HTML5 Microdata and Schema.org

On June 2, 2011, Bing, Google, and Yahoo! announced the joint effort Schema.org. When the big search engines talk, Web site authors listen. This article is an introduction to Microdata and Schema.org. The first section describes what HTML5, Microdata and Schema.org are, and the problems they have been designed to solve. With this foundation in place section 2 provides a practical tutorial of how to use Microdata and Schema.org using a real life example from the cultural heritage sector. Along the way some tools for implementers will also be introduced. Issues with applying these technologies to cultural heritage materials will crop up along with opportunities to improve the situation.

By Jason Ronallo

Foundation

HTML5

The HTML5 standard or (depending on who you ask) the HTML Living Standard has brought a lot of changes to Web authoring. Amongst all the buzz about HTML5 is a new semantic markup syntax called Microdata.

HTML elements have semantics. For example, an ol element is an ordered list, and by default gets rendered with numbers for the list items. HTML5 provides new semantic elements like header, nav, article, aside, section and footer that allow more expressiveness for page authors. A bunch of div elements with various class names is no longer the only way to markup this content.

These new HTML5 elements enable new tools and better services for the Web ecosystem. Browser plugins can more easily pull out the text of the article for a cleaner reading experience. Search engines can give more weight to the article content rather than the advertising in the sidebar. Screen reader software can use the structural elements such as nav to make textual content more accessible to people with disabilities.

While these new elements provide extremely useful extra information about the sections of content, they do not really describe what the HTML document is about. For example HTML5 does not provide a book element for use in Web pages being delivered by library catalog. Most Web applications sit on top of databases that contain lots of structured data. However, the trip this data takes from the database into HTML is often lossy, as the structured data is converted to HTML for display. Maybe a human can read your field labels on the page to understand your metadata, but that meaning is lost on machines.
Fortunately HTML5 includes a syntax called Microdata, that allows web publishers to layer richly structured metadata directly into their Web presentations. But before we jump into how Microdata works, we'll take a brief tour through previous attempts at making structured data available on the Web.

To Embed Or Not To Embed

One way to communicate structured metadata in HTML is to link your HTML presentation to an alternate representation of your data. Here's a simple example of making RIS formatted citation data available:

```html
<link rel="alternate" type="application/x-research-info-systems" href="/search?q=cartoons&format=ris" />
```

This pre-HTML5 approach can work in many cases, such as in the area of Web syndication with RSS and Atom...but it has some problems. Techniques like this require consumers of your data to have out of band knowledge of where to look for this particular invisible content. It also adds a layer of complexity by requiring a site to have APIs or metadata gateways that can be burdensome to setup and expensive for organizations to maintain.

Another approach which avoids some of those problems is to embed the data directly in the HTML. The HTML representation of a resource is most visible to users, so it is also the HTML code which gets the most attention from developers. Little-used, overlooked APIs or data feeds are easy to let go stale. If the Website goes down, you are likely to hear about it from multiple sources immediately. If the OAI-PMH gateway goes down, it would probably take longer for you to find out about it.

Hidden services and content are too easy to neglect. Data embedded in visible HTML helps keep the representations in sync so that page authors only have to expose one public version of their data. This insight has lead to a number of different standards over time which take the approach of embedding structured data along with the visible HTML content. Microdata is just one of the syntaxes in use today.

A Short History of Structured Data in HTML

Other efforts that came before Microdata have addressed this same problem of marking up the meaning of content. Microformats are one of the earliest efforts to provide “a general approach of using visible HTML markup to publish structured data to the Web.” Some Microformat specifications like hCard, hCalendar, and rel-license are in common use across the Web. Development of the various small Microformat standards takes place on a
community wiki. Simply put, Microformats usually use the convention of standard class names to provide meaning on a Web page. The latest version microformats-2 simplifies and harmonizes these conventions across specifications significantly.

RDFa, a standard of the W3C, has the vision of “Bridging the Human and Data Webs.” The idea is to provide attributes and processing rules for embedding RDF (and all of its graph-based, Linked Data goodness) in HTML. With all that expressive power comes some difficulty, and implementing RDFa has proven to be overly complex for most Web developers. Google has supported RDFa in some fashion since 2009, and over that time has discovered a large error rate in the application of RDFa by webmasters. Simplicity is a central reason for the development of Microdata and the search engines preferring it over RDFa. In part a reaction to greater adoption of Microdata, a simplified profile of RDFa has been created. RDFa Lite 1.1 provides simpler authoring guidelines that mirror more closely the syntax of Microdata.

It also bears mentioning that in the library space, we have developed specifications which tried to solve similar problems of making structured data available to machines through HTML. The unAPI specification uses a microformat for exposing (using the deprecated abbr method) the presence of identifiers which may resolve to alternative formats through a Web service. COinS uses empty spans to make OpenURL context objects available for autodiscovery by machines. While there has been some deployment of unAPI and COinS in the library community, it in no way approaches the use that Microformats and RDFa have seen in the larger Web ecosystem.

So what is Microdata?

