Film/Movie finder: Searching and Accessing Movies Using DBpedia.

Robert Beirne
B.A.I. Engineering
Final Year Project April 2012
Supervisor: Mary Sharp

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
Acknowledgements

First, I would like to thank Mary Sharp and Melike Sah for their help and guidance during this project. Their patience and support is much appreciated.

I would like to thank my parents for the opportunities they have given me, especially those of which I have taken for granted.

I would like to thank Dr. Richie Porter and Dr. Laurence Gill for fighting my corner when I needed it most.

I would like to thank Michael Walsh for his time and patience.

I would like to thank Rory O’Sullivan for instilling in me a love and deep appreciation for film.

I would like to thank Stephen Cantrell, Helen Molony, Eveanne Slattery and Fionán Keenan for keeping me sane in less than sane circumstances.

Finally, I would like to thank James Judge, Rebekah Clarke, Joshua Montgomery, CP O’Reilly and Rudi O’Reilly Meehan, without whom I would never have made it this far.
DECLARATION:
I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

______________________________   ____________________
Robert Beirne                        4th of April 2012
III

Permission to Lend:

I agree that the Library and other agents of the college may lend or copy this thesis upon request.

____________________________________  ________________
Robert Beirne  4th of April 2012
IV

Abstract:

The aim of this project is to perform sophisticated queries of relationships between Linked Data found in the DBpedia knowledge base.

This will be done using a Servlet so that a user can simply enter keywords into a search form and select their preferences to perform the query, rather than having to write an actual SPARQL query themselves.

This means that anybody would be able to query the DBpedia knowledge base. No knowledge of programming would be necessary.
Contents

I Acknowledgements 2

II Declaration 3

III Permission to Lend 4

IV Abstract 5

V Contents 6

VI Table of Figures 9

1 Introduction 13

2 Objective 20
   2.1 Required Features 20
      2.1.1 User Interface Form 20
      2.1.2 SPARQL Queries 21
2.1.3 Java Servlet 21

2.2 Report Overview 21

3 Background 22
3.1 Semantic Web 22
3.2 Linked Data 25
3.3 DBpedia 26
3.4 SPARQL 30
3.5 Java Servlet 32

4 Design 33
4.1 Introduction 33
4.2 Design of HTML Form 35
4.3 Design of SPARQL Queries 38
  4.3.1 Prefix Declarations 38
  4.3.2 Dataset Definition 38
  4.3.3 Result Clause 39
  4.3.4 Query Pattern 39
  4.3.5 Query Modifiers 39
  4.3.6 Example SPARQL Query 40
4.4 Design of Servlet 42
5 Implementation 46
  5.1 Preparation for Implementation 46
  5.2 Apache Tomcat 51

6 Conclusions 56
  6.1 Successes 56
  6.2 Failures 58
  6.3 Potential for Future Work 60
  6.4 What I learned 61

7 References 62

8 Appendix 67
Table of Figures

1.1 Total number of articles on the English edition of Wikipedia. 14

1.2 Monthly growth of the English edition of Wikipedia. 16

1.3 DBpedia page for “A Film With Me In It”. 19

3.1 Timeline of W3C’s Semantic Web standards. 23

3.2 Web of Linked Data. 27

3.4 SPARQL Explorer for querying DBpedia. 31

4.1 Design of Project. 34

4.2.1 HTML Form. 35

4.2.2 HTML Code for form. 37
4.2.3 HTML Code for form (Continued). 37

4.3.1 Prefix Declarations of SPARQL query. 38

4.3.3 Result Clause of SPARQL query. 39

4.3.4 Query Pattern for SPARQL query. 39

4.3.5 Query Modifiers for SPARQL query. 39

4.3.6 Example SPARQL query. 40

4.3.7 Results of Example SPARQL query. 41

4.4.1 How to import Jena files and libraries into Servlet. 42

4.4.2 Servlet name definition. 43
4.4.3 Proxy settings for Servlet. 43

4.4.4 HTML form entry into Servlet. 44

4.4.5 SPARQL query entry into Servlet. 45

5.1.1 ‘lib’ folder of Jena file. 47

5.1.2 How to add CLASSPATH. 48

5.1.3 Network Settings for Virtual Machine. 49

5.1.4 Proxy settings for Mac OS. 50

5.2.1 Location of ‘lib’ folder in Tomcat 7.0. 51

5.2.2 ‘lib’ folder in Tomcat 7.0. 52

5.2.3 How to run Tomcat 7.0. 53

5.2.4 How to start Tomcat service. 54
5.2.5 How to call the Servlet. 56

6.1.1 Another example SPARQL query. 57

6.1.2 Results for 2nd example SPARQL query. 58

6.2 Result when servlet is run in IE. 60
Chapter 1

Introduction

Wikipedia is the largest online encyclopedia in existence today. It consists of around 21 million articles (3.9 million of which are in English) and has around 100,000[4] regularly active contributors who update and edit the site and who keep it up to date with concise, accurate information on a wide range of topics. At the time of writing, there are editions of Wikipedia in 284 different languages.[3] It has an estimated 365 million readers worldwide[5] and receives, on average, an estimated 2.7 billion page views every month from the USA alone[6]. This makes it the 6th most viewed website in the world. In other words, it is the single largest and most popularly used general reference work on the internet. It is essentially the largest encyclopedia in existence.
Figure 1.1 below shows the total number of articles on the English edition of Wikipedia since it was founded in January 2001.[12]

![Figure 1.1](image)

