ODF and Web Mashups

Basic techniques

Rob Weir, IBM
robert_weir@us.ibm.com

2009-11-05
16:15

Agenda

- Why it is hard to use ODF in a web app
- Two techniques for accessing ODF on the web
  - Atom
  - JSON
- A couple of demos
- Some areas for exploration.

- Note: although the examples give use ODFDOM, the same techniques could be used with other ODF libraries on other languages.

How ODF Relates to the Web

- Documents stored on the web, retrieved with HTTP.
- Launch platform editor
- Browser Plugins to view documents
- HTML convertors (Stellent)
- Full text indexing (Google)

- But none of these techniques allow sharing of data from ODF documents with web applications...

So what is the problem?

- ODF is XML-based, but not XML
- ODF is a ZIP-archive containing multiple XML files and associated resources (images, etc.).
- Web client's cannot get inside the ZIP given functionality provided by HTML/CSS/ JavaScript, etc.
- Could get around this with Java applets or ActiveX, but that has security implications and is not as portable.

Two Approaches

- Convert ODF to Atom
- Convert ODF to JSON

- Small, fast, on-the-fly transformations from ODF to standard formats that are broadly consumable on the web.

Atom Syndication Format

- Created in 2003, goal was:
  - “100% vendor neutral,
  - implemented by everybody,
  - freely extensible by anybody, and
  - cleanly and thoroughly specified.”

- IETF RFC 4287 (2005)
- Successor/rival of RSS
- Generally used for blog, podcast and similar feeds.
- But has some broader uses as well.

Basic Structure of Atom

```
<?xml version="1.0" encoding="utf-8"?>

<feed xmlns="http://www.w3.org/2005/Atom">

        <title>Example Feed</title>
        <subtitle>A subtitle.</subtitle>
        <link href="http://example.org/feed/" rel="self" />
        <link href="http://example.org/" />
        <id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>
        <updated>2003-12-13T18:30:02Z</updated>
        <author>
                <name>John Doe</name>
                <email>johndoe@example.com</email>
        </author>

        <entry>
                <title>Atom-Powered Robots Run Amok</title>
                <link href="http://example.org/2003/12/13/atom03" />
                <link rel="alternate" type="text/html" href="http://example.org/2003/12/13/atom03.html"/>
                <link rel="edit" href="http://example.org/2003/12/13/atom03/edit"/>
                <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
                <updated>2003-12-13T18:30:02Z</updated>
                <summary>Some text.</summary>
        </entry>

</feed>
```

Mapping ODF to Atom

- dc:creator → name
- dc:date → updated
- dc:title → title
- dc:description → summary

- “dc” refers to Dublin Core Element Set

Example: Metadata Aggregator

- Point to a directory full of ODF docs
- Use ODFDOM to extract the metadata
- Generate an Atom feed from this information

Let's look at the code and have a demo

Enhancements and Uses

- Run on server, via a cron job
- Atom feed via a servlet call: http://foo.com/odf2atom?base=dir
- Read ODF documents from a document repository
- Apply same technique to individual documents, to publish individual slides in a presentation, or rows in a spreadsheet.
- Send updates whenever new document is uploaded to your web site
- Use TwitterFeed to publish updates to Twitter
- Use feed in IBM Mashup Center or in Yahoo Pipes
- Scan incoming attachments, extract them and add to personal feed of incoming documents
- Can help with widespread problem of two metadata views, that are unsynched, the intrinsic metadata in the document versus the metadata tracked by the content manager.

JSON JavaScript Object Notation

- IETF 4627 (2006)
- Alternative to XML for an interchange data format.
- Is a subset of JavaScript's data declaration syntax.
- Can be very easily consumed by JavaScript applications via call to eval(), but can also be used by other languages, via libraries.

JSON Example

```
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumbers": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

Example: Spreadsheet Publisher

- Start with a spreadsheet with rows of records.
- First row is treated as the header row and used as labels.
- We then iterate over remaining rows, writing a subobject for each row.
- End result is a JSON object representing the spreadsheet.

Let's look at the code and have a demo

Security Note

- “Same origin” security restrictions require that the HTML page using the JSON object be in the same domain as the domain of the server hosting the JSON object.
- Can get around this somewhat with frames, but this then gets you into cross-frame scripting issues.

Enhancements and Uses

- Run on server, via a cron job
- JSON == a servlet call: http://foo.com/json?doc=foo.ods
- Could be used to aggregate data across documents

- Great for mashups
  - Identify a column as a location value, use Google Maps API to geocode and place marker on a map.

- Can we go in the other direction?
  - ODS → JSON → Interactive modifications → JSON → ODS
  - Would be a killer feature. Let's do it!

Thank you for your attention.

Any questions?