Microdata came out of a long thread about incorporating RDFa into HTML5. Because RDFa simply was not going to be incorporated into HTML5, something else was needed to fill the gap. Out of that thread and collected use cases Ian “Hixie” Hickson, the editor of the HTML5 specification, showed the first work on Microdata on May 10, 2009 (the syntax has changed some since then). The syntax is designed to be simple for page authors to implement.

According to Microdata terminology the things being described in an HTML page are items. Each item is made up of one or more key-value pairs: a property and a value. The Microdata syntax is completely made up of HTML attributes. These attributes can be used on any valid HTML element. The core of the data model is made up of three new HTML attributes:

- **itemscope** which says that there is a new item within
- **itemtype** which specifies the type of item
- **itemprop** which gives the item properties and values

A First Example
Here is a simple example of what Microdata looks like:

```html
<div itemscope itemtype="unorganization">
  <span itemprop="eponym">code4lib</span>
</div>
```

The user of a browser would only see the text “code4lib” on the page. The snippet provides more meaning for machines by asserting that there is an “unorganization” with the “eponym” of “code4lib.”

The `itemscope` attribute creates an item and requires no value. The `itemtype` attribute asserts that the type of thing being described is an “unorganization.” This item has a single key-value pair—the property “eponym” with a value of “code4lib.”

To make it clear to someone who thinks in JSON, here is what the item looks like:

```json
{
  "type": ["unorganization"],
  "properties": {
    "eponym": ["code4lib"]
  }
}
```

Those are the only three attributes necessary for a complete understanding of the Microdata model. You’ll learn about two more optional ones later. Pretty simple, right?

What is Schema.org?

While the above snippet is completely valid Microdata, it uses arbitrary language for its `itemtype` and `itemprop` values. If you only need to communicate this information within a tight community and do not need anyone else to ever understand what your data means, that may be just fine. But for the most part you probably want many other machines (like Web crawlers) to understand the meaning of your content. To accomplish this you need to use a shared language so that page authors and consumers can cooperate on how to interpret the meaning.

This is where the Schema.org vocabulary comes in. The search engines (Bing, Google, Yahoo!) created Schema.org and have agreed to support and understand it. It is unrealistic for them to try to support every vocabulary in use. Schema.org is an attempt to define a broad, Web-scale, shared vocabulary focusing on popular concepts. It stakes a position as a “middle” ontology that does not attempt to have the scope of an “ontology of everything” or go into depth in any one area. A central goal of having such a broad schema all in one place
is to simplify things for mass adoption and cover the most common use cases. Understandably, the vocabulary does seem to have a bias towards search engine and commercial use cases.

The type hierarchy presented on this site is not intended to be a ‘global ontology’ of the world. It only covers the types of entities for which we (Microsoft, Yahoo! and Google), think we can provide some special treatment for, through our search engine, in the near future. (Schema.org Data Model)

You can browse the full hierarchy of the vocabulary to get a feel of the bounds of the world according to search engines. Schema.org defines a hierarchy of types all descending from Thing. Thing has four properties (description, image, name, url) which are inherited by all other types. Child types can add their own properties and in turn can have their own children types. Each property name has the same meaning when found in any type in the vocabulary. We will get to other specifics later in the tutorial.

Microdata and Schema.org have a tight connection, though each can be used without the other. The search engines are currently the main consumers of Schema.org data and have a stated preference for Microdata. The Schema.org examples are written using the Microdata syntax. Both are designed and work well together to make adoption simple (and less error-prone) for HTML authors.

Here is the above Microdata example rewritten to use the Schema.org type (and make more sense).

```html
<div itemscope itemtype="http://schema.org/Organization">
  <span itemprop="name">code4lib</span>
</div>
```

Rich Snippets

The most obvious way that the search engines are currently using Microdata is to use the embedded data on pages to display rich snippets in search results. If you have done a Google search in the past two years, you have probably seen some examples of rich snippets showing up in your search results. You can see good examples of how powerful this is by doing a Google Recipe Search for “vegan cupcakes”:
This search result snippet for a recipe includes an image, reviews, cooking time, calorie count, some of the text introducing the recipe, and a list of some of the ingredients needed. This gives a lot more information to the user to help them decide whether to click on a particular result. Snippets with this kind of extra information are reported to increase click through rates.

Google started presenting rich snippets in 2009. Using embedded markup like microformats, RDFa, or Microdata, page authors can influence what may show up in a search result snippet. Both Microformats and RDFa were promoted for rich snippets in the past using various vocabularies and continue to be supported. Google added support for Microdata for Rich Snippets in early 2010. After RDFa Lite was created, the Schema.org partners agreed to support that syntax as well.

Before Schema.org, rich snippets were constrained to being triggered by the few types defined by data-vocabulary.org, which prefigured the Schema.org approach. Reviews, products, and breadcrumbs were the kinds of common types of data that could be marked up. At the time of this writing there is still not rich snippet support for all, or even most, of the Schema.org types. The number of supported types and example rich snippets has been slowly growing. The promise is that many more of the Schema.org types will begin to trigger rich snippets.

Microdata with Schema.org is the latest search-engine preferred method for exposing structured data in HTML, and this is the main reason to use it. While other consumers for structured data using Microdata and/or Schema.org may appear in the future, the most compelling use cases are currently from the search engines, especially for rich snippets. By providing the search engines with more data on your pages, it improves the search experience of users and can draw them to your site. Since most of the users of your site likely come through the search engines, this could be a powerful way to draw more users to your resources. The search engines may also start using structured data as signals for relevance, though they are unlikely to say much publicly about this.

