Land & Water

Summer Internship 2015/16



Developing Environmental Vocabularies – Technical Report

By Xavier Butcher. Co-Authors: Jonathan Yu, Simon Cox, Bruce Simons. 29/01/16

Abstract:

Traditional vocabularies in natural language formats like PDFs and CSVs are an ineffective way of displaying concepts, definitions and relationships. This report explains why it is important to create and collate online vocabularies using SKOS and RDF principles. These vocabularies are published on the Linked Data Registry. Detailed below is the method which I have created for efficient conversion from simple tables stored in an online database to fully functioning semantic linked data vocabularies. This project mostly focussed on vocabularies in the soil domain. Furthermore, all of the data references, vocabulary manuals and programs used are comprehensively covered.

Contents

[1 Keywords 2](#_Toc441491942)

[2 Introduction 2](#_Toc441491943)

[3 Materials 3](#_Toc441491944)

[4 Method 4](#_Toc441491945)

[4.1 Summarised Method 4](#_Toc441491946)

[4.2 Full Method 5](#_Toc441491947)

[4.2.1 Background Reading 5](#_Toc441491948)

[4.2.2 Database 5](#_Toc441491949)

[4.2.3 Excel (CSV) 8](#_Toc441491950)

[4.2.4 RDF123 12](#_Toc441491951)

[4.2.5 Text Editor (notepad++) 15](#_Toc441491952)

[4.2.6 TopBraid Composer 18](#_Toc441491953)

[4.2.7 Post Processing (Text) 22](#_Toc441491954)

[4.2.8 Publishing 24](#_Toc441491955)

[5 References 27](#_Toc441491956)

# Keywords

Concept, CSIRO, Database, Editors, Environment, Excel, Internship, Methods, Notepad, RDF, Registry, Semantics, SKOS, Soil, SPARQL, SQL, Taxonomy, TopBraid, URI, Vocabulary, Wiki.

# Introduction

The aim for this project was to create and collate a number of online vocabularies into a standard format and location. Vocabularies are often hosted online, and are used to support environmental applications. They contain labels, definitions and other related information for a particular concept or registry. This project mostly focused on soil and land operations. Currently, a number of vocabularies are unique to their individual application, and are often accessed through different formats and mediums (text, csv, pdf, or lists). Having a definitive vocabulary on a subject is critical, as the same term could have different definitions or meanings in different vocabularies.

It is also necessary to introduce semantic relationships, which essentially add another layer of detail to vocabulary. This means different terms can be placed in a hierarchical order (broader and narrower), as well as relating similar terms. These vocabularies follow the guidelines of SKOS, the Simple Knowledge Organization System, which is an industry standard of rules and guidelines across online vocabularies which aims to make publication and use of vocabularies an easy and standard process [1].

SPARQL the SPARQL Protocol and RDF Query Language is, as the acronym suggests, a Resource Description Framework (RDF) query language [2]. SPARQL is the web standard query language for RDF and allows the user to query semantic data in databases, and retrieve and manipulate RDF data.

There were 4 main tasks relating to this project, which were completed successfully:

•          Source relevant existing vocabularies

•          Convert the vocabulary content using semantic web technologies (RDF/SKOS)

• Harmonising vocabulary content with existing vocabularies where applicable

•          Publish vocabularies to SPARQL triple stores where applicable

# Materials

The following is a list of all the materials used in the creation of these vocabularies. Three books were used, these formed the basis of the data itself, and the definitions and terms used. The SITES book [3] contained all of the tables, lookups and views that these vocabularies were created from. The “Yellow Book” [4] and “Green Book” [5] were used in the gathering of definitions, notations and symbols for the terms found in the tables. Two wiki’s [6] [7] were used for guidance in web technologies like SKOS, RDF and SPARQL. Vocabularies were published to the Linked Data Registry [8]. The programs used will be discussed in the method.

* National Committee on Soil and Terrain (2009), ‘Australian soil and land survey field handbook (3rd edn).’ (CSIRO Publishing: Melbourne)
* ‘Soil Information Transfer and Evaluation System (SITES) – Database design and exchange protocols (version 2.0) (2012).’ Jacquier, D; Wilson, P; Griffin, T; Brough, D. (CSIRO Publishing: Canberra)
* ‘Soil Chemical Methods – Australasia’ (2011). Rayment, G; Lyons, D. (CSIRO Publishing: Melbourne).
* The CSIRO environmental informatics Wiki - <https://wiki.csiro.au/display/EI/Environmental+Informatics+Home>
* The CSIRO vocabularies and vocabulary services Wiki - <https://wiki.csiro.au/display/VOCAB/Home>
* The Linked Data Registry (refer to my sandbox: /sandbox/student/xavier) [http://registry.it.csiro.au](http://registry.it.csiro.au/sandbox/student/xavier)
* Microsoft office suite
* Notepad++
* TopBraid Composer
* RDF123 - <http://ebiquity.umbc.edu/project/html/id/82/RDF123>
* Microsoft SQL Server Management Studio

# Method

This report will explain the method used to create a vocabulary. This can be seen in Figure 1. There are different programs that achieve the same outcome, however, the steps presented below documents our approach and includes supporting notes on the wiki. The following description builds on prior work and documentation by a previous vocabularies student project by adding a step by step method with screenshots and annotations. This can be found on the wiki [10].

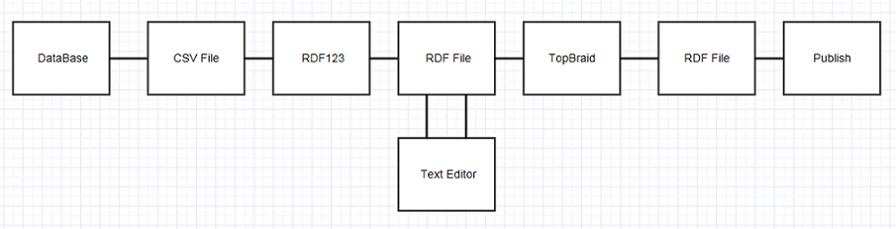


Figure 1: Process diagram

## Summarised Method

Firstly, there are a number of documents, papers, manuals and wiki’s online that are very useful for getting up to speed on vocabularies [6] [7]. This method describes how to transform vocabulary terms denoted in a database into RDF/SKOS vocabularies and deployed as Linked Data. For this project, the natsoils database was used as source information. In this database, a collection of tables are kept which can be exported as a CSV. Following this, Excel can be used to open the CSV. This is where columns will need to be created and modified so to capture all of the properties to be maintained in the vocabulary. The most common manipulations will be the creation of a URI and the creation of a Label/Definition. Subsequently, the CSV file can be exported into RDF123. This program essentially converts the tabulated data into an RDF format. This is also where many of the links between the data are created. After a quick tidy up in Notepad, the file can be exported into TopBraid. TopBraid is a program that creates semantic relationships and adds further necessary properties to make the vocabulary function better. Finally, after another tidy up in Notepad, the vocabulary is ready to be published to the registry.

## Full Method

Section 5.2 gives an in depth look at a step by step method for creating a vocabulary.

### Background Reading

There are a number of Wiki’s, both from previous interns who had worked in this area previously, as well as those managed by the Environmental Informatics team [6] [7]. The most useful notes are in the Vacation Scholarship Readings [11], especially: Tom Baker's Presentation, the SKOS Primer, Megan’s AGIFT spreadsheet to RDF, and Jane Frazier's SKOS Manual. This vocabulary work has a steep learning curve.

### Database

1. Download and Install Microsoft SQL Server Management Studio
2. When prompted to connect to server, input these fields:

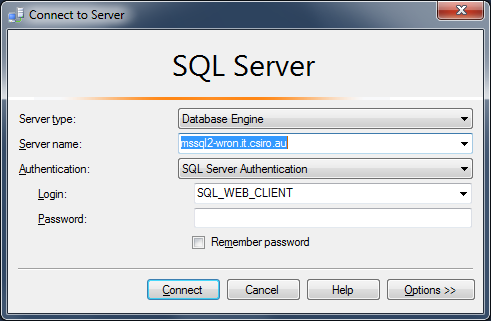


Figure 2: SQL Login

1. Find the Table folder in the Natsoil database:

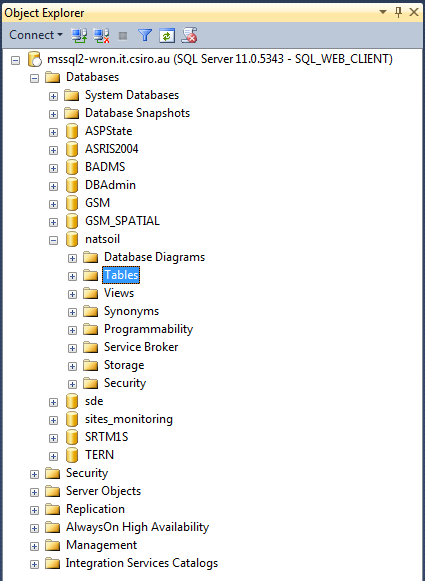


Figure 3: Menu Navigation

1. View the Top 1000 Rows of the table you're interested in (if there are more than 1000 entries, in the SQL code, just change ‘Select Top 1000’ to ‘Select’).

