Guide to preparing data for RES

This article is about preparing data for indexing by RES. To keep things simple, I assume that you are familiar with the RES project, RDF, and the principles of Linked Open Data. If you need an introduction to these topics, Inside Acropolis is a good place to start.

Unlike Inside Acropolis, this article provides a worked example of exposing existing web pages and other digital assets to RES for indexing. I suggest a workflow for converting existing assets, and guidance on how to structure and annotate your data so that RES can make the best of it.

The raw material for the conversion task comes from an imaginary castle museum ("Museum of The Castle of Otranto") with its own website (http://castleofotranto.gothic/). The website contains images, articles and videos about the castle. The owners of the museum want the website data to be indexed by RES so that it can be used by teachers, students and academics.

I'll focus on how to make "RES-friendly" data for one page of the castle website, http://castleofotranto.gothic/items/the-helmet. This page is about a helmet in the museum collection with this content:

- **Identifier:** C001
- **Label:** The Golden Helmet
- **Physical description:** Large gold helmet crowned with a plume of black feathers.
- **Dimensions:** 3m diameter
- **Materials:** 18 carat gold, swan feathers
- **Origin:** Created in the 15th century by an unknown Spanish armourer.
- **History:** The helmet spontaneously fell from its mount on a wall in the main hall in 1529; Conrad, son of Lord Manfred Otranto, was due to wed Isabella that day, but was instead crushed to death by the falling helmet. The helmet was then lost for many years, until, in 1749, it was rediscovered by archaeologist Fuchsia Otranto in the ruins of the old chapel.
- **Display location:** On permanent display in the Arms and Armour Wing.

The page also has an image of the helmet embedded in it:

- http://castleofotranto.gothic/images/the-helmet-photo.jpg A digital photograph of the helmet as it is displayed in the museum. This is credited to Vlad Dracula and was taken in 2016.

By the end of this article, we will have produced a machine-readable version of the data on this web page as RDF. This data could then be indexed by RES, so that the resources in the original web page (the data about the helmet and the image of it) can be found and reused by others.

All of the RDF examples in the article are presented in Turtle format. For the sake of brevity, the RDF namespaces and corresponding prefixes used in the examples are summarised here:

```
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix dct: <http://purl.org/dc/terms/> .
```

```
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix formats: <http://www.w3.org/ns/formats/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix mrss: <http://search.yahoo.com/mrss/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix unit: <http://qudt.org/vocab/unit#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

Where an RDF example introduces a new prefix, that prefix will be included in the example.

The RES data model

RES has a fairly constrained view of the world, and isn't interested in the fine detail of data models: it is primarily a high-level index of the resources it has "ingested". But it is intensely interested in a few key properties of resources, and in the links between resources. It is especially keen on multimedia resources of all shapes and sizes.

The following sections explain this view of the world in more detail, focusing on how to improve RES's indexing of your resources by applying the following techniques:

1. **Use RES categories** RES looks for particular types of resource when indexing, and organises resources of those types into special categories. By making your resources fit these categories, you stand more chance of them being clearly visible in the right place in RES.
2. **Use standard ontologies** Using common RDF ontologies to define the properties of your resources means that they can be displayed properly by RES, and that your data can be easily interpreted by users of RES.
3. **Link with other resources** RES attempts to combine resources from different datasets into aggregate resources, so that consumers can see all of the data relating to a resource in one place. For example, a resource about Shakespeare's "Twelfth Night" may contain data aggregated from DBPedia, the BBC Genome pages, and the BBC Shakespeare dataset. Relating your resources to well-known datasets like DBPedia makes it easier for RES to build these aggregates and incorporate your data into them.
4. **Provide licensing and format metadata** By declaring the licences on your description resources (as a minimum), you give RES permission to read those descriptions and index them.
5. **Organise data into datasets** Once you've defined your resources, you can add "meta" resources which gather them into a browsable set, by using VoID.
6. **Publish the datasets on the web** With the resources defined, licensed and formalised as datasets, you can put them on the web (using a web server) so that RES and other Linked Open Data users can access them.
7. **Ask RES to index your data** Finally, you can ask for your data to be crawled, so that it is included in RES's index.

1. Use RES categories

RES divides the world into four main categories of resource:

1. **Real-world things** (people, creative works, agents, physical objects, places, events etc.) These can

be physical ("William Blake", "London"), purely conceptual ("The Theory of Evolution"), imaginary ("Narnia"), or anything in-between.
2. **Descriptions of things** Examples in the real world are entries in a library catalogue, or the text on a card next to a museum exhibit. In the context of the web, this includes web pages about real-world things.
3. **Digital assets** (photos, videos, web pages) relating to topics (or to other digital assets) This covers resources which are mostly experienced through a computer, and which can potentially be accessed or reused by other people: "a digital photograph of Barack Obama", "a video about the Mona Lisa". Note that descriptions of things can also be digital assets, e.g. "a web page describing The Louvre" is both a description of a thing and a digital asset.
4. **Collections** (i.e. sets of topics, descriptions, digital assets, or even other collections) This allows grouping of "things" according to their aspects or facets: for example, "a collection of novels by Victorian novelists", "suits of armour worn by King Henry VIII".

We can use this list to categorise resources on the museum website. Looking at the web page we're modelling, we can identify real-world things by looking for nouns in its content. Doing this results in the following (at least), any of which could be described using RDF:

1. A *physical thing*, the helmet in the real world, which is the main topic of the page.
2. Conrad, the *person* killed by the helmet (the groom).
3. Isabella, the *person* who was Conrad's betrothed.
4. Lord Manfred Otranto, a *person* who is Conrad's father.
5. The *event* of the helmet falling on Conrad.
6. The *materials* from which the helmet is made.
7. Fuchsia Otranto, the *person* who found the helmet.
8. Vlad Dracula, the *person* who took the photo of the helmet.
9. The ruined chapel, the *place* where the helmet was found.
10. Spain, the *place* where the helmet was created.
11. An unnamed blacksmith, the *person* who made the helmet.
12. The Arms and Armour wing, the physical *place* in the museum where the helmet resides now. This is also a *collection* which contains the helmet plus other museum exhibits.

The image of the helmet is best treated as a *digital asset+*, as it is a reusable, useful piece of media.

Finally, there is the original web page itself, which is a *description* of all of the above as HTML, as well as being a *digital asset*.

So how do we decompose the data from the web page to model the resources identified above as RDF?

A first step is to take the existing structure of the page and convert it directly (and naively) to RDF with little translation. To do this, we create a single resource for the main topic of the page (the helmet) and do a very basic conversion of its "fields" (i.e. sections of the web page which describe its properties and related resources) to a kind of "pidgin RDF", as described below.

1.1. Pidgin RDF

The aim here is to create the crudest mapping of the web page into RDF, with no regard for semantics or logical consistency. This is quick and dirty; but it is useful, too, as it highlights issues

we need to address to produce a "proper" RDF model.

Before we can do even this, though, we have to start thinking about the URIs for our resources. As in all RDF modelling, each resource is identified by a unique URI to distinguish it from all other resources. For the purposes of this article, we'll put the RDF resources on a sub-path of the main site, at http://castleofotranto.gothic/data/. We'll use a further URI path segment to divide the data site further: any physical items in the museum are under "/data/items/". (If you are familiar with building RESTful web services, this strategy should be familiar.)

The next part of the URI is derived from the museum database's identifier for the item, "C001", which should provide a fairly future-proof URI (assuming the museum identifiers won't change). If your dataset includes identifiers (e.g. if you use a database to power your website), they make good candidates for this.

Finally, because the resource we're creating stands in for an object in the "real world", we append "#id" onto the end of it. The final URI for the resource representing the helmet therefore is:

`http://castleofotranto.gothic/data/items/C001#id`

What is the reason for the "#id"? This is to do with the types of resource that can be identified using URIs, and how those resources are accessed on the web. Recall that we are creating resources for "things" which exist outside the web (also known as *non-information resources*). The "essence" of these things can't be delivered over the web: if I put the URI `http://castleofotranto.gothic/data/items/C001#id` into my web browser, I wouldn't expect the *actual* helmet to be returned to me; at best, I would expect to get a representation of the helmet, e.g. some HTML or RDF. The "#id" is a way to deal with this: because the fragment part of a URI is ignored by the server, requesting the non-information resource `http://castleofotranto.gothic/data/items/C001#id` (aka *dereferencing the URI*) will actually return the information resource at `http://castleofotranto.gothic/data/items/C001` (no hash); in our case, this will resolve to an RDF representation of the helmet. For more information about this distinction, see Cool URIs for the Semantic Web and Dereferencing HTTP URIs.

The resource at `http://castleofotranto.gothic/data/items/C001#id` contains statements about the "real" helmet, replicating the data in the web page:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .

<http://castleofotranto.gothic/data/items/C001#id>
  mco:identifier "C001" ;
  mco:label "The Golden Helmet"@en-gb ;
  mco:physical_description "Large gold helmet crowned with a plume of black
feathers."@en-gb ;
  mco:dimensions "3m diameter"@en-gb ;
  mco:made_of "18 carat gold, swan feathers"@en-gb ;
  mco:origin "Created in the 15th century by an unknown Spanish armourer."@en-gb ;
  mco:history "The helmet spontaneously fell from its mount on a wall in the main
hall in 1529; Conrad, son of Lord Manfred Otranto, was due to wed Isabella that day,
but was instead crushed to death by the falling helmet. The helmet was then lost
```

```
for many years, until, in 1749, it was rediscovered by archaeologist Fuchsia
Otranto in the ruins of the old chapel."@en-gb ;
  mco:display_location "On permanent display in the Arms and Armour wing."@en-gb .
```

The `mco` prefix refers to an ontology for the museum's RDF, identified by the URI `http://castleofotranto.gothic/data/schema#`; this is purely imaginary at the moment. Each piece of data attached to the helmet resource uses a hypothetical property from this ontology. This is quick and dirty, but it means we don't have to worry about looking up the right ontologies to use to be able to rough out the RDF. We use string literals for all the objects of the statements for the same reason.