From a developer's perspective there are many considerations beyond rich snippets for choosing a particular syntax. Microdata has a natural fit with HTML and is designed for
simplicity and ease of implementation. Schema.org simplifies the documentation and choices to make. In my own experience implementing Microdata and Schema.org was much simpler than a past failed attempt at implementing RDFa using various vocabularies on a similar site.

Tutorial

This tutorial will move through implementing Microdata and Schema.org on a pre-existing site for exposing digitized collections. The example is based on NCSU Libraries' Digital Collections: Rare and Unique Materials. Each section will lead you through decisions that are made and common problems that are encountered in implementing Microdata and Schema.org. Along the way are tips, tricks, tools, and a closer look at the specifications.

Before Microdata

Here's a screenshot of the page to mark up. As we add Microdata markup, the appearance of the page will not change at all. The page uses a grid system to place the image on the left and the metadata to the right.

These students are jumping with excitement to learn more about HTML5 Microdata and Schema.org!
Students jumping in front of Memorial Bell Tower

Photograph Information

Created Date
circa 1981
Subjects
Buildings
Students
Genre
Architectural photographs
Publicity photographs
Digital Collection
University Archives Photographs

Building Information

Building Name
Memorial Tower
Description
Memorial Tower honors those alumni who were killed in World War I. The cornerstone was laid in 1922 and the Tower was dedicated on November 11, 1949.
Address
2701 Sullivan Drive
Raleigh, NC 27607
Latitude, Longitude
35.786098, -78.663498

Source Information

Repository
Special Collections Research Center at NCSU Libraries
Collection
Memorial Tower (UA023.005.063)
Rights
Reproduction and use of this material requires permission from North Carolina State University.

Here is the basic structure of the main content of the page with some sections and attributes removed with ellipses for brevity.
Students jumping in front of Memorial Bell Tower

Photograph Information
- Created Date: circa 1981
- Subjects: Buildings, Students
- Genre: Architectural photographs, Publicity photographs
- Digital Collection: University Archives Photographs

Adding a WebPage
When you are adding Microdata there is a sense in which you are always just describing a Web page. Ian Hickson, the editor of the Microdata specification, has said that Microdata items exist “in the context of a page and its DOM. It does not have an independent existence outside the page” (Ian Hickson on public-vocabs list). This is different than the way RDFa may think about embedding structured data in HTML as part of a graph which links items together across the Web. Microdata is not so much Linked Data as it is a description of a single page. There are efforts to serialize Microdata as RDF that allow Microdata to be used in a Linked Data environment.

When using the Schema.org vocabularies, every page is implicitly assumed to be some kind of WebPage, but the advice is to explicitly declare the type of page. When picking an appropriate type, it is best to choose the most specific type that could accurately represent the thing to be marked up. In this case it seems appropriate to use ItemPage which is a subclass of WebPage. ItemPage adds no new properties to WebPage, but it communicates to the search engines that the page refers to a single item rather than a search results page or other type of page.

In the HTML snippet below the ItemPage is added to the div#main on the page and some properties are added within the scope of that div#main.

```html
<div id="main" class="container_12" itemscope itemtype="http:schema.org/ItemPage">
  <h2 id="page_name" itemprop="name">
    Students jumping in front of Memorial Bell Tower
  </h2>
  <div class="grid_5">
    <img id="main_image" alt="Students jumping in front of Memorial Bell Tower" src="/images/bell_tower.png" itemprop="image">
  </div>
  <div id="metadata" class="grid_7">
    ...
  </div>
</div>
```

Here we apply the itemprop and itemtype attributes at a level in the DOM that surrounds the item being described in the page. Since we are describing the page, we could instead add the itemprop and itemtype at a higher level. If we need to use some metadata in the head of the document, say the title element, we could apply this to the body or even the html element, though there are some limitations to using Microdata within head.

An itemprop is added as an attribute to the element which contains its value. Within div#main we add the two itemprop attributes for the “name” and “image” properties. Different elements take their itemprop value from different places. In this case the name property is taken from the text content of h2 element. For the img element the value is taken from the src attribute, which is then resolved into an absolute URL. The a element
uses the absolute URL from the `href` attribute. Like `img` and `a`, several elements provide their values through an attribute rather than the text content. If you start using Microdata, you will want to consult this list from the spec which details how property values are determined for different elements.

Microdata JSON and DOM API

One of the cool features of Microdata is that it is designed to be extracted into JSON. You can copy any HTML snippet with Microdata markup into Live Microdata to see what the JSON output would look like. Live Microdata uses MicrodataJS, which is a Javascript (jQuery) implementation of the Microdata DOM API.

