Service based API

Restructuring the DSpace API

www.atmire.com
OVERVIEW

- Introduction
- How it works
- What it does
- What it doesn't do
- Implications
- Current state
INTRODUCTION

- The DSpace API is in need of a restructure
- No structural changes were made since the beginning of DSpace
- Changes were focussed on plugins, UI, features, maven, …
- But DatabaseManager, Context, Item, Bitstream, … classes still have a very similar signature compared to the earliest DSpace releases
INTRODUCTION

• New design: split the API into three layers
  
  • Service layer: This layer would be our top layer, will be fully public and used by the CLI, JSPUI & XMLUI, ...

  • Data access layer: It contains no business logic and it's sole responsibility is to provide database access for each table (CRUD (create/retrieve/update/delete)). This layer is not exposed to the CLI, JSPUI & XMLUI, ...

  • Database objects layer: Each object in this layer is a single database table. This layer is exposed to the CLI, JSPUI & XMLUI so it can use its getters/sets
INTRODUCTION

- New design: replace all custom database queries by Hibernate
- DatabaseManager is completely dated
- Hibernate can resolve all database specific quirks
- Avoid any database queries outside the Data access layer
HOW IT WORKS

Service layer

MetadataFieldService

MetadataFieldServiceImpl

Database Access layer

MetadataFieldDAO

MetadataFieldDAOImpl

Database Object Layer

MetadataField

<<Interface>>

<<Interface>>

<<Interface>>

<<Interface>>

Public

Internal

«Uses»

«Uses»

«Uses»
HOW IT WORKS: SERVICE LAYER

• Consists entirely out of interfaces which will receive their implementation from spring configuration

• Business logic block services
  • Replaces DSpace Manager classes
  • Contains business logic for other classes
HOW IT WORKS: DATA ACCESS LAYER

- Database access layer is only accessed from service layer
- Consists entirely out of interfaces which will receive their implementation from spring configuration
- Database based services
  - Interact with the database
  - Business logic methods for the database objects
- Hibernate is chosen JPA implementation, but can be replaced easily
HOW IT WORKS: DATABASE OBJECTS LAYER

- Each object corresponds to a single database table
- No business logic
- No database queries
- Only getters & setters and JPA annotations to link the table & columns to the variables
WHAT IT DOES

• Separation of concerns: Clear split between “business logic”, “database access” & “database representation” of an object

• Easier API adjustments: Easier adjustments of DSpace api components (overwrite/add methods without altering existing classes)

• Developers’ Productivity: Once you understand the hibernate principles, linking objects & writing queries requires considerably less time. No need to write all sql queries
WHAT IT DOES

- **Modularization support**: Easily replace a service & put another in place without altering any existing code.

- **Effective cross database portability**: No need to write “if postgres query X, if oracle query Y” anymore, hibernate takes care of all of this.

- **Database access flexibility**: Easy replacement of the database layer (if hibernate ever becomes obsolete, just reimplement the DAO interfaces).
WHAT IT DOES

- Improved caching mechanism: Hibernate auto caches queries (session based, configurable application wide). This will replace the caching in the Context

- Performance: Hibernate offers lazy loading to only retrieve certain linked objects at the moment they are requested. This can replace retrieving the bundle, bitstreams, bitstreamformat, file extensions, ... at once
WHAT IT DOESN'T DO

- **Refactor state-full classes**: Only stateless classes have been refactored into services. API classes containing a constructor with arguments which are required for the processing have not yet been turned into services.

- **Make modules from all features**: It becomes much easier to replace certain functionality, but not everything has been modularized in such a manner.

- **Change the DSpace business logic**: All concepts are unchanged, the item, bundle, bitstream, ... concepts, the command line tools, ... are all still present.
IMPLICATIONS

Changes made to the DSpace API:

- **Triple layer api** (for database objects): Every database object class representation has been split into multiple classes (this implies every class in DSpace using these objects will need to be adjusted)

- **Service class** that consists of all business logic (consists of an interface & an a default implementation)

- **Database Access Object** class that contains all our database access calls (consists of an interface & a default hibernate powered implementation)

- **Database Object** class that contains all the setters & getters for the database object
IMPLICATIONS

• **Service based managers**: Each static “manager” (e.g. WorkflowManager, AuthorizeManager, ...) has been split into an interface and a default implementation, making it easier to make local changes & even to replace a "manager". The static methods have all been dropped.

• **DSpace Object identifier change**: All DSpace objects main identifier has been changed from an integer to a UUID. These UUID's are stored in a main DSpaceObject tabel and the objects implementing it will reference this table (Hibernate doesn't support the "type + id" references).

• **Site is now a DSpaceObject**: It will now be possible to add metadata to the site object (but there is currently no code that makes use of this functionality).
IMPLICATIONS

- **Bundle Bitstream linking**: Since the Bundle2Bitstream table contains an additional column order, a new object BundleBitstream was created. When requesting all bitstreams from a bundle, "BundleBitstream" objects will be returned (containing the bitstream).

- **Context no longer contains database connection**: Database connection is bound to a thread, at the moment the context is still passed along to every method (because it does contain other details).

- **Database browse removed**: Hibernate does not easily support "dynamic" database tables, therefore only discovery based browse is supported.

- **Intermittent context commit no longer allowed**: When you commit a DB connection, all objects will be flushed from hibernate. The context.commit() method is gone, the database connection can only be completed.
CURRENT STATE

- dspace-api module:
  - completely refactored to work with the new service based api
  - code compiles
  - The unit test compile, and 99% succeeds
  - There is a complete list of all class/method modifications for each class (this will make the porting a lot easier).
CURRENT STATE

Next actions:

- 6 unit tests are still failing, this is still work in progress
- Migration script is only present for in memory DB and PostgreSQL
- Add DSpace license to all new files
- Cleanup some of the comments
- Finish the XMLUI work (flowscript)
- Adjust the OAI, SWORD, JSPUI, … to use the new DSpace api (This is the majority of the remaining work, few weeks of work)
- Create tutorials on how to port/adjust/create code in the new api
- Automate the deployment: make a local deployment as easy as other DSpace versions
READ MORE

• https://wiki.duraspace.org/pages/viewpage.action?title=DSpace+Service+based+api&spaceKey=DSPACE
Thank you

www.atmire.com