How far does this get us? The good news is that we already have some RDF; the bad is that we haven't defined precisely what the helmet resource is, so that RES can index it properly. To do that, we assign a common type to the helmet so that RES treats it as a specific kind of thing, as described next.

1.2. Apply a RES-friendly category to the main topic

Previously, I mentioned how RES organises resources into broad categories, one of which is "real-world things". The helmet, as mentioned above, is the real-world thing which is the main topic of the web page.

But, more specifically, the helmet is a physical thing. RES has a series of datasets, used as containers for resources it has indexed; you can see the list of these at http://acropolis.org.uk/. One of these datasets contains physical things, at http://acropolis.org.uk/things, which seems a suitable place for the helmet.

To be included in the physical things dataset, a resource must either have the type `crm:E18_Physical_Thing`, or a type which maps onto it. (RES's mappings between types can be seen at https://github.com/bbcarchdev/spindle/blob/develop/twine/rulebase.ttl.) One type which is mapped to `crm:E18_Physical_Thing` is `crm:E22_Man-Made_Object`; applying this type to the helmet resource is as simple as adding a single statement:

```
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  ...
```

`crm:E22_Man-Made_Object` is a term from the CIDOC-CRM vocabulary. This vocabulary was designed for cultural heritage data, so it is ideally suited to describing things in The Museum of the Castle of Otranto.

When this RDF is indexed, the helmet will now be included in the "physical things" dataset in RES.

Inclusion of a resource in a RES dataset is determined by whether a resource has a particular type,

or can be mapped to one of these types, as shown below:

- **Agents (typically used for agents who aren't real people, e.g. characters in books):** use *foaf:Agent*
- **Audiences:** use *odrl:Group*
- **Collections:** use *dcmitype:Collection*
- **Concepts:** use *skos:Concept*
- **Creative works (e.g. novels, plays, articles):** use *frbr:Work*
- **Digital assets:** use *foaf:Document*
- **Events:** use *event:Event*
- **Groups:** use *foaf:Group*
- **People (i.e. actual people who lived or are living):** use *foaf:Person*
- **Physical things:** use *crm:E18_Physical_Thing*
- **Places:** use *geo:SpatialThing*

It's not essential for a resource to end up in one of these datasets, but if it does, it will be more visible to a user browsing RES's content.

1.3. Split out other topics and categorise them

The helmet resource has properties which reference various other "things". Rather than tucking these away in properties of the helmet, they could be treated as resources in their own right. Going back to the topic list above, we have the following resources we could potentially add:

1. People: "Conrad", "Isabella", "Lord Manfred", "Fuchsia Otranto", "unknown blacksmith", "Vlad Dracula".
2. Materials which make up the helmet: "18 carat gold", "swan feathers".
3. Places: "the ruined chapel", "Spain", "the Arms and Armour Wing".
4. Events: "the helmet falling on Conrad", "the wedding", "the helmet being made".
5. Collections: "the Arms and Armour Wing".

How do we decide which of these things should be converted into resources in their own right? The main question to ask is whether it would be useful to have a separate resource for a thing. To understand what "useful" means in this context, compare the following examples.

Example 1: using a string for the display location

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  mco:label "The Golden Helmet"@en-gb ;
  mco:display_location "On permanent display in the Arms and Armour wing."@en-gb ;
  ...
```

Example 2: using a geo:SpatialThing for the display location

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  mco:label "The Golden Helmet"@en-gb ;
  mco:display_location <http://castleofotranto.gothic/data/locations/L001#id> ;
  ...

<http://castleofotranto.gothic/data/locations/L001#id>
  a geo:SpatialThing ;
  mco:label "Arms and Armour Wing"@en-gb .
```

Example 1 is an abbreviated version of what we already have, using a textual description of the location. Example 2 instead adds another resource for the location, of type `geo:SpatialThing`, and sets the `display_location` property of the helmet to point at it. Is this new `geo:SpatialThing` resource useful?

It could be, if we want to describe more items which are housed in the Arms and Armour Wing. These items could each have a `mco:display_location` property set to the "Arms and Armour Wing" resource URI (`<http://castleofotranto.gothic/data/locations/L001>`). This would enable a RES user to easily find all of the items in the "Arms and Armour Wing" by querying for any resource where `mco:display_location == <http://castleofotranto.gothic/data/locations/L001>`.

Consider trying to do the same thing if the location is stored as a free-text string. This might work if every item in the "Arms and Armour Wing" had an identical `mco:display_location` string. However, because free-text strings are not part of a "controlled vocabulary", there is significant scope for discrepancy: one item might be "On display in the Arms and Armour wing", another might be "In the Arms and Armour wing", and a third may be in "arms and armer" (sic). All of these can be understood as the same location by a human reader, but a machine would have to do additional work (potentially a lot of it) to match the strings and recognise them as equivalent. Using a URI removes this ambiguity, as it assigns a unique name to the location which can be referenced from any other resource.

Also, from the perspective of RES, having a separate resource with type `geo:SpatialThing` means that the "Arms and Armour Wing" is added to RES's "Places" dataset (at http://acropolis.org.uk/places). This brings an additional benefit of making that place more visible to RES users.

Multiple types

Although we specified the Arms and Armour Wing as having type `geo:SpatialThing`, we could also specify it as having a "collection" type, as it contains a related group of items in the museum. RDF places no restriction on the number of types which a resource can have, providing they are logically compatible with each other.

Applying the additional `dcmitype:Collection` type on the resource will expose it as part of another dataset in RES ("Collections", at http://acropolis.org.uk/collections) and make the RDF look like this:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .

<http://castleofotranto.gothic/data/locations/L001#id>
  a geo:SpatialThing, dcmitype:Collection ;
  ...
  mco:label "Arms and Armour Wing"@en-gb .
```

1.4. Decompose free-text fields into resources

The next step is to (optionally) decompose free-text fields into more RDF resources. For example, the materials, descriptions and origin fields of the museum items could be processed this way.

While this can be a tricky and time-consuming task, it may be relatively simple where the free-text fields are "semi-formatted" already, i.e. have a consistent layout wherever they are used. Straightforward string manipulation can be used to convert such fields into resources. For example, in the case of the materials found in museum items (the `mco:made_of` field), we could automate extraction of resources as each field contains comma-separated phrases.

However, automatic processing may be prohibitively difficult. For example, different records within a single dataset may have wildly varying amounts and formats of text in the same free-text field; some may contain very little, while others may have a huge chunk. For example, the helmet's description is:

"The helmet spontaneously fell from its mount on a wall in the main hall in 1529; Conrad, son of Lord Manfred Otranto, was due to wed Isabella that day, but was instead crushed to death by the falling helmet. The helmet was then lost for many years, until, in 1749, it was rediscovered by archaeologist Fuchsia Otranto in the ruins of the old chapel."

But it could just as easily have been:

"A gold helmet found in the castle."

It's possible that there is no simple algorithm which can decompose free-text with as much variation as even these two examples. In such cases, there are two choices:

1. Leave the free-text data as is, and just store it as a literal in the RDF.
2. Do a manual decomposition of the free-text data into additional RDF resources.

The approach you take depends on the time you have available to do the conversion to RDF. For the purposes of this article, we'll leave the `mco:history` and `mco:origin` properties as literals.

Returning to the example, the text strings "18 carat gold" and "swan feathers" are extracted to generate two new "material" resources. Because we are already using the CIDOC-CRM vocabulary

to specify the helmet's type as `crm:E18_Physical_Thing`, we use another type from the same ontology, `E57_Material`, as the type of these new resources:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  ...
  mco:made_of <http://castleofotranto.gothic/data/materials/M001#id>,
    <http://castleofotranto.gothic/data/materials/M002#id> .

<http://castleofotranto.gothic/data/materials/M001#id>
  a crm:E57_Material ;
  mco:label "18 carat gold"@en-gb .

<http://castleofotranto.gothic/data/materials/M002#id>
  a crm:E57_Material ;
  mco:label "swan feathers"@en-gb .
```

Again, using separate resources for the materials used in museum items enables reuse: other items made of the same stuff can reference the materials resources, and a consumer of the data can more easily find all of the things made of a particular material.