The vast amount of knowledge and information contained in Wikipedia is unprecedented. Never before has so much information been so readily available in one place. It is available to anyone with access to the internet and is completely free to use. However, the sheer size of Wikipedia is not its greatest asset. It is constantly updated by users in order to keep the information accurate and up to date. The ability for
users to update information and even create articles for new, important events as they unfold is invaluable to the success of wikipedia. With regards to film, this means that as soon as the details of the cast, budget, release date, etc. of a film are announced, this information can be, and inevitably is, added to the Wikipedia page for said film. This allows a reader to access up-to-date information at all times. Articles on Wikipedia are not owned by anyone and are agreed on by consensus.[10]

Although the number of articles in the English edition continues to grow at an astonishing rate (on average around 1,000 new articles were created every day in 2010) the growth of the English edition of Wikipedia has slowed down considerably since its peak in early 2007 (at which time an average of around 1,800 new articles were being created on a daily basis).[11]

This does not, however, indicate a waning interest in Wikipedia. This slowed growth can, in fact, be easily explained. As the Wikipedia project gained momentum and awareness of this ambitious venture grew, many people began to create articles. Eventually it got to the stage where Wikipedia had essentially caught up with society. This decrease in the growth of Wikipedia was therefore inevitable as contributors had to begin to wait for new topics to arise which would merit an article. It stands to reason that articles about more obscure topics would be created less regularly than articles on the more obvious topics, simply because more people have knowledge on these topics.

Figure 1.2 shows the monthly growth of the English edition of Wikipedia since it was founded in January 2001.
Decisions such as the one announced in March 2012 that the 15th edition of the oldest encyclopedia in the English language, Encyclopædia Britannica, would be its last print edition only serve to further the belief that the Wikipedia model is the way forward.

One of the major criticisms of Wikipedia is that in allowing anyone to edit the information on the site, it is vulnerable to misinformation being posted and to online vandalism. What this essentially means is that at any given time, on any given article, there is a chance that some misinformation is present (be it deliberate or accidental). This is opposed to printed encyclopedias which pride themselves in being written by people who are considered experts in their respective fields. However, the median time between when any misinformation is posted
and when it is corrected is generally a matter of minutes.[7] In addition to this, a study carried out in 2005 by the science journal *Nature* found that the science articles which they investigated and compared to equivalent articles in the Encyclopædia Britannica contained a similar amount of what they deemed to be “serious errors” as the highly regarded printed encyclopedia.[8]

Wikipedia articles consist largely of text which the user can read to find the information they are looking for. What this essentially means is that a user can search for certain keywords on Wikipedia.org and the search will return a number of related articles. The user can then read these articles in order to find what, specifically, they are looking for. This is fine for human users, but when a machine tries to access this information it finds it much more difficult to extract meaningful and useful data.

Wikipedia also, however, contains different types of structured information which is embedded in the articles. Some examples of this type of structured information are infobox templates, categorisation information and links to external webpages. DBpedia extracts this structured information and combines it in order to create a very large knowledge base which covers many different domains. This knowledge base is available under free licenses. This means that anyone can reuse the dataset to perform sophisticated queries on the information stored in the knowledge base.

DBpedia is described on its website as “a community effort to extract structured information from Wikipedia and to make this information available on the web”[1]. In essence, what this means is that DBpedia
allows you to access the information stored on Wikipedia programmatically, and to query this information and the relationships and properties associated with Wikipedia resources. DBpedia also allows us to link other data sets on the web to data found on Wikipedia.

Since DBpedia accesses its information directly from Wikipedia, its knowledge base covers many domains. This knowledge base is also constantly changing and evolving along with Wikipedia, meaning the information remains up-to-date and accurate. DBpedia’s links to Wikipedia also mean that the information accessed by DBpedia represents real community agreement. It contains over 1 billion pieces of information. This number is made up of around 385 million pieces of information which have been extracted from the English edition of Wikipedia and around 665 million which have been extracted from other language editions of Wikipedia and links to external datasets.[1] This demonstrates another advantage which using DBpedia has over using other knowledge bases: DBpedia is multi-lingual, and when querying DBpedia, the user can filter which language editions of Wikipedia they would like to query. In fact, DBpedia extracts various different types of information from 97 different language editions of Wikipedia.

I decided to base my project on the Film domain of DBpedia. I have a very strong interest in film and found there to be a very wide vocabulary in DBpedia for film. I thought it would be an interesting domain to study and there would be plenty of examples of interesting relationships to query between pieces of Linked Data. Figure 1.3 below
Figure 1.3 shows the vocabulary for the DBpedia page for an Irish Film called “A Film With Me In It”. 