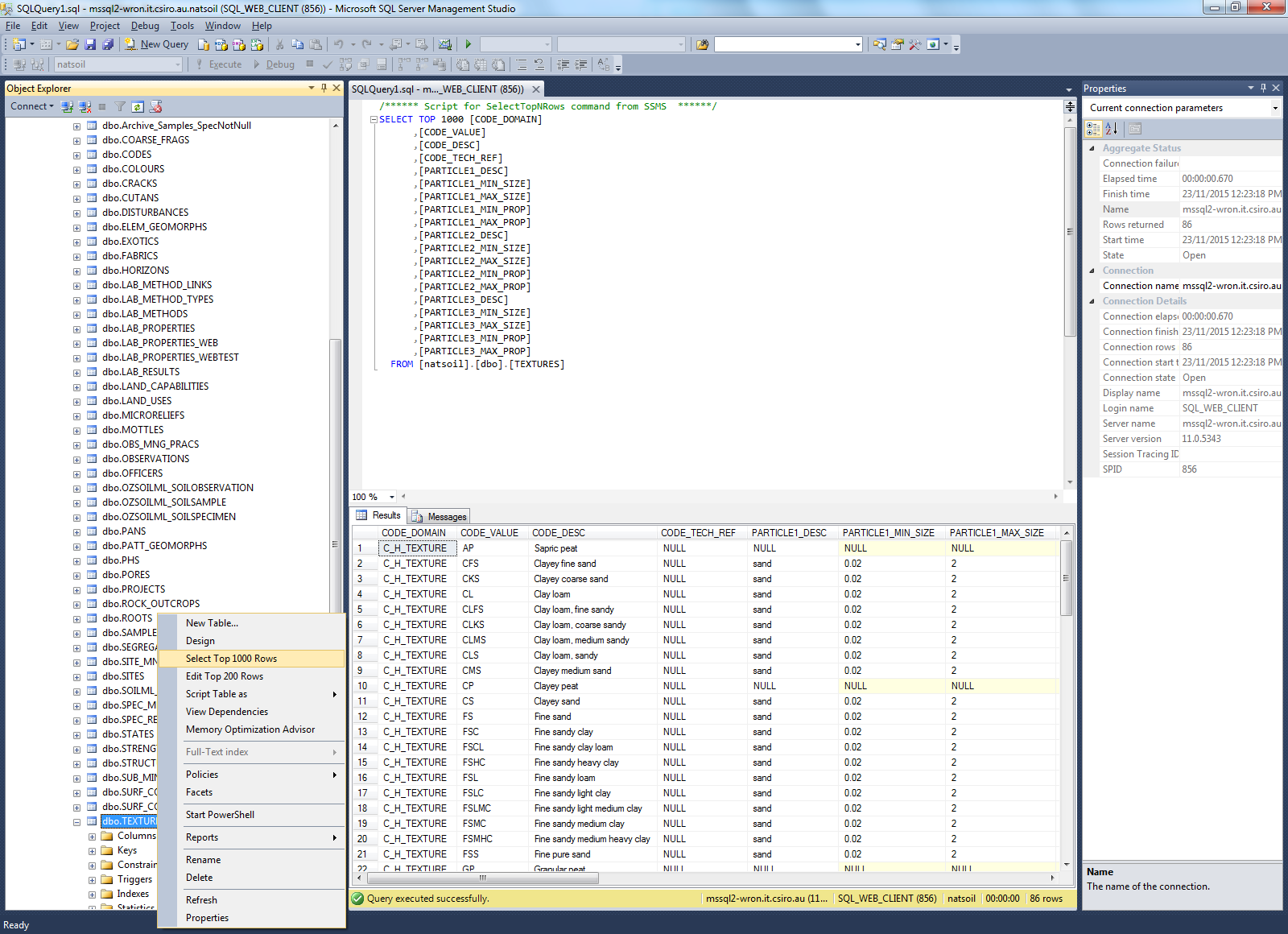


Figure 4: Top 1000 rows

1. Using the SQL code in the top half of the screen, eliminate the columns that are not needed. Then Save the Results as a .CSV:

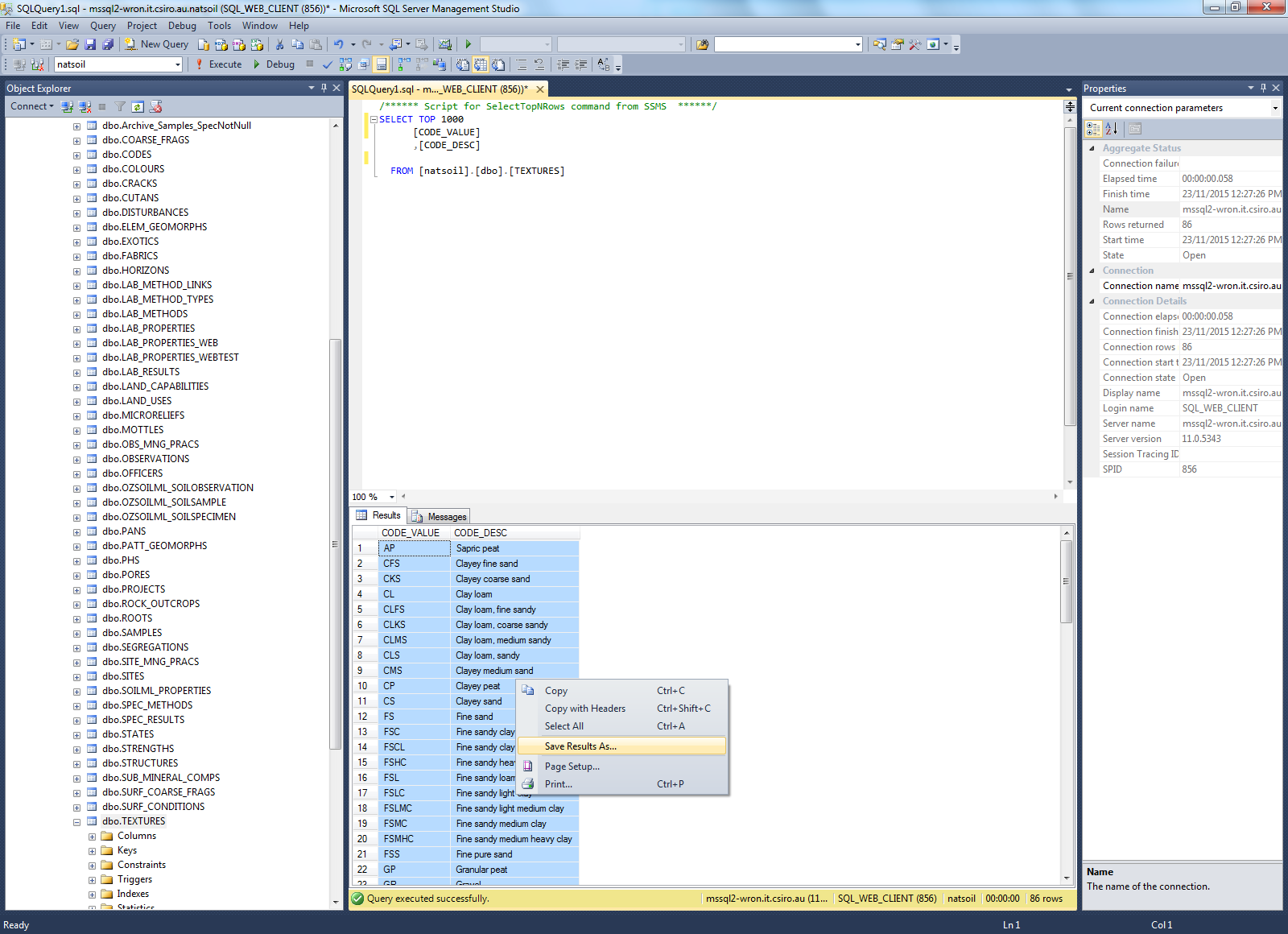


Figure 5: Save results

### Excel (CSV)

1. Establish best practice URI rules. In my case I used dashes for spaces, only used lowercase and used terms instead of codes
2. Decide on a base URI on which everything will be based off. Eg: http://registry.it.csiro.au/def/soils/texture/. Type it and Fill down.

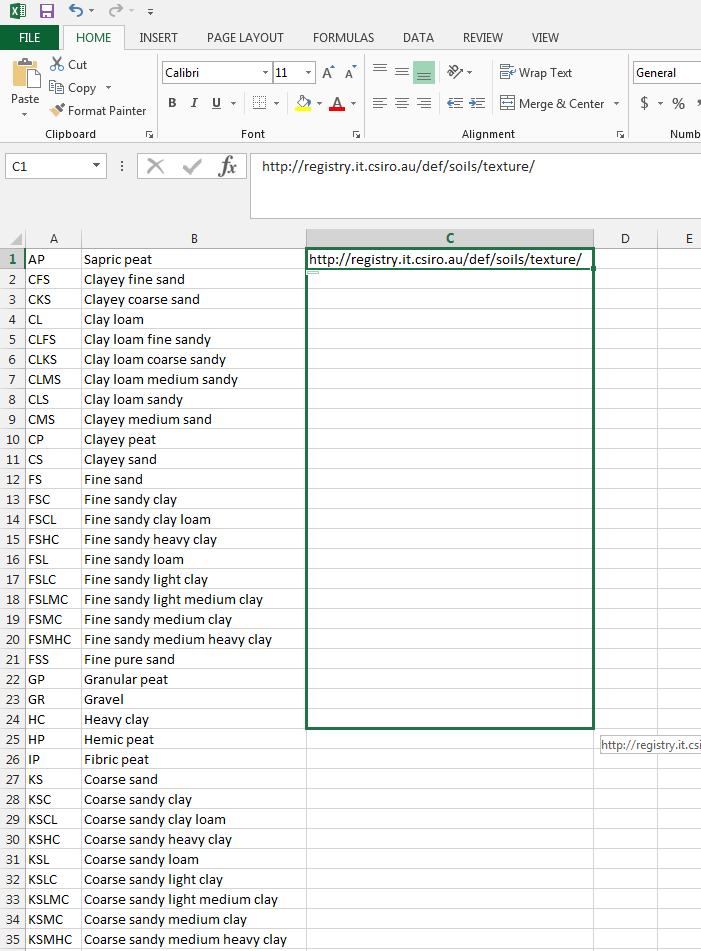


Figure 6: URI Filldown

1. Using a formula is the most efficient way to convert the terms in column B into a URI which follow the rules set out in point 1. The first function to use is the substitute formula, which replaces certain characters with something else. Eg: =SUBSTITUTE(B1, " ", "-") . Fill down.

