Librairies ADQL, UWS et TAP du CDS



Librairies complémentaires

Librairie TAP

- exécuter des requêtes ADQL
- décrire les tables disponibles
- importer des tables

Librairie ADQL

- interpréter de l'ADQL
- manipuler de l'ADQL
- > traduire en SQL

Librairie UWS

- exécuter des tâches
- gérer une file d'attenteLister et décrire les tâches



Librairie ADQL

- **Objectif :** Interpréter, manipuler et traduire des requêtes ADQL
- **Dernière version :** 1.0 Bêta (sources non disponibles jusqu'à la version stable)
- Site: http://cdsportal.u-strasbg.fr/adqltuto
- Développement requis : aucun

ADQL ? Astronomical Data Query Language est une extension du langage SQL permettant l'interrogation par coordonnées.



Exemple

Objectif: Traduire de l'ADQL en SQL

Contraintes :

- i. Si erreur : récupérer le numéro de ligne et de colonne en plus du message
- ii. Vérifier l'existence des colonnes et tables utilisées :

DATA

id: int8

name : varchar(100)

ra: float8 dec: float8

coord : spoint

type: varchar(32)

iii.Restriction des systèmes de coordonnées à seulement ICRS [barycenter]



RAPPEL: ADQL vs SQL

- Différences par rapport à SQL :
 - LIMIT à la fin du SQL → TOP juste après le SELECT
 - Pas d'UNION et d'INTERSECT
- Pas de sensibilité à la casse sauf entre "..."
- Types supplémentaires :
 - POINT, CIRCLE, BOX, POLYGON, REGION
- Fonctions supplémentaires :
 - CONTAINS, INTERSECTS, DISTANCE, AREA, ...



Gestion des fonctions géométriques

- Problème : conversion en SQL et exécution sur une BdD
- Solution dépendante du SGBD :
 - PostgreSQL:
 - Q3C (http://code.google.com/p/q3c/)
 - PgSphere (http://pgsphere.projects.postgresql.org/)
 - SQLite
 - Spatialite (http://www.gaia-gis.it/gaia-sins/)



Autres exemples

- Exemple du tutoriel ADQL:
 - Validateur d'ADQL (http://cdsportal.u-strasbg.fr/adqltuto/validator.html)
- Simbad-TAP:
 - Vérification intégrée à un formulaire HTML grâce à Javascript
- Interface TAP de Topcat



A faire!

- Gestion de plusieurs erreurs lors de l'interprétation
- Améliorer la gestion des systèmes de coordonnées (intégrer STC?)



Librairie UWS

- Objectif: Faciliter la mise en place et la gestion d'un service UWS
- Dernière version: 3.0
- Site: http://cdsportal.u-strasbg.fr/uwstuto
- Développement requis : 2 classes

UWS ? Universal Worker Service pattern est un modèle de gestionnaire de tâches.



Exemple

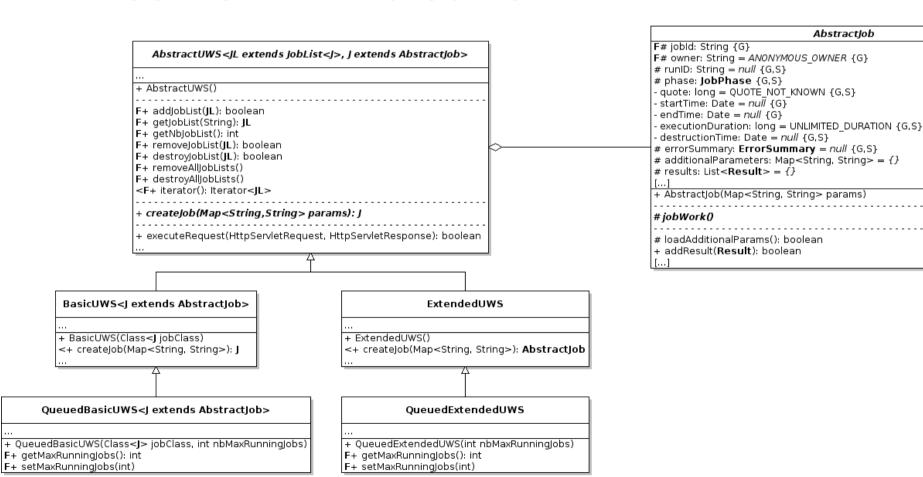
- Objectif : construire un service UWS d'exécution de requêtes ADQL
- Contraintes:
 - Identification par adresse IP
 - File d'attente limitée à 10 requêtes simultanées
 - Page d'accueil personnalisée

http://localhost:8080/asov/uws



Exemple (suite)

- 1 classe abstraite à étendre : AbstractJob
- 1 servlet HTTP à écrire





Autres exemples

- Exemples du tutoriel UWS :
 - UWSTimers (http://cdsportal.u-strasbg.fr/uwstuto/basic.html)
 - UWSAlgorithms (http://cdsportal.u-strasbg.fr/uwstuto/extended.html)

 Service du Crossmatch du CDS (http://cdsxmatch.u-strasbg.fr/xmatch)



Fonctionnalités

- Personnalisation de la page d'accueil
- Identification des utilisateurs
- File d'attente
- Destruction et annulation automatiques
- Plusieurs formats de sortie
- Sauvegarde/Restauration (v3.2)



Versions prévues!