The Microdata DOM API is another neat feature of HTML5 Microdata which allows you to extract Microdata client-side. You can test whether your browser implements the Microdata DOM API by running the Microdata test suite created by Opera. If you open a page containing Microdata with Opera Next (at the time of this writing version 12.00 alpha, build 1191) and open up the console (Ctrl+Shift+i), you can play with the Microdata DOM API a bit. `document.getItems()` will return a `NodeList` of all items. The `NodeList` will contain only the top-level items; in this case, one element. It is possible to get all items of a particular type by specifying the type or types as an argument like `document.getItems('http://schema.org/ItemPage')`. This API may change in the future “to match what actually gets implemented.”

```javascript
>>> var itemPage = document.getItems('http://schema.org/ItemPage')
undefined
>>> itemPage
NodeList [{html itemscope="" itemtype="http://schema.org/ItemPage" lang="en">}
>>> itemPage[0].properties
HTMLPropertiesCollection [{h2 id="page_name" itemprop="name">,
  <img itemprop="image" id="main_image" alt="Students jumping in front of Memorial Bell Tower" src="/images/bell_tower.png"/>}
```

ItemPage about a Photograph

Now that we have some basic Microdata about the page, let's try to describe more items. One valid property for an `ItemPage`, is `about`, inherited from `CreativeWork`. `ItemPage` inherits different properites from `Thing`, `CreativeWork`, and `WebPage`. In some cases child types add in new properites, but in the case of `ItemPage` no new properties are defined.

One way to express what the page is about would be to add `itemprop="about"` to the `div#metadata` containing all of the metadata. What would be extracted from the page is just the text content within the `div#metadata` with line breaks and all. Microdata processing never maintains markup. (If you need to maintain some snippet of markup, then
consider using microformats or RDFa, which have the ability to maintain XML literals.) Unlike XML, Microdata is designed to be tolerant of errors. So in this case a processor may still be able to do something useful with even just the text content contained by the div#metadata. Processors and consumers should expect bad data (Conformance).

Looking at how the about property of an ItemPage is defined, the proper value of the about property is another Thing (not text content). In this way Schema.org suggests how to nest items within other items forming a tree. In this case specifying Thing as a valid value of the about property means that any type in the Schema.org hierarchy can be selected to create a new item as the value of the about property. The Photograph type is most appropriate for this example.

The way to show relationships between items on a page is by nesting items. Nesting is implemented by using all three attributes (itemprop, itemscope, itemtype) on the same element. Doing this states that the value of a property is a new item of a particular type. Since all of the metadata describes the photograph, these attributes will be applied to div#metadata which contains all of the metadata.

At this point the subjects and genres can be added as properties of the Photograph. Subjects map well enough to the keywords property of Photograph. The older practice of using invisible meta keywords that had nothing to do with the content in the body were used to try to game the search engines, so they were ignored. The keywords on the example page are visible to users, so the values are less likely to be used to try to trick search engines to rank for particular terms. Google advises page authors to not mark up non-visible content on the page, but to stick to adding Microdata attributes to what is visible to users.

We could attach the “genres” itemprop to the dd element, but then a processor would extract all the text contained within as a single chunk of text. The genres have some spaces within each term, so rather than leaving it up to a post processor to handle that, we apply the same itemprop to each term separately. This ensures that the multiword genres remain intact. At this time regardless of the use of singulars or plurals for property names in the Schema.org documentation, it is allowable to repeat all Schema.org properties like this. When these repeated properties are parsed, the values are merged under a single property key to form a list/array of values.

The “keywords” itemprop cannot be applied to the a element that surrounds each term. The gotcha here is that the property value of an a element is the value of its href attribute. To work around this we use the common pattern of adding some extra spans around the text. Adding the itemprop to the spans allows the text content to be selected instead.

Here is the markup so far:
You can see what has been implemented in Microdata so far as JSON.

Picking Types and Properties (and More Nesting)

After nesting a Photograph within an ItemPage, the nesting can continue further. The photograph has the Memorial Bell Tower at North Carolina State University in the background. The Photograph is about the Memorial Bell Tower. In this case, LandmarksOrHistoricalBuildings seems to work well enough as a type.
Once we add an item for the building, it is easy to add name and description properties for the building. We also have the data to add the address (as a PostalAddress) and geographic location (geo property with a value of a GeoCoordinates) as further nested data items.

Picking appropriate types is an important issue for describing the historic record. At NCSU Libraries we are describing lots of buildings and making related drawings accessible. Some of these buildings are on the National Register of Historic Places, but many are buildings of lesser note. Some were never built or have since been demolished. It could be that for most of the records of buildings in an architectural collection, that a more generic item type like Place would be a more appropriate fit. For other buildings there are definitely more specific types like Airport, CityHall, Courthouse, Hospital, and Church.

Using a more specific type, it would still be impossible for the search engines to recognize that the items refer to the historic record of these places rather than to their current services. We could be giving exact geographic coordinates for a hospital which was demolished or no longer in operation! Also, this is a new metadata facet that may not already be captured in a way that makes it easy to convert to Schema.org types. Some retrospective and ongoing work would be needed to choose the correct type for every building. For now NCSU has made the decision, that will likely get revisited, to always pick LandmarksOrHistoricalBuildings for all the buildings we describe as something of a compromise. It almost communicates that the resource is part of the historic record, but it does lose on the specificity we might like to have.