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbpedia-owl:abstract</td>
<td>A Film With Me In It is a 2008 Irish black comedy film directed by Ian Fitzgibbon, written by and starring Mark Doherty. A Film With Me In It is a thriller de comedia. Foi realizado em 2008, e é dirigido por Ian Fitzgibbon, escrito por Mark Doherty, com Dylan Moran, Mark Doherty e Amy Huberman nos principais papéis.</td>
</tr>
<tr>
<td>dbpedia-owl:releaseDate</td>
<td>2008-10-17 (xsd:dateTime)</td>
</tr>
<tr>
<td>dbpedia-owl:runtime</td>
<td>4980.0 (xsd:double)</td>
</tr>
</tbody>
</table>
| dbpedia-owl:starring | dbpedia:Keith_Alan_(actor)  
  dbpedia:David_D_O'Hearty  
  dbpedia:Dylan_Moran  
  dbpedia:Mark_Doherty  
  dbpedia:Mark_Doherty  |
| dbpedia-owl:thumbnail | http://upload.wikimedia.org/wikipedia/commons/thumb/3/3D/A_Film_with_my_in_it.png/200px-A_Film_with_my_in_it.png |
| dbpedia-owl:writer | dbpedia:Mark_Doherty                                                                 |
| dbpedia:cinematography | Seamus Deasy                                                                  |
| dbpedia:director | Ian Fitzgibbon                                                                 |
| dbpedia:distributor | Parallel Films                                                               |
| dbpedia:editor | Tony Cranagh                                                                     |
| dbpedia:language | English                                                                        |
| dbpedia:movie | A Film With Me In It                                                            |
| dbpedia:producer | Alan Moloney                                                                   |
| dbpedia:released | 2008-10-17 (xsd:dateTime)                                                   |
| dbpedia:runtime | 4980.0 (xsd:double)                                                          |
| dbpedia:starring | dbpedia:Keith_Alan_(actor)  
  dbpedia:David_D_O'Hearty  
  dbpedia:Dylan_Moran  
  dbpedia:Mark_Doherty  
  dbpedia:Mark_Doherty  |
| dbpedia:studio | dbpedia:Irish_Film_Board                                                        |
| dbpedia:wikiPageLink | Template:info_box_film                                                          |
| dbpedia:wordnet_type | http://www.w3.org/2000/01/rdf-schema#instance/synset-movie-movien-1 |
| dbpedia:writer | Mark Doherty                                                                   |
Chapter 2

Objective

The objective of my project was to use Jena to search the SPARQL endpoint of DBpedia in the Film domain and to display the results of my queries in a Servlet. Although DBpedia allows the user to query the information extracted from Wikipedia directly, the user must have at least a basic knowledge of SPARQL to do this. My project aims to allow a user to simply enter keywords into a form in order to query DBpedia. The project should also have the capacity for future expansion.

2.1 Required Features

There were three main features which I had to design in order for my project to reach its objective. These features were as follows:

2.1.1 User Interface Form

This was designed to allow the user to enter keywords which would then be used to query DBpedia.

2.1.2 SPARQL Queries

These were designed to query DBpedia to get the results that the user is looking for.
2.1.3 Java Servlet

This was designed to combine the form and the SPARQL queries. It is also where we planned to return the results for the queries to the user.

2.2 Report Overview

Chapter 3: Background
In this chapter I aim to give a basic outline of the technologies used in the project. I will also attempt to give reasons why I used these technologies as opposed to others.

Chapter 4: Design
In this chapter I will discuss the design of my project as a whole and how I designed each of the different parts of the project.

Chapter 5: Implementation
In this chapter I will describe how I implemented the design of the project in order to reach my objective. I will discuss the problems I faced and the difficulties I encountered.

Chapter 6: Conclusions
In this chapter I will look at the final results of the project and reflect on the successes and failures of project. I will also discuss the potential for future work on this project and the lessons learned from carrying out this project.
Chapter 3

Background

3.1 Semantic Web

The Semantic Web is a collaborative movement led by the W3C (World Wide Web Consortium) that promotes common formats for data on the world wide web. The W3C also oversee the development of proposed Semantic Web Standards. The W3C say of Semantic Web that “The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries”. [19] It is essentially a web of data. The aim of semantic web is to convert the current web of unstructured information into a web of structured and, most importantly, interlinked information. This is done through the inclusion of semantic data on webpages which can be easily extracted by a machine and linked to other pieces of data.

Where the current web of unstructured data concentrates largely on the exchange of documents, Semantic Web uses common formats in order to integrate and combine information extracted from many different sources. Semantic Web also uses its own language in order to keep track of how the extracted data relates to objects in the real world. By doing this, Semantic Web allows a user to start off in one database, and move through a never ending set of databases which are all related to each other by having the same linked information as each other. The Semantic Web is based on the Resource Description Framework (RDF).
Figure 3.1 shows the timeline of W3C’s Semantic Web Standards.[20]

Figure 3.1

The Web as we know it today is a hugely valuable resource for mankind and it is very easily navigated by human users. However, machines cannot navigate the current web efficiently without human direction. This is because the vast majority of webpages have been designed to be read easily by humans, with no thought to whether or not a machine can access the information as easily. The Semantic Web movement is the driving force behind the evolution of the current web into a web of structured information which can be readily interpreted by machines. By structuring information in this way, human users can allow machines to perform the more tedious work of carrying out searches for the information the human user requires, as opposed to the user reading through page after page of unstructured data in order to find what they are looking for.
The aim is to have a system which allows computers to understand and respond to complex requests from human users and carry out these requests automatically. This can not be done unless the relevant sources of information are correctly structured using Semantic Web standards. Essentially, the ultimate aim of the Semantic Web movement is to one day have the same knowledge base accessible to machines as to humans.