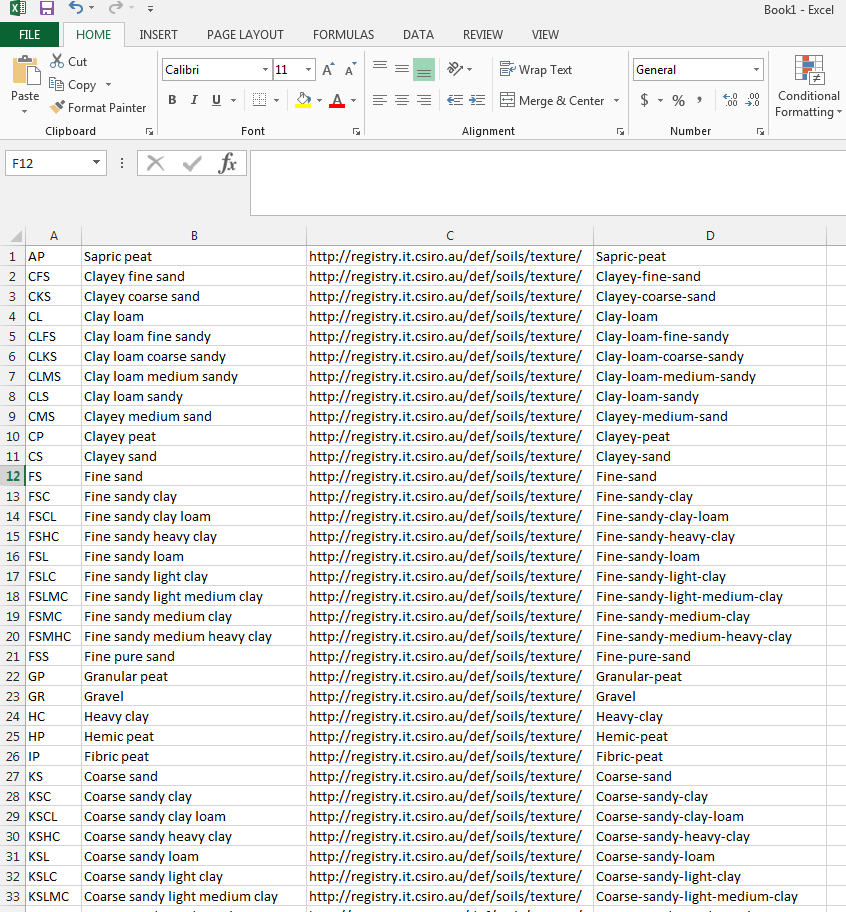


Figure 7: CSV Dashes

1. The next function will be the lower formula. This function changes everything in the cell to lower case. Eg: =LOWER(D1) . Fill down.

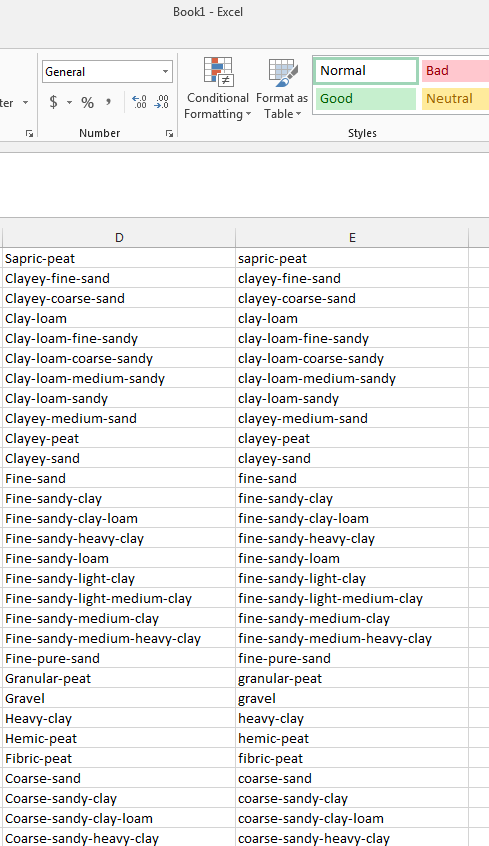


Figure 8: CSV Lower

1. To merge the base URI in column C with the unique URI prefix in column E, we need to use the concatenate function. This merges text from different cells and places it in a new one. Eg: =CONCATENATE(C1, E1) . Fill down.

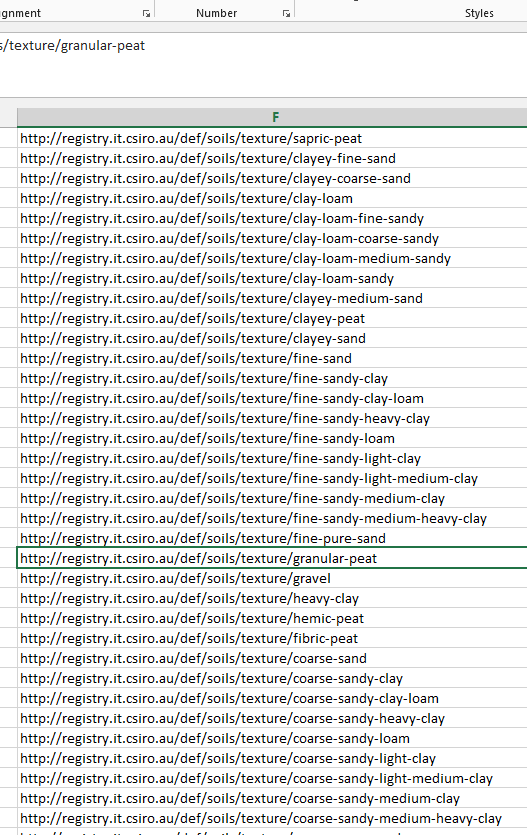


Figure 9: Concatenate

1. Next, we want to create the prefLabel. First, type 2 apostrophe's in a cell (Note: Not quotation marks). Fill down. Then type '@en' in the next cell. Fill down. We then want to concatenate these. Eg: =CONCATENATE(G1, B1, G1, H1) . Fill down.

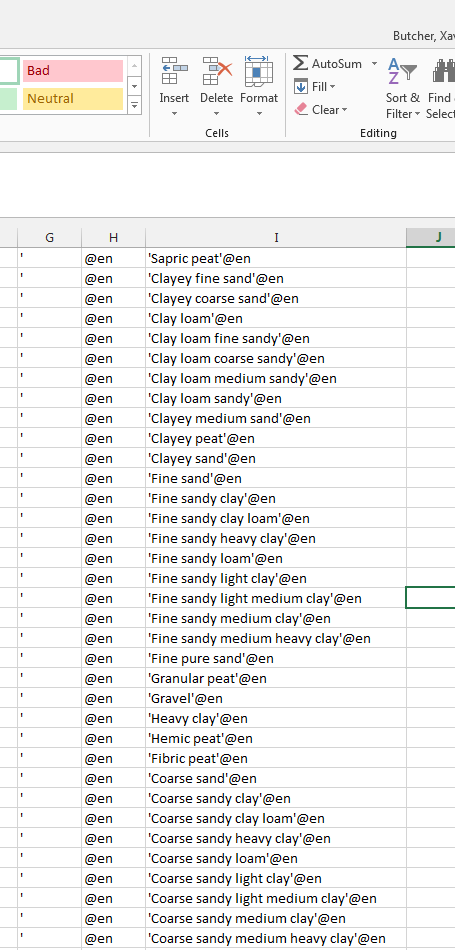


Figure 10: prefLabel

### RDF123

RDF123 is a powerful RDF editor, but the user interface takes some time to learn [12]

1. In the prefix definition window, you will need to add or amend these definitions (make sure there are no extra spaces after the URLs):

|  |  |
| --- | --- |
| **Title** | **URL** |
| Base | http://registry.it.csiro.au/def/soils/texture/ |
| mapBase | http://registry.it.csiro.au/def/soils/texture/ |
| Skos | http://www.w3.org/2004/02/skos/core# |
| Dct | http://purl.org/dc/terms/ |

1. The next step is to import the CSV file you created in excel. In the spreadsheet window, open your spreadsheet. Your screen should look like this so far:

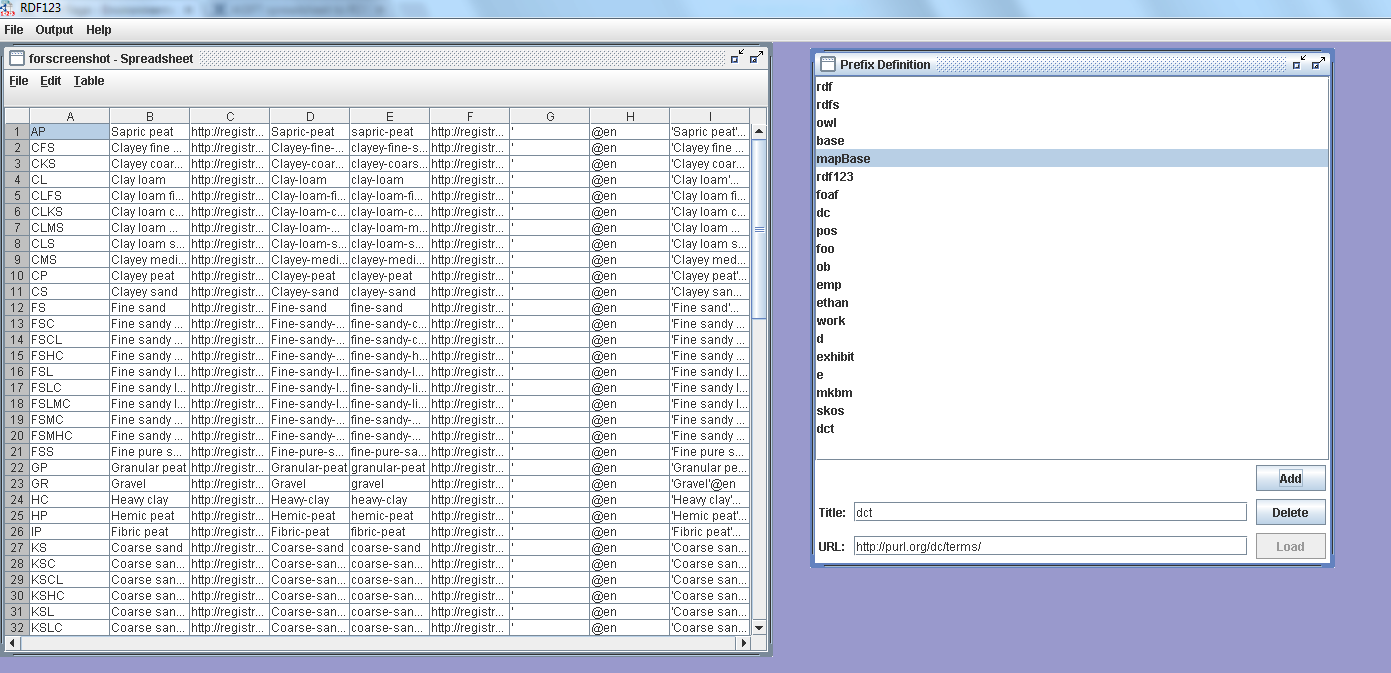


Figure 11: CSV Import

1. The next step is to use the 3rd window (map graph) to create a sort of mind map of what the data is trying to say. The first step is to create vertices and edges. If there is data already in the map area, just delete them, start with a blank canvas. In this case we will need 5 vertices and 4 edges. Don't worry about labels for now. The edges should radiate out of 1 central point to the other 4 vertices. It should look something like this:

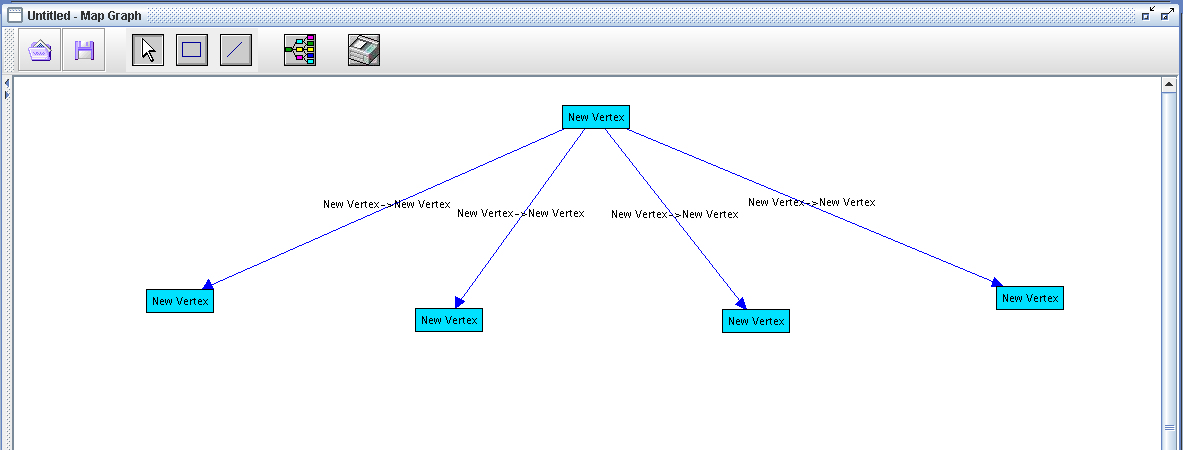


Figure 12: Blank Map

1. Now we need to start labelling the vertices. Firstly, the central vertex must be the unique URI. To tell the program that each individual URI is located in column F we need to use the notation "Ex:$6". The other columns you need are A, B and I (which are Ex:$1, 2 and 9 respectively). The remaining box must be called a "skos:Concept" to let the program know that each row is a separate concept.

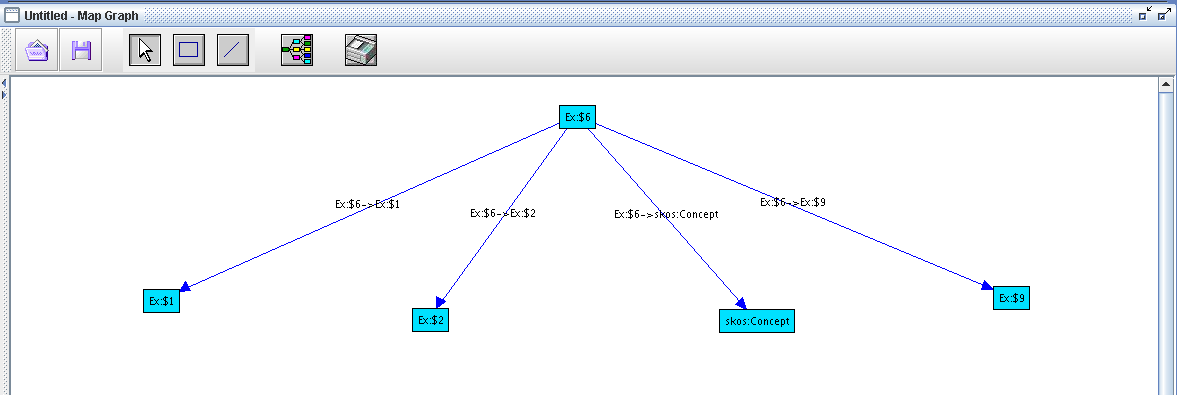


Figure 13: Map Labels

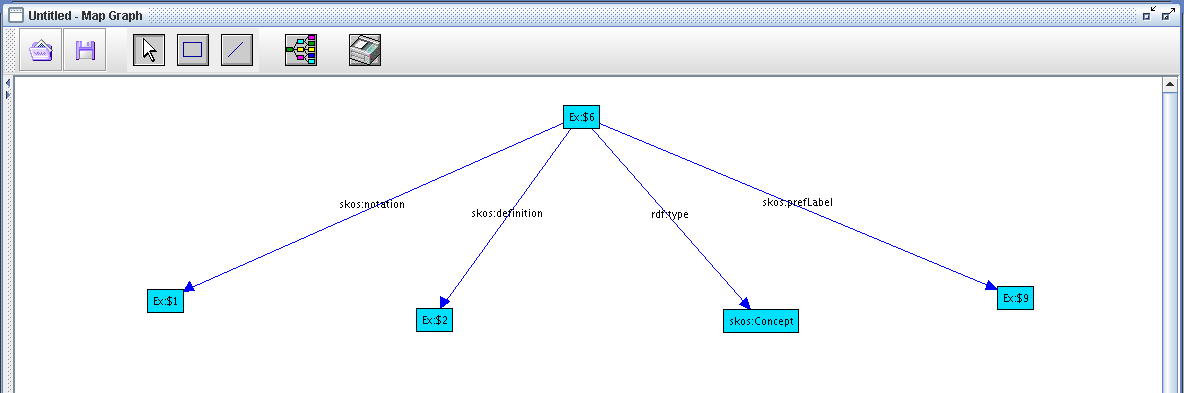
1. Next we must label the edges. Column 1 is a 'skos:notation', column 2 is a 'skos:definition', skos:Concept is an 'rdf:type' and column 9 is a 'skos:prefLabel'. Label the edges as such.
2. 

Figure 14: Map edges

1. You can now go to the overall window, and click the output menu at the top. Then click 'view spreadsheet in RDF'. Make the display type 'N3' as it’s easier to read. Copy this text output and paste it into notepad++.

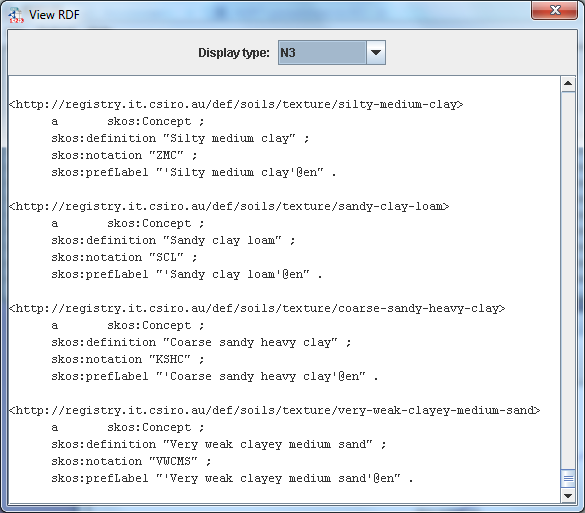


Figure 15: Output

### Text Editor (notepad++)

Check the output from RDF123 as it is usually not error free. Some modifications to the text file will need to be made. As it is coding language, even the smallest discrepancy in the text will cause an error. Patience is key!

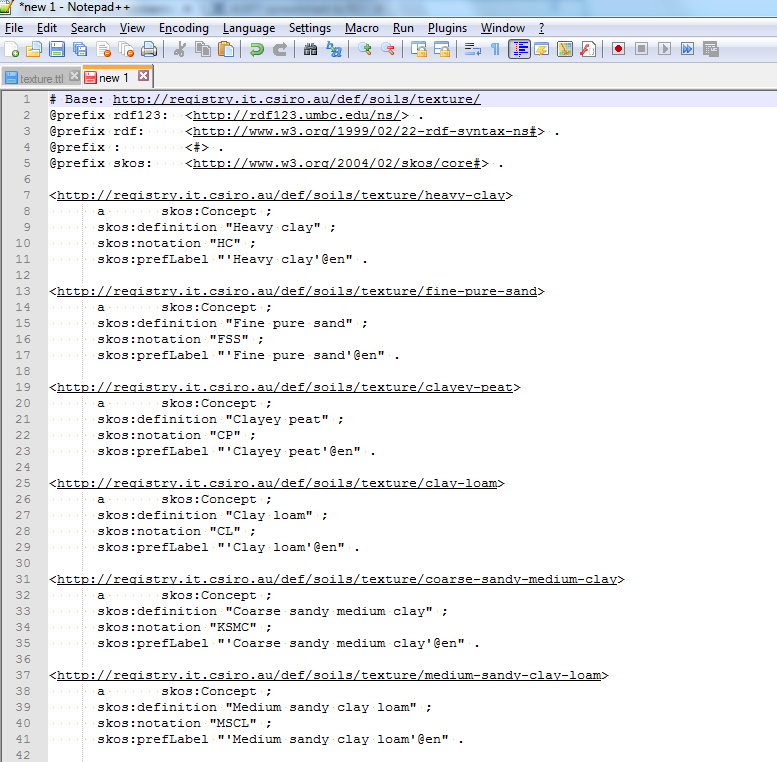
1. Initially it should look like this:  
   

Figure 16: N++ input

1. You're going to want to save it as a .TTL file so that it can be read by TopBraid.
2. The first thing you'll need to correct is in the first few lines:

|  |  |
| --- | --- |
| **Bad output from RDF123** | **Change to:** |
| # Base: <http://registry.it.csiro.au/def/soils/texture/> | #baseURI: <http://registry.it.csiro.au/def/soils/texture/> |
| @prefix : <#> . | DELETE |

1. The next thing that needs to be changed is that the prefLabel for each concept has not come across properly, due to RDF not recognising the '@en' as a separate suffix that isn't part of the concept name. To do this, use the Replace tool (in the Search menu). In this case for example, we will need to change <"'Heavy clay'@en"> to <"Heavy clay"@en>. This needs to be done throughout the whole document. The easiest way to do this is to replace all:

|  |  |
| --- | --- |
| **Find** | **Replace** |
| "'(quotation then apostrophe) | "(quotation) |
| '@en" | "@en |