Picking appropriate types is one area where Schema.org does not provide anything near the level of granularity at which archives and museums often specify the types of objects they describe. Schema.org has types for Painting, Photograph, and Sculpture. Some types of objects like drawings, vases, and suits of armor may have to move back up the hierarchy to use the generic CreativeWork type. With the thousands of types of objects that may be held by libraries, archives, museums and historical societies (LAMs), it is not feasible to add every type to Schema.org. One suggestion made by Charles Moad, Director of the Indianapolis Museum of Art Lab, is to extend CreativeWork with an “objectType” property (private correspondence). If this “objectType” property were part of Schema.org, then the LAM community could suggest that the value of the property come from a vocabulary that covers the kinds of objects LAMs collect. This could go a long way towards expanding the options for LAMs to describe their materials.
This pattern of using external vocabularies maintained by domain experts is used in other places within Schema.org. The JobPosting type includes an occupationalCategory property which suggests the use of the BLS O*NET-SOC taxonomy (though the exact format this value should take is underspecified).

This still leaves open the question of what free, open vocabulary would allow for the kind of extensibility LAMs need to cover all of their object types at a good enough level of specificity. It is not feasible to create a vocabulary to cover and keep up with every type of thing LAMs may collect. The Product Ontology provides a possible model for how to create such an extensible vocabulary by reusing the names of Wikipedia articles to identify types of objects. Something like the following code could be done to simply reuse the name of the Plate_armour article to note that the type of object in the collection is plate armour. The URL http://objectontology.org/id/Plate_armour could resolve to useful information and a tutorial similar to the Product Ontology page for Plate armour. An object in a LAM collection differs enough from a product for sale to need a new namespace to explain the difference, especially to machines. This kind of link also begins to make Microdata more linkable data.

  <span itemprop="name">Gorget (neck defense) and Cuirass (torso defense), for use in the field</span><br/>
  Item Type: <link itemprop="objectType" href="http://objectontology.org/id/Plate_armour"> Plate armour </link>
</div>

Although it could be argued that the addition of objectType is not actually necessary since the Microdata data model allows for itemtype to be used with Wikipedia URLs. If compatibility with Schema.org is not a concern the following is another option:

  <span itemprop="name">Gorget (neck defense) and Cuirass (torso defense), for use in the field</span><br/>
</div>

Another possibility that would maintain compatibility with Schema.org is the following based on a Product Ontology example. The example objectontology.org URL is used here, but the Wikipedia page URL may work as well.

  <span itemprop="name">Gorget (neck defense) and Cuirass (torso defense), for use in the field</span><br/>
  Item Type: <link itemprop="http://www.w3.org/1999/02/22-rdf-syntax-ns#type" href="http://objectontology.org/id/Plate_armour"> Plate armour </link>
</div>

Because there are multiple possibilities to express more specific types of objects, each with
their own strengths, this is definitely an area where the cultural heritage community could come to some agreement and promote a shared approach to the problem.

Result so far

Here's what our marked up snippet looks like so far:

```html
<div id="main" role="main" class="container_12" itemscope itemtype="http:schema.org/ItemPage">
  <h2 id="page_name" itemprop="name">
    Students jumping in front of Memorial Bell Tower
  </h2>
  <div class="grid_5">
    <img itemprop="image" id="main_image" alt="Students jumping in front of Memorial Bell Tower" src="/images/bell_tower.png">
  </div>
  <div id="metadata" class="grid_7" itemprop="about" itemscope itemtype="http://schema.org/Photograph">
    <div id="object" class="info">
      <h2>Photograph Information</h2>
      <dl>
        <dt>Created Date</dt>
        <dd>circa 1981</dd>
        <dt>Subjects</dt>
        <dd>
          <a href="/s/buildings">Buildings</a><br>
          <a href="/s/students">Students</a><br>
        </dd>
        <dt>Genre</dt>
        <dd>
          <a href="/g/architectural_photos">Architectural photographs</a><br>
          <a href="/g/publicity_photos">Publicity photographs</a><br>
        </dd>
        <dt>Digital Collection</dt>
        <dd><a href="/c/uapc">University Archives Photographs</a></dd>
      </dl>
    </div>
    <div id="building" class="info" itemprop="about" itemscope itemtype="http://schema.org/LandmarksOrHistoricalBuildings">
      <h2>Building Information</h2>
      <dl>
        <dt>Building Name</dt>
        <dd><a href="/b/memorial_tower">Memorial Tower</a></dd>
        <dt>Description</dt>
        <dd>Memorial Tower honors those alumni who were killed in World War I.</dd>
      </dl>
    </div>
  </div>
</div>
```
The cornerstone was laid in 1922 and the Tower was dedicated on November 11, 1949.

