**Exercise:**

**Enrich TEI with Dublin Core**

**Summary**

TEI p5 includes a new tag for other metadata [<xenoData>](http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-xenoData.html). In this exercise, we will use OpenRefine to extract metadata from TEI header tags, convert to Dublin Core, and embed in the <xenoData> tag of the document.

<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-xenoData.html>

**Exercise**

OpenRefine works with many file formats including XML. We can directly import our sample TEI files and work with the metadata within. In the following steps we will create a project with our sample records and limit to specific nodes as part of the import process.

1. Start OpenRefine and create a new project
2. Select the 10 sample Bierce TEI files from github link
3. On the *Select Files to Import* screen make sure that all ten files are checked and click *Configure Parsing Options*
4. In *Click on the first XML Element* dialog, select the `fileDesc` node and leave the rest of the options unchanged.
5. Click *Create Project*

We created an OpenRefine project with records for each **fileDesc** element from the TEI header for our letters. Our next steps will be to filter out blank rows and cleanup the data.

1. At the top of each column containing text, select *Edit Column > Rename this column* and enter a brief descriptive name for the column
2. Click the dropdown in any one of the columns that contains text and select Facet > Text Facet
3. In the facet control that appears to the left, click *include* next to the blank facet
4. At the top of the facet control, click *invert* (this hides all of the blank rows)
5. At the top of each empty column, select *Edit Column > Remove this column*

Some of the dates have an errant angle bracket. Let’s remove these with a GREL expression.

<https://github.com/OpenRefine/OpenRefine/wiki/GREL-String-Functions>

1. On the `fileDesc – publicationDate` column, select *Edit Cells > Transform*
2. In the *Expression* box, input value.chomp(‘>’) In the preview box you should see column stripped of all angle brackets, years remaining in text
3. Click OK to execute the change.

Now that we have the fileDesc data cleaned up, we can do some schema alignment using the RDF plugin to prep for export as Dublin Core/RDF

1. Beside *Extensions,* click the *RDF* dropdown and select *Edit RDF Skeleton*
2. Edit Base URI to something like `http://localhost/`
3. Click *manage prefixes,* next to *Available Prefixes*
4. In *New Prefix* dialog, add `dc` to prefix field and the URI field should auto populate. Click `Ok` and `dc` should now appear under available prefixes
5. The list of defined prefixes should match Figure 1

