ADQL, UWS and TAP libraries

Grégory Mantelet

Centre de Données astronomiques de Strasbourg

3 libraries in 1

TAP library

- execute ADQL queries
- describe available tables
- import user tables

+

ADQL library

- parse ADQL
- manipulate ADQL
- translate in SQL

+

UWS library

- execute asynchronous tasks
- manage execution queue
- list and describe tasks

ADQL library

- **Goal:** Parse, manipulate and translate ADQL queries
- **Last release :** 1.0 Bêta *(sources not available until a stable release)*
- **WebSite :** http://cdsportal.u-strasbg.fr/adqltuto
- **Required development :** none

ADQL ? *Astronomical Data Query Language is an extension of the SQL language and which includes functions to interrogate a database by astronomical coordinates.*

Example

- **Goal:** Translate ADQL into SQL

- **Constraints :**

  i. If error : get the line and column numbers in addition to the error message

  ii. Check the existence of the following table and columns

DATA
id : *int8*
name : *varchar(100)*
ra : *float8*
dec : *float8*
coord : *spoint*
type : *varchar(32)*

  iii. Limit coordinate systems to only ICRS [barycenter]

REMINDER : ADQL vs SQL

- Differences with SQL :
  - LIMIT at the end of SQL → TOP just after the SELECT
  - No UNION and INTERSECT
- No case sensitivity except between "…"
- Additional types :
  - POINT, CIRCLE, BOX, POLYGON, REGION
- Additional functions :
  - CONTAINS, INTERSECTS, DISTANCE, AREA, ...

Management of the geometrical functions

- **Problem :** translation into SQL and execution on the database
- **DB dependant solution :**
  - PostgreSQL :
    - Q3C *(http://code.google.com/p/q3c/)*
    - PgSphere *(http://pgsphere.projects.postgresql.org/)*
  - SQLite
    - Spatialite *(http://www.gaia-gis.it/gaia-sins/)*

Other examples

- The CDS ADQL tutorial:
  - *ADQL validator (http://cdsportal.u-strasbg.fr/adqltuto/validator.html)*
- Simbad-TAP:
  - *Checks are done with Javascript in the HTML form*
- TAP interface/form of Topcat

To do !

- Management of several (minor) errors during the parsing
- Improve the coordinate system and STC-S management

UWS library

- **Goal :** Make easier the configuration and management of a UWS
- **Last release :** 3.0
- **WebSite :** http://cdsportal.u-strasbg.fr/uwstuto
- **Required development :** 2 classes

UWS ? *Universal Worker Service pattern is a pattern for an asynchronous tasks management system.*