<dt>Address</dt>
<dd itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
  <span itemprop="streetAddress">2701 Sullivan Drive</span>
  <span itemprop="addressLocality">Raleigh</span>,
  <span itemprop="addressLocality">NC</span> <span itemprop="postalCode">26707</span>
</dd>

<dt>Latitude, Longitude</dt>
<dd itemprop="geo" itemscope itemtype="http://schema.org/GeoCoordinates">
  <span itemprop="latitude">35.786098</span>, <span itemprop="longitude">-78.663498</span>
</dd>

All told there are five Microdata items on the page (ItemPage, Photograph, LandmarksOrHistoricalBuildings, PostalAddress, GeoCoordinates). In part, this snippet now basically says something like this in English:

This page is an ItemPage about a Photograph. The Photograph is about a LandmarksOrHistoricalBuildings. The Itempage has a “name” of “Students jumping in front of Memorial Bell Tower” and an “image” at “http://example.com//images/bell_tower.png.” The Photograph has some “keywords” and “genre.” The LandmarksOrHistoricalBuildings has a “name” of “Memorial Tower” and a “description.” The LandmarksOrHistoricalBuildings has an “address” which is a PostalAddress item (with its own properties), as well as, a “geo” property which is a GeoCoordinates item.

The nested item types could be represented by this image:
You can see the extracted JSON at Live Microdata under the JSON tab.

**itemref**

So far the image is not associated with the Photograph or any child items on the page. That needs to be fixed, because a rich snippet for a Photograph is unlikely to show up without a value for the `image` property. The problem is that the image is not nested within the same `div` where the Photograph is defined.

Valid HTML is particularly important in pages that contain embedded markup. All methods of embedding data within HTML use the structure of the HTML to determine the meaning of the additional markup. ([Choosing an HTML Data Format](http://example.com))

While we have the content on our page relatively well organized to contain our items, our layout and grid system result in the image of the photograph not being nested within the same `div` as the metadata about the photograph. When coding new pages, it is important to think about how to structure the page with both presentation and data in mind. It makes marking up data easier if the properties of a thing are grouped together. While in most cases it will be easier to arrange our markup so that the properties of an item are all within the same block on the page, there are times like this where the style and layout of the page determines the structure. Microdata provides a rather simple mechanism for including properties that are outside of an item's scope.

Microdata uses the `itemref` attribute to make this more convenient.

Note: The itemref attribute is not part of the Microdata data model. It is merely a syntactic construct to aid authors in adding annotations to pages where the data to be annotated does not follow a convenient tree structure. ([Microdata specification itemref attribute](http://example.com))
In our example above the `img` already has an `id` of `main_image` and an `itemprop` with the value `image`. All that we need to do to use that image property for the Photograph is add `itemref="main_image"` to `div#metadata`. The `itemref` adds elements with that id attribute to the queue of locations in the DOM to check for properties for the item created on the same element. Here is the relevant markup:

```html
<img id="main_image" alt="Students jumping in front of Memorial Bell Tower"
     src="/images/bell_tower.png" itemprop="image">
...
<div id="metadata" class="grid_7" itemprop="about" itemscope
     itemtype="http://schema.org/Photograph" itemref="main_image">
    ...
</div>
```

The `div#metadata` which contains all of the information about the photograph now uses four Microdata attributes. The same `itemref` value could be added to the building to associate the page image with that item as well.

### Problems with Rich Snippets

No rich snippet would show up in search results or the Rich Snippets Testing Tool for this example right now. Even though the Microdata is valid and multiple Schema.org items properly marked up, at the time of this writing, Google would not use data from any of the possible items to create a search result snippet. Google currently only supports Rich Snippets for several of the Schema.org types (Applications, Authors, Events, Movie, Music, People, Products, Products with many offers, Recipes, Reviews, TV Series– not all of which are Schema.org types). For each of these item types Google requires that certain properties be present in order for a rich snippet to have a chance of showing up. Even then a rich snippet may only show up for a page if an item is relevant to the users query. But using the Structured Data Linter we get this possible preview for the LandmarksOrHistoricalBuildings item.

```
Memorial Tower
2701 Sullivan Drive, NC Raleigh, 26707
an actual search result may display other content relating to your search terms here.
```

This is exactly the kind of attractive rich snippet we want users to see for digitized resources. The snippet includes the image and address to make a more clickable target. Hopefully the search engines will begin showing snippets for some of the item types cultural heritage
organizations are most likely to be making available. There may also be an opportunity for
cultural heritage organizations to build their own local and cross-institutional search
engines that take more advantage of more detailed Microdata.

**time and Datatypes**

Another piece of data which would be good to add to the Photograph is the date created using the `dateCreated` property with an expected value of `Date`.

The Microdata specification (unlike RDFa and Microformats-2) does not provide a generic mechanism for specifying data types. Instead Microdata relies on a vocabulary to define expected data types. Schema.org has four basic types: Boolean, Date, Number, and Text, but none of them are very well documented.

HTML5 has some new elements to allow inclusion of machine-readable data in HTML and these can be used for the Schema.org datatypes. The `time` and `data` elements have been added to HTML5. The specification of `time` has not been stable. In fact the `time` element was removed from HTML5 with some strong objections and lots of exciting specification drama.

So while the `time` element is in the specification again, the Schema.org and Google documentation add confusion to the matter by using the `meta` element instead when the content is a date. Lots of the examples look like this:

```html
<meta itemprop="startDate" content="2016-04-21T20:00"> 
Thu, 04/21/16 
8:00 p.m. 
```

Using the `meta` and `link` elements within the `body` element of a document is allowed in HTML5 when used with an `itemprop` attribute. These elements can sometimes be useful for Microdata in expressing the meaning of content or a URL which has no usefulness to a human reader. The `meta` element above is used to give the machine readable date near the date visible to the user. It is hidden on the page, so goes against the general recommendation to not use hidden markup for Microdata. Further, the `meta` element does not surround the free text version of the date disassociating them from each other.

