tr>
<td>Updating Courses</td>
<td>PUT</td>
<td>courses.com/</td>
</tr>
<tr>
<td>Get information about a specific course</td>
<td>GET</td>
<td>courses.com/courseid</td>
</tr>
<tr>
<td>Update Information about a Specific course</td>
<td>PUT</td>
<td>courses.com/courseid</td>
</tr>
<tr>
<td>Delete a specific course</td>
<td>DELETE</td>
<td>courses.com/courseid</td>
</tr>
<tr>
<td>Get information about a specific student in the course</td>
<td>GET</td>
<td>courses.com/courseid/studentid</td>
</tr>
<tr>
<td>Update Information about student in a Specific course</td>
<td>PUT</td>
<td>courses.com/courseid/studentid</td>
</tr>
<tr>
<td>Drop a student from the course</td>
<td>DELETE</td>
<td>courses.com/courseid/studentid</td>
</tr>
<tr>
<td>Add a new student to the course</td>
<td>POST</td>
<td>courses.com/courseid/studentid</td>
</tr>
</tbody>
</table>
One should notice that each URI can map to more than one verb and also some URI cannot map to certain verbs such as courses.com/ cannot map to HTTP delete because as a user, you don’t want all the data in the database to be deleted.

Now a series of XML documents are created to represent the resources of that the Restful Courses Service will support. XML is the standard format that helps in machine-machine communication in context of business transactions. For instance a course can be defined using the XML as listed below.

```xml
<courses name="Introduction to Computer Applications" Credit Hours="4" id="cs101">
  <uri> /courses/cs101 </uri>
</courses>
```

Note the course has an ID which identifies the course uniquely and the most important part is each course has a URI which is key aspect of REST. This URI will give us the RESTful representation of the course and also this XML file can be used to process the GET request to /courses/cs101. This XML can also be used to create or add a new course to the course listing using it with the POST request.

After the course is added to the course listing, XML file that deals with adding students to the Course should be developed so that it can processes the student registration.
Restlet applications are very light weight applications which for the most part extend from two basic and important classes: Application and Resource. The application classes will map the URI to a Resource instances and Resource classes handle CRUD commands.

To start of building up a Restlet the first step is extending the Application class to define the resources that respond to the URIs. This process is aided by Router class in the Rest API. Let us say all the CRUD operation of the /courses and /course/courseid are handles by the Courses.java program code for assigning the object will be
Router router = new Router (this.getContext());
router.attach("/courses", Courses.class);
router.attach("/courses/courseid" Courses.class);

In the example the /courses and /courses/courseid is now mapped to Courses class which in turn will extend Resource class.

Mapping for entire example:

public class RaceApplication extends Application{
    public RaceApplication(Context context) {
        super(context);
    }
    public Restlet createRoot() {
        Router router = new Router(this.getContext());
        router.attach("/courses", Courses.class);
        router.attach("/courses/courseid" Courses.class);
        router.attach("/race/courseid/studentid", StudentResource.class);
        return router;
    }
}

Now the Courses and StudentResourse class handle all the CRUD requests for the URI mapped to them.
6.2.2 Courses Class

After the Application instance to handle all the URI has been mapped the resources, the next step that should be implemented is to develop the Resources. Any High level Resource has four methods that are to be overridden. These four methods implement the four HTTP CRUD commands for the specific class. Frame work requires the user to implement paired methods like allowDelete() methods that return true should precede the delete() method. But default allowPut(), allowDelete() and allowPost() return false and allowGet( ) methods will return true. The courses class will have other methods that will handle the incoming request and generate the required output documents in the one of the standard formats such as JSON or XML.

In this section we have discussed how the web service can be developed using the REST API. The web service in our architecture is responsible for processing a request from the Flex front and forwarding it to the Java Middle layer. In the next section we shall see how the Flex front end will send the request and how it processes the incoming XML file.
Chapter 7 Visualization using Adobe Flex

7.0 What are Rich Internet Applications?

According to Wikipedia “Rich internet applications are web applications that have most of the characteristics of desktop applications, typically delivered either by a standards based web browser or via a browser plug-in”. Traditionally web applications that allowed user interactions constantly kept in touch with the server for new information based on the every single user input. [27] These web apps suffer from performance issues and the performance of the applications was dependent on how quickly the client-server communication happens and how quickly the server generated the response to the request received from the client. Rich internet application which are also web application try to overcome this problem by downloading a runtime engine which will reduce the number of interactions with the server and will make sure apps communicates with the server only when it is necessary. The run time engine will make sure the UI interactions and related activity happen on the client side and data manipulation and data manipulation and other operations happens on application server side.

