Development of Predictive Toxicology Applications

An OpenTox Workshop

19 Sep 2010, Rhodes, Greece

Application Programming Interfaces

presented by Nina Jeliazkova
(Ideaconsult Ltd., Bulgaria)

OpenTox

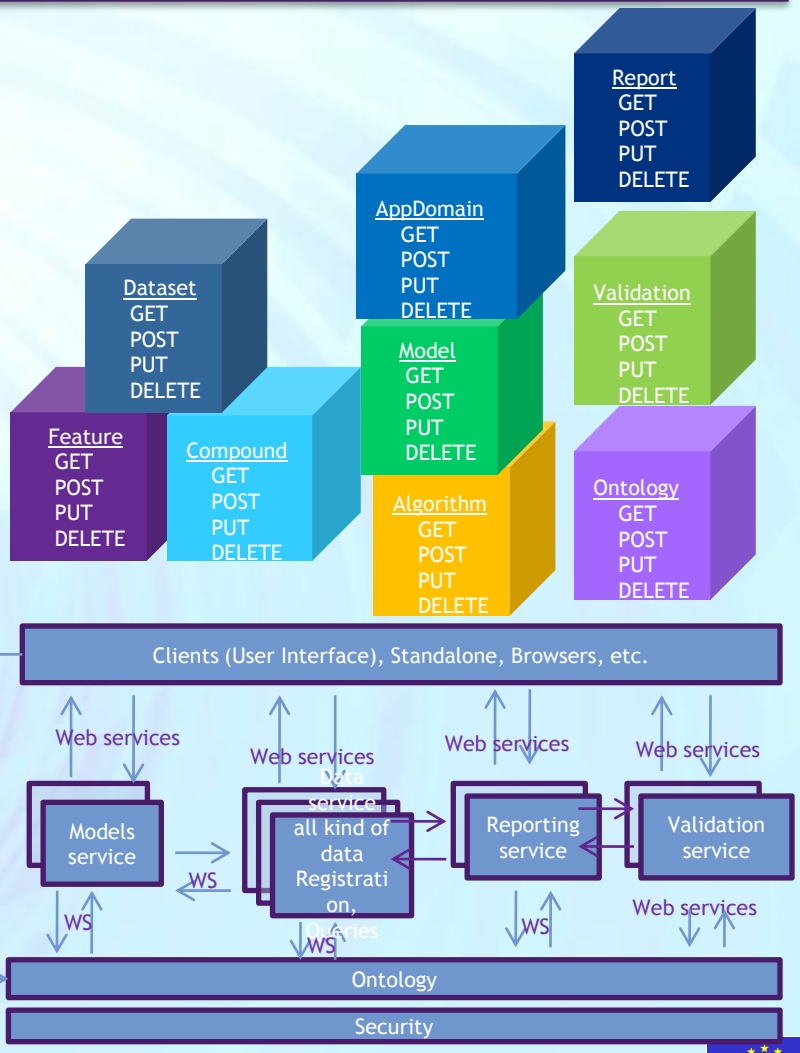
SEVENTH FRAMEWORK PROGRAMME

Framework design rationales

User Requirements		Software Requirements
Umambiguous data	⇒	*formal way of representing information about **data***
Unambiguous access	⇒	*well-defined interfaces*
Transparency of computational tools	⇒	*formal way of representing information about **methods**, well-defined interfaces*
Variety of user groups	⇒	*simplicity and modularity of design*
Need to integrate various resources (e.g., databases, prediction methods, models, …) to make meaningful predictions	⇒	*distributed architecture, interoperability*
Need to integrate biological information	⇒	*again, modularity of design, extensibility*

The framework

- OpenTox API
  - The way applications talk to each other
  - The way developers talk to applications
  - http://opentox.org/dev/apis/api-1.1
- The basic building blocks:
  - data, chemical structures, algorithms and models.
- Functionality offered
  - build models,
  - apply models,
  - validate models,
  - access and query data in various ways.
- Technologies
  - REST style web services
  - RDF for description of resources
  - Links to existing and newly developed ontologies (mainly to describe metadata) about resources



Representational State Transfer (REST)