The same data as above could be marked up using `<time>` like so:

```html
<time itemprop="startDate" datetime="2016-04-21T20:00"> 
Thu, 04/21/16 
8:00 p.m. 
</time>
```

The `time` element has the correct semantics for the Photograph:
I have made no attempt to handle the approximate date (“circa”). The current processing rules in the specification do not handle many valid ISO8601 dates. As dates and ambiguity about dates is important for describing cultural heritage materials, hopefully the HTML5 processing rules can be adjusted to handle more valid ISO8601 dates. It seems as if the WHATWG has accepted a proposal to support year only dates, which is a start.

**itemid**

Some items have unique identifiers or canonical representations elsewhere on the Web that can be used to link resources together. The Memorial Bell Tower is a unique landmark that could be linked together with other representations of the same place. This linking could help machines to make connections between resources or users to follow their nose to related representations.

In Microdata the *itemid* attribute can be used to associate an item with a globally unique URL identifier, “so that it can be related to other items on pages elsewhere on the Web.” This is the main mechanism by which Microdata natively supports something like Linked Data. When serialized to RDF the *itemid* of an item would become the global identifier used as the subject of RDF assertions. The other URLs used as values throughout Microdata do not provide this linkability, and the assumption is that consumers will have a built-in knowledge of the vocabulary used for item types and properties in order to make sense of the data. Microdata does not adhere to the “follow-your-nose principle, whereby vocabulary authors are encouraged to provide a machine-readable description of classes and properties at the URL used for the class or property,” like RDF does (HTML Data Guide, W3C Editor’s Draft 08 January 2012).

```
| Item types are opaque identifiers, and user agents must not dereference unknown item types, or otherwise deconstruct them, in order to determine how to process items that use them.  
(HTML Microdata: Items) |
```

The meaning of the *itemid* attribute is determined by the vocabulary. Unfortunately, Schema.org does not document any use or support for the *itemid* attribute at this time, though they “strongly encourage the use of itemids.” The semantics of *itemid* in the Schema.org context seems uncertain and overlapping with the consistent use of url property names.

We'll add an *itemid* in any case. The Memorial Bell Tower can be represented by this Freebase URL: [http://www.freebase.com/m/026twjv](http://www.freebase.com/m/026twjv)
Extending Schema.org

Another kind of property a cultural heritage organization might like to add to a landmark or building like the Memorial Tower are the events related to the building. In this case the cornerstone was laid in 1922 and the tower dedicated on November 11, 1949. Other buildings could have events in their history like the dates they were designed or dates of renovations, derived from the drawings and project records. Museums may be interested in various events in the history of a painting including provenance and restorations. History museums and historical societies may also want to refer to various historical events that relate to their exhibits. Each of these institutions may also want to promote various future events like movie screenings, limited time exhibits, and tours. So it may be important to be able to disambiguate whether an event is in the future or of some historical significance.

Schema.org has an Event type defined as: “An event happening at a certain time at a certain location.” That seems broad enough to apply it to either future or historic events. But the message from Google's support page on Rich Snippets for events gives more specific guidance:

The goal of events snippets is to provide users with additional information about specific events not to promote complementary products or services. Event markup can be used to mark up events that occur on specific future dates, such as a musical concert or an art festival.

Google has a different view of the world than cultural heritage organizations and historians. Because of the focus on the future, we may not want to try to mark up historic events, as rich snippets may be unlikely to show up for past events.

Another option for publishing data about this kind of event in the life of an object may be to use the Schema.org Extension Mechanism to try to make it clearer to the search engines that certain types of events are different. As Schema.org is intended as a Web-scale schema, there is no possibility of having it fit every kind of data on the Web. The basic mechanism for extending an item type is to take any Schema.org type URL, add a forward slash to the end, and then add the camel cased name of the extension. So one possibility to handle historic events would be to extend Event:

http://schema.org/Event/HistoricEvent
At least the search engines will understand that these items are some type of Event. If enough other folks use the same extension and the search engines notice, then the search engines may start using the data in a meaningful way. This is one way to grow the schema organically with actual use influencing the vocabulary, though there are those who question this extension mechanism and there are no assurances that extensions will get used. Schema.org lacks a good, public, formal process for sharing extensions and advocating for their inclusion in the vocabulary. Work still needs to be done to build up a clear central location to share extensions or have a community process to work out new extensions.

Dan Brickley has noted that the properties available in the Schema.org types for describing museum objects are a little “sparse” compared to some controlled vocabularies used by museums. For properties there are two options for mixing in a new property for an existing (or extended type). Schema.org prefers page authors to add the new property name as if it were defined by Schema.org. So our rights statement could be given this markup:

<dd itemprop="rights">
  Reproduction and use of this material requires permission from North Carolina State University.
</dd>

The other option provided by the Microdata specification is to add a full URL for the property like this:

<dd itemprop="http://purl.org/dc/elements/1.1/rights">
  Reproduction and use of this material requires permission from North Carolina State University.</dd>

So while anyone can easily extend Schema.org types and properties, there needs to be some community using or consumer understanding the same extensions in a consistent way for the extensions to have any usefulness. Do not expect the search engines to do anything with property names that are not documented on Schema.org.