Note that we are still using our own property `mco:made_of` to associate the helmet with its materials. This will be rectified later, when we look at replacing our own predicates with those from standard ontologies.

1.5. Add digital assets

A key use case (possibly *the* key use case) for RES is surfacing digital assets (images, videos, web pages) which are reusable in other contexts. The web page we're working on has an image, `http://castleofotranto.gothic/images/the-helmet-photo.jpg`, which could be used this way.

When producing RDF for digital assets, the URI for the resource is typically the same as the URI of the digital asset file itself. Note that because URIs for digital assets are actual URIs for information resources (resources which which can be conveyed over the web), they don't use the "#id" URI form we've used previously.

As a minimum, the description should contain a type for the asset, as well as a `dct:format` statement describing its MIME type. For example:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://castleofotranto.gothic/images/the-helmet-photo.jpg>
  a foaf:Image, dcmitype:StillImage ;
```

```
    dct:format <http://purl.org/NET/mediatypes/image/jpeg> ;
    mco:label "Photograph of the golden helmet in the museum"@en-gb ;
    mco:photographer "Vlad Dracula"@en-gb ;
    mco:date_taken "2016" .
```

I've used the pidgin RDF approach to get a first version of the description of the image data, again using predicates from the fictitious `mco` ontology to speed things up. This will be refined in later sections.

As with topic types, it's important to use the correct types for digital assets. In particular, you should assign a type to each digital asset, using a value from the DCMI Types vocabulary, as follows:

- **Photograph, digital scan, painting, static picture:** use *dcmitype:StillImage* and *foaf:Image*.
- **Raw video file (AVI, MPEG etc.), video player embedded in a web page:** use *dcmitype:MovingImage*.
- **Web page, digitised book, PDF:** use *dcmitype:Text* and *foaf:Document*.
- **Audio file:** use *dcmitype:Audio*.

Adding these types is important, because RES has special facilities for querying assets by `dcmitype`. For example, this URL will give you a list of "things" whose label and/or description contain the key word "disease":

```
http://acropolis.org.uk/?q=disease
```

We can filter this set of results to only return things (still matching "disease") which also have a related image:

```
http://acropolis.org.uk/?q=disease&media=http://purl.org/dc/dcmitype/StillImage
```

Note that the `dct:format` statement uses a value from the media types vocabulary. To determine this value yourself, append the MIME type of the digital asset to "http://purl.org/NET/mediatypes/" and use that as the object for `dct:format`. Some examples:

- AVI: http://purl.org/NET/mediatypes/video/avi
- HTML: http://purl.org/NET/mediatypes/text/html
- MP3: http://purl.org/NET/mediatypes/audio/mpeg3
- PNG: http://purl.org/NET/mediatypes/image/png
- WAV: http://purl.org/NET/mediatypes/audio/wav

Next, we associate the image with the thing it depicts (the helmet). Again, RES recognises a special set of predicates used to associate things with media depicting them, which enables all of the media for a topic to be correctly aggregated:

- **Relating a thing to a moving image:** use *mrss:player*.
- **Relating a thing to a still image or audio:** use *mrss:content*.
- **Relating a thing to a web page or other electronic document:** use *foaf:page*.

In our case, we are relating the helmet to an image of it, so we use `mrss:content`.

Finally, we associate the image back to the thing it depicts using a fictitious relationship, `mco:depicts`. The resulting RDF looks like this:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix mrss: <http://search.yahoo.com/mrss/> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  mrss:content <http://castleofotranto.gothic/images/the-helmet-photo.jpg> ;
  ...

<http://castleofotranto.gothic/images/the-helmet-photo.jpg>
  a foaf:Image, dcmitype:StillImage ;
  dct:format <http://purl.org/NET/mediatypes/image/jpeg> ;
  mco:label "Photograph of the golden helmet in the museum"@en-gb ;
  mco:photographer "Vlad Dracula"@en-gb ;
  mco:date_taken "2016" ;
  mco:depicts <http://castleofotranto.gothic/data/items/C001#id> .
```

2. Use standard ontologies

We've already used types from some ontologies which RES has a special understanding of, to ensure that our resources end up in the correct datasets within RES. Next, we take this further, attempting to remove as many of our own "imaginary" ontology terms (under the `mco` prefix) as we can, instead using terms from standard ontologies. This has two benefits:

1. It improves how our resources are displayed in RES.
2. It enables consumers of our data to more easily grasp the structure of our data, providing they are familiar with the ontologies we use.

Here is the full RDF for what we've modelled so far in the previous sections:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix mrss: <http://search.yahoo.com/mrss/> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  mco:identifier "C001" ;
  mco:label "The Golden Helmet"@en-gb ;
  mco:physical_description "Large gold helmet crowned with a plume of black
```

```
feathers."@en-gb ;
  mco:dimensions "3m diameter"@en-gb ;
  mco:made_of <http://castleofotranto.gothic/data/materials/M001#id>,
    <http://castleofotranto.gothic/data/materials/M002#id> ;
  mco:origin "Created in the 15th century by an unknown Spanish armourer."@en-gb ;
  mco:history "The helmet spontaneously fell from its mount on a wall in the main
hall in 1529; Conrad, son of Lord Manfred Otranto, was due to wed Isabella that day,
but was instead crushed to death by the falling helmet. The helmet was then lost
for many years, until, in 1749, it was rediscovered by archaeologist Fuchsia
Otranto in the ruins of the old chapel."@en-gb ;
  mco:display_location <http://castleofotranto.gothic/data/locations/L001#id> ;
  mrss:content <http://castleofotranto.gothic/images/the-helmet-photo.jpg> .

<http://castleofotranto.gothic/data/locations/L001#id>
  a geo:SpatialThing, dcmitype:Collection ;
  mco:label "Arms and Armour Wing"@en-gb .

<http://castleofotranto.gothic/data/materials/M001#id>
  a crm:E57_Material ;
  mco:label "18 carat gold"@en-gb .

<http://castleofotranto.gothic/data/materials/M002#id>
  a crm:E57_Material ;
  mco:label "swan feathers"@en-gb .