A software architecture style, defined by Roy Fielding in his PhD thesis (2000). Many services worldwide offer REST API. There are (currently) **no standards** for RESTful applications, but merely **design guides.**

Design principles:

- Resource oriented
  - Every object (resource) is named and addressable (e.g. HTTP URL) Example: **http://example.opentox.com/model/myBestModel** , **http://example.opentox.com/compound/50-00-0**
  - RESTfull API design starts by identifying most important objects and groups of objects, supported by the software system and proceeds by defining URL patterns.
- Transport protocol
  - HTTP is the most popular choice of transport protocol, but other protocols can be used as well
- Operations
  - All resources (nouns) support the same fixed and universal number of operations (verbs). HTTP (GET, POST, PUT, DELETE) operations are the common choice, when the transport protocol is HTTP.
- Hypermedia as the Engine of Application State
  - All resources should be reachable via a single (or minimum) number of entry points into RESTful applications. Thus, a representation of a resource should return hypermedia links to related resources
- Error codes (for each resource/operation pair)
  - HTTP status codes (e.g. 200 OK, 400 Bad Request, 404 Not found, etc.) are usually used

OpenTox resources (1)

OpenTox considers the following set of entities as essential building blocks:

- Structures of **chemical compounds**
- **Properties and identifiers** of chemical compounds
- **Datasets** of chemical compounds and various properties (measured or calculated)
- **Algorithms**
  - Data processing algorithms
    - Algorithms generating certain values, based on chemical structure (e.g. descriptor calculation)
    - Data preprocessing (e.g. Principal component analysis, feature selection)
    - Structure processing (e.g. structure optimization)
    - Algorithms, relating set of structures to another set of structures (e.g. similarity search or metabolite generation)
  - Machine learning algorithms
    - Supervised (e.g. Regression, Classification)
    - Unsupervised (e.g. Clustering)
  - Prediction algorithms, defined by experts (e.g. series of structural alerts, defined by human experts , not derived by learning algorithms)

OpenTox resources (2)

- **Models** are generated by respective algorithms, given specific parameters
  - Statistical models are generated by applying statistical/machine learning algorithms to specific dataset and parameters
  - Models can be other than statistical, e.g. expert defined rules, quantum mechanical calculations, metabolite generation, etc. The intention of the framework is to be generic enough to accommodate varieties of predictive models.
- **Validation** provides procedures independent of model building facilities (e.g. crossvalidation) and generates relevant statistics.
- **Reports**
  - Various types of reports might be generated, using building blocks above (e.g. validation report can be generated using validation object, a model and a dataset).
- In addition, the following components are introduced:
  - **Task** (asynchronous processing of computationally intensive tasks)
  - **Authentication and authorization** (Ensuring secure access to sensitive resources)
  - **Ontology service** (provides an RDF storage and SPARQL endpoint for resources registration)

Resources identification

All resources are identified via unique web address, assigned according to the URL templates

Component	Description	URL Template (example)
Compound	Representations of chemical compounds	http://host:port/compound/{compoundid}
Feature	Properties and identifiers	http://host:port/feature/{featureid}
Dataset	Encapsulates set of chemical compounds and their property values	http://host:port/dataset/{datasetid}
Model	OpenTox model services	http://host:port/model/{modelId}
Algorithm	OpenTox algorithm services	http://host:port/algorithm/{algorithmid}
Validation, Report	A validation corresponds to the validation of a model on a test dataset.	http://host:port/validation/{validationid} http://host:port/report/{reportid}
Task	Asynchronous jobs are handled via an intermediate Task resource. A resource, submitting an asynchronous job should return the URI of the task.	http://host:port/task/{taskid}
Ontology service	Provides storage and SPARQL search functionality for objects, defined in OpenTox services and relevant ontologies	http://host:port/ontology
Authentication and authorisation	Granting access to protected resources for authorised users	http://host:port/opensso http://host:port/opensso-pol