Another way forward for the cultural heritage sector?

These are just some of the ways in which Schema.org may not fit the needs of cultural heritage organizations very well. Other communities have already voiced their concerns that their domains are not adequately specified.

For instance, the rNews standard of the International Press and Telecommunications Council was in large part added to Schema.org. This was the first industry organization to work with the Schema.org partners to publish an expansion of Schema.org to make it more robust and useful in a particular domain. The adoption of much of rNews has resulted in news and publishing-related types being more expressive and better meeting the needs of news organizations.
Another change to the schema was the result of a collaboration between Schema.org and the United States Office of Science and Technology Policy to add support for job postings. These additions were immediately put to use to create a job search widget for government Web sites to highlight job listings from employers who commit to hiring veterans. (See the Veterans Job Bank.)

So this kind of partnership with domain experts seems like a way forward for other groups. Libraries, museums, archives, and other cultural heritage organizations could start to enumerate the places where the vocabulary could be modified to better fit their data and use cases. There has been some suggestion for future discussions with the cultural sector to this end. One suggested location for this kind of activity is the W3C wiki, which points to the simplicity of the submission for job postings. Another place this work is being done is in the Schema.org Alignment Task Group of the Dublin Core Metadata Initiative, which has a multi-decade history of working internationally with Web metadata in the cultural heritage sector.

Some of this work may already be available. Work is ongoing to map other vocabularies to Schema.org. This provides a simpler way for organizations to expose their data through Microdata and Schema.org while still maintaining their data in their current schema. In each of these mappings there are certainly some areas where there is not overlap, so there is potential for expanding Schema.org in those directions.

Conclusion

Microdata and Schema.org provide a relatively simple way for libraries, archives, and museums to begin to expose their data in new ways and make their collections more discoverable and useful. The semantic markup of data embedded in HTML is a rapidly changing area, and much of what is written here is likely to change. While this is challenging for implementers, it also provides a chance for cultural heritage organizations to enter the conversation, and build new tools that actually use the newly available structured data. There is a huge opportunity to have an impact on these technologies to improve the discoverability and use of our collections and services.

Appendix: Resources

Examples from Cultural Heritage Organizations

- **NCSU Libraries’ Digital Collections: Rare and Unique Materials** The example in this tutorial is based on this site.
- **Indianapolis Museum of Art** Uses CreativeWork and extends the type by adding the following properties: accessionNumber, collection, copyright, creditLine, dimensions, materials.
- **Biodiversity Heritage Library** Adds an “OCLC” property for a Book.
- **Sudoc French academic union catalogue** Seems to only show the Microdata representation to crawlers? [more information](#)
- **Sindice Search** This search can be adjusted to find a specific schema.org type (class here). Faceting by the Microdata does not always seem to find sites that predominantly use Microdata rather than RDFa.

## Tools

These are tools which I have regularly used.

- **Rich Snippets Testing Tool** Note that while it may not currently show a *rich* snippet example for every schema.org type, you can use the data at the bottom of the page to insure that your Microdata is being parsed as you intended. The format here breaks every item out and shows references between them in a flat way.
- **Structured Data Linter** The best feature of this tool is the way that it displays your nested Microdata as nested tables, making it easy to spot problems. If the Rich Snippets Testing Tool doesn't show a rich snippet for your content, this is a good alternative to see what your snippets *might* look like. The snippets here do not cover every type either, but they cover a few different types from what the Rich Snippets Testing Tool does, for instance it will show images for more types. The code is open source, so you can run your own instance to be able to check your syntax while you are in development. This is written by folks who have been part of the conversations around Web vocabularies and structured data in HTML.
- **Live Microdata**
  A good open-source tool for testing snippets of HTML marked up with Microdata. The [MicrodataJS source code](#) allows you to implement the Microdata DOM API on your own site, similar to how this tutorial outputs the JSON from parsing the page.
- **HTML5 Living Validator**
- [schema.rdfs.org list of tools and libraries](#)

## Mappings

There are many efforts to map Schema.org to other vocabularies. If you maintain your metadata in one of these vocabularies, you can use these to help expose your data in a way that the search engines understand.

## Other Tutorials

- **Google Rich Snippets Videos**: This series of short tutorial videos uses Microdata and Schema.org.
• Getting started with schema.org The Schema.org examples have been reported to include some bugs.

• HTML Data Guide: This guide helps producers and consumers determine which structured data syntax to use. It covers Microformats, RDFa, and Microdata, and is highly recommended.

• Why rNews? is a good tutorial on how organizations often have good structured data that when made accessible through HTML loses its meaning.

• Dive Into HTML5: “Distributed,” “Extensibility,” And Other Fancy Words

• Spoonfeeding Library Data to Search Engines

• Extending microdata vocabularies

Discussions

About the author

Jason Ronallo is Associate Head of Digital Library Initiatives, NCSU Libraries. He also writes on his blog Preliminary Inventory of Digital Collections.