<http://castleofotranto.gothic/images/the-helmet-photo.jpg>
  a foaf:Image, dcmitype:StillImage ;
  dct:format <http://purl.org/NET/mediatypes/image/jpeg> ;
  mco:label "Photograph of the golden helmet in the museum"@en-gb ;
  mco:photographer "Vlad Dracula"@en-gb ;
  mco:date_taken "2016" ;
  mco:depicts <http://castleofotranto.gothic/data/items/C001#id> .
```

To improve this, we replace our `mco` terms with generic terms from common vocabularies. In particular, we use the RDFS (prefix: `rdfs`), FOAF (Friend Of A Friend (prefix: foaf), and Dublin Core (DC) Terms (prefix: `dct`) vocabularies to make some initial substitutions:

- If a resource has a title (e.g. a proper name), use `dct:title`.
- If a resource has a long description, use `dct:description` or `rdfs:comment`.
- If a resource has a summary/synopsis, use `dct:abstract`.
- If a resource has an identifier specific to its dataset (e.g. a unique catalogue number in a museum collection), use `dct:identifier` to record it.
- If a resource has a creator (e.g. author, artist, developer, designer), use `dct:creator`.
- If a resource doesn't have a proper name, provide a short name via `rdfs:label`.
- If a resource depicts another resource, use `foaf:depicts`.

`rdfs:label`, `rdfs:comment` and `dct:description` are particularly useful, as the RES user interface uses these for its human-readable description of a resource; for example, this resource for Twelfth Night gets its title "Twelfth Night" from the `rdfs:label` and its longer description from `rdfs:comment`.

The DC Terms, FOAF and RDFS ontologies are very general, widely-used, and well understood. If possible, terms from these ontologies should be your first choice for replacing custom terms from your own ontologies. However, where there are no suitable terms in these ontologies, terms from other ontologies can be used instead. This is where data modelling becomes a craft rather than a science: choosing relevant vocabularies and applying them correctly is a skill in itself, and can be daunting.

One way to approach the task is to use a very generic but broad ontology, such as schema.org. This provides a wide range of different terms for describing common types of resource, and covers a considerable range of modelling tasks. However, schema.org can be too generic for specialised modelling. If you find that you need a more expressive ontology, this list of commonly-used vocabularies provides some alternatives.

In the case of items in the Museum of the Castle of Otranto, schema.org doesn't provide the granularity we need in our model: for example, it doesn't have terms for associating materials with a physical object, or for describing dimensions. But we have already used terms from the CIDOC-CRM ontologies to define the types for the helmet and its materials; this ontology is specialised for describing cultural heritage objects, so it makes sense to use more terms from it to replace our `mco` properties.

With judicious use of RDFS, DC Terms and CIDOC-CRM, we can replace almost all of our custom `mco` terms, as follows:

- Replace `mco:made_of` with `crm:P45_consists_of`. Note that `crm:P45_consists_of` expects a `crm:E18_Physical_Thing` as its subject and an `E57_Material` as its object, which is what we have here.
- Replace `mco:dimensions` with `crm:P43_has_dimension`. Because this property expects a full-fledged `crm:E54_Dimension`, we introduce a new resource for that, `<http://castleofotranto.gothic/data/items/C001#diameter>`. Note that, because this resource is only relevant to the helmet, we make it a "secondary resource" related to the helmet resource by using a hash URI ("#diameter"). This highlights the dependency between the two resources: the diameter resource doesn't make much sense as a resource in its own right at a URI with a different path, as it only exists by virtue of the helmet resource. To define the properties of the new dimension resource `<http://castleofotranto.gothic/data/items/C001#diameter>`, we use more terms from the CIDOC-CRM ontology:
  - `crm:P2_has_type`, which defines the type of dimension. We use a value from the British Museum's vocabulary of types of dimension for this, `<http://collection.britishmuseum.org/id/thesauri/dimension/diameter>`.
  - `crm:P90_has_value`, which defines the value for the dimension (a number in this case).
  - `crm:P91_has_unit`, which defines the unit which applies to the value. We use a value from the qudt.org unit vocabulary as the object for this property, `unit:Meter`.
- Replace `mco:display_location` with another term from CIDOC-CRM, `crm:P55_has_current_location`. This needs a `crm:E53_Place` as its object; therefore, we need `<http://castleofotranto.gothic/data/locations/L001>` to be a `crm:E53_Place`, so we add a

type statement to satisfy this requirement.
- For the image resource, we can create a `foaf:Person` resource for "Vlad Dracula" (the photographer) at the URI `<http://castleofotranto.gothic/data/people/P001>`, using `foaf:givenName` and `foaf:familyName` to define his name. We then make this resource the `dct:creator` of the photograph. We replace `mco:date_taken` with the standard `dct:date` property to specify when the photograph was taken. Finally, we replace `mco:depicts` with `foaf:depicts`, to associate the image with the "thing" it is a representation of.

Making the above changes to the RDF for the helmet results in:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix mrss: <http://search.yahoo.com/mrss/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix unit: <http://qudt.org/vocab/unit#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  dct:identifier "C001" ;
  dct:title "The Golden Helmet"@en-gb ;
  dct:description "Large gold helmet crowned with a plume of black feathers."@en-gb
;
  crm:P45_consists_of <http://castleofotranto.gothic/data/materials/M001#id>,
    <http://castleofotranto.gothic/data/materials/M002#id> ;
  crm:P43_has_dimension <http://castleofotranto.gothic/data/items/C001#diameter> ;
  mco:origin "Created in the 15th century by an unknown Spanish armourer."@en-gb ;
  mco:history "The helmet spontaneously fell from its mount on a wall in the main
hall in 1529; Conrad, son of Lord Manfred Otranto, was due to wed Isabella that day,
but was instead crushed to death by the falling helmet. The helmet was then lost
for many years, until, in 1749, it was rediscovered by archaeologist Fuchsia
Otranto in the ruins of the old chapel."@en-gb ;
  crm:P55_has_current_location
<http://castleofotranto.gothic/data/locations/L001#id> ;
  mrss:content <http://castleofotranto.gothic/images/the-helmet-photo.jpg> .

<http://castleofotranto.gothic/data/items/C001#diameter>
  a crm:E54_Dimension ;
  crm:P2_has_type
<http://collection.britishmuseum.org/id/thesauri/dimension/diameter> ;
  crm:P90_has_value "3"^^xsd:integer ;
  crm:P91_has_unit unit:Meter .

<http://castleofotranto.gothic/data/locations/L001#id>
  a geo:SpatialThing, crm:E53_Place, dcmitype:Collection ;
  rdfs:label "Arms and Armour Wing"@en-gb .

<http://castleofotranto.gothic/data/materials/M001#id>
  a crm:E57_Material ;
  rdfs:label "18 carat gold"@en-gb .
```

```
<http://castleofotranto.gothic/data/materials/M002#id>
  a crm:E57_Material ;
  rdfs:label "swan feathers"@en-gb .

<http://castleofotranto.gothic/images/the-helmet-photo.jpg>
  a foaf:Image, dcmitype:StillImage ;
  dct:format <http://purl.org/NET/mediatypes/image/jpeg> ;
  rdfs:label "Photograph of the golden helmet in the museum"@en-gb ;
  dct:creator <http://castleofotranto.gothic/data/people/P001#id> ;
  dct:date "2016"^^xsd:gYear ;
  foaf:depicts <http://castleofotranto.gothic/data/items/C001#id> .

<http://castleofotranto.gothic/data/people/P001#id>
  a foaf:Person ;
  rdfs:label "Vlad Dracula"@en-gb ;
  foaf:givenName "Vlad"@en-gb ;
  foaf:familyName "Dracula"@en-gb .
```

The only remaining terms from the `mco` ontology are `mco:history` and `mco:origin`. For the sake of simplicity, we'll leave these as they are. Remember that it is OK to retain your own terms if there are no available replacements, or if replacing them or decomposing them into more resources would prove too time-consuming and/or add complex.

If you intend to keep your own ontology, you should consider formalising and publishing it, either as an RDF schema or an OWL ontology. However, this is beyond the scope of the current article.

3. Link to other resources

Now that the resources are described using standard terms as far as possible, the next step is to make them more useful by connecting them to other datasets. Depending on the type of data you're dealing with, this may be more or less difficult:

- If your data are about things which are globally recognisable or especially culturally significant, it's simple enough to find other datasets to link them to. Examples might be general data about ancient civilisations, or articles about popular Victorian authors. Well-known datasets like DBPedia and Wikidata contain masses of useful data you can link to in this case.
- If your data are highly specialised or very specific to your organisation, it may be harder to find datasets with a lot of comparable data. Extreme examples might be datasets about people working for a small company on a remote island, or about specialised products (e.g. bathroom fittings). If general datasets don't contain much linkable stuff, you may be aware of more specialised datasets which are applicable and can be linked to. It may even be that your organisation is the authority on a particular area of knowledge. If this is the case, your dataset may be considered the authoritative source of data in that area, and other people may need to link to you, rather than you linking to them.

In either case, the steps for linking to other datasets are the same:

1. Identify a source resource A in your dataset which could relate to a target resource B in another dataset. Good candidates are "things" in your dataset which belong to one or more of the

categories RES recognises, namely:
    - Concepts
    - Creative works
    - Digital assets
    - Events
    - People
    - Physical things
    - Places In the case of the Museum of the Castle of Otranto, "Vlad Dracula" is a particularly well-known person who is probably referred to in other datasets. (Other candidates might be the castle itself, and the materials the helmet is made from.)
2. Identify the target resource B that you want to link to. To do this, you will need to make use of the APIs provided by other datasets to find related resources. These differ between datasets: some may provide a search interface, others a SPARQL endpoint, others a RESTful API. But some of the most well-known datasets are organised in such a way that you can intuitively find related resources using simple shortcuts; one such dataset is DBPedia. The quick method for finding related DBPedia resources is as follows:
    1. Take the natural title or label (`name`) of your resource, e.g. "Vlad Dracula".
    2. Replace any spaces in `name` with underscores, e.g. "Vlad_Dracula".
    3. Construct a `test URL` of the form "http://dbpedia.org/resource/" + `name`, e.g. "http://dbpedia.org/resource/Vlad_Dracula".
    4. Using a browser, open `test URL` and see whether it returns a resource which looks relevant. If it does, you can relate your resource to `test URL`. If not, you can try other variations (e.g. http://dbpedia.org/resource/Dracula) or resort to the search interface.
3. Identify the type of relationship between the source and the target. Because RES is an aggregator, relationships between datasets are especially valuable in creating RES's index. These relationships allow resources from two or more datasets to be correlated in RES's index: if resource A is described as being the same as resource X, and resource B is also the same as resource X, RES can combine data from A and B into a new "proxy" resource, containing the data from both. There are three main relationships which are useful for linking to other datasets, and which also help RES form its proxy resources:
    1. `rdfs:seeAlso`: This term from RDFS specifies a general "you might also be interested in" kind of relationship between two resources. It makes no statement about whether two resources are equivalent. However, if resources A and B are both related by `rdfs:seeAlso` to resource X, it's likely that resources A and B are also relevant to each other in some (albeit possibly distant) way.
    2. `owl:sameAs`: This term from the OWL ontology expresses a much stronger relationship between two resources. If A is `owl:sameAs` B, it means that A and B are identical to each other: both refer to the same resource. This is very useful and generic, but should be used carefully: the types of the two resources should be similar, at least to the extent that they are not logically inconsistent. For example, if resource A is a `frbr:Work` it probably cannot be `owl:sameAs` resource B which is a `foaf:Person` (unless resource B represents a person who is a creative work somehow...).
    3. `skos:exactMatch`: This term from the SKOS ontology relates two SKOS concepts together. For this to be applicable, resources A and B should both have the type `skos:Concept`, as required by the SKOS ontology's definition of `skos:exactMatch`. This makes the term less generic, but is especially useful when working with datasets based on SKOS.