OpenTox REST operations

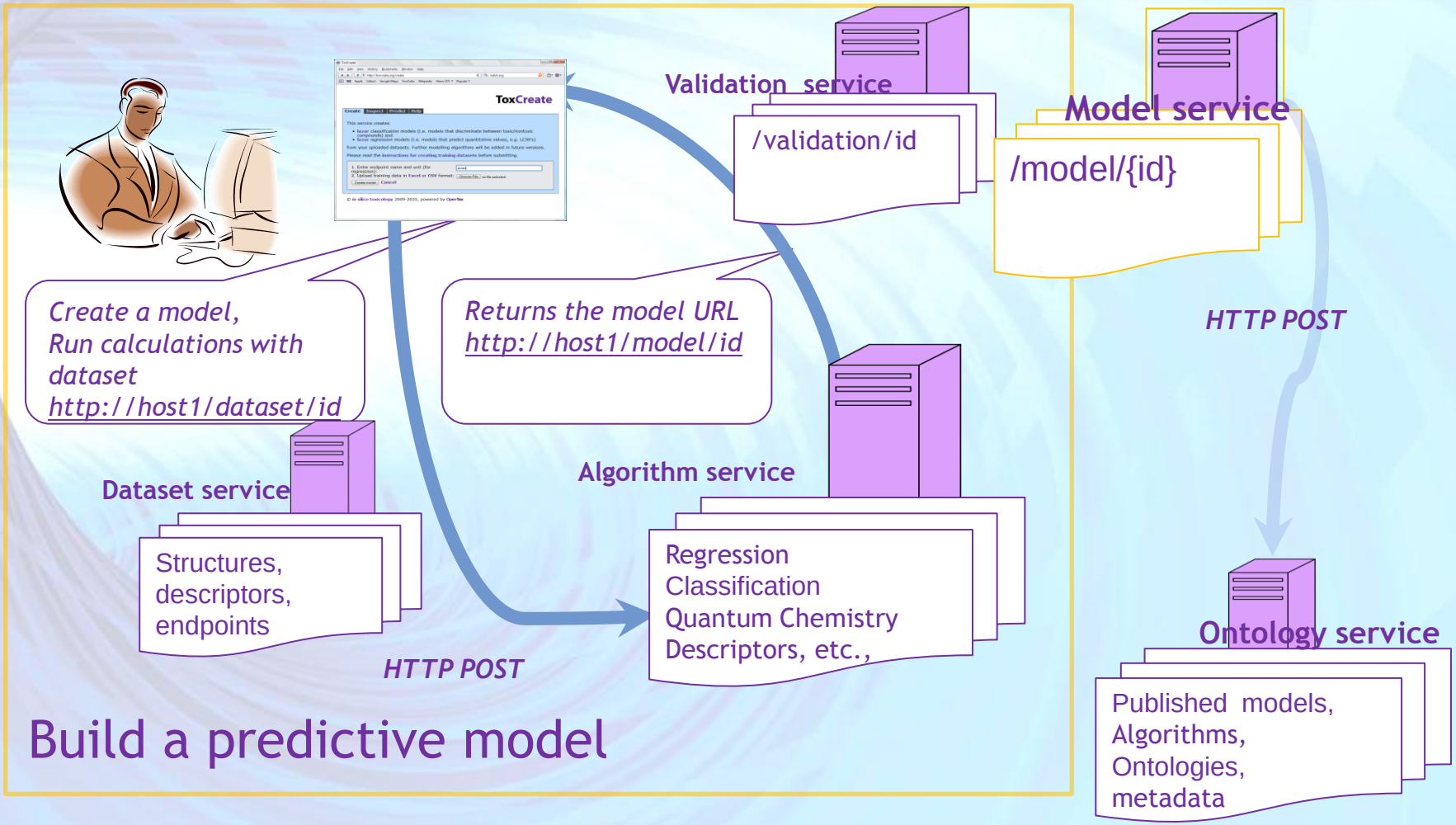
Individual resources (e.g. a dataset or a model)

- URI template http://host:port/{resource}/{resourceid} , e.g. http://host:port/model/{model_id} or http://host:port/dataset/{dataset_id}
- GET – retrieve representation of the resource
- PUT – update representation of the resource
- POST :
  - replace representation of the resource with a new one (e.g. replace the dataset with new content)
  - initiate calculations, based on this resource (e.g. submit dataset URI to an algorithm resource and obtain a model URI as a result)
- DELETE – delete the resource

Collections of resources (e.g. list of all available models, or datasets)