- Version 3.1 :
 - Correction de bugs mineurs
- Version 3.2:
 - Possibilité de sauvegarder et restaurer un UWS
 - Amélioration de la gestion des utilisateurs (1 utilisateur = 1 classe)
 - Ajout de la possibilité d'uploader des fichiers
- Version 4.0 = Simplification de l'API :
 - UWS = 1 seule classe non-abstraite = 1 servlet
 - Donner la main totale aux développeurs sur la gestion des paramètres des jobs ainsi que des threads



Librairie TAP

- **Objectif**: Faciliter la mise en place et la gestion du protocole TAP.
- **Dernière version**: 1.0 Bêta (sources non disponibles jusqu'à la version stable)
- Site: http://cdsportal.u-strasbg.fr/taptuto
- **Développement requis** : 4 classes + 1 par format de sortie

TAP ? Table Access Protocol est un protocole permettant l'exécution de requêtes ADQL sur un service d'accès à des données astronomiques.



Ressources d'un service TAP



Liste les différentes fonctionnalités et particularités du service.

TAP

lavailability

Dit si le service est disponible et dans le cas contraire explique pourquoi.

/tables

Liste toutes les tables et colonnes utilisables dans une requête ADQL.

Isync

Exécution synchrone/immédiate d'une requête ADQL.

lasync

Soumet une requête ADQL au gestionnaire de tâches asynchrones (**UWS**).



Modification en BdD

- Ajout du schéma TAP_SCHEMA :
 - Création de 6 tables dans ce schéma
 - Remplissage de ces tables avec la description de toutes les tables et colonnes disponibles
- (Optionnel) Ajout du schéma TAP_UPLOAD
 - Importation de tables d'utilisateur dans ce schéma
 - Suppression immédiate juste après exécution de la requête



Exemple

 Objectif: Créer un service TAP permettant l'interrogation de la BdD suivante :

DATA

<u>id</u> : *int8*

name : *varchar(100)*

ra: float8 dec: float8 coord: spoint

type: varchar(32)

Contraintes:

- Exécution limitée à 1h
- Destruction après 7 jours
- Identification des utilisateurs par adresse IP
- Page d'accueil personnalisée

http://localhost:8080/asov/tap

«DBConnection < R>»

- + startTransaction()
- + cancelTransaction()
- + endTransaction()
- + executeQuery(String, ADQLQuery): R
- + close()

www.www.FOR UPLOAD www.www

- + createSchema(String)
- + dropSchema(String)
- + createTable(TAPTable)
- + dropTable(TAPTable)
- + insertRow(SavotTR, TAPTable)

«ServiceConnection < R > »

- + getProviderName(): String
- + getProviderDescription(): String
- + isAvailable(): boolean
- + getAvailability(): String
- + getRetentionPeriod(): int[]
- + getExecutionDuration(): long[]
- + getOutputLimit(): int[]
- + getOutputLimitType(): LimitUnit[]
- + getUserIdentifier(): UserIdentifier
- + uploadEnabled(): boolean
- + getUploadLimit(): int[]
- + getUploadLimitType(): LimitUnit[]
- + getUploadDirectory(): String
- + getTAPMetadata(): TAPMetadata
- + getFactory(): TAPFactory<R>
- + getOutputFormats(): OutputFormat<R>[]
- + getOutputFormat(String): OutputFormat<R>
- + deleteResults(ADQLExecutor<R>)
- + log(LogType, String)

«TAPFactory<R>»

- + createExecutor(Map<String, String>): ADQLExecutor<R>
- + createQueryChecker(TAPSchema): QueryChecker
- + createADQLTranslator(): ADQLTranslator
- + createDBConnection(): DBConnection<R>
- + createUploader(): Uploader

AbstractTAPFactory<R>

F# service: ServiceConnection<R>

- # AbstractTAPFactory(ServiceConnection<R>)
- <+ createExecutor(Map<String, String>): ADQLExecutor<R>
- <+ createQueryChecker(TAPSchema): QueryChecker</p>
- <+ createUploader(): Uploader</p>

«OutputFormat < R>»

- + getMimeType(): String
- + getShortMimeType(): String
- + getDescription(): String
- + getFileExtension(): String
- + writeResult(R, OutputStream, ADQLExecutor<R>)
- + writeResult(R, ADQLExecutor<R>): Result



Fonctionnalités

TAP Features			Managed ?
TAP	languages	ADQL	V
		PQL	V
	query executions	synchronous	
		asynchronous	
	resources	availability	
		capabilities (with TAPRegExt)	
		tables	V
	parameters	request=doQuery	V
		request=getCapability	V
		version	
		query	\checkmark
		format	
		maxRec	
		runld	
		upload (inline)	\checkmark
		upload (http)	
	TAP_UPLOAD (db schema)		
	metadata		V
	TAP_SCHEMA (db schema)		V
	parse		\checkmark
	Execute	PostgreSQL+PgSphere	V
ADQL		other DBMs	V
ADQL		others	V
	coordinate system		V
	check with DB		V

Legend			
	Fully managed		
	Specific extension required		
	Not yet managed		
	No generic implementation possible		

A faire!

- Gestion de /tables :
 - Lecture de TAP_SCHEMA
 - OU Génération de TAP_SCHEMA

- Implémentations par défaut adaptées à une BdD :
 - TAPMetadata
 - DBConnection
 - OutputFormat pour VOTable, CSV, TSV et JSON

Liens utiles

- Librairie ADQL
 - http://cdsportal.u-strasbg.fr/adqltuto
- Librairie UWS
 - http://cdsportal.u-strasbg.fr/uwstuto
- Librairie TAP
 - http://cdsportal.u-strasbg.fr/taptuto
- Questions, critiques et suggestions?
 - gregory.mantelet@astro.unistra.fr