Architecture

Overview

The DSpace system is organized into three layers, each of which consists of a number of components.

- **Application Layer** - All external/public facing interfaces/tools. These include the Web User Interface, REST API, OAI-PMH, RDF, and SWORD (v and v2) interfaces. Also includes the Command Line interface, and various tools that can be used to import/export data to/from DSpace.
- **Business Logic Layer** - Primarily the Java API layer ([dspace-source]/dspace-api and dspace-services), which provides the core business logic for all the various application interfaces.
- **Storage Layer** - A subset of the dspace-api (*org.dspace.storage.* classes)* whose role is to manage all content storage (metadata, relationships, bitstreams) for all business layer objects. This layer is provides access to a relational database (Postgres or Oracle usually) via Hibernate ORM & using FlywayDB for migrations/updates. It also defines a custom BitStoreService for storing files (bitstreams) via storage plugins (currently supporting filesystem storage or Amazon S3 storage).

DSpace System Architecture

The storage layer is responsible for physical storage of metadata and content. The business logic layer deals with managing the content of the archive, users of the archive (e-people), authorization, and workflow. The application layer contains components that communicate with the world outside of the individual DSpace installation, for example the Web user interface and the Open Archives Initiative protocol for metadata harvesting service.

Each layer only invokes the layer below it; the application layer may not use the storage layer directly, for example. Each component in the storage and business logic layers has a defined public API. The union of the APIs of those components are referred to as the Storage API (in the case of the storage layer) and the DSpace Java API (in the case of the business logic layer), and the DSpace REST API (in the case of the application layer). In the Application Layer, it's worth noting that the Web User Interface only accesses DSpace via the REST API.

It is important to note that each layer is *trusted*. Although the logic for *authorising actions* is in the business logic layer, the system relies on individual applications in the application layer to correctly and securely *authenticate* e-people. If a 'hostile' or insecure application were allowed to invoke the Java API directly, it could very easily perform actions as any e-person in the system.

The reason for this design choice is that authentication methods will vary widely between different applications, so it makes sense to leave the logic and responsibility for that in these applications.

The source code is organized to cohere very strictly to this three-layer architecture.

The storage and business logic layer APIs are extensively documented with Javadoc-style comments. Generate the HTML version of these by entering the [dspace-source]/dspace directory and running:

```
mvn javadoc:javadoc
```

The resulting documentation will be at *[dspace-source]dspace-api/target/site/apidocs/index.html*. The package-level documentation of each package usually contains an overview of the package and some example usage. This information is not repeated in this architecture document; this and the Javadoc APIs are intended to be used in parallel.

The REST API provides not only JavaDocs, but also a public contract. See REST API.

Each layer is described in a separate section:

- Storage Layer
  - RDBMS
  - Bitstream Store
- Business Logic Layer
  - Core Classes
  - Content Management API
  - Workflow System
  - Administration Toolkit
  - E-person/Group Manager
  - Authorisation
  - Handle Manager/Handle Plugin
  - Search
  - Browse API
  - History Recorder
  - Checksum Checker
- Application Layer
  - Web User Interface
  - OAI-PMH Data Provider
  - Item Importer and Exporter
  - Transferring Items Between DSpace Instances
  - Registration
  - METS Tools
  - Media Filters

- Sub-Community Management