Looking at the resources we've defined so far, we can identify a few opportunities for linking:

- The person "Vlad Dracula" can be related to http://dbpedia.org/resource/Vlad_Dracula. This resource represents exactly the same person, so `owl:sameAs` is applicable.
- The helmet appears in the novel *The Castle of Otranto*, which has a DBPedia resource at http://dbpedia.org/resource/The_Castle_of_Otranto. The appropriate relationship is `rdfs:seeAlso`, as the DBPedia resource contains data pertinent to the helmet (but isn't the same thing as the helmet).
- The "18 carat gold" used in creating the helmet can be related to the DBPedia resource about gold, http://dbpedia.org/resource/Gold. The appropriate relationship is `rdfs:seeAlso`, as the DBPedia data is pertinent to the material used in the helmet.
- For the "swan feathers", a link can be added which points at a DBPedia resource about the symbolism of black swan feathers, http://dbpedia.org/resource/Black_swan_emblems_and_popular_culture. Again, `rdfs:seeAlso` is the appropriate relationship.

Adding these statements to our resources gives us:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  dct:title "The Golden Helmet"@en-gb ;
  ...
  rdfs:seeAlso <http://dbpedia.org/resource/The_Castle_of_Otranto> .

<http://castleofotranto.gothic/data/materials/M001#id>
  a crm:E57_Material ;
  rdfs:label "18 carat gold"@en-gb ;
  rdfs:seeAlso <http://dbpedia.org/resource/Gold> .

<http://castleofotranto.gothic/data/materials/M002#id>
  a crm:E57_Material ;
  rdfs:label "swan feathers"@en-gb ;
  rdfs:seeAlso <http://dbpedia.org/resource/Black_swan_emblems_and_popular_culture>
.

<http://castleofotranto.gothic/data/people/P001#id>
  a foaf:Person ;
  rdfs:label "Vlad Dracula"@en-gb ;
  ...
  owl:sameAs <http://dbpedia.org/resource/Vlad_Dracula> .
```

As RES indexes datasets, it may find other resources for "Vlad Dracula" which are `owl:sameAs` the DBPedia resource. In this case, RES will be able to tie the museum's "Vlad Dracula" to those resources, via their `owl:sameAs` relationships. The `rdfs:seeAlso` relationships are less useful for aggregation, but make the data in RES richer: they link the museum dataset to the wider web of data, and provide RES users with a way to find useful information elsewhere.

4. Provide licensing and format statements

In the previous sections, we mostly concentrated on describing *non-information resources*: the "real things" like the helmet, the materials it's made from, where it is in the museum, and who photgraphed it. In each case, we used a "#id" URI to denote the thing as "real"; or, in other words, as a resource which can't be conveyed over the web directly.

So what happens when someone requests the URI for one of these real things, e.g. the helmet's URI `http://castleofotranto.gothic/data/items/C001#id`? Given today's technology, we wouldn't expect to receive the actual helmet over the web; we would instead expect some kind of description of the helmet in a format which our web browser (or other client) can understand. (You may hear this referred to as *dereferencing* a URI: "...the act of retrieving a representation of a resource identified by a URI is known as dereferencing that URI" - from *Dereferencing HTTP URIs*.)

By convention, when the URI of a non-information resource is requested, a server should deliver the information resource which describes it. Requesting the URI `http://castleofotranto.gothic/data/items/C001#id` should return the resource at `http://castleofotranto.gothic/data/items/C001` instead (removing the hash part of the URI to get the URI of the information resource).

We don't have a resource with this URI yet. However, we do already have a set of statements describing the helmet, associated with the helmet resource itself. So what we need now is a resource which is explicitly defined as the description of the helmet, and which is accessible at `http://castleofotranto.gothic/data/items/C001`. Usefully, this description can also carry additional data about who wrote the description, how the description is licensed, when it was written etc.

> Note that this distinction between *information resources* and *non-information resources* is contentious, and there is no complete agreement on the correct terminology or definitions. See this wiki page or P.J. Hayes' *In Defence of Ambiguity* for some discussion. But, for the purposes of keeping this article on track, I'll do some hand-waving (rather than writing a dissertation on semantics) and assume that we can distinguish between "real world" and "digital" things.

First, we create the resource representing the description. We keep to RES conventions and create this as a `foaf:Document`, the type used for information resources which act as descriptions (note that this also makes the description resource a digital asset in its own right):

```
<http://castleofotranto.gothic/data/items/C001>
  a foaf:Document ;
  rdfs:label "Description of 'The Golden Helmet'"@en-gb .
```

Next, we associate the description with the thing it describes (the helmet) via `foaf:primaryTopic`:

```
<http://castleofotranto.gothic/data/items/C001>
  a foaf:Document ;
  rdfs:label "Description of 'The Golden Helmet'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/items/C001#id> .
```

Then we specify the licence *which applies to the description* using `dct:license`:

```
<http://castleofotranto.gothic/data/items/C001>
  a foaf:Document ;
  rdfs:label "Description of 'The Golden Helmet'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .
```

The licence statement is vital, as RES will only index description resources which carry an open licence. See the full list of acceptable licences for details. If the descriptions of things in your dataset are not licensed correctly, RES will refuse to index them. Also note that the licence on the description can be different from the licence on the thing which is being described. For example, a gallery may have a painting in its collection which is "copyright Nosferatu 1956"; but the description of that painting may be released under a "Public Domain" licence. RES is only interested in the latter while crawling your data (though it will index the former and present it to users, too).

The preferred licence for data from UK institutions is the Open Government Licence (OGL), as this makes particular reference to UK law. However, Creative Commons and similar open licences are also acceptable.

The final step is to define the formats in which the description is available. At the beginning of this article, we started constructing RDF using a web page as a basis. Recall that the web page we started from is itself a description, in HTML, of the helmet. So one format we have for the description of the helmet is the original web page. We associate this web page with the description resource using `dct:hasFormat`, as our original web page is a HTML-formatted representation of the data about the helmet. We also add some statements about the web page resource, including its licence:

```
<http://castleofotranto.gothic/data/items/C001>
  a foaf:Document ;
  rdfs:label "Description of 'The Golden Helmet'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/items/the-helmet> .

<http://castleofotranto.gothic/items/the-helmet>
  a dcmitype:Text ;
    dct:format <http://purl.org/NET/mediatypes/text/html> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
```

```
    rdfs:label "Description of 'The Golden Helmet' as HTML"@en .
```

Note that the HTML representation of the description is defined with type `dcmitype:Text` and with a `dct:format` triple formed as for digital assets (see earlier).

To finish this off, we add a Turtle format resource for the description, available at `http://castleofotranto.gothic/data/items/C001.ttl`, again associating it with the description resource via `dct:hasFormat`:

```
@prefix formats: <http://www.w3.org/ns/formats/> .

<http://castleofotranto.gothic/data/items/C001>
  a foaf:Document ;
  rdfs:label "Description of 'The Golden Helmet'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/items/the-helmet>,
    <http://castleofotranto.gothic/data/items/C001.ttl>  .

<http://castleofotranto.gothic/items/the-helmet>
  a dcmitype:Text ;
  dct:format <http://purl.org/NET/mediatypes/text/html> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'The Golden Helmet' as HTML"@en .

<http://castleofotranto.gothic/data/items/C001.ttl>
  a dcmitype:Text, formats:Turtle ;
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'The Golden Helmet' as Turtle"@en .
```

Other "#id" resources representing "real things" were created earlier in this article, namely:

- The Arms and Armour Wing (place): http://castleofotranto.gothic/data/locations/L001#id
- 18 carat gold (material): http://castleofotranto.gothic/data/materials/M001#id
- Swan feathers (material): http://castleofotranto.gothic/data/materials/M002#id
- Vlad Dracula (person): http://castleofotranto.gothic/data/people/P001#id

Each of these will also need description and format resources with appropriate licensing statements. In each case, the description follows the pattern given above. One difference is that we can't associate these descriptions with the original web page, only with an RDF description in Turtle. For example, here's the description resource for "swan feathers":

```
<http://castleofotranto.gothic/data/materials/M002>
  a foaf:Document ;
  rdfs:label "Description of 'swan feathers'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/materials/M002#id> ;
```

```
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/data/materials/M002.ttl>  .

<http://castleofotranto.gothic/data/materials/M002.ttl>
  a dcmitype:Text, formats:Turtle ;
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'swan feathers' as Turtle"@en .

