DSpace Backend as One Webapp

The initial version of this feature has been merged into "master" and will be released in 7.0 final.

- JIRA Ticket: Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- Initial PR: htt

Background

In the past we've had several discussions/attempts at making DSpace easier to install (e.g. Installer Prototype, Installer Brainstorms).

However, with DSpace 7.0, the new REST API (dspace-spring-rest module) is now a Spring Boot web application. Spring Boot's entire purpose is to provide tools to make easy-to-install web applications.

This proposal is to begin to utilize the tools in Spring Boot to turn DSpace's "backend" webapps (REST API, OAI-PMH, SWORD, SWORDv2, RDF) into a **si ngle, easy to install webapp**. *At this time, I'm concentrating only on the backend webapps, as they are all Java webapps. So, it does not include the new Angular UI. I'm also specifically excluding the **old** REST API (v4-6), as it is deprecated in DSpace 7 (and will be removed in DSpace 8).*

How would we achieve this?

All of the separate, backend web applications (namely SWORDv1, SWORDv2, OAI-PMH, and RDF) would be merged into a single Spring Boot web application (i.e. the REST API v7 webapp). This can be achieved as follows:

1. Keep each as a separate Maven module
2. Turn each Maven module into a **JAR** (i.e. update each module's POM)
3. Add a new dependency on Spring Boot (spring-boot-starter-web), to allow each module to use Spring Boot annotations/APIs.
4. Replace each module's `web.xml` webapp configuration with a Spring `@Configuration` class (defining several @Beans). This is a direct replacement for the web.xml concept.
    a. For example, in PR#2265, SWORDWebConfig.java replaces SWORDv1's web.xml
        i. Notice how the class defines a series of @Beans, all of which can be enabled/disabled via DSpace configuration (in this case a "sword-server.enabled" property).
        ii. Additionally, this class can pull in other DSpace configuration/settings to allow for path customization (see the "swordPath" variable in this example, which reads from the "sword-server.path" configuration)
5. Remove the Overlay folder for the module (from `[dspace-src]/dspace/modules/`). The Maven Overlay concept is only supported by WARs, and therefore, all Overlays would be applied to the Spring Boot webapp.

The implementation of this can be found in the initial PR: https://github.com/DSpace/DSpace/pull/2265

What is the benefit?

There are several known benefits:

- Deploying the DSpace backend is easier, as it's a single web application within Tomcat
- Server resource usage is minimized, as all modules share the same Java classes/libraries
    - Only one initialization of Hibernate (instead of one per webapp)
    - Only one initialization of Flyway (instead of one per webapp)
- Arguably, enabling/disabling various "modules" (SWORD, OAI, RDF) is clearer, as it's all based on configuration (and modules you want enabled can be specified in your local.cfg file).
    - *KEEP IN MIND, this same "module" concept could be extended to DSpace third-party modules.* In this case, all DSpace "modules" are JARs (that optionally use Spring annotations/APIs to add subpaths/behavior) backed by DSpace configuration (which can be used to easily enable/disable/configure the module). So, installing a new module could be simply dropping in a new JAR (or adding a new dependency) and adding configuration to your local.cfg.
- (In the future, post-7.0) We also could consider using Spring Boot's capability to **embed Tomcat** directly, so that deployment is as simple as running the application.
    - However, this would not be available in DSpace 7.0, and would require more discussion on whether it's the right direction for DSpace.

Other questions to consider

Would I still be able to disable specific features? For example, what if I don't need or want to install SWORD

Yes, it's now a simple configuration.

All of these modules come out-of-the-box within DSpace. You can choose whether to enable or disable specific ones in your local.cfg.

For example, enabling SWORDv1 in PR#2265 just requires adding this to your local.cfg:

```
sword-server.enabled = true
```

Commenting out that setting or setting it to "false" will disable SWORDv1.

Can I still overlay specific classes within a module (e.g. in OAI-PMH or SWORD)?

While this still needs to be tested out further, you should be able to simply move those overlays to overlay the main DSpace Spring Boot web application. So, overlays should still be possible, but they would all now exist in a single place (i.e. you'd always be overlaying the "dspace-spring-rest" webapp, as that's the webapp that loads all other modules)

Can I still change the URL or path of OAI-PMH or SWORD?

Yes, it's now a simple configuration.

While each module has a default path, you can easily override it within your local.cfg.

For example, in PR#2265, the SWORDv1 module defaults to using the "sword" path (e.g. http://localhost:8080/spring-rest/sword/). However, if you wanted to change it to use the "swordv1" path, you can simply set the following in your local.cfg:

```
sword-server.path = swordv1
```

Keep in mind that this path will always be a **subpath of the root webapp (i.e. spring-rest)**. So, you need to be careful not to set two modules to use the same subpath (e.g. setting both SWORDv1 and SWORDv2 to use "sword" won't work). Additionally, the "api" path is **reserved**, as it's used by the REST API (e.g. http://localhost:8080/spring-rest/api/).

However, if you install the root webapp at the root path, then this behavior is similar to current DSpace (v6). For example, if the "spring-rest" WAR is accessible at `https://mydspace.edu/`, then all modules would be a subpath of that:

- `https://mydspace.edu/api/` (REST API v7)
- `https://mydspace.edu/sword/` (SWORDv1)
- `https://mydspace.edu/oai/` (OAI-PMH)

If the backend is all one webapp, what should this webapp be called?

This was discussed in a DSpace 7 Working Group meeting, and it was decided that the single webapp will be named "dspace-server", as it comprises the primary "server-side" components of DSpace 7.

(NOTE: Technically, there is also a server-side portion to the Angular UI to precompile UI pages in support of SEO, etc. However, as the Angular UI is primarily a client side user interface, we still feel this single webapp is most appropriately called the "server" webapp.)