Tim Berners-Lee, creator of the world wide web, said of the Semantic Web movement:
“I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.”[13]
3.2 Linked Data

The use of linked data is a key component in the success of DBpedia. Linked data is used to share information in a way that can be easily read by machines. It is essentially a way of publishing structured data so that it can be linked to other datasets. This allows data from different sources to be connected and queried when required.[13]

There are 4 main principles of linked data:
1. Use URIs (Uniform Resource Identifiers) to identify pieces of information. A URI is a string of characters used to identify a name or a resource.
2. Use HTTP URIs so that people can look up the information identified by the URI.
3. When a URI is looked up (dereferenced), provide useful information using standard formats such as RDF and SPARQL.
4. Include links to other, related URIs to help the user discover other related information on the web.[14]

RDF (Resource Description Framework) is a family of W3C (World Wide Web Consortium) specifications. It is a data model which is based upon the idea of making statements about web resources in the form of subject-predicate-object expressions, known as triples (In general, RDF is not limited to the description of web resources but for the purposes of this project this will always be the case).[15] RDF is, for all intents and purposes, a directed, labelled graph data format for representing information on the web.
Everything described by RDF expressions are called ‘resources’. A resource could be an entire webpage, part of a web page or a collection of web pages. Everything used to describe a resource are called ‘properties’. A property can be a specific aspect, characteristic, attribute or relation used to describe a resource. An RDF statement is made up of a specific resource, a named property for that resource and a value for the named property. These parts of the statement are known as the subject, predicate and object respectively, as mentioned above.

The subject of an RDF statement denotes the resource, the predicate denotes traits or aspects of the resource and also expresses a relationship between the subject and the object. The subject of an RDF statement is either a URI or a blank node (in which case the resource is known as an anonymous resource) both of which indicate resources. The predicate is also a URI and also denotes a resource, representing a relationship. The object is a URI, blank node or a Unicode string literal.

The aim of linked data is to create a web of structured information which is all linked by the relationships between these pieces of information. This web currently consists of several billion RDF triples.[1]
DBpedia is right at the centre of this web as shown in Figure 3.2.[16]
3.3 DBpedia

The DBpedia dataset is linked with many other open data datasets on the web on the RDF level. This means that its data is enriched by the data from all of these other datasets. DBpedia represents the information it extracts from Wikipedia and external links using RDF. Sir Tim Berners Lee, creator of the world wide web, described DBpedia as an integral part of the linked data project.[17] It is essentially an RDF version of information from Wikipedia.

The DBpedia knowledge base is served as linked data on the web. As mentioned above, many data providers have begun to set RDF links to DBpedia from their own datasets. This is the reason that DBpedia is now the central interlinking-hub[1] of the vast web of structured data which is now available to us. It uses RDF to represent extracted information and to publish it on the web.

The DBpedia knowledge base currently describes around 60,000 films. The film domain of DBpedia has its own vocabulary which allows us to query the dataset effectively. Everything in this dataset is identified by a URI reference. Each resource in DBpedia is described by a label, a short and a long English abstract and a link to the corresponding Wikipedia page.[18] The DBpedia ontology is a shallow, cross-domain ontology, which has been manually created based on the most commonly used pieces of structured information within Wikipedia itself.

The DBpedia ontology aims to be a common scheme for articles in Wikipedia. It is a collection of unique names and identifiers for the
strings used in infoboxes on Wikipedia. These identifiers are known as the vocabulary of the domain. The aim of the DBpedia ontology is to bring together the information that belongs together and to separate the strings which have different meanings. It provides a single name for each piece of information. This makes it much easier for computers to extract this information. This in turn allows the user to query this information and the relationships that this piece of information has with other pieces of information in the dataset much more effectively. As well as names for properties the ontology also contains classes. These classes provide a strict categorisation system for articles. The DBpedia ontology adheres to common Semantic Web standards like RDF and OWL.

What DBpedia allows us to do is to query the relationships between pieces of information in ways which are impossible using only Wikipedia. By querying DBpedia we can locate more specific information of a subject, more efficiently. What this means is that the linking and reasoning abilities of RDF and OWL (Web Ontology Language) can be utilised and queries for specific information can be made using SPARQL. OWL is a family of knowledge representation languages for authoring ontologies. Each infobox template in Wikipedia can be assigned an OWL class from the DBpedia ontology. In addition to this, each parameter in the template can be assigned to an ontology property.
3.4 SPARQL

SPARQL (SPARQL Protocol And RDF Query Language) is an RDF query language for databases, similar to SQL. It is used to locate and manipulate data stored in RDF format. It consists of a standard query language, a data access protocol and an RDF based data model.[23] It is the standard query language for the Semantic Web and it can be used to express queries over a wide range of diverse data sources. SPARQL is capable of carrying out queries consisting of triple patterns and optional patterns, as well as their conjunctions and disjunctions. In other words, SPARQL is capable of performing both “AND” and “OR” queries. SPARQL queries can return results as either result sets or RDF graphs.[24]

SPARQL allows a user to extract values from structured and semi-structured data. It then allows the user to explore this data by querying unknown relationships between datasets. SPARQL also allows the user to query a wide range of databases at the same time using a single, simple query. This means that the query can return results from many different datasets. It also allows the user to transform the RDF data which it finds from one vocabulary to another.

There are several tools online which allow a user to query DBpedia using the SPARQL endpoint. The SPARQL endpoint allows users (be it human or machine) to query a knowledge base (in this case DBpedia) using the SPARQL language.[25] However, a SPARQL endpoint usually returns results in a format which is easy for a machine to process rather than a human. This means that a SPARQL endpoint is generally thought to be a machine-friendly interface towards a knowledge base.
Figure 3.4 shows a SPARQL Explorer for querying DBpedia.[26]

![SPARQL Explorer for http://dbpedia.org/sparql]

We must therefore come up with a way to process the results returned by the SPARQL endpoint and return to the human user in a way which will be easy for them to understand and interpret. We must also find a way for a human user to easily construct a query without a detailed knowledge of the SPARQL language. Ideally, both the formulation of the queries and the human-readable results should be implemented by the calling software.[25] In other words, the human user should not have to worry about building the query or interpreting the results; this should be done for them by some program, in this case a Java Servlet.
3.5 Java Servlet

A servlet is a Java application that runs in a Web server or application server and provides server-side processing; in this case accessing a database. Servlets are often used for the processing of data on the web. They are designed to handle HTTP requests although they can respond to any type of request. They are not limited to a specific client-server protocol but they are usually used with the HTTP protocol. They are most commonly used to extend the applications hosted by Web servers.[27]

Servlets are often used to process or store data that was submitted from a HTML form and to provide dynamic content such as the results of a database query. In this project, the Servlet is used to carry out these two functions.