- URI template http://host:port/{resource} , (e.g. http://host:port/model or http://host:port/dataset)
- GET – retrieve representation of multiple resources (e.g. retrieve all available algorithms)
- PUT - N/A
- POST – create new resource and return its URI (e.g. create a new dataset by submitting new dataset content to the dataset service)
- DELETE – N/A

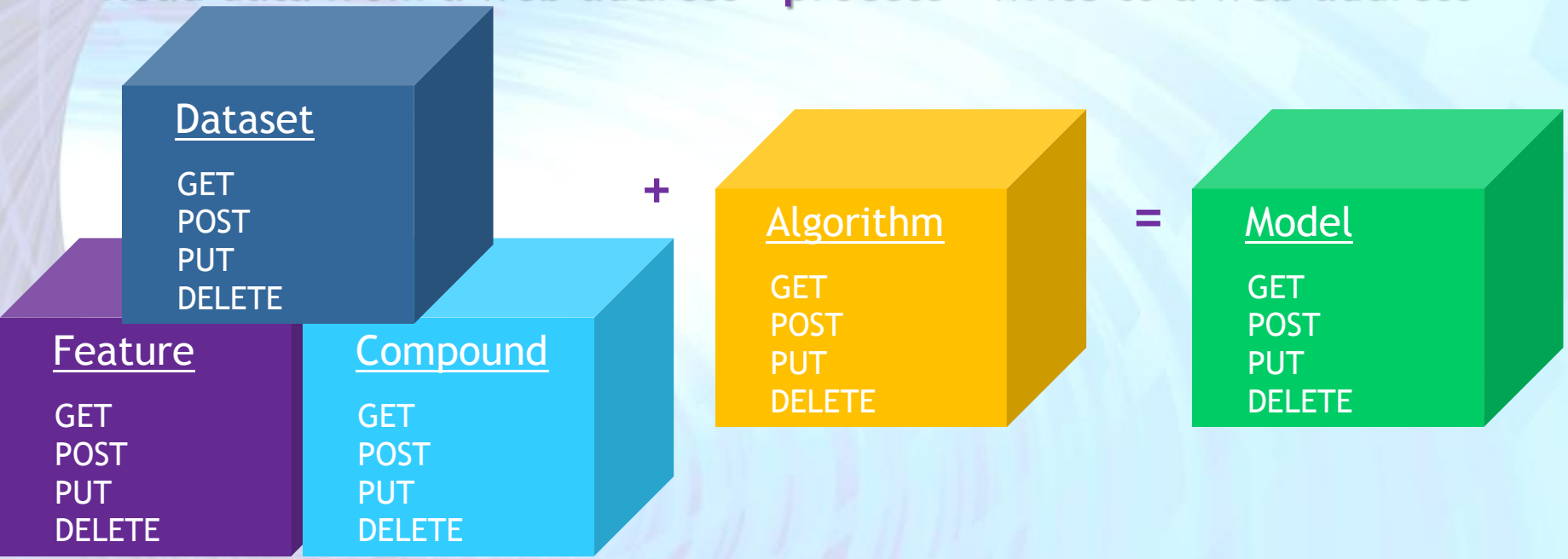
Build a predictive model



OpenTox

Uniform approach to models creation

Read data from a web address - process - write to a web address



http://myhost.com/algorithm/neuralnetwork

http://myhost.com/dataset/trainingset1

http://myhost.com/model/predictivemodel1

Use an algorithm to build a model

- An algorithm is applied by submitting HTTP POST to the algorithm URI and providing required parameters.
- A common required parameter is **dataset_uri=http://host:port/dataset/{datasetid}**, which specifies the data set to be operated on.
- HTTP POST in REST style services returns URI of the result, and not the content of the result.
- The algorithm services are designed to store the results into a dataset service and return the URL of the resulted dataset.
- In case of slow calculations a Task URI,