<http://castleofotranto.gothic/data/materials/M002#id>
  a crm:E57_Material ;
  rdfs:label "swan feathers"@en-gb .
```

Note that we don't have an HTML representation of the description of "swan feathers", so we just link to the Turtle description for now. If we went back later and added an HTML page which describes "swan feathers", we could add another resource for this, and link it with `dct:format` to the `http://castleofotranto.gothic/data/materials/M002` resource.

Licensing images

Because the image is already an information resource (it can be conveyed over the web), we don't need an additional information resource for its description. However, we do need to add a licensing statement to the existing resource:

```
<http://castleofotranto.gothic/images/the-helmet-photo.jpg>
  a foaf:Image, dcmitype:StillImage ;
  dct:format <http://purl.org/NET/mediatypes/image/jpeg> ;
  rdfs:label "Photograph of the golden helmet in the museum"@en-gb ;
  dct:creator <http://castleofotranto.gothic/data/people/P001#id> ;
  dct:date "2016"^^xsd:gYear ;
  foaf:depicts <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .
```

Note that RES would still be able to index this metadata about the image, even if the licence applied to it were proprietary: the licence is just here to tell a RES user how they can reuse the image.

5. Organise the data into datasets

Once the data has been transformed to RDF, we need a way to expose it to RES, so that it can be fetched and indexed. The usual way of doing this is via a VoID (Vocabulary of Interlinked Datasets) dataset resource. This provides metadata about all of the data in your dataset, and gives RES a single entry point from which it can reach every resource you are publishing.

As a minimum, you need a VoID dataset resource, along with a dataset description for it. Here's an example which shows these, but doesn't show how to include your resources in the description (we'll get to that later):

```
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix void: <http://rdfs.org/ns/void#> .

<http://castleofotranto.gothic/data/void>
  a void:DatasetDescription ;
  rdfs:label "Description of 'Data from The Museum of the Castle of Otranto'"@en-gb
;
  foaf:primaryTopic <http://castleofotranto.gothic/data/void#dataset> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .

<http://castleofotranto.gothic/data/void#dataset>
  a void:Dataset ;
  rdfs:label "Data from The Museum of the Castle of Otranto"@en-gb .
```

Important points to note:

- The URI for the dataset is `http://castleofotranto.gothic/data/void`. This uses a common pattern for specifying VoID resources within a dataset.
- There are two resources: a description of the dataset with type `void:DatasetDescription`, and the dataset itself with type `void:Dataset`.
- The description has a `foaf:primaryTopic` with the dataset as its object.
- A licence is specified on the description using `dct:license`. This is key, because RES will inspect this licence before indexing the dataset. If an open licence is not specified on the dataset description, RES will not index the related dataset.

For a typical dataset, which may contain anywhere from hundreds to millions of resources, it is often essential to define a dataset as a collection of smaller subsets. If a single dataset were used to contain millions of resources, a client retrieving the dataset would have a massive, impractical download to deal with. (For even larger datasets, which may contain billions of triples, a single dataset becomes a practical impossibility.) Dividing a dataset into subsets makes interacting with it less resource-intensive, as it can be fetched in manageable chunks.

In the case of the museum dataset, we can use the categories we defined earlier to divide the dataset into logical parts: collection items (the helmet), locations with the museum (the Arms and Armour Wing), people (Vlad Dracula), materials (gold and swan feathers), and digital assets (the image of the helmet). Each of these parts becomes a dataset in its own right, with its own description; and those parts become subsets of the main dataset (via a `void:subset` predicate). An example is shown here for the people subset; the pattern is the same for each of the other subsets (collection items, locations, materials, digital assets):

```
<http://castleofotranto.gothic/data/void#dataset>
  a void:Dataset ;
  rdfs:label "Data from The Museum of the Castle of Otranto"@en-gb ;
  void:subset <http://castleofotranto.gothic/data/people#dataset> .
```

```
<http://castleofotranto.gothic/data/people>
  a void:DatasetDescription ;
  rdfs:label "Description of 'People data from The Museum of the Castle of
Otranto'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/people#dataset> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .

<http://castleofotranto.gothic/data/people#dataset>
  a void:Dataset ;
  rdfs:label "People data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class foaf:Person .
```

The `void#dataset` resource is connected to `people#dataset` via `void:subset`; `people#dataset` is described in a way which is similar to the main `void#dataset`, but with two more statements:

1. A `void:inDataset` which provides a back-link from the people dataset to the root dataset.
2. A `void:class` statement which specifies a type for all resources within this dataset. In this case, we are stating that every resource within the people dataset will have the type `foaf:Person` (NB they could have other types, too).

We haven't yet associated any of the museum's resources to the datasets. This is again done via `void:subset`, e.g. to associate "Vlad Dracula" with our people dataset:

```
<http://castleofotranto.gothic/data/people#dataset>
  a void:Dataset ;
  rdfs:label "People data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class foaf:Person ;
  void:subset <http://castleofotranto.gothic/data/people/P001#id> .
```

Note that the relationship is from the `void:Dataset` resource to the "#id" URI for the person: in other words, we put the resource representing the person, rather than the resource for their description, into the dataset.

We can also modify the resource for "Vlad Dracula", adding a statement that he is a member of `people#dataset` via `void:inDataset`:

```
<http://castleofotranto.gothic/data/people/P001#id>
  a foaf:Person ;
  rdfs:label "Vlad Dracula"@en-gb ;
  foaf:givenName "Vlad"@en-gb ;
  foaf:familyName "Dracula"@en-gb ;
  owl:sameAs <http://dbpedia.org/resource/Vlad_Dracula> ;
  void:inDataset <http://castleofotranto.gothic/data/people#dataset> .
```

The datasets for other parts of the museum data follow a similar pattern; their datasets could be defined at these URIs:

```
<http://castleofotranto.gothic/data/items#dataset>
<http://castleofotranto.gothic/data/locations#dataset>
<http://castleofotranto.gothic/data/materials#dataset>
```

In some cases, it may be necessary to further sub-divide a subset into manageable pieces. For example, if you had thousands of resources in `people#dataset`, you could divide it again, alphabetically, e.g.

```
<http://castleofotranto.gothic/data/people#dataset>
  a void:Dataset ;
  rdfs:label "People data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class foaf:Person ;
  void:subset <http://castleofotranto.gothic/data/people/A#dataset>,
    <http://castleofotranto.gothic/data/people/B#dataset>, ... .
```

where `http://castleofotranto.gothic/data/people/A#dataset` is a dataset of people whose family name begins with "A" etc.

Or perhaps into "pages", grouped by ID:

```
<http://castleofotranto.gothic/data/people#dataset>
  a void:Dataset ;
  rdfs:label "People data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class foaf:Person ;
  void:subset <http://castleofotranto.gothic/data/people/P0001-P1000#dataset>,
    <http://castleofotranto.gothic/data/people/P1001-P2000#dataset>, ... .
```

where `http://castleofotranto.gothic/data/people/P001-P1000#dataset` is another dataset for people with IDs in the range P0001 to P1000 inclusive etc.

This section shows the minimal amount of statements you should attach to your VoID resources so that RES knows what to do with them. But the VoID specification includes various other terms for adding metadata about your datasets, such as the number of triples they contain, who maintains them, when they were created or modified and so on. You can use these as you see fit, though they are not necessary from RES's perspective.

6. Publish the datasets on the web

The process of converting the museum resources to RDF is now complete. An example of the full RDF for the museum resources, along with their description and formatting resources and the datasets which contain them, is given at the end of this article.

This brings us to the final stage in making our data "RES ready": putting the RDF onto the web somewhere, so that RES can fetch and index it. However, this stage is one of the hardest to

generalise about, as we have to consider implementation details; more specifically, how do we put the dataset on the web so that resources are accessible at the URIs we've given them?

There are many ways this could be managed; see Best Practice Recipes for Publishing RDF Vocabularies and Linked Data: Evolving the Web into a Global Data Space for suggestions). We'll look at three common approaches below.

Option 1: Publish as static files

The simplest approach is to create a plain text file for each resource, containing its RDF in an appropriate format. As we've already used Turtle in the examples, we'll choose Turtle as the format for the text files.

First, we create a directory structure to hold the files, which can be mapped onto the paths in the URIs we want to serve. In the case of the museum website, all the paths are under the `/data/` path, so that becomes our top-level directory; under that, we create a directory for each sub-path, giving us:

```
/data/
/data/items/
/data/people/
/data/materials/
/data/locations/
```

Next, we create a Turtle file with a .ttl suffix for each resource URI, including the VoID resources we created in the previous section:

```
/data/void.ttl => /data/void (VoID RDF for the whole dataset)
/data/items.ttl => /data/items (VoID RDF for the items dataset)
/data/items/C001.ttl => /data/items/C001 (RDF for the helmet)
/data/people.ttl => /data/people (VoID RDF for the people dataset)
/data/people/P001.ttl => /data/people/P001 (RDF for the person "Vlad Dracula")
/data/materials.ttl => /data/materials (VoID RDF for the materials dataset)
/data/materials/M001.ttl => /data/materials/M001 (RDF for the material "18 carat
gold")
/data/materials/M002.ttl => /data/materials/M002 (RDF for the material "swan
feathers")
/data/locations.ttl => /data/locations (VoID RDF for the locations dataset)
/data/locations/L001.ttl => /data/locations/L001 (RDF for the location "Arms and
Armour Wing")
```

(The text after `=>` shows the mapping from the directory to a resource URI, as well as a summary of the resource it represents.)

What goes into the Turtle files? The guideline here is to include any resources whose URI path, minus any hash, matches the path of the file, minus its ".ttl" suffix. For example, for the file `/data/locations/L001.ttl`, we would include any resources whose URI path (without the hash)

matches `/data/locations/L001`; this would include `http://castleofotranto.gothic/data/locations/L001#id` (the location resource), `http://castleofotranto.gothic/data/locations/L001` (the description of the location resource) and `http://castleofotranto.gothic/data/locations/L001.ttl` (the representation of the description); which gives us this content for `/data/locations/L001.ttl`:

```
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix formats: <http://www.w3.org/ns/formats/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix void: <http://rdfs.org/ns/void#> .