Servlets are easier to use and more powerful than traditional CGIs (Common Gateway Interfaces). This is because when a HTTP request is made to a CGI, a new process is created for each time the CGI script is called. However, when a HTTP request is made to a Servlet, each request is dealt with by a separate Java thread. This means that if there are many CGI requests running simultaneously, the CGI script is copied and loaded into memory as many times as there are requests. Although there are the same amount of threads as there are requests when a HTTP request is made to a servlet, there is only one copy of the servlet class created in memory. This one copy also remains in memory in between requests.[28]
Chapter 4

Design

4.1 Introduction

The design of this project required three components. The first component was the HTML form into which the user would submit the keywords for the query they wished to carry out. The second component was the set of SPARQL queries themselves. The third component was the Java Servlet which would use the first two components together in order to return the information which the user was seeking by querying DBpedia.

The project was designed in such a way that it takes a keyword (or keywords) which have been entered into the HTML form by the user. It then searches DBpedia for the chosen keyword(s), returning an RDF in the form of a DBpedia URI. It then sends a HTTP request to the servlet. The Servlet then carries out the relevant SPARQL queries for the search and displays the results in the Servlet. This process is illustrated in Figure 4.1 on the next page.
Figure 4.1
4.2 Design of HTML Form

The design of the HTML form had to be simple, attractive, and easy to use. I decided to use a very minimal design from which the user could select their own preferences for the search. The user would then enter keywords into the space provided beside the option they choose. If the user wishes to carry out a more specific search, they can select an option in the “Refine Search” field.

Figure 4.2.1 shows the the HTML form which is presented to the user when they run the application.

![Film Search Form](image)

**Figure 4.2.1**

For example, if the user wished to search for a list of films starring Jack Nicholson, they would select the “Actor” option in the “Keyword Search” section of the form and then in the empty “Actor” field, they would enter the words “Jack Nicholson” and then press the “Search” button at the bottom of the form.
If the user wished to perform a more detailed query of the DBpedia knowledge base, they would, for example, select the “Actor” option in the “Keyword Search” section and enter “Jack Nicholson” into the empty “Actor” field, and then select the “Year of Release” option in the “Refine Search” section and enter “1980” into the empty “Year of Release” field. The user would then press the “Search” button to submit the query.

Figure 4.2.2 and Figure 4.2.3 on the next page show the HTML code used to design this form. The code in Figure 4.2.3 follows on directly from the code in Figure 4.2.2 and is part of the same HTML file.
Figure 4.2.2

Figure 4.2.3
4.3 Design of SPARQL Queries

For this project it was necessary to design several different SPARQL queries. When the user enters their preferences into the HTML form, the relevant query can then be chosen and carried out by the servlet in order to return the information that the user requires from the DBpedia knowledge base. SPARQL queries are usually made up of the following parts (in order):

4.3.1 Prefix Declarations

Prefix declarations are used for abbreviating URI namespaces. An example is shown below in Figure 4.3.1.

```sparql
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

Figure 4.3.1

4.3.2 Dataset Definition

The dataset definition states which RDF graph(s) are being queried. If we are querying an unknown RDF graph then there is no need for a dataset definition to be included.
4.3.3 Result Clause
The result clause identifies which information is to be returned to the user from the query. An example of a result clause is shown below in Figure 4.3.3.

```
```

Figure 4.3.3

4.3.4 Query Pattern
The query pattern specifies what to query for in the underlying dataset. An example of a query pattern is shown below in Figure 4.3.4.

```
WHERE {
  ?subject rdf:type <http://dbpedia.org/ontology/Film>.
  FILTER(lang(?abstract) = "en" && lang(?label) = "en").
  FILTER(xsd:date(?released) < "1980-01-01"^^xsd:date).
}
```

Figure 4.3.4

4.3.5 Query Modifiers
Query modifiers serve to limit the amount of results, order the results of the query or otherwise rearrange the query results. An example of some query modifiers is shown below in Figure 4.3.5.

```
ORDER BY ?released
LIMIT 20
```

Figure 4.3.5

4.3.6 Example SPARQL Query
When the different components described above come together they make up a SPARQL query. Anything that starts with a ‘?’ in SPARQL is a variable. Figure 4.3.6 shows an example of SPARQL query performed by this project. It is, in fact, the second example query described in section 4.2.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core>;

WHERE {
  ?subject rdf:type <http://dbpedia.org/ontology/Film>.
  FILTER(lang(?abstract) = "en" && lang(?label) = "en").
  FILTER(xsd:date(?released) <= '1980-01-01'^^xsd:date).
}
ORDER BY ?released
LIMIT 20
```

Figure 4.3.6

This query should return a list of films which starred Jack Nicholson and which were released before 1980. It should only return results from the English version of Wikipedia and these results should be ordered starting with the oldest film and then returning films in the order in which they were released chronologically. For demonstration purposes the query is limited to return only the first 20 results it finds.

