Fedora and Django at Memorial Sloan-Kettering Cancer Center

Here at Memorial Sloan-Kettering Cancer Center (MSKCC), we are using Fedora, at least initially, as an image repository. Fedora's flexibility in associating multiple sources and kinds of metadata with an object have been a major factor in the increasing adoption within MSKCC. We've been using Fedora to increase the availability of assets, making them available for re-use throughout the center, rather than keeping these assets confined within the originating department. Fedora allows us to overcome the natural tendency of departments to create inaccessible silos of information.

Our initial use has been in the Department of Surgery, where images created by one service are made available to clinicians in other services, with security and patient privacy as foremost drivers. Our second use is within the Public Affairs office, which has a large set of images from their work on our website and print publications, which they are making available to our internal community. Our third use, under development, is the High Throughput Screening (HTS) Core Facility, a service which offers researchers the ability to screen large numbers of chemical compounds for potential therapeutic uses very quickly. For HTS, we will be ingesting approximately 200GB of images per day, and making those images and their analyses available to the researchers. Previously the researchers would need to come to the HTS facility to obtain the screening results.

My presentation will cover how MSKCC became aware of Fedora, and our initial set of use cases, including Surgery, the Medical Library, and HTS, Public Affairs, Biomedical Communications and Strategic Planning. Within the Department of Surgery, we used Fedora 2.2.3 with Muradora for our front-end. We contracted for consulting expertise with MediaShelf for project planning and UI design, and with 2DC/MELCOE for additional functionality within Muradora. Due to that initial engagement, we became comfortable with Fedora itself, and aware of the technologies surrounding the project. We also became aware that we lacked the internal expertise necessary to mold Muradora to our satisfaction.

Our second use of Fedora is within the Public Affairs Department, and this time we chose to build our own web user interface. Due to my success with Python and Django on Synapse, a bibliographic database for the Medical Library (http://synapse.mskcc.org/), I was able to convince our management to take a chance on Django for the web UI for Public Affairs. Initially there was significant resistance to using Python, but management asked me how long it would take to implement a particular feature in Muradora using Java, and I gave them an estimate. I then told them I could recreate the entire Muradora UI, plus the added feature and a few more feature requests in the same amount of time, and they were sold on the idea of using an agile language and framework.

Public Affairs has a couple of terabytes of image assets in various formats -- JPG, TIF, EPS, PNG -- which they've collected over the years of adding content to our main website (http://www.mskcc.org/) and of printing various publications. Their main use
case is to make these assets available to the rest of the MSKCC community. Their secondary use case is to improve their own ability to find images for re-use in other situations.

Our active constraints were a three month deadline for the project, and that it must be based on the design we'd already put in production for Surgery. We need not limit ourselves to just recreating the Surgery UI.

We found that using Django and Python provided several advantages, and a few disadvantages. One major advantage is the ecosystem of available applications for Django. There is a very strong emphasis within the Django community that code contributions to the community should be reusable, self-contained chunks of code that address a very specific need. The end result is that there are thousands of such tiny applications that provide units of functionality that can be easily composed into the needs of a particular site or web application.

Another major advantage of using Python is inherent in the language. Python is designed to maximize programmer productivity. By eliminating as much unnecessary verbiage as possible, and by emphasizing the readability of code, Python promotes rapid development. Since Python is easy to read and easy to learn, basing a project on Python makes it relatively simple to bring on new programmers. Since Django is very Pythonic in its approach to code, it is easy for someone versed in Python to pick up quickly. Furthermore, Django is a full-stack web framework, meaning that nearly all of the boilerplate code to make a web application is already written for you. The result is that the programmer need spend time only on the domain-specific code.

There are a few potential disadvantages to using Python and Django. One is the dearth of Python programmers. In our case, we found that training a programmer in Python was quick enough to obviate the problem.

Another potential problem is that Django is designed to work with a relational database, which Fedora is not. While Fedora makes use of a relational database to provide certain functionality, that use is limited. I worked around this problem by writing a client library to access Fedora, and writing a custom Django management command that inspects the repository and creates Django model objects and stores them in the database.

The third potential problem we faced is that of Python's performance. Python's runtime performance is benchmarked at about 5-10 times slower than Java. In practice, we found this problem to be entirely theoretical. Our Django UI responds immediately, while fetching images from Fedora often takes a measurable amount of time. The light weight of the Django UI compares extremely favorably to the heaviness and latency of Surgery's Muradora stack.

For Public Affairs I wrote a custom ingest script, which uses various imaging libraries to extract EXIF and XMP metadata from the various image files as they are ingested.
Public Affairs spent a fair amount of time beforehand tagging each image with metadata using Adobe Bridge. The ingest script creates the necessary FOXML and Dublin Core, and derives screensize, thumbnail and watermarked images from the original image, adding them as datastreams in the object. The script is easily extensible to create additional datastreams as needed, and can handle different kinds of objects by adding additional classes defining the object structure.

For the Django web application, we are using LDAP authentication against Active Directory. We are also performing role-based access control based on the organizational unit (OU) in Active Directory. We define roles for system administrators and for object administrators, where object administrators are Public Affairs staff, who have permissions to grant access to objects to individual users.

We are using django-granular-permissions, a reusable application, to provide per-object permissions checking for access to the original datastream. If users do not have the proper permissions, they are allowed to view only the watermarked datastream. If Public Affairs gives them access rights to an image, they can then view the non-watermarked screensize version and can download the original datastream.

For search, we chose Solr. Unlike the Solr integration provided by Muradora, we are not using Solr to index Fedora directly. Instead, we chose to use solango, a reusable application that integrates Solr with Django. Solango indexes Django's underlying PostgreSQL database instead, and is extremely easy to use and customize.

Additional features include a new browse view powered by AJAX calls to Django, tagging with controlled vocabularies, and a download cart. We make extensive use of jQuery to make these features snappy and easy to use within the browser.

My presentation will conclude with a demo and a quick question-and-answer session. Given additional time, I can expand the demo and go deeper into various features as audience interest dictates.