<http://castleofotranto.gothic/data/locations/L001>
  a foaf:Document ;
  rdfs:label "Description of 'Arms and Armour Wing'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/locations/L001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/data/locations/L001.ttl>  .

<http://castleofotranto.gothic/data/locations/L001.ttl>
  a dcmitype:Text, formats:Turtle ;
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'Arms and Armour Wing' as Turtle"@en .

<http://castleofotranto.gothic/data/locations/L001#id>
  a geo:SpatialThing, crm:E53_Place, dcmitype:Collection ;
  rdfs:label "Arms and Armour Wing"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/locations#dataset> .
```

However, there are some resources for which we have created RDF descriptions, but which we won't deliver RDF for directly; specifically, existing digital assets, like the original web page and the image of the helmet. In this case, we would include the RDF for these resources in the Turtle file for other resources which reference them. For the image and the web page about the helmet, this means including their resources in the Turtle file for the helmet at `/data/items/C001.ttl`:

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix formats: <http://www.w3.org/ns/formats/> .
@prefix mrss: <http://search.yahoo.com/mrss/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix unit: <http://qudt.org/vocab/unit#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<http://castleofotranto.gothic/data/items/C001>
  a foaf:Document ;
  rdfs:label "Description of 'The Golden Helmet'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/items/the-helmet>,
    <http://castleofotranto.gothic/data/items/C001.ttl>  .

...

<http://castleofotranto.gothic/images/the-helmet-photo.jpg>
  a foaf:Image, dcmitype:StillImage ;
  dct:format <http://purl.org/NET/mediatypes/image/jpeg> ;
  rdfs:label "Photograph of the golden helmet in the museum"@en-gb ;
  dct:creator <http://castleofotranto.gothic/data/people/P001#id> ;
  dct:date "2016"^^xsd:gYear ;
  foaf:depicts <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .
```

Although the files used in this article were hand-cranked, they could be generated from an existing database if you have one. In this situation, when generating the Turtle files, an algorithm could be used to determine which resources to include in each file. For the above case, you might first find the set of resources which will definitely be in the file, due to their URIs matching; then, include any additional resources which have either a `dct:hasFormat` or `mrss:content` relationship with that set.

For more suggestions about deciding which resources to deliver for a particular URI, see this section of *How to Publish Linked Data on the Web*.

Hosting static RDF files

The process of hosting static files on the web is well-understood and standardised. There are many options for hosting a Linked Open Data site composed of static files, from the venerable Apache web server, to servers custom-built on top of libraries like Express, to RES's own Linked Open Data publication engine, Quilt.

Whichever solution you use, though, the main consideration is how to deliver those static files using *content negotiation*. The underlying idea of content negotiation is that when a client requests a resource, it should receive a representation of that resource which is appropriate to its capabilities. In practice, this means that the server should map the request URI onto a file which represents the resource in a format that the client understands.

The mechanism used for finding out what the client "understands" is to look at the `Accept` header sent in the request. This defines the MIME types which the client would prefer to get in the server's response; if the server can't supply any of those formats, the client won't understand the response. For example, to request a Turtle representation of the Golden Helmet, a client might make an HTTP request like this (common request headers have been removed for brevity):

```
GET /data/items/C001 HTTP/1.1
Host: castleofotranto.gothic
Accept: text/turtle
```

To create a response for this request, the server tries to build a Turtle representation for the resource with the URI `http://castleofotranto.gothic/data/items/C001`. How this is done depends on the server technology being used; but, in general, web servers are configured to serve static files from a directory (e.g. called the `DocumentRoot` in Apache) at a particular host name (`castleofotranto.gothic` in this case). Any request coming in to the server maps the path of the request URI (`/data/items/C001`) to a path inside the document root: so if the document root in this case were the directory `/www/castleofotranto.gothic/`, the server would attempt to find a file at `/www/castleofotranto.gothic/data/items/C001`. If no file exists at this path, rewrite rules on the server adjust the requested path to look for similarly-named files; for example, by appending filename suffixes which marry up with the MIME types the client can accept. Here, this would mean looking for a Turtle file with a ".ttl" suffix, e.g. `/www/castleofotranto.gothic/data/items/C001.ttl`. As this file exists in our directory structure, its content can be returned in the response.

Also note that requests for a non-information resource (typically with a "#" in its path) will resolve to that resource's URI, minus the hash. For example, a request for `http://castleofotranto.gothic/data/items/C001#id` (the helmet) will receive the RDF for the resource at `http://castleofotranto.gothic/data/items/C001` (the information resource which describes the helmet). This is because when a client requests a URI from the server, it shouldn't send the fragment identifier (the hash part); the fragment identifier is only significant on the client side, e.g. to enable the browser to locate the user within a section of an HTML page once it is rendered.

Hopefully, it should be obvious that a static file server could be configured to serve files from our RDF file structure. Using the algorithm above, the path from a client's HTTP request, combined with an appropriate `Accept` header, could be mapped onto one of our specific Turtle files. The content of that Turtle file would then be returned as a response.

For more information about content negotiation, with some examples of how to configure Apache to use it, see Serving Linked Data as Static RDF/XML Files and Perform content negotiation when requests are received for item URIs.

Option 2: Publish from a database

If you already have your dataset in a database somewhere, perhaps in an electronic catalogue or as the driver for a Content Management System (CMS), you could generate RDF dynamically from that database. When a resource is requested, the application queries the database, then creates an RDF representation from the query results (similar to how many database-driven websites work, except you're generating RDF rather than HTML).

How you do this again depends on your environment: for example, if your website runs from a CMS,

you may be able to use the CMS's APIs to develop a data site alongside the main website. Alternatively, you may be able to develop a custom server-side application which uses the same database as your catalogue or CMS, but which is served from different infrastructure. Standard web development languages could be used for this (Python, Ruby, PHP etc.), as for any other database-driven web application.

There is at least one de-facto standard tool for serving RDF from a relational database in this mode: D2R. D2R enables an existing relational database to be converted into RDF and served over the web, using a mapping file to translate from database tables and columns to RDF resources and their properties. This might be an option if you have a database you want to quickly convert into a Linked Open Data site.

Finally, instead of publishing dynamically from a database, you could periodically generate static RDF files from the database. These could then be served by an HTTP server, as described in the previous section.

Option 3: Publish from an RDF store

A dedicated RDF platform is a good way to serve a Linked Open Data site, especially if you have large amounts of data. Using such a platform provides storage for the RDF, usually alongside a server for publishing it on the web and tools for working with it, such as RESTful APIs, editors, and SPARQL endpoints.

There are many options for storing RDF, including these open source ones:

- Apache Jena has two components which are relevant: TDB (triple store) and Fuseki (SPARQL end-point). TDB provides RDF storage, while Fuseki enables its content to be queried over the web.
- OpenLink Virtuoso is a commercial product, but has an open source edition.
- You could write your own custom RDF storage application and provide a SPARQL endpoint on top of it. The aforementioned Apache Jena is one Java library for doing this; RDF4J is another. There are also libraries in numerous other languages, such as Redland for Python and Ruby-RDF for Ruby.

(Note that inclusion in the list above should not be taken as an endorsement by me or the BBC.)

Once you have your RDF in a triple store, you need a way to publish it as Linked Open Data: when a resource URI is requested from a server, the correct resource(s) should be retrieved from the triple store, then serialised into an RDF format acceptable to the client. Some of the above platforms, like Virtuoso, have this as core functionality; others, like the Apache Jena components, may need another layer on top. Pubby is one such layer, able to expose a SPARQL endpoint as Linked Open Data.

7. Ask RES to index your data

To get RES to index your data, you need to contact the RES team, supplying the URI of your main VoID resource (e.g. `http://castleofotranto.gothic/data/void` in the above examples) and

requesting a crawl of your data.

Your dataset will then be added to RES's queue. Once it has been crawled, you will be notified that your data has been indexed in RES.

Conclusion

This completes conversion of the web page we started with, http://castleofotranto.gothic/items/the-helmet, to RDF, published as Linked Open Data.

Once you've gone through the steps in this article, you should understand the structures and processes you need to publish your own data on the web, including:

- How to decide on URI patterns for publishing resources.
- Techniques for identifying resources which could be published.
- Which vocabularies to use.
- Ways to link your resources to other datasets.
- Creating VoID resources to introduce your data to RES.
- An overview of platforms for publishing your data.