As you are writing the queries it is very helpful to test them using an online SPARQL Explorer to query DBpedia.[26] By doing this you can make sure that the queries you are writing return the correct results.
This will save a lot of time later on when you are writing your Servlet because if the Servlet is not returning the results you desire, you will know that it is nothing to do with the queries themselves but rather how they are being implemented.

Figure 4.3.7 below shows the results of the example query we looked at above in Figure 4.3.6.

As you can see, the correct set of SPARQL results have been returned by this query. As well as this the results have also been correctly filtered and ordered. We can now move on to designing our next SPARQL query as we now know for certain that this one returns the results we were looking for from the DBpedia knowledge base.
4.4 Design of Servlet

The servlet was designed using Jena. Jena is an open source Semantic Web framework for Java. It is a Java toolkit for developing semantic web applications based on W3C recommendations for RDF and OWL. It provides an API (Application Programming Interface) to extract data from and write data to RDF graphs. The graphs are represented as an abstract “model”. A model can be sourced from files, databases, URLs or a combination of these. These models can also be queried using SPARQL.[30] Jena also provides support for OWL. Jena can be downloaded for free from http://jena.sourceforge.net/downloads.html.[34]

The first thing that must be done in the Servlet is to import all of the Jena files and libraries. This is necessary if we want to use the Jena framework to develop a semantic web application to query DBpedia. Figure 4.4.1 below shows how to import all of these Jena files and libraries into the Servlet.

```
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import com.hp.hpl.jena.ontology.*;
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.query.*;
import java.nio.charset.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import java.net.URL;
import java.net.URLConnection;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
//import java.nio.charset.Charset;
/*import org.json.simple.JSONObject;
import org.json.simple.JSONArray;*/
```

Figure 4.4.1
The name of the servlet must now be defined. This is done by adding the code in Figure 4.4.2 below.

```java
@webServlet(urlPatterns = { "/FilmSearchServlet3" })
public class FilmSearchServlet3 extends HttpServlet {

    // Figure 4.4.2

If you want the servlet to access the internet while connected to the COMPSCIwireless network, proxy settings must be set in the servlet. This is shown in Figure 4.4.3 below. If you are outside the school of computer science, these lines can be commented out.

```java
System.setProperty("http.proxyHost", "www-proxy.cs.tcd.ie");
System.setProperty("http.proxyPort", "8080");
```

    // Figure 4.4.3

The HTML form and SPARQL queries which we designed earlier must now be put into the servlet. The SPARQL queries are separated by the “Union” function which acts as a disjunction between the queries.
The entry of the HTML form is shown in Figure 4.4.4 below.

```
String Form = "<DOCTYPE HTML PUBLIC "+//W3C//DTD HTML 4.0 Transitional//EN">
  "<HTML>"
  "<HEAD>"
  "<STYLE>
   BODY {BACKGROUND-COLOR: #EFEFEF;}
   "<BR><HR>
   "<H1 ALIGN="CENTER">Film Search</H1>"
  "<FORM ACTION="/Servlet/Film.Search" METHOD="POST">"
  "<BR><BR>"<HR>"
  "<H3>Keyword Search:</H3>"
  "<BR><BR>"<INPUT TYPE="RADIO" NAME="SearchName""
    VALUE="Name">Name"
  "<BR>"
  "<INPUT TYPE="TEXT" NAME="SearchName">"
  "<BR>"
  "<INPUT TYPE="RADIO" NAME="SearchDirector""
    VALUE="Director">Director"
  "<BR>"
  "<INPUT TYPE="TEXT" NAME="SearchDirector">"
  "<BR>"
  "<INPUT TYPE="RADIO" NAME="SearchActor""
    VALUE="Actor">Actor"
  "<BR>"
  "<INPUT TYPE="TEXT" NAME="SearchActor">"
  "<BR>"
  "<INPUT TYPE="RADIO" NAME="SearchProducer""
    VALUE="Producer">Producer"
  "<BR>"
  "<INPUT TYPE="TEXT" NAME="SearchProducer">"
  "<HR>"
  "<H3>Refine Search:</H3>"
  "<BR><BR>"<INPUT TYPE="RADIO" NAME="SearchYear""
    VALUE="Year">Year of Release"
  "<BR>"
  "<INPUT TYPE="TEXT" NAME="SearchYear">"
  "<BR>"
  "<INPUT TYPE="RADIO" NAME="Runtime""
    VALUE="Runtime">Run Time"
  "<BR>"
  "<INPUT TYPE="TEXT" NAME="Runtime">"
  "<BR>"
  "<HR>"
  "</CENTER>"
  "<INPUT TYPE="SUBMIT" VALUE="Search">"
  "</CENTER>"
  "</FORM>"
  "</BODY>"
  "</HTML>"
```

Figure 4.4.4
The entry of the SPARQL queries is shown in Figure 4.4.5 below.