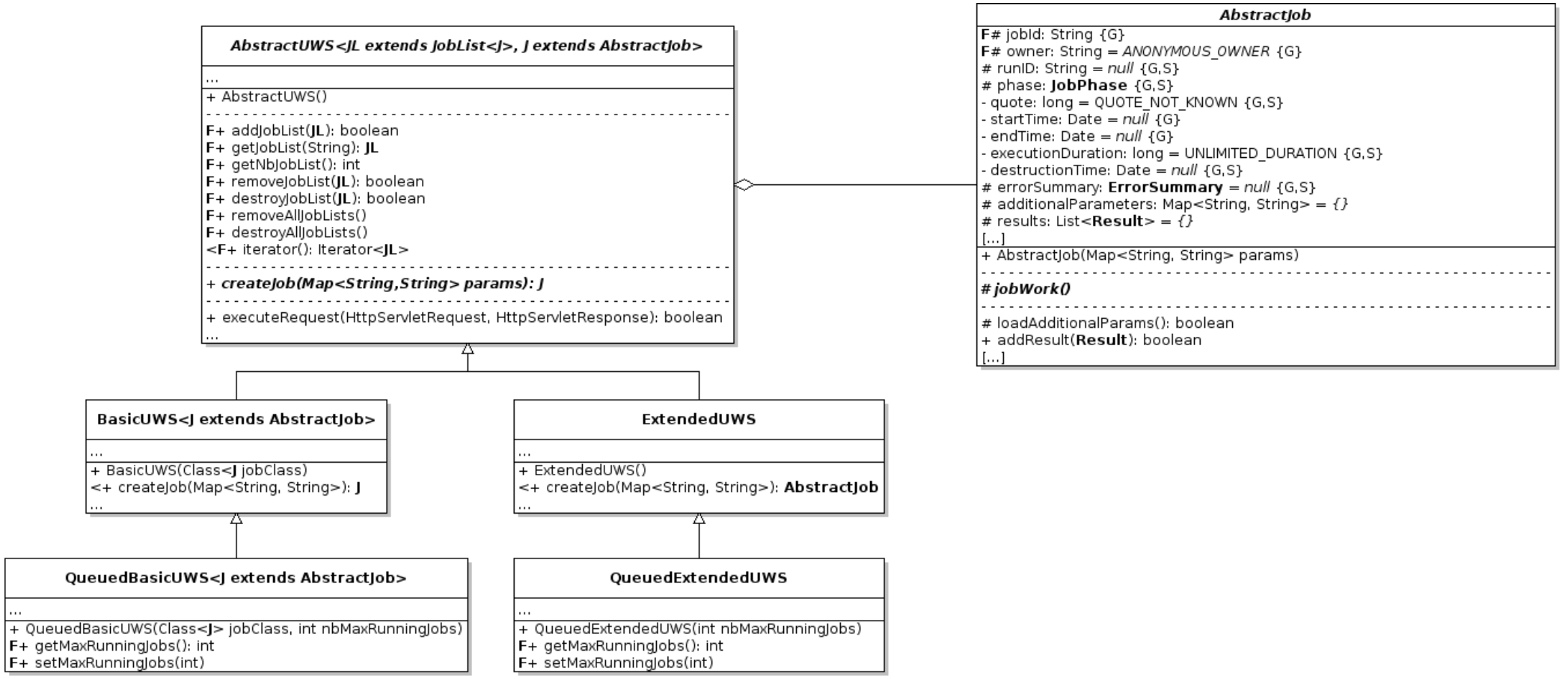
Example

- **Goal :** build a UWS service able to execute ADQL queries
- **Constraints :**
  - Authentication by IP address
  - Execution queue limited to 10 queries
  - Custom welcome page

http://localhost:8080/asov/uws

Example (2)

- 1 abstract class to extend: AbstractJob
- 1 HTTP servlet to write



Other examples

- The CDS UWS tutorial :
  - UWSTimers *(http://cdsportal.u-strasbg.fr/uwstuto/basic.html)*
  - UWSAlgorithms *(http://cdsportal.u-strasbg.fr/uwstuto/extended.html)*

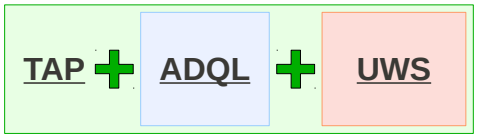
- The CDS Crossmatch service *(http://cdsxmatch.u-strasbg.fr/xmatch)*

Functionalities

- Welcome page customization
- User authentication
- Execution queue
- Automatic task abortion and destruction
- Several output formats
- Backup/Restoration *(v3.2)*

Next releases !

- Version 3.1 :
  - Minor bugs fixed
- Version 3.2:
  - Possibility to save and restore a UWS
  - Improvement of the users management (1 user = 1 class)
  - File upload
- Version 4.0 = API simplification :
  - UWS = only 1 NON-abstract class = 1 servlet
  - Make easier the management of task parameters and thread