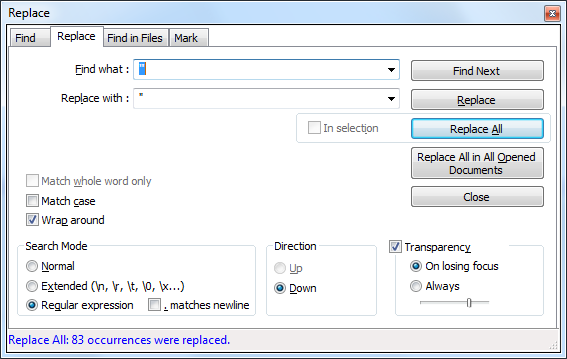


Figure 17: Replace text

1. This should be all that needs to be edited in notepad++, however there may be more depending on how it exported. It should look like this:

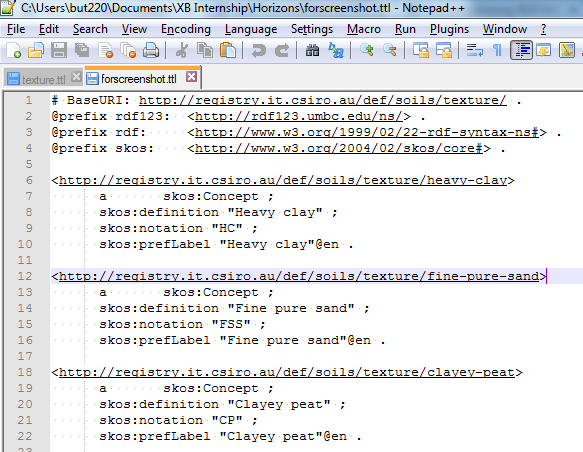


Figure 18: N++ Output

As a footnote, this table did not have a date created column. But if yours does, these are the steps needing to be completed, as dates come across poorly out of RDF:

**Editing**[**dct:created**](http://dctcreated/)**and**[**dct:modified**](http://dctmodified/)

Dates were not in the format required in RDF. There were a number of issues,   
1. Dates contained 0:00 at the end of the string  
2. RDF 123 picked up the text as a string  
3. Dates were in the form dd/mm/yyyy, but need to be yyyy-mm-dd

**Regex to fix these issue**

|  |  |
| --- | --- |
| **1.** | **Remove the 0:00 and recognize number as a date** |
| FIND | "([/0-9]+) 0:00" |
| REPLACE | "$1"^^[xsd:date](http://xsddate/) |

|  |  |
| --- | --- |
| **2.** | **Re-format so that date reads yyyy-mm-dd** |
| FIND | "([0-9]+)/([0-9]+)/([0-9]+)"\^\^[xsd:date](http://xsddate/) |
| REPLACE | "$3-$2-$1"^^[xsd:date](http://xsddate/) |

|  |  |
| --- | --- |
| **3.** | **Ensure that single digits have a 0 in front ( eg 9 --> 09)** |
| FIND | -([1-9])"\^\^xsd |
| REPLACE | -0$1"^^xsd |

### TopBraid Composer

1. To import your .TTL file into TopBraid, click on File, then Open your .TTL file and it should open.
2. You must also import the SKOS core .TTL file and the Dublin core .TTL file (dc-1.1.TTL). Go to the imports tab at the bottom (little blank page with a plus), and import the local file. Here it is a good idea to sort both the properties tab and the classes tab by Namespace. This can be done by clicking the small button in the bottom left of each tab. Classes looks like three brown dots, Properties looks like 3 blue rectangles.

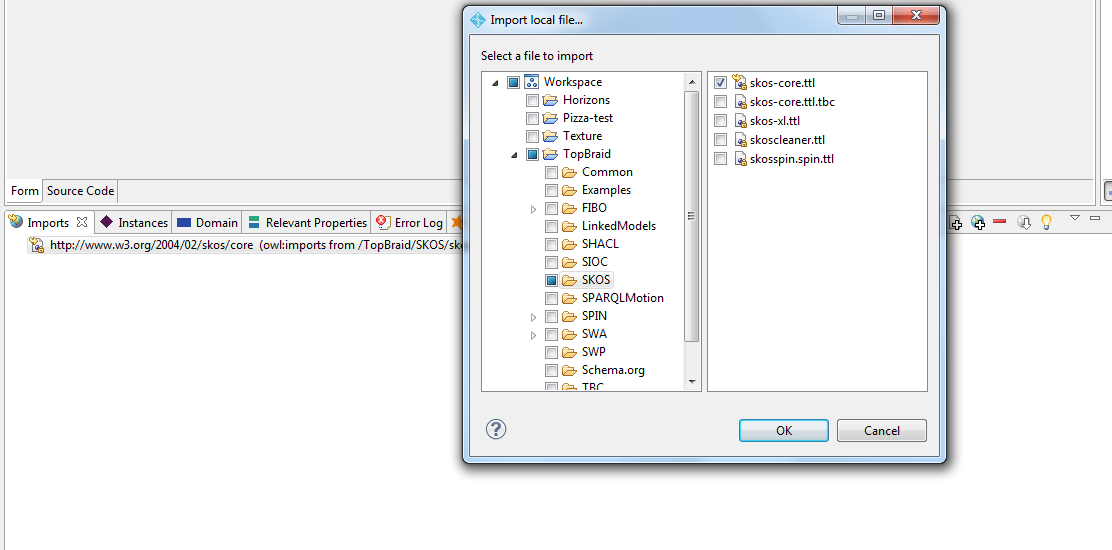


Figure 19: TopBraid Import

1. If the editing in notepad++ was done successfully, then each concept should appear in the Instances tab at the bottom, and in the skos:Concept tab in the Namespace at the top left.

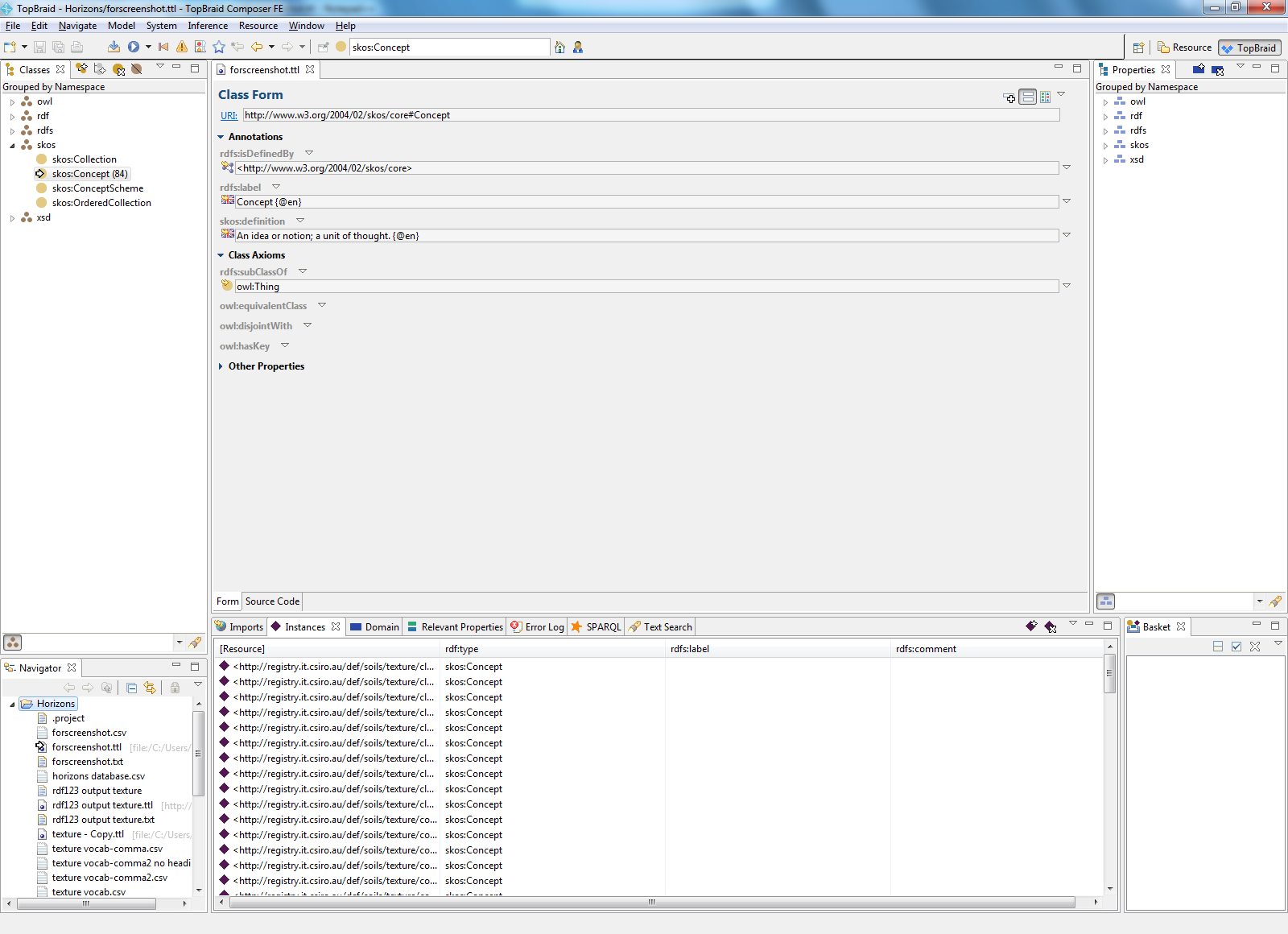


Figure 20: TopBraid Input

1. The next step is to add an rdfs:label to each concept. This is done by using SPARQL code found below:

INSERT{ ?s rdfs:label ?label }

WHERE {

{ ?s a skos:Concept . } UNION { ?s a skos:ConceptScheme . } UNION { ?s a skos:Collection . NOT EXISTS { ?s rdfs:label ?l }

OPTIONAL { ?s skos:prefLabel ?pl . }

OPTIONAL { ?s dc:title ?t . }

OPTIONAL { ?s dct:title ?tt . }

BIND( REPLACE(str(?s), '^.\*(#|/)', "") AS ?localname)

BIND( REPLACE(str(?s), '[^/^#]+$', "") AS ?namespace)

BIND( REPLACE(str(?namespace), '(#|/)$', "") AS ?ns)

BIND( REPLACE(str(?ns), '^.\*(#|/)', "") AS ?nsfrag)

BIND ( STR ( COALESCE ( ?tt, ?t, ?pl, IF( STRLEN(?localname), ?localname, ?nsfrag) ) ) AS ?label )

}

1. The dct:description must also be added, using this SPARQL code.

INSERT { ?s dct:description ?desc }

WHERE {

{ ?s a skos:Concept . } UNION { ?s a skos:ConceptScheme . } UNION { ?s a skos:Collection . }

NOT EXISTS { ?s dct:description ?d }

OPTIONAL { ?s skos:definition ?def }

OPTIONAL { ?s skos:scopeNote ?note }

OPTIONAL { ?s rdfs:comment ?com }

OPTIONAL { ?s skos:prefLabel ?plab }

?s rdfs:label ?lab .