```
String Query -
"PREFIX owl: <http://www.w3.org/2002/07/owl#>
"PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
"PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
"PREFIX foaf: <http://xmlns.com/foaf/0.1/>"
"PREFIX dc: <http://purl.org/dc/elements/1.1/>"
"PREFIX : <http://dbpedia.org/resource/>
"PREFIX dbpedia2: <http://dbpedia.org/resource/>
"PREFIX dbpedia: <http://dbpedia.org/>
"PREFIX skos: <http://www.w3.org/2004/02/skos/core#>"

"WHERE {
""FILTER(xsd:date(?released) < "1980-01-01"^^xsd:date)."
""}"
"ORDER BY ?released"
"LIMIT 20"
"}
"UNION"

"WHERE {
""FILTER(xsd:date(?released) < "1980-01-01"^^xsd:date)."
""}"
"ORDER BY ?released"
"LIMIT 20"
```

Figure 4.4.5
Chapter 5

Implementation

5.1 Preparation for Implementation

In order to implement the Servlet for the project, a web server and servlet container must be used. We decided to use Apache Tomcat as the web server for this project. This meant that I was going to have to install Windows on to my Macbook since Apache Tomcat does not run on the Mac OS.

I downloaded Windows 7 from the Microsoft Developer Network[35] and burned it on to a rewritable DVD. I then partitioned the hard disk on my Mac using Boot Camp Assistant. I tried to install Windows 7 on to the partitioned hard disk but was unable to do so. I then managed to obtain a Windows XP installation disk from a friend and installed Windows XP on to my Mac.

However, I was unable to connect Windows XP to the internet and was forced to uninstall Windows XP and to delete the partition on the hard disk of my computer. I contacted the technical staff in college and was advised by Michael Walsh to try and install the version of Windows 7 that I had previously downloaded using a Virtual Machine. I managed to install Windows 7 using Oracle VM Virtual Box. This meant that I could run Windows 7 parallel to the Mac OS and could switch between the two without the need to restart my computer.
Once Windows 7 was up and running and I had installed Jena, the next step was to add the Jena files and libraries into the Classpath of Windows 7. This is done by first going to the downloaded Jena file, then going into the ‘lib’ folder. The ‘lib’ folder is shown in Figure 5.1.1 below.

![Figure 5.1.1](image)

You must then access the control panel, select “System and Security” settings, then click on “System” and select “Advanced system settings”. Once this has been done a dialog box will appear. Click on “Environment Variables...”. When this is done another dialog box will appear. Under the “System Variables” section click on “New...”.

47
Another dialog box appears asking you to enter a variable name and a variable value.

In the “Variable name” field type “CLASSPATH”. In the “Variable value” field you must type the name of each of the Jena files found in the lib folder shown in Figure 5.1.1 including their location. For example: “C:\Lib\annotations-api.jar”. Each file should be separated by a semicolon. I found it to be necessary to create a new folder (named Lib) containing all of these files on the Local Disk (C:). This is because if you try to enter all of these files using the location shown in Figure 5.1.1, not all the files will fit in the space designated for the “Variable value”.

Figure 5.1.2 below shows where this is to be done.
Another problem I encountered in preparing to implement the project was connecting the virtual machine to the internet while I was in college. Again, I contacted the technical staff in Trinity and was again referred to Michael Walsh. We eventually managed to connect the virtual machine to the “COMPSCIwireless2” network.

To do this, go to the Virtual Box VM settings and then go to the “Network” settings. Click on the drop down menu for “Attached to:” and select “Bridged Adaptor”. Ensure that “en1: Airport” is selected in the drop down menu for the “Name:” field. Figure 5.1.3 below shows what the correct Network settings should look like.

![Figure 5.1.3](image-url)
Then go to the System Preferences on the Mac OS and click on “Network”. Click on “Advanced...” and then select “Proxies”. in the table below where it says “Select a protocol to configure:” ensure that “Auto Proxy Discovery” is checked and nothing else. Then connect to the “COMPSCIwireless2” network as usual using your username and password. The virtual machine should now be connected to the internet. Figure 5.1.4 below shows what the correct Proxy settings should look like.

![Network Preferences](image)

Figure 5.1.4

5.2 Apache Tomcat
Apache Tomcat is an open source web server and servlet container. It was developed by the Apache Software Foundation and is free to download from http://tomcat.apache.org/. It implements the Java Servlet and JavaServer Pages technologies. The specifications for these technologies are developed under the Java Community Process.[33] It allows Java code to run in a “pure Java” HTTP web server environment.

Once Jena and Tomcat have been downloaded and installed, all of the Jena files and libraries must be added into Tomcat. This is done by going to the downloaded Jena file, going into the ‘lib’ folder and copying all of the files in this folder. This ‘lib’ folder is the same as the one shown in Figure 5.1.1.

These files must then be added into Tomcat. The destination for these files is the ‘lib’ folder in your webapp. Figure 5.2.1 on the next page shows how this folder can be found.
Once this folder has been located, the files are simply pasted into it as shown in Figure 5.2.2 below.

![Figure 5.2.2](image)

Once the Jena files and libraries have been added to the relevant locations and the Servlet has been compiled, run Tomcat. I found that to do this it was necessary to right-click on “Configure Tomcat” in the Start menu and select to “Run as administrator”. This is shown in Figure 5.2.3 below.
When you open Tomcat you will be asked if you want to allow Tomcat to make changes to your computer. Click “Yes”. Once you have done this, a dialog box entitled “Apache Tomcat 7.0 Tomcat7 Properties” will appear as in Figure 5.2.4 below. Click “Start”. The Service Status should change from “Stopped” to “Started”. It is important to
remember to stop Tomcat if you are making any changes to your Servlet, as the changes will not apply until Tomcat has been restarted.

![Apache Tomcat 7.0 Tomcat7 Properties](image)

**Figure 5.2.4**

Once Tomcat has started, open a web browser such as Internet Explorer. Assuming your Servlet is named “FilmSearchServlet” and is saved in a folder named “Film” in the “webapps” folder of Tomcat 7.0, enter the following URL in the address bar of your web browser: [http://localhost:8080/Film/FilmSearchServlet/](http://localhost:8080/Film/FilmSearchServlet/)
This is shown in Figure 5.2.5 below.

![Figure 5.2.5](image)

This should access and run your Servlet, presenting the HTML form in your web browser. The user should then be able to enter the keywords for their search and select the preferences they want to use for the search. Using this information, the servlet should then query DBpedia using SPARQL and return the results of the query in the web browser.
Chapter 6

Conclusions

6.1 Successes

The SPARQL queries I designed worked very well. When we used the online SPARQL Explorer for querying DBpedia, we found that the queries not only returned the correct set of results, but the results were also correctly filtered and ordered. We saw an example of this in Figure 4.3.7. Another of example query is shown below in Figure 6.1.1 with the results of this query shown in Figure 6.1.2.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX dbpedia-owl2: <http://dbpedia.org/ontology/>
PREFIX dbo: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

  FILTER(lang(?label) = "en")
}

ORDER BY ?runtime

Figure 6.1.1
```
This query returns a list of films directed by Stanley Kubrick which are more than 100 minutes long. The list of films is ordered from shortest to longest. The query only returns results extracted from the English version of Wikipedia. In general, my SPARQL queries returned the correct results.

