Review of Existing Dspace REST API Frameworks

Existing REST Modules

Project	Read-only?	Unit tests?	DB specific?	DSpace version?	Technology	Notes
https://github.com/hedtek/dspace-rest	Yes	Yes	Yes, Postgres for testing	Yes, pegged at 1.8.2	Sakai	Was developed by a consultancy for a client who ended up not using it. The POM is pegged to DSpace 1.8.2, but I haven't had issues pegging it to 3.0 May not be Oracle-friendly.
https://github.com/wijiti/dspace-rest-api	No	No	No	No	Sakai	Uses the Sakai Project's Entity Bus REST framework. Still very rough around the edges (e.g., search provider comments out support for user supplied parameters).
https://github.com/DSpace/dspace-rest	No	No			Sakai	Was a Google summer of code project. Really just a proof of concept, not production ready.

Sakai

All of the existing DSpace REST API frameworks use the Sakai Entity-Bus. Sakai is an architecture that shields services from their implementation.

Wiring an endpoint to SAKAI looks like:

```
this.entityProviders = new Vector<AbstractBaseProvider>();
this.entityProviders.add(new BitstreamProvider(entityProviderManager));
this.entityProviders.add(new CommunitiesProvider(entityProviderManager));
this.entityProviders.add(new CollectionsProvider(entityProviderManager));
this.entityProviders.add(new ItemsProvider(entityProviderManager));
```

For CollectionsProvider, it then sets its endpoint prefix:

```
public String getEntityPrefix() {
    return "collections";
 }
```

And then you have to know that getEntity(...) is to get a single instance of this resource, and getEntities(...) is to get a list of this resource. For the most part it has entities/providers that shield you from the implementation, and then it also has to implement a bunch of business logic to fetch resources, get their relations, and other CRUD, that the API doesn't care about, but has no alternative, other than to do it itself.

JAX-RS 1

JAVA has a standard for building REST API's, with JSR-311, aka JAX-RS1, aka JAVA API for RESTful Web Services. The reference implementation of JAX-RS1 is JERSEY, there are other implementations that are probably more pleasant. If you use JAX-RS1, then you have a large amount of tools, guides, a community of users using this that you can build off of. JAX-RS2 has come out, and its approved, giving even more features for JAVA web services.

Below is some sample JERSEY code of how you wire up resources, choose to serialize to HTML, JSON, or XML. And between display single-entity vs display list-of-entities.

```
@Path("/collections")
public class CollectionsResource {

    @GET
    @Path("/")
    @Produces(MediaType.TEXT_HTML)
    public String listHTML() {...}


    @GET
    @Path("/")
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    public org.dspace.rest.common.Collection[] list(@QueryParam("expand") String expand) {...}
```

```
    @GET
    @Path("/{collection_id}")
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    public org.dspace.rest.common.Collection getCollection(@PathParam("collection_id") Integer collection_id,
@QueryParam("expand") String expand) {...}
```

There was no central ProviderRegistry that you have to declare your path. Your free to use @annotations to get your code to respond to requests, and there are helpful parameter helpers to extract parameters into Java variables.

Some analyses between wijiti and hedtek APIs

- They both are based on the same underlying technology (Sakai entitybus) - which in my opinion makes it pretty hard to do any developments with them.
- They both use database ids rather than handles -- we (Jorum) changed that locally so that we can have either db ids or handles
- Hedtek does not have any kind of authentication, which would not be suitable for places with restricted access. Wijiti seems to want a user name password for everything But username passwords are transmitted in the header or as part of the request. What about anonymous use of the GET parts?
- For people who have to report on the usage of their repository, you have to note that neither API submits usage stats to solr (or anywhere else)
- For the Hedtek API not all the endpoints are implemented which are 'advertised' (i.e. search and I have only looked at the once we wanted to use)
- wijiti seems to return wrong error messages and (or) does not work as the documentation suggests:
    - http://.../rest/communities.xml?user=xxx@xxx.de&pass=yyy
      works, but
    - http://../rest/communities/2.xml?user=xxx@xxx.de&pass=yyy
      returns a 403 Bad username or password -- username password same as on first request and community with id 2 exists
- wijiti itself states that "PLEASE NOTE: This DSpace REST API implementation and its associated documentation is a work-in-progress. Some documentation is still missing and some REST API endpoints are either not completed or have not been implemented. It is recommended you do not use this API in production environments unless fully tested."