BIND ( COALESCE ( ?def , ?note , ?com , ?plab , ?lab ) AS ?desc )

FILTER ( (datatype(?desc) = xsd:string) || ( lang(?desc) = "en" ) || (lang(?desc) = "EN") )

}

1. Once both of those search queries have been run, your instance panel should look like this:

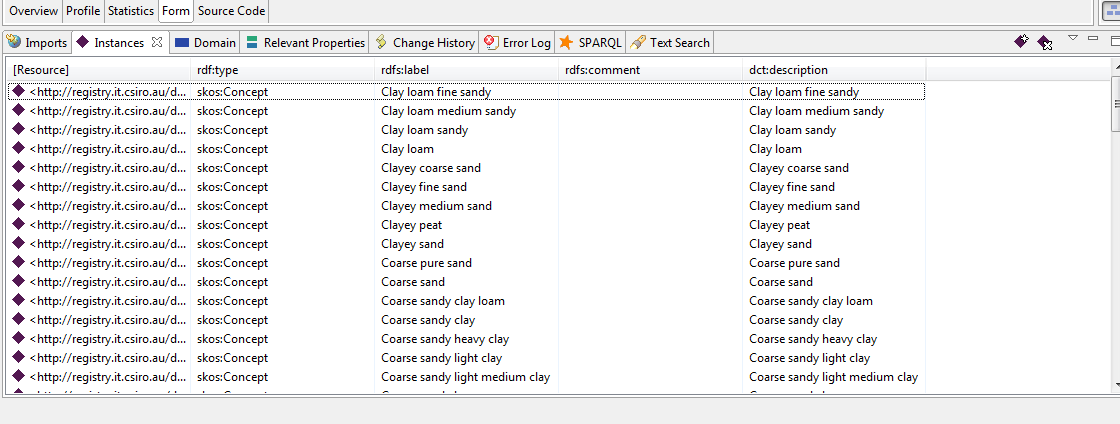


Figure 21: Description Added

1. The next step is to create a prefix for the concepts. Click the home button, and then click the overview tab. Add the prefix you would like to use, and then the base URI. I used texture in this case.

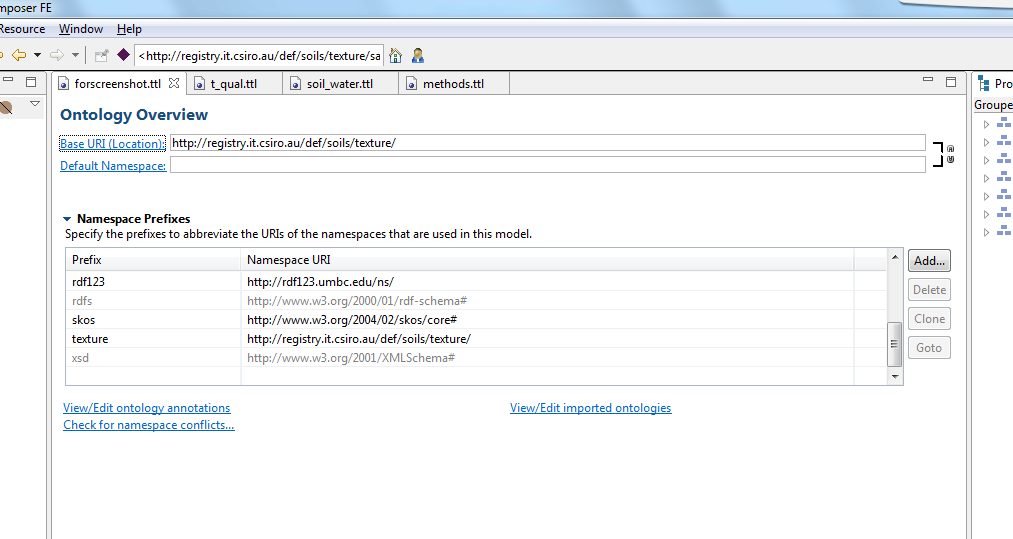


Figure 22: Prefixing in TopBraid

1. Now is the time to do final quality control and checking of your concept data and make sure that all the properties you need have been filled. We must now do some final editing of the register in TopBraid before we can start uploading the vocabulary.
2. There are 2 options of achieving this next step. Using the Form tab or the Source Code. Firstly, click on the home screen, but this time navigate to the Form tab. Here we will need to add some necessary properties. The ones you'll need at a minimum are: dc:source, rdfs:comment, rdfs:label, dct:created, dct:creator, dct:description. Note: change the data type from string to date for date created (little white arrow at end of box). A purple diamond means the input should be a URI, whilst a blue square means the input should be a text string. If you do not have a URI for a particular field, you can get around this by firstly inserting a blank URI (<>) and then going to the source code and changing the <> to a " ". This will change the form to accept a text string instead. Secondly, copy and paste this code and change the values where appropriate:

method:

  rdf:type owl:Ontology ;

  dc:source "natsoil CSIRO database - Lab Methods table"^^xsd:string ;

  dct:created "2015-12-17"^^xsd:date ;

  dct:creator <mailto:xavier.butcher@csiro.au> ;

  rdfs:comment "The different methods for examining the type and abundance of certain compounds."^^xsd:string ;

  rdfs:label "Lab Methods"^^xsd:string ;

  rdfs:seeAlso "National Committee on Soil and Terrain (2009), 'Australian soil and land survey field handbook (3rd edn).' (CSIRO Publishing: Melbourne)"^^xsd:string ;

  owl:imports dc: ;

  owl:imports <http://www.w3.org/2004/02/skos/core> ;

.

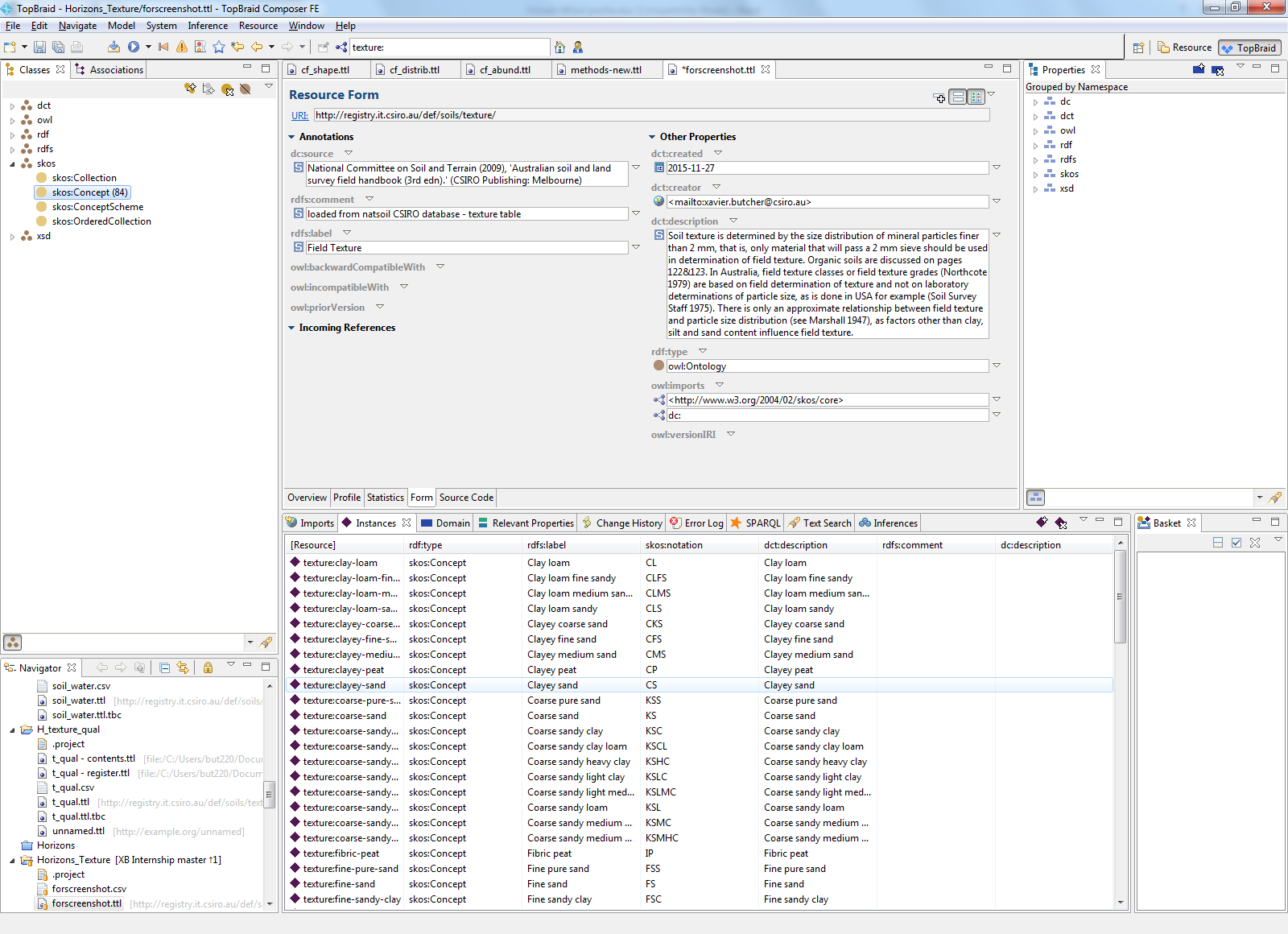


Figure 23: Registry Properties

1. This should be the end of the editing you need to do in TopBraid, however now we must use notepad++ again to do some final post processing.