Figure

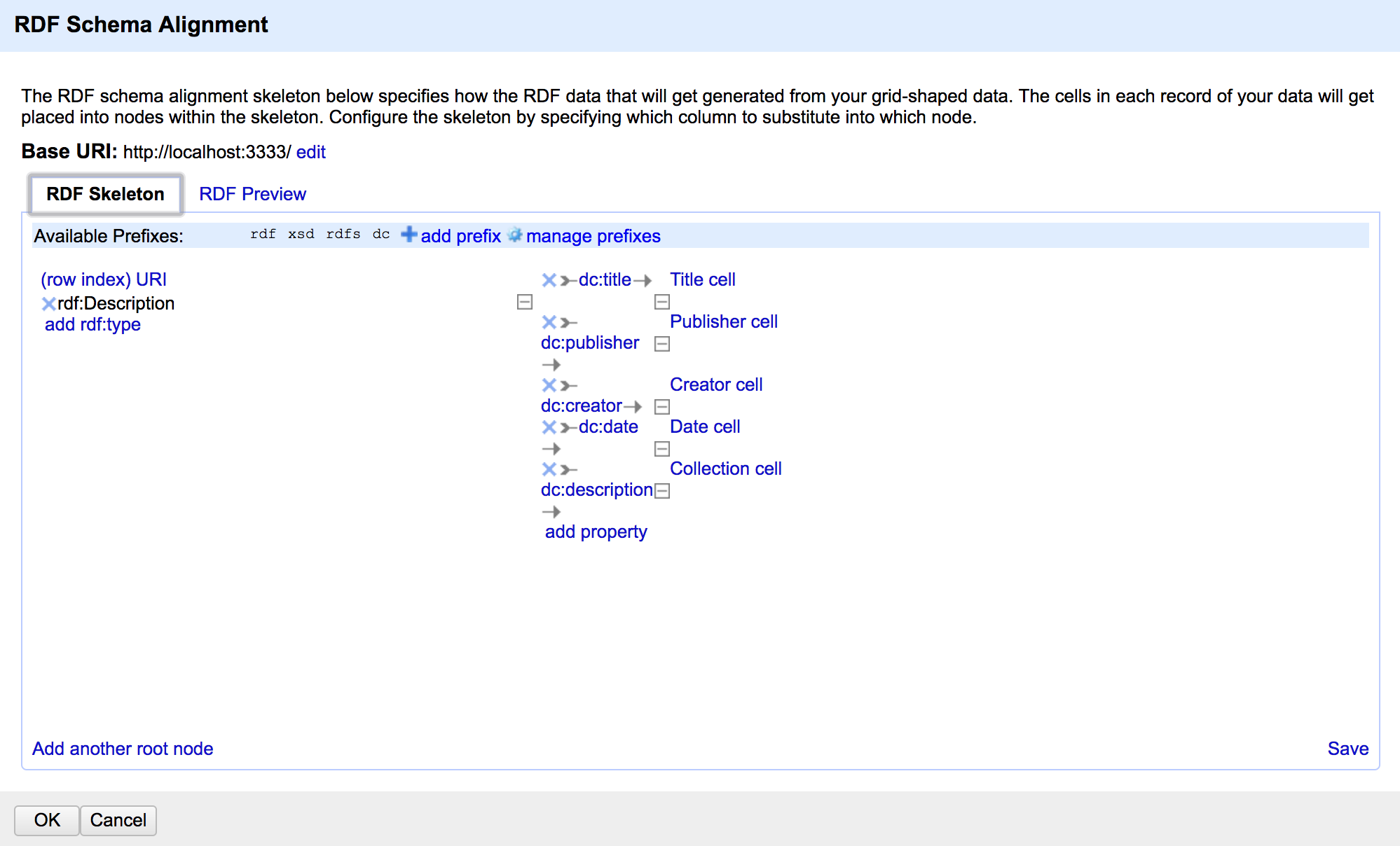
1. Follow the steps from above for all other prefixes in Figure 1. Make sure to delete any prefixes that are not present in the Figure 1

Now we can begin to fill out the RDF skeleton: we align our renamed fields TEI Header - fileDesc with appropriate Dublin Core terms. We’ll be using RDF to make machine readable statements about each entry in the set. With RDF we use URIs to represent concepts. We use Dublin Core terms to point to text from our records: **dc:date** identifies `2015` as the date of release for the item.

1. On the left of the RDF skeleton, under `URI`, click *add rdf:type* and type “rdf:Description” into *search for property* and select or hit enter to apply (if you don’t see it as an option in the dropdown, press shift+enter to add it.)
2. On the right side of the RDF skeleton, click *property?* and type “dc:publisher” into *search for property* and select or hit enter to apply (if you don’t see it as an option in the dropdown, press shift+enter to add it.)
3. Repeat the above action for each of the terms in the following table:

|  |  |
| --- | --- |
| dc:title | Title |
| dc:creator | Contributor |
| dc:description | Collection |

1. Once finished, click OK (figure 2) \*Toggle the RDF Preview tab if OK button not showing.
2. The complete skeleton should resemble Figure 3.



Figure

Now that we have defined the RDF schema for our data, we can use the RDF plugin to export for use in other applications. The RDF plugin allows to export either as RDF/XML or Turtle (ttl) format. We will export in turtle and use the file in another exercise for data reconciliation (also with the RDF OpenRefine plugin.

1. Click the *Export* dropdown and select *RDF as RDF/XML*
2. This should prompt and file save dialog with a .rdf file
3. Insert the text of each RDF item into the TEI, within a <xenodata> tag set.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  xmlns:dc="http://purl.org/dc/elements/1.1/">  <rdf:Description rdf:about="http://localhost:3333/0">  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Description"/>  <dc:title>Letter 01: Bierce to Elizabeth (Lily) Walsh, undated</dc:title>  <dc:publisher>University of Cincinnati</dc:publisher>  <dc:creator>Archives and Rare Book Library</dc:creator>  <dc:date>2015</dc:date>  <dc:description>Letters of Ambrose Bierce to Myles Walsh, 1895-1911</dc:description>  </rdf:Description> |

Figure 3