To develop a Rich internet applications vendors round the globe have designed and released various frame works into market. Listed below are a few of them

- Adobe Flex
- Microsoft Silverlight
- Google Web Toolkit
• JavaFX

In our Vijjana Frame work we have Adobe Flex with Prefuse Flare API to create the Visualization agent.

7.1 Visualization

According to Wikipedia Visualization is “any techniques for creating image, diagrams, or animations to communicate a message”. Right from ancient history human relied on visuals to communicate and preserve information. A good example for this is cave paintings which helped modern historians to understand the life of the early man.

There are various applications of visualization which include scientific visualization, educational visualization, information visualization etc. In the Vijjana system we have use the Information visualization. Information visualization was defined as “The use of computer-supported, interactive, visual representations of abstract data to amplify cognition.” [28]

7.2 Why use Flex? [29]

• One of the Biggest advantages of Flex is a code written in Flex can run on any web browser with a Flash Player Plug-in. According to a recent study published by Adobe the Flash player in now installed on 99% of the World's Internet capable PC's.[30] So any applications that are developed in Flex will have wider audiences and will run without any hassle.
• Flex was developed based on existing standards such as MXML and Action Script and it is an object oriented framework. So learning or adapting the Flex platform should not be a big problem for developers familiar with these standards.

• Front-end Flex platform can be associated with any back-end technology ranging from Java to .NET.

• Flex provides developers with a rich set of components and Flash effects which will help in developing applications that offer rich and highly fluid experiences to end users.

• Any program that is developed in Flex will not have cross-browser compatibility issues like applications developed in AJAX.

7.3 How does Flex Work? [31]

Originally most web developers and also casual designers used Adobe Flash. To a developer selecting components that are needed, specifying the time line and also dealing with other visual problem is something they were not used to. So Adobe Flex was designed to help programmer work in an environment they are used to, like an IDE, classes that can be extended and designs that can be created from MXML documents. Flash uses rich set of components along with a time line for creating the layout of RIA and Action Script for adding functionality to the User Interface. Flex extends this concept and it uses MXML for specifying the layout and Action Script to specify the functionality of the Application. When a Flex program is compiled, both the MXML code and Action Script code are converted to Action Script and then the Action Script code is converted to SWF files. These SWF files can be saved on to the server and these files can be accessed by any user. The Flex application can also request web server/service for data and this application server contacts the database and this data in relayed back to the Flex application which will then be used to generate the UI for the user.
7.4 Action Script and MXML

7.4.1 MXML

MXML was introduced as a part of Flex programming language. MXML is a XML based markup language is much like HTML used to layout the user interface of a Flex Application. MXML has more structured syntax when compared with HTML and also MXML had richer set of tags when compared to HTML. Another advantage that MXML enjoys is it allows the user to create custom tags. MXML when compiler will create a user interface rendered by flash player and this offers rich user experience when compared HTML-based page-centric web applications. Being a markup language MXML does not have adequate support for developing business logic for application. In Flex all event handlers are coded in Action Scripts which based on ECMA -262 Standards

Example for MXML

```xml
<?xml version="1.0" encoding="utf-8"?>
  <mx:Button x="264" y="156" label="I am a Button" id="Button" enabled="true" labelPlacement="top"/>
</mx:Application>
```

This application is compiled; output will be a SWF file with a button in it.
The next section will deal with Action Script, which will help a developer write event handlers.

7.4.2 Action Script

Action script is an object-oriented programming language based on ECMA Script. Action Script was primarily used to develop web applications along with Adobe Flash to specify the business logic. But when Adobe released Flex for developers they actually replaced the components and timelines with MXML and reused Action script to provide business logic to the components created in MXML. It can be used along with MXML to perform various activities that a user needs in his applications. One of the important functions of action script is to define event listeners inside MXML event attributes.
7.5 Data Access in Flex

Flex supports many different ways of accessing server-side data. Depending on the type of interface the server side implements Flex can be connected to the server-side application using one of following techniques. [32]

- HTTP Get and POST by using the HTTPService component.
- SOAP complaint web services using Web Services component.
- Adobe Action Message Format remoting services by using the Remote Object component.

7.5.1 Data Access used in Vijjana’s Visualization Agent

We have used HTTPService component along with Rest Web service to provide access data access. The reason why HTTP Service component was used is HTTP service is

- Rest Web service generates XML instead of HTML. Http component in flex aided by excellent XML-handling capabilities helps us manipulated data for display in the Flex application’s user interface.
- Use of Rest services along with Flex frame work will help us scale up the Vijjana Frame work in future. Adding additional functionality will be easy as the Rest API provides XML files when it received Http Request. By changing the arguments in the HTTP Request different XML files can be generated when can then be used to develop various other tools. Since XML files can be used with in any frame work making our entire application platform independent. So tools that can be developed based on this frame
work that we are creating can be created in any platform making it attractive to developers.