### Post Processing (Text)

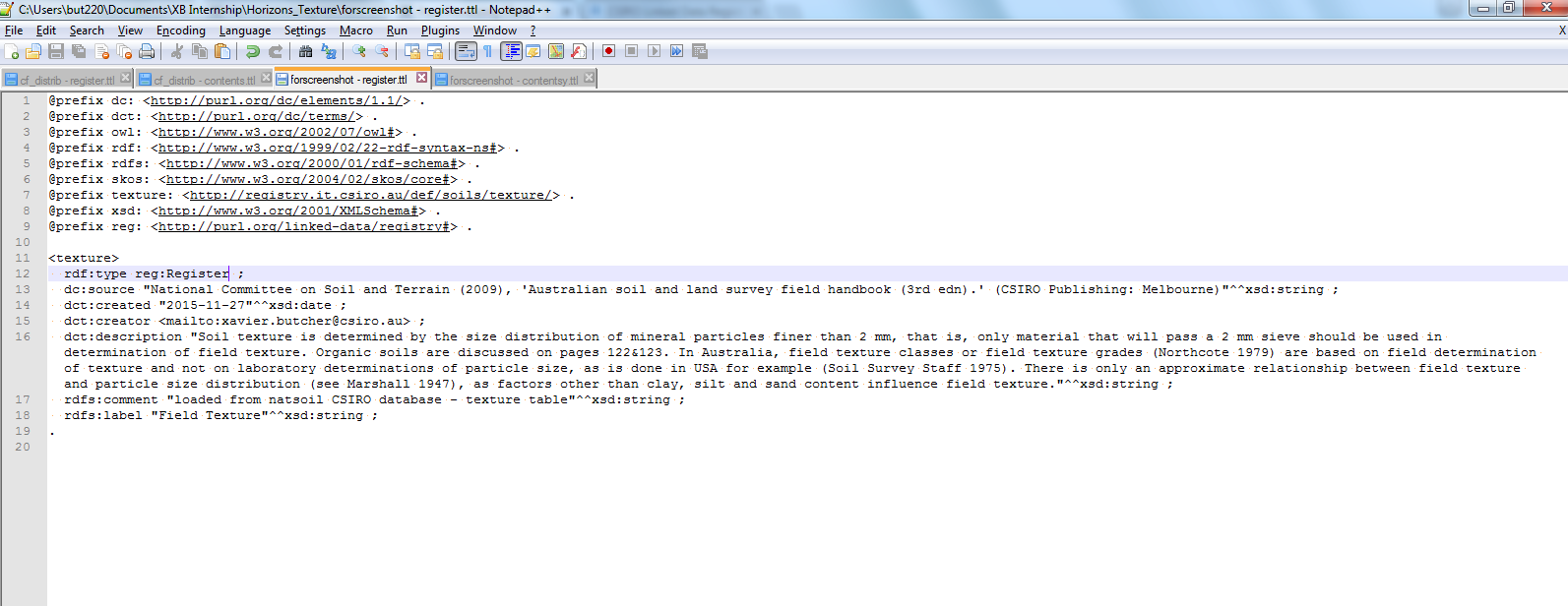
1. The first thing to do is create 2 copies of your .TTL file. One called registry, the other called contents.
2. We will start working with the register file. Firstly, delete the top 4 lines (baseURI, imports 1 and 2 and a blank line).
3. Next, we will delete all of the 'contents', leaving only the register information.
4. Following this, delete line 5 (the rdf123) prefix. Then add another prefix: @prefix reg: <[http://purl.org/linked-data/registry#](http://purl.org/linked-data/registry)> .This just tells the program that this particular file is part of the registry.
5. We will then change "owl:ontology" to "reg:register".
6. We will then change line 11 from "texture:" to "<texture>”. This is just the end of your base URI.
7. Then we will delete lines 19 and 20, the 2 owl imports.
8. That is it for the registry file, we will now move onto the contents file. The final registry file should look like this:  
   

Figure 24: Registry

1. Open the contents file.
2. Much like the registry file, we will delete the top 4 lines. We will also delete the prefix rdf123 line. But this time we will delete the registry information, and keep the contents information.
3. We then need to do a bulk search and replace. We need to change "texture:clay-loam" to "<clay-loam>". The regular expression code you need for this is:

|  |  |
| --- | --- |
| **Find** | **Replace** |
| texture:([a-zA-Z0-9\_\-]+) | <$1> |

1. The text editing should now be done. The contents document should look like this:

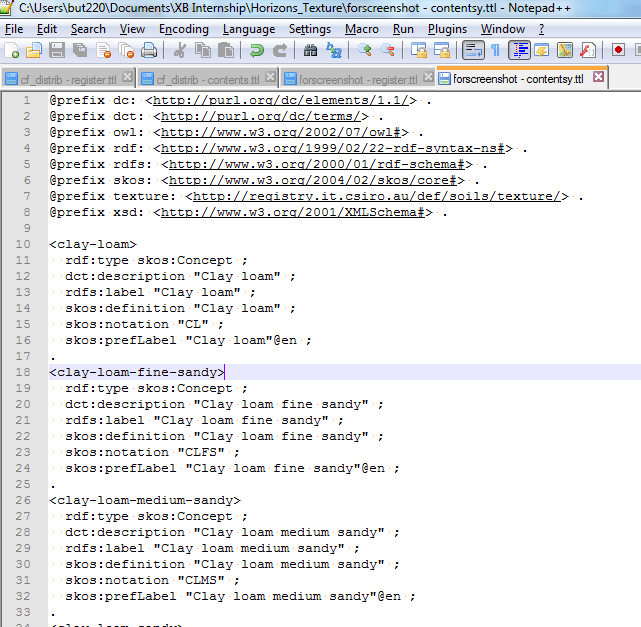


Figure 25: Contents File

### Publishing

1. Confirm your registry and contents files are correct, as once uploaded to the Linked Data Registry (LDR), it is hard/impossible to remove/add/modify.
2. Navigate to your registry location. You will need to create an account/login if you haven't already. Mine is: http://registry.it.csiro.au/sandbox/student/xavier
3. The page should look like:

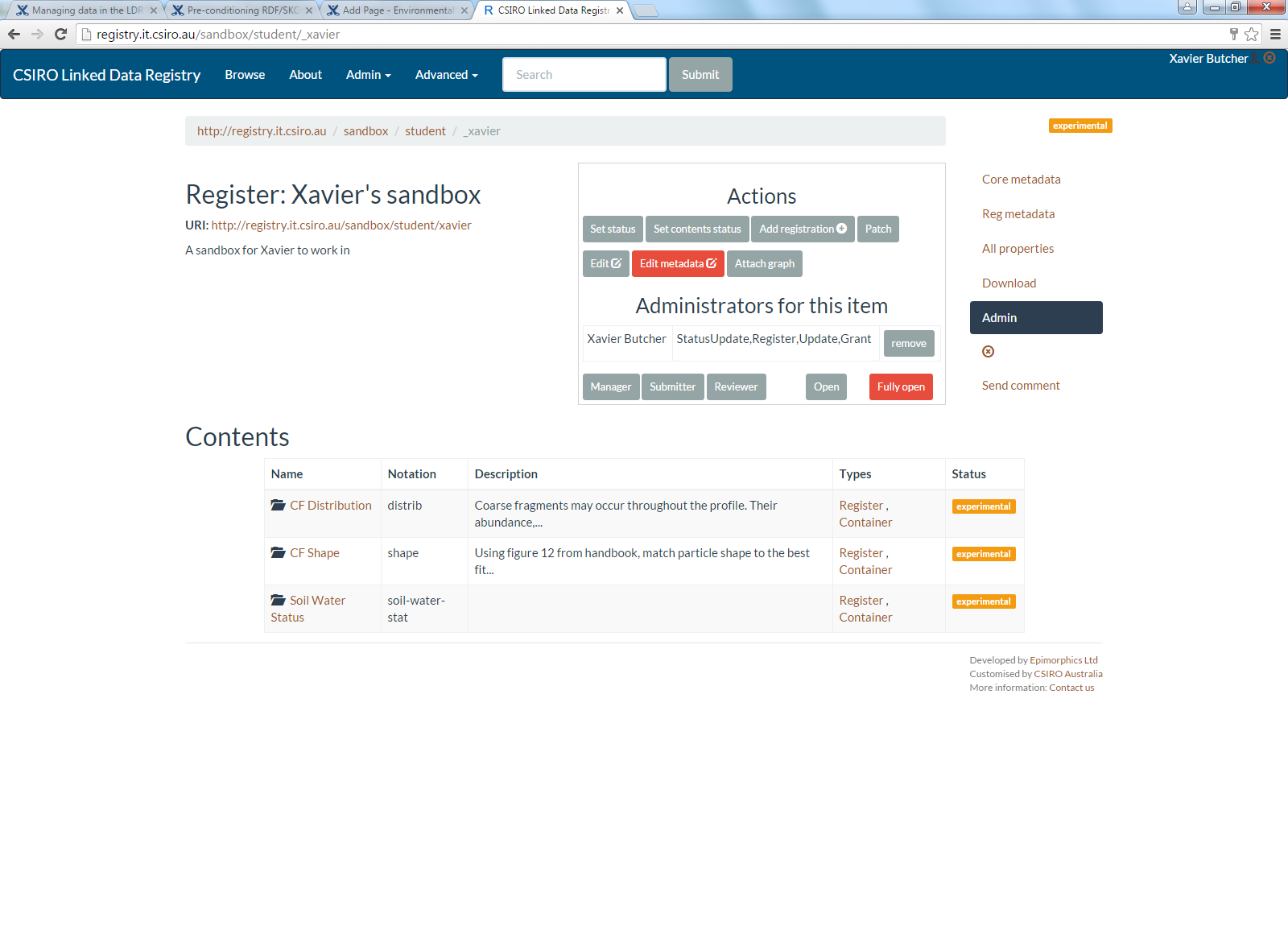


Figure 26: Navigation

1. Then click on admin, and select add registration.
2. Next, click upload, and choose your registry file.
3. It should look like this:

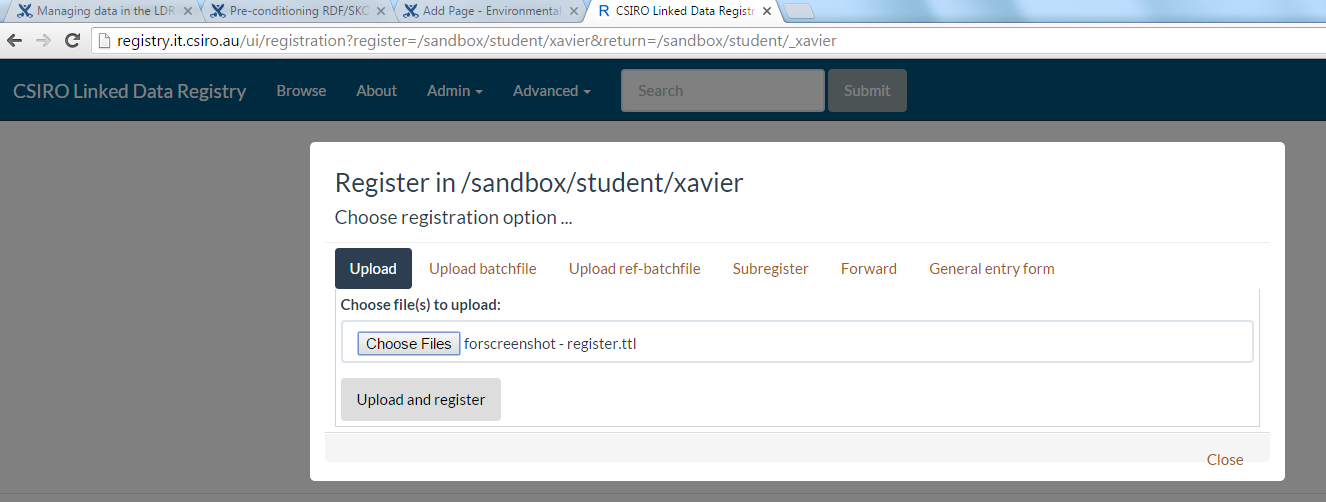


Figure 27: Register upload

1. Once uploaded, you should see the new register appear. Click on it, and you should see the register you uploaded, with labels and descriptions. However, there shouldn't be any contents - yet.
2. When inside the register, using the same method as before, upload the contents file. It should look like this:

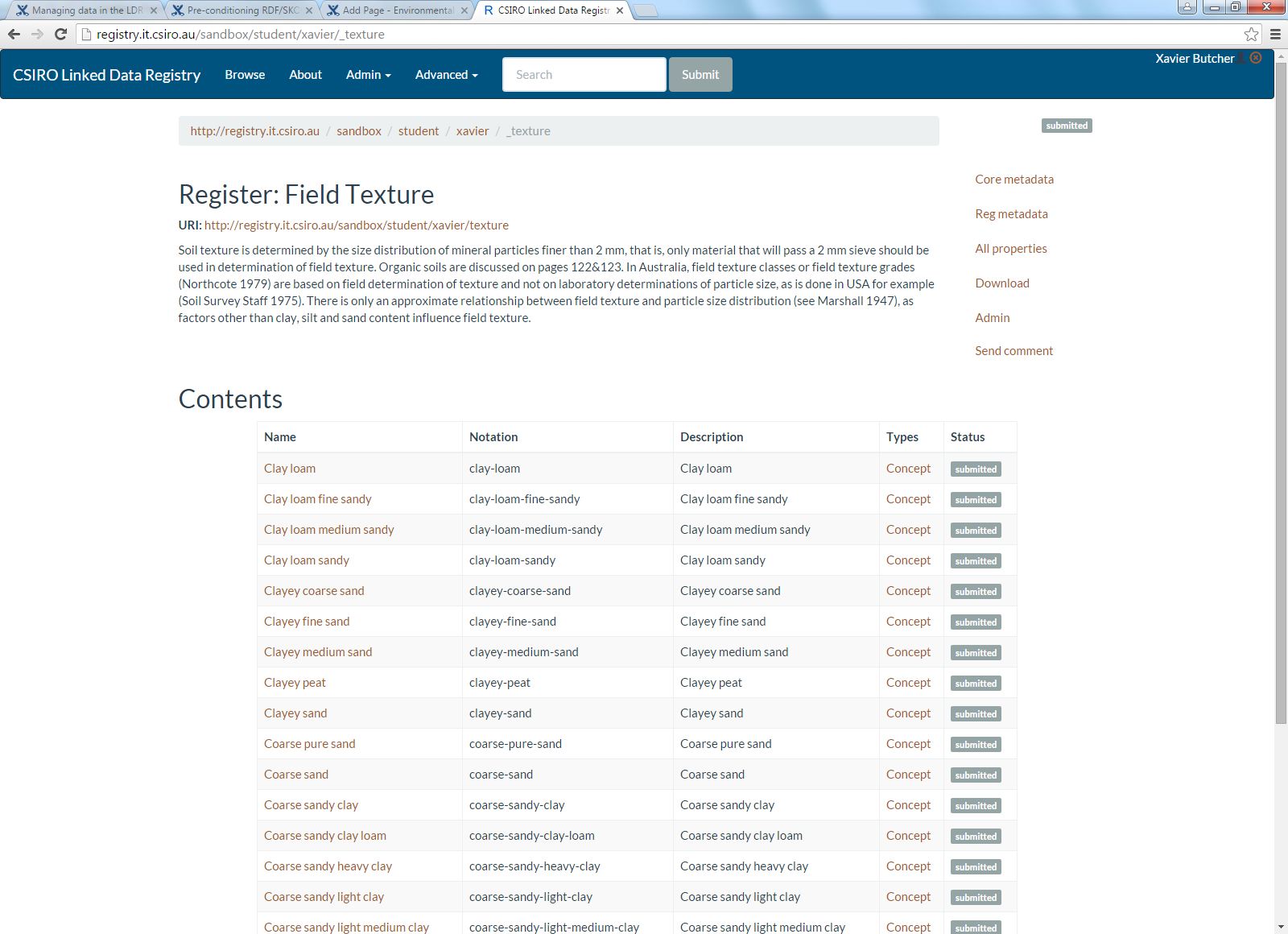


Figure 28: Contents Upload

1. You will notice the register and all the contents say submitted. This means only you can see them. To 'turn on' your vocabulary, you must make the status of both the register and the contents "Experimental".
2. This is done by clicking on admin, then 'set status'. A box should come up; click on the yellow experimental button. Do this process again for the 'set contents status' button. It should look like this:

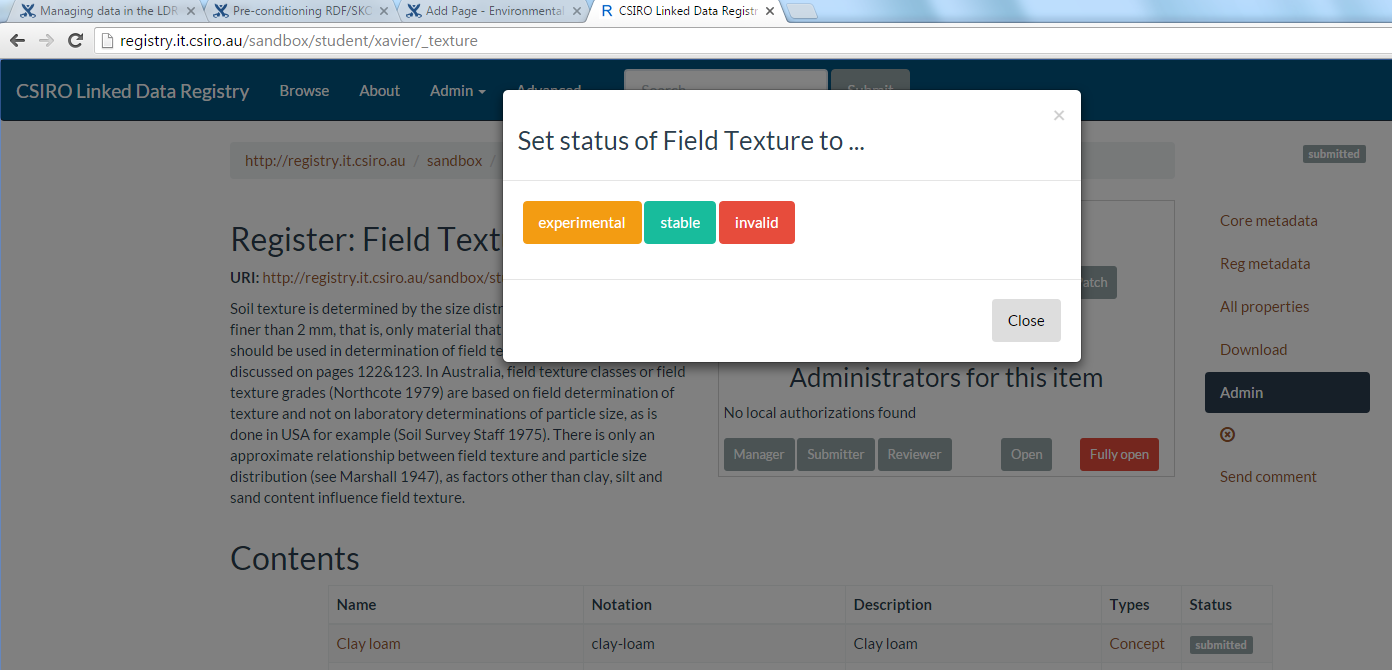


Figure 29: Set Status

1. Your vocabulary is now complete!

# References

[1] <https://www.w3.org/2004/02/skos/>

[2] <https://www.w3.org/RDF/>

[3] ‘Soil Information Transfer and Evaluation System (SITES) – Database design and exchange protocols (version 2.0) (2012).’ Jacquier, D; Wilson, P; Griffin, T; Brough, D. (CSIRO Publishing: Canberra)

[4] National Committee on Soil and Terrain (2009), ‘Australian soil and land survey field handbook (3rd edn).’ (CSIRO Publishing: Melbourne)

[5] ‘Soil Chemical Methods – Australasia’ (2011). Rayment, G; Lyons, D. (CSIRO Publishing: Melbourne).

[6] <https://wiki.csiro.au/display/EI/Environmental+Informatics+Home>

[7] <https://wiki.csiro.au/display/VOCAB/Home>

[8] <http://registry.it.csiro.au/sandbox/student/xavier>

[9] <http://ebiquity.umbc.edu/project/html/id/82/RDF123>

[10] <https://confluence.csiro.au/pages/viewpage.action?pageId=154009631>

[11] <https://confluence.csiro.au/display/EI/Vacation+scholarship+program>

[12] ‘RDF123: a mechanism to transform spreadsheets to RDF (2007).’ Han, L; Finin, T; Parr, C; Sachs, J; Joshi, A. (University of Maryland) <http://ebiquity.umbc.edu/_file_directory_/papers/370.pdf>

[13] <http://www.qudt.org/>

[14] <http://registry.it.csiro.au/def/environment/unit>

[15] <http://environment.data.gov.au/def/op>