The design of the HTML form was also a success. It was a simple, functional design which was very easily understood. It allowed the user to select their own preferences for the search they wanted to carry out and to enter keywords for the search. When the user pressed “Search” the keywords were taken to be passed into the SPARQL queries. The HTML form can be seen in Figure 4.2.1 on page 34 of this report.
In the end, I was unable to get the Servlet working. This meant that the components of the project could not be connected as I had hoped. Essentially what this meant was that in order to obtain a set of results from DBpedia, the user would have to run the SPARQL code using an online SPARQL Explorer tool. This was very disappointing as I had hoped that this project would be user-friendly to people with no background in computer science. I had hoped that a user would simply be able to enter keywords in a certain place and have the results returned to the same place. I wanted the machine to do the work for the user but unfortunately, I fell short of reaching this objective.

I would attribute this failure largely to my inexperience in designing and writing Java code (this was the first time I had ever used Java). I had a lot of difficulty in compiling the Java file for the Servlet and when I tried to run the servlet in internet explorer, it did not perform the tasks I had hoped it would. I also think that the proxies on the college network contributed to my difficulty in trying to access DBpedia through the Servlet. I encountered problems with the college network again and again throughout the duration of my project. On more than one occasion I had part of the project working at home, only to find that it would not function when I tested it in college. This also meant that I was unable to test any changes I made to the servlet while I was in college.

Figure 6.2 below shows the result when I tried to call my servlet in the web browser.
6.3 Potential for Future Work
I believe that this a very interesting concept with great potential. The idea of getting a machine to do all of the tedious work in searching for information makes a lot of sense. Ideally, the user should be able to search for keywords using preferences they have set themselves and have the information they desire returned to them efficiently and without the user having to do any work.

If the servlet was carrying out this function, it would be relatively easy to expand the project. The user could be given a larger choice of preferences for their search and be allowed to search for even more specific relationships between data. This could easily be done by adding more options to the HTML form and by designing queries based on these new options. Ideally, this would eventually give the user the capacity to learn extremely specific information on films without doing any more work than entering keywords into one simple search.

The project could also be expanded into other domains. For example, music would be a logical domain to integrate into the film search web app. Users could then search for soundtracks of films, composers of films’ scores and eventually search for specific relationships in the music domain which have nothing to do with film.

The possibilities for expansion of this project are seemingly limitless once the servlet is carrying out the search correctly.

6.4 What I learned.
I found this project to be very interesting. I already knew a little bit about the Semantic Web movement but through my research for this project I have learned a lot more. I had never heard of DBpedia before but now feel as though I would like to contribute to the DBpedia community in their efforts to extract the structured information found in Wikipedia and make it more freely available on the web. I find it to be a fascinating subject which I am sure I will explore further in the future.

Before this project I had never programmed in Java before. Although I would still consider myself far from an expert, I now feel I have a solid base in this language. As Java is one of the most commonly used programming languages in the world, I have no doubt that this will stand to me in the future. I also feel like I now have a basic understanding of the structure behind the development of web based apps, which is a field I have always been interested in and have often considered exploring further.

Although I had done an SQL project in the 1st semester of this year I had never used SPARQL before. The two are quite similar and I found that my basic grounding in SQL was very useful to me in learning SPARQL. I enjoyed querying DBpedia using SPARQL and found the concept of querying relationships between linked data to be very interesting.
Chapter 7

References


[2] JAVA: How to Program by Harvey M. Deitel & Paul J. Deitel


[7] Reid Priedhorsky, Jilin Chen, Shyong (Tony) K. Lam, Katherine Panciera, Loren Terveen, and John Riedl (GroupLens Research, Department of Computer Science and Engineering, University of Minnesota) (November 4, 2007). "Creating, Destroying, and Restoring
Value in Wikipedia" (PDF). Association for Computing Machinery GROUP '07 conference proceedings (Sanibel Island, Florida).

http://www.nature.com/nature/journal/v438/n7070/full/438900a.html


[27] http://www.pcmag.com/encyclopedia_term/0,2542,t=servlet&i=51191,00.asp


[34] http://jena.sourceforge.net/downloads.html


Chapter 8

Appendix

All the code for my project is on a CD attached to this document.