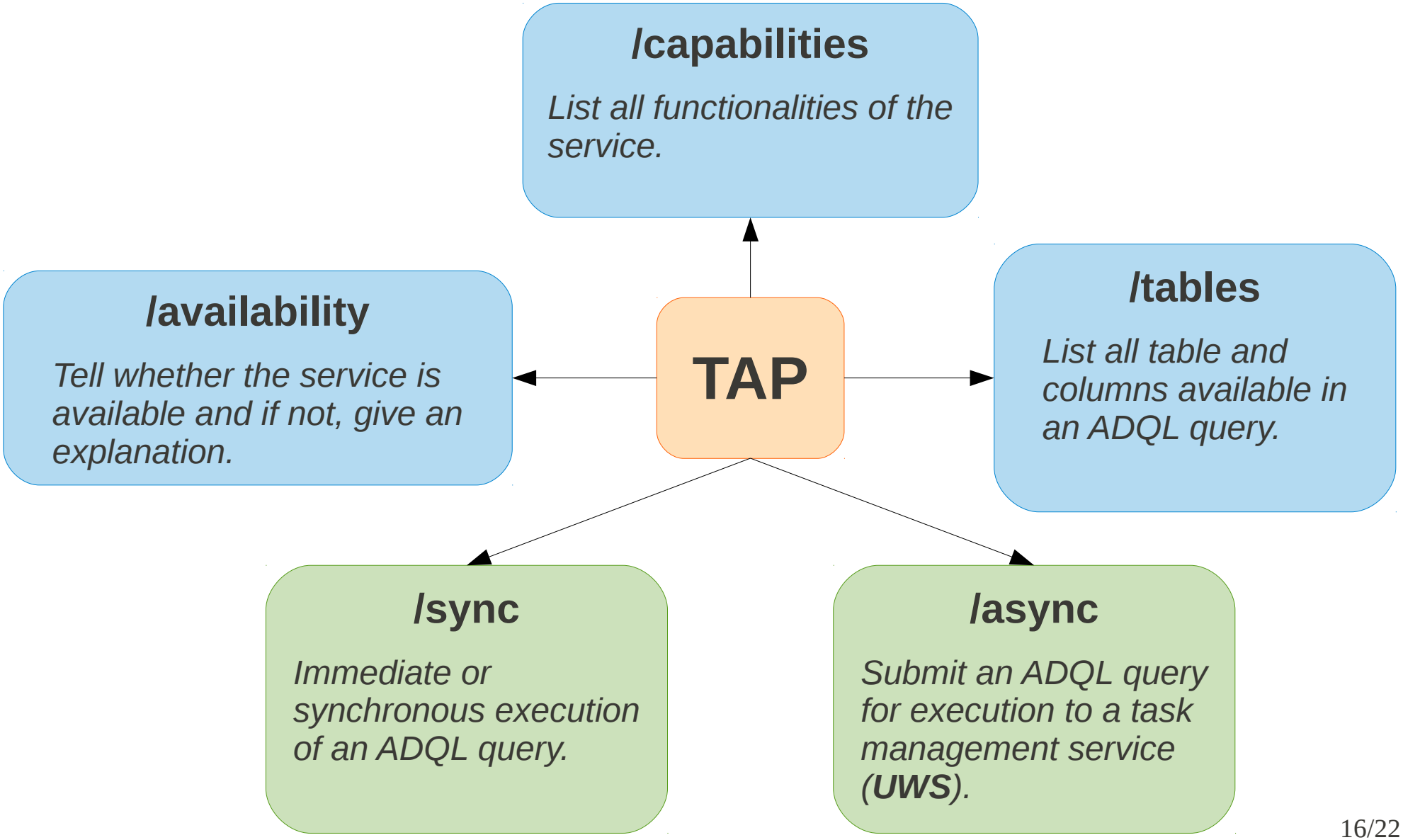


TAP libraries

- **Goal :** Make easier the configuration and management of the TAP protocol.
- **Last release :** 1.0 Bêta *(sources not available until a stable release)*
- **WebSite :** http://cdsportal.u-strasbg.fr/taptuto
- **Required development :** 4 classes + 1 for each output format

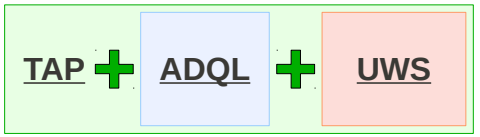
TAP ? *Table Access Protocol is a protocol which lets executing ADQL queries on an astronomical service.*

Resources of a TAP service



Database modifications

- Add a schema TAP_SCHEMA :
  - Creation of 6 tables in this schema
  - Filling of these tables with the description of all tables and columns available in ADQL queries
- (Optional) Add a schema TAP_UPLOAD
  - Importation of all user tables in this schema
  - Immediate destruction of these tables after execution