instead of the dataset URI is returned

```
$ curl -H "Accept:text/uri-list" -X POST -d
'dataset_uri=http://apps.ideaconsult.net:8080/ambit2/dataset/1037' -d
'prediction_feature=http://apps.ideaconsult.net:8080/ambit2/feature/26
701' -d
'dataset_service=http://apps.ideaconsult.net:8080/ambit2/dataset'
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-
dev/algorithm/J48 -iv
* Connected to opentox.informatik.tu-muenchen.de (131.159.28.16) port
8080 (#0)
➢POST /OpenTox-dev/algorithm/J48 HTTP/1.1
➢> Host: opentox.informatik.tu-muenchen.de:8080
➢Accept: */*
> Content-Type: application/x-www-form-urlencoded
< HTTP/1.1 202 Accepted
< Date: Sat, 31 Jul 2010 14:46:38 GMT
< Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-
dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5
< Accept-Ranges: bytes
< Server: Noelios-Restlet-Engine/1.1.snapshot
< Content-Type: text/uri-list;charset=ISO-8859-1
< Content-Length: 99
<
* Connection #0 to host opentox.informatik.tu-muenchen.de left intact
* Closing connection #0
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-
dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5
```

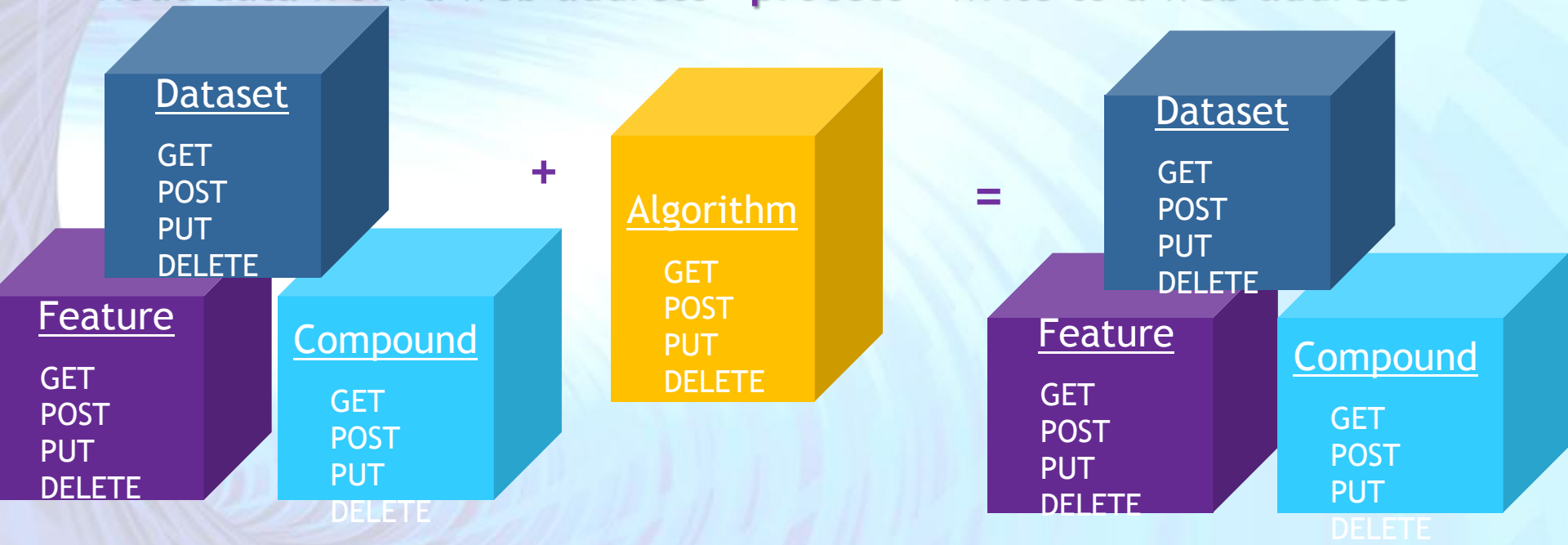
Resources: The model

- When task URI is returned , the returned status code is HTTP 202 Accepted, instead of HTTP 200 OK.
- This tells the client the processing is not completed and the client need to poll the task URI until OK code is returned
- The final result, returned by Example 25 is the URI of the new model http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48 .
- To obtain prediction results POST a dataset to the model URI

```
$ curl -iv -H "Accept:text/uri-list" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c
* About to connect() to opentox.informatik.tu-muenchen.de