INTERNATIONAL VIRTUAL OBSERVATORY ALLIANCE
US National Virtual Observatory

The DALServer Service Framework

D. Tody (NVO, NRAO)

The DALServer Service Framework

- Purpose and Scope
  - Provides reference implementation of the DAL services
  - Production service framework for community use
    - Especially important for 2ndGen VO technology
- Common Implementation
  - Family of DAL services share a largely common implementation
    - GDS, svc i/f, query response, VOTable, GWS, etc. largely or entirely common
- Software Features
  - Core DALServer classes provide generic service implementations
    - These can be used directly in many cases, eg SIA with table
    - Subclassing used in more complex cases
  - Logical service functionality and transport are separated
    - Service code is transport independent
  - Service implementations are all data model based
    - Service code writes to DM; framework handles serializations
  - Data model is machine readable; keyword factory

Current Status

- Has been in use for 2 years or so now.
  - First appeared as part of SSA development as ref impl
  - Both ready to use Java WAR as well as source code
- Current Service Classes Complement
  - SSA, SIAV1, SCS, SLAP
  - SLAP implementation used for Splatalogue (ALMA)
- Near Term Plans
  - Reference implementations for TAP (ADQL+PQL), SIAV2
    - Once we have SIAV2 WD far enough along plan to do all of these
  - Grid functionality
    - TAP and SIAV2 will include integrated Grid capabilities
  - Hope to develop this as a more polished product for VAO
  - Will use NRAO and ALMA as our local use case

http://trac.us-vo.org/project/nvo/wiki/DALServer

Purpose and Scope

- Standards Development
  - Provides reference implementation of the DAL services
  - We use this within NVO/VAO to prototype new standards
- Community Support
  - Being developed as a software product for community use
  - Reduces effort for data providers
    - Can be quite modest in the most common cases
    - But is fully customizable via subclassing framework
  - Greatly increases chance of a quality user data service
  - Correctness, completeness, robustness, etc.
    - Especially important for 2ndGen VO services

How it Works

- Concept
  - What we have is a family of closely related DAL services
    - Generic Dataset, service interface, data model
    - Much of the implementation can be shared among services
- Common infrastructure
  - VOTable and output formatting
    - This is all generic and shared, based upon DM (see below)
  - Grid infrastructure
    - SSO, UWS/async/jobs, VOSpace etc.
    - (We don't have this in yet, but it will be common infrastructure)

How it Works

- Software Structure
  - Core DALServer classes provide generic service implementations
  - These can be used directly in many cases
    - e.g., with SIA metadata in a DBMS table
    - Just configure and data can be served directly
  - Subclassing used in more complex cases
    - Data provider subclasses selected methods of generic class
    - Replace generic method with a custom one

- Unit tests
  - Unit tests are built directly into the generic classes
  - Null query, or test query on built-in data
  - Built in Web interface for testing

How it Works

- Protocol Abstraction
  - Logical service functionality and transport are separated
  - Service methods know nothing about transport (eg HTTP)
  - Can expose via multiple transports if desired
- Data Model
  - Service implementations are all data model based
    - Service code modifies data model provided as abstract class
    - Serialization reads data model and writes ouput
    - Many output formats can be provided (VOTable, XML, text, CSV etc)
  - Data model is machine readable
    - Keyword factory ensures correct DM metadata
    - In some cases code is autogenerated from DM

How it Works

- Packaging
  - DALServer code is all in Java
  - Distributed with directly install-and-go WAR file
  - Configuration and customization is then possible, but basic functionality is immediately available.

Status

- Has been in use for 2 years or so now.
  - First appeared as part of SSA development as ref impl
- Current Service Classes Complement
  - SSA, SIAV1, SCS, SLAP
  - SLAP implementation used for Splatalogue (ALMA)
- Near Term Plans
  - Reference implementations for TAP (ADQL+PQL), SIAV2
    - Once we have SIAV2 WD far enough along plan to do all of these
  - Grid functionality
    - TAP and SIAV2 will include integrated Grid capabilities
  - Hope to develop this as a more polished product for VAO
  - Will use NRAO and ALMA as our local use case