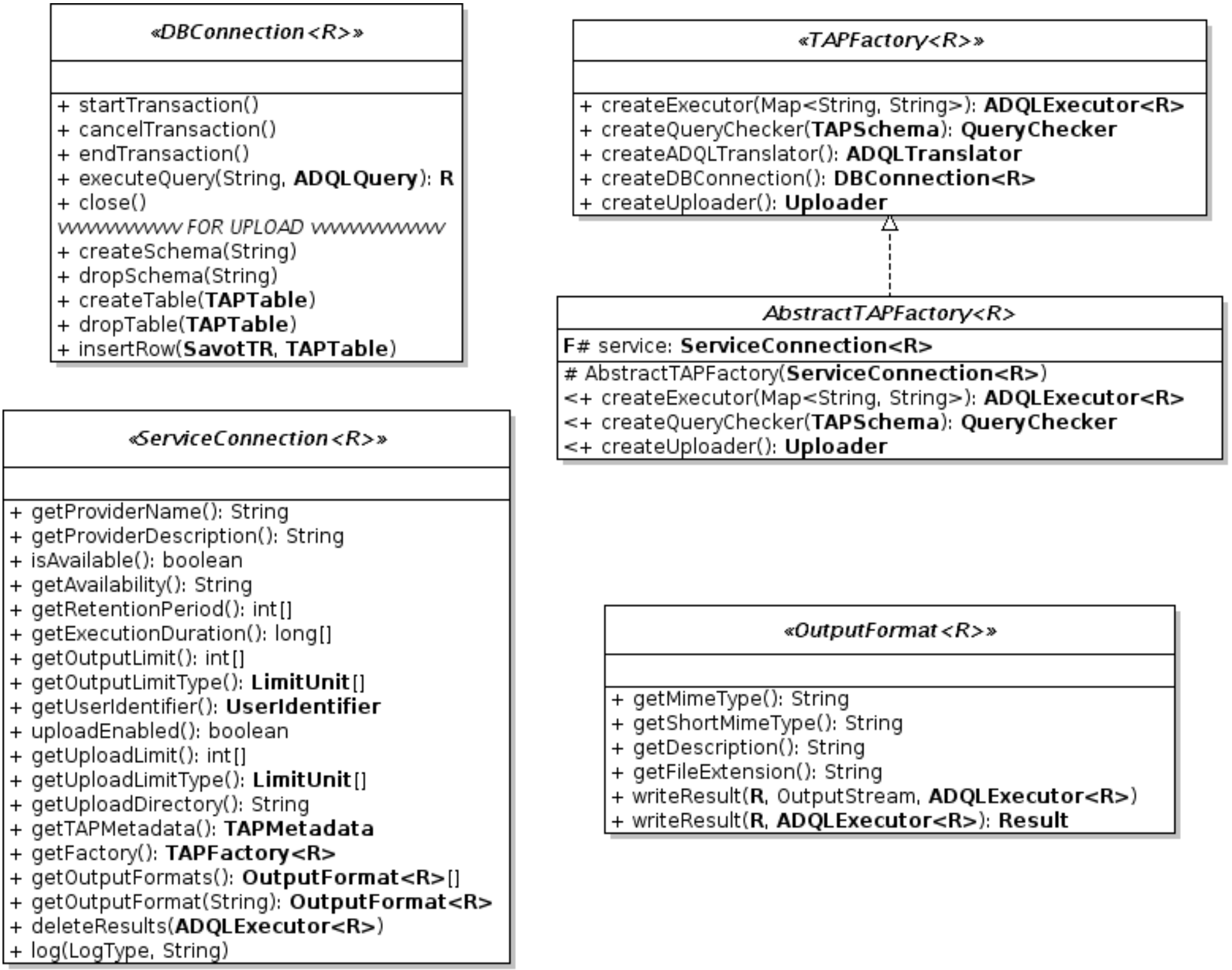
Example

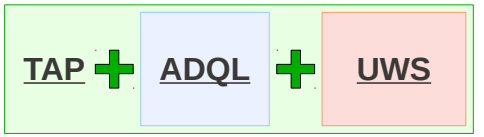
- **Goal :** Create a TAP service on the following database :

DATA
id : *int8*
name : *varchar(100)*
ra : *float8*
dec : *float8*
coord : *spoint*
type : *varchar(32)*

- **Constraints :**
  - Execution limited to 1h
  - Destruction after 7 days
  - User authentication by IP address
  - Custom welcome page

http://localhost:8080/asov/tap


«DBConnection<R>»
+ startTransaction()
+ cancelTransaction()
+ endTransaction()
+ executeQuery(String, ADQLQuery): R
+ close()
wwwwwwww FOR UPLOAD wwwwwwww
+ createSchema(String)
+ dropSchema(String)
+ createTable(TAPTable)
+ dropTable(TAPTable)
+ insertRow(SavotTR, TAPTable)
«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
<+ createUploader(): Uploader
«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)
«OutputFormat<R>»
+ getMimeType(): String
+ getShortMimeType(): String
+ getDescription(): String
+ getFileExtension(): String
+ writeResult(R, OutputStream, ADQLExecutor<R>)
+ writeResult(R, ADQLExecutor<R>): Result



Functionalities

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

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

To do !

- Management of /tables :
  - Reading of TAP_SCHEMA
  - OR Automatic generation of TAP_SCHEMA
- DB oriented default implementations :
  - TAPMetadata
  - DBConnection
  - OutputFormat pour VOTable, CSV, TSV et JSON

Some links

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