Although the process is quite involved, and can seem labour intensive, this article should help you reach the goal of publishing Linked Open Data. Hopefully, it has equipped you to make decisions about how to deal with your data and present it as RDF in such a way that RES can index it.

Appendix: Sample RDF for the Museum of the Castle of Otranto

This section contains the complete RDF for the examples in this article, presented in Turtle format.

```
@prefix mco: <http://castleofotranto.gothic/data/schema#> .
@prefix crm: <http://www.cidoc-crm.org/cidoc-crm/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dcmitype: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix formats: <http://www.w3.org/ns/formats/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix mrss: <http://search.yahoo.com/mrss/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix unit: <http://qudt.org/vocab/unit#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://castleofotranto.gothic/data/void>
  a void:DatasetDescription ;
  rdfs:label "Description of 'Data from The Museum of the Castle of Otranto'"@en-gb
;
  foaf:primaryTopic <http://castleofotranto.gothic/data/void#dataset> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .
```

```
<http://castleofotranto.gothic/data/void#dataset>
  a void:Dataset ;
  rdfs:label "Data from The Museum of the Castle of Otranto"@en-gb ;
  void:subset <http://castleofotranto.gothic/data/items#dataset>,
    <http://castleofotranto.gothic/data/people#dataset>,
    <http://castleofotranto.gothic/data/materials#dataset>,
    <http://castleofotranto.gothic/data/locations#dataset> .

<http://castleofotranto.gothic/data/items>
  a void:DatasetDescription ;
  rdfs:label "Description of 'Item data from The Museum of the Castle of
Otranto'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/people#dataset> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .

<http://castleofotranto.gothic/data/items#dataset>
  a void:Dataset ;
  rdfs:label "Item data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class crm:E22_Man-Made_Object ;
  void:subset <http://castleofotranto.gothic/data/items/C001#id> .

<http://castleofotranto.gothic/data/people>
  a void:DatasetDescription ;
  rdfs:label "Description of 'People data from The Museum of the Castle of
Otranto'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/people#dataset> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .

<http://castleofotranto.gothic/data/people#dataset>
  a void:Dataset ;
  rdfs:label "People data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class foaf:Person ;
  void:subset <http://castleofotranto.gothic/data/people/P001#id> .

<http://castleofotranto.gothic/data/materials>
  a void:DatasetDescription ;
  rdfs:label "Description of 'Materials data from The Museum of the Castle of
Otranto'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/people#dataset> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .

<http://castleofotranto.gothic/data/materials#dataset>
  a void:Dataset ;
  rdfs:label "Materials data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class crm:E57_Material ;
  void:subset <http://castleofotranto.gothic/data/materials/M001#id> ,
    <http://castleofotranto.gothic/data/materials/M002#id> .

<http://castleofotranto.gothic/data/locations>
  a void:DatasetDescription ;
  rdfs:label "Description of 'Locations data from The Museum of the Castle of
```

```
Otranto'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/locations#dataset> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .

<http://castleofotranto.gothic/data/locations#dataset>
  a void:Dataset ;
  rdfs:label "Locations data from The Museum of the Castle of Otranto"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/void#dataset> ;
  void:class crm:E53_Place ;
  void:subset <http://castleofotranto.gothic/data/locations/L001#id> .

<http://castleofotranto.gothic/data/items/C001>
  a foaf:Document ;
  rdfs:label "Description of 'The Golden Helmet'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/items/the-helmet>,
    <http://castleofotranto.gothic/data/items/C001.ttl>  .

<http://castleofotranto.gothic/items/the-helmet>
  a dcmitype:Text ;
  dct:format <http://purl.org/NET/mediatypes/text/html> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'The Golden Helmet' as HTML"@en .

<http://castleofotranto.gothic/data/items/C001.ttl>
  a dcmitype:Text, formats:Turtle ;
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'The Golden Helmet' as Turtle"@en .

<http://castleofotranto.gothic/data/items/C001#id>
  a crm:E22_Man-Made_Object ;
  dct:identifier "C001" ;
  dct:title "The Golden Helmet"@en-gb ;
  dct:description "Large gold helmet crowned with a plume of black feathers."@en-gb
;
  crm:P45_consists_of <http://castleofotranto.gothic/data/materials/M001#id>,
    <http://castleofotranto.gothic/data/materials/M002#id> ;
  crm:P43_has_dimension <http://castleofotranto.gothic/data/items/C001#diameter> ;
  mco:origin "Created in the 15th century by an unknown Spanish armourer."@en-gb ;
  mco:history "The helmet spontaneously fell from its mount on a wall in the main
hall in 1529; Conrad, son of Lord Manfred Otranto, was due to wed Isabella that day,
but was instead crushed to death by the falling helmet. The helmet was then lost
for many years, until, in 1749, it was rediscovered by archaeologist Fuchsia
Otranto in the ruins of the old chapel."@en-gb ;
  crm:P55_has_current_location
<http://castleofotranto.gothic/data/locations/L001#id> ;
  mrss:content <http://castleofotranto.gothic/images/the-helmet-photo.jpg> ;
  rdfs:seeAlso <http://dbpedia.org/resource/The_Castle_of_Otranto> ;
  void:inDataset <http://castleofotranto.gothic/data/items#dataset> .

<http://castleofotranto.gothic/data/items/C001#diameter>
```

```
  a crm:E54_Dimension ;
  crm:P2_has_type
<http://collection.britishmuseum.org/id/thesauri/dimension/diameter> ;
  crm:P90_has_value "3"^^xsd:integer ;
  crm:P91_has_unit unit:Meter .

<http://castleofotranto.gothic/data/locations/L001>
  a foaf:Document ;
  rdfs:label "Description of 'Arms and Armour Wing'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/items/L001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/data/locations/L001.ttl>  .

<http://castleofotranto.gothic/data/locations/L001.ttl>
  a dcmitype:Text, formats:Turtle ;
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'Arms and Armour Wing' as Turtle"@en .

<http://castleofotranto.gothic/data/locations/L001#id>
  a geo:SpatialThing, crm:E53_Place, dcmitype:Collection ;
  rdfs:label "Arms and Armour Wing"@en-gb ;
  void:inDataset <http://castleofotranto.gothic/data/locations#dataset> .

<http://castleofotranto.gothic/data/materials/M001>
  a foaf:Document ;
  rdfs:label "Description of '18 carat gold'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/materials/M001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/data/materials/M001.ttl>  .

<http://castleofotranto.gothic/data/materials/M001.ttl>
  a dcmitype:Text, formats:Turtle ;
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of '18 carat gold' as Turtle"@en .

<http://castleofotranto.gothic/data/materials/M001#id>
  a crm:E57_Material ;
  rdfs:label "18 carat gold"@en-gb ;
  rdfs:seeAlso <http://dbpedia.org/resource/Gold> ;
  void:inDataset <http://castleofotranto.gothic/data/materials#dataset> .

<http://castleofotranto.gothic/data/materials/M002>
  a foaf:Document ;
  rdfs:label "Description of 'swan feathers'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/materials/M002#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/data/materials/M002.ttl>  .

<http://castleofotranto.gothic/data/materials/M002.ttl>
  a dcmitype:Text, formats:Turtle ;
```

```
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'swan feathers' as Turtle"@en .

<http://castleofotranto.gothic/data/materials/M002#id>
  a crm:E57_Material ;
  rdfs:label "swan feathers"@en-gb ;
  rdfs:seeAlso <http://dbpedia.org/resource/Black_swan_emblems_and_popular_culture>
;
  void:inDataset <http://castleofotranto.gothic/data/materials#dataset> .

<http://castleofotranto.gothic/data/people/P001>
  a foaf:Document ;
  rdfs:label "Description of 'Vlad Dracula'"@en-gb ;
  foaf:primaryTopic <http://castleofotranto.gothic/data/people/P001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  dct:hasFormat <http://castleofotranto.gothic/data/people/P001.ttl>  .

<http://castleofotranto.gothic/data/people/P001.ttl>
  a dcmitype:Text, formats:Turtle ;
  dct:format <http://purl.org/NET/mediatypes/text/turtle> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> ;
  rdfs:label "Description of 'Vlad Dracula' as Turtle"@en .

<http://castleofotranto.gothic/data/people/P001#id>
  a foaf:Person ;
  rdfs:label "Vlad Dracula"@en-gb ;
  foaf:givenName "Vlad"@en-gb ;
  foaf:familyName "Dracula"@en-gb ;
  owl:sameAs <http://dbpedia.org/resource/Vlad_Dracula> ;
  void:inDataset <http://castleofotranto.gothic/data/people#dataset> .

<http://castleofotranto.gothic/images/the-helmet-photo.jpg>
  a foaf:Image, dcmitype:StillImage ;
  dct:format <http://purl.org/NET/mediatypes/image/jpeg> ;
  rdfs:label "Photograph of the golden helmet in the museum"@en-gb ;
  dct:creator <http://castleofotranto.gothic/data/people/P001#id> ;
  dct:date "2016"^^xsd:gYear ;
  foaf:depicts <http://castleofotranto.gothic/data/items/C001#id> ;
  dct:license
<https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> .
```