APIs and Services

Overview

Documentation on the REST APIs.

REST APIs

Representing business logic of the application, the REST APIs comprise the set of functionality needed to completely support the client user interface. They are well organized-functionally and do not ever refer to each other. Thus, each service can stand on its own as a collection of functionality to suport some major feature set of the application (e.g. "refset editing", or "release process").

For each REST API there is also a fully-functional client that can be used for integration testing against that service.

The REST APIs were designed with some general principles in mind

- Each REST method takes an authToken that can be authorized by the security service
- Each REST method validates the token and then authorizes the user's action based on role.
  - Some methods consider "application role"
  - some methods consider "project specific role"
- Error handling is consistently managed at the REST layer
  - Print a stack trace
  - Send an email to an administrator list (unless its a known condition that does not require notification, such as a failure to authenticate)
  - Return an HTTPS 401 or 500 error with an appropriate error message
- Each REST method opens one or more internal JPA services, performs their operations, and then closes the resources at the end.
- Each REST method is designed to provide information to clients at a level that generally does not require chained callbacks. In some cases model objects have "transient" fields that are not persisted but are populated by the REST layer to ensure the client has enough information from a single call to avoid other calls.
- Generally getter methods use GET
- Generally write methods use PUT, POST, or DELETE.
- "Get" methods take identifiers and return single objects.
- "Find" methods take a "pfs" parameter, handle paging/filtering/sorting and return a corresponding List object that has a "total count" attribute.
- Lists are wrapped classes designed to persist nicely into JSON for a Javascript client (e.g. see ConceptListJpa).
- Application paging is done on the server side. The javascript requests a page of content that includes the total count (so the number of pages can be easily computed). A request for the next page always makes a server callback.

Internal APIs

Rest APIs are backed by an internal layer of APIs that are "Jpa-enabled". They actually manage connections to the database, transactions and all the details of adding, updating, and removing objects.
While the REST APIs represent the "business logic layer" of the application, these are the services that actually directly perform the unit operations needed to make the application function. There is a relative 1-1 correspondence between many of the REST layer APIs and the Jpa layer APIs, but not exactly where REST calls need to be compositions of more complex logic (such as cloning a refset).
Internal APIs follow a hierarchy, so that successive services add functionality to earlier ones. For example a "translation service" is capable of also handling all calls related to a refset. Thus, for most REST API calls, only a single service needs to be opened (in addition to the security service) in order to fully handle the call. This goes a long way to reducing complex transaction and visibility issues when interacting with the database.

Service APIs

Following are individual service APIs (both internal and REST):

- Security Service
- Project Service
- Release Service
- Refset Service
- Translation Service
- Workflow Service
- Validation Service

References/Links

- n/a