So the main objective of using RestAPI along with HTTP Service Flex is to provide future users of Vijjana and also web developers, a frame work on which an entire eco system of tools can be developed adding additional functionality to Vijjana Web application.
7.6 Prefuse Flare

Prefuse flare [5] is extensible software framework which helps developers in creating interactive visualization applications in action script. The biggest advantage of this API is enables people to create interactive web applications as it runs in Adobe Flash player. One the tops of all the advantage this API is an open source and under the BSD license can used in developing both commercial and non-commercial applications.

Toolkit Structure

Source Data: Source data is data set that the user wants to represent in form a visual. Source data in the Vijjana System is available in the Vijjana Database.

Data Tables: The source data in then converted to data table which is an internal representation of the data that should be visualized. The process of generating data table from source data includes reading data from a database or extracting data from formatted files (Ex GraphML files)

Visual Abstraction: In this step data in the data tables will be used to create a visual abstraction. In this step the user will create a data model by specifying the data model that includes visual effects such as layout, coordinated, shapes and colors to be used.

Views: The visualization is finally reflected in the views and this provides the user with a good interface to interact with the application. The view should be so coded that any change in the visual representation such as rotation, changing type of graph should not change the underlying data.
Figure 11: Prefuse Architecture [5]
Chapter 8: Implementation

8.0 Implementation

The first step in the implementation process is a generic XML schema is designed. The XML schema designed will make sure the final XML schema generated will adhere to the GraphML format. Using this XML schema with JAXB will allow us generate the Data objects.

When a user logs into the Vijjana system and requests for a visual of the knowledge network, Flex front end will send a HTTP request to web service using HTTP service. The web service will process the request and will forward it to the Java Middle layer.

The Java Middle layer will then query the database for information using Hibernate and will populate the data objects generated on Step 1. The Hibernate frame work will use Hibernate query language to query the database. The data objects generated on step 1 will be used along with JAXB using the marshaling technique will generate the XML file. The XML file will forwarded to the Flex part through the Web service. The XML received will be parsed using the XML parsing tools with in the Flex frame work and along with Prefuse Flare frame work will create the visual of the knowledge network.
1: XML schema was designed. The same schema is used
for all users. Using XML schema the Java data objects are
generated. (Using JAXB unmarshalling)
2: The Flex Front End will send a HTTP request to
the Web service(Restlet)
3: The webservice will request the Java Middle Layer
to process the request received from Flex front end.
4 & 5: The database will query the database for the
required information. The Java middle layer
communicates with the database using O/R libraries
like Hibernate.
6: The Java middle layer will use data from the
database and will generate the XML file using JAXB
(Marshalling)
7 & 8: The Java middle layer send the XML file to the
Flex front end through the Web service.

Figure 12: Implementation
Chapter 9
Future Work and Conclusions

9.1 Future Work

Vijjana framework is in its very early stage of development. One of the biggest limitations of the current framework is that it does not have a mechanism to build up taxonomy automatically. For the framework to function, the Vijjana team uses the Taxonomy provided by Open Source Project in order to build the system. The visualization agent discussed in this paper also uses the same taxonomy. Building up a system that can automatically extend the taxonomy based on URL’s marked up could be one of the biggest improvements to the current framework.

The Web service developed for the Visualization agent can be used to develop various tools. By changing arguments of the HTTP request (sent to the web service) XML files can be generated. These XML files can be used along with various applications to develop tools which perform a variety of functions based on data in the Vijjana database. Since the web service generated XML files, these tools can be developed using any platform ranging from Java to .NET. So any application that can send out HTTP requests and process XML files it receives should be able to use the architecture described in this report.
Now we are providing support for just one type of view. The existing application can be modified to provide the user with various other view such as a tree view and hyper graph.

**9.2 Conclusions**

Visualization agent was developed which will help in visualizing the knowledge network for each user. The graph will be in form of a Flash object can user can interact with the object to find the required URL.
References


14. **Bray, T, Paoli, Initials, Maler, E, & Sperberg-McQueen, C.M.** *Extensible markup language (xml)*. Retrieved from [http://www.w3.org/TR/xml/](http://www.w3.org/TR/xml/)

15. **Fallside, D.C, & Walmsley, P.** *Xml schema part 0: second edition*. Retrieved from [http://www.w3.org/TR/xmlschema-0/](http://www.w3.org/TR/xmlschema-0/)


27. **Leahy, P.** *Definition for the term: ria*. Retrieved from [http://java.about.com/od/r/g/ria.htm](http://java.about.com/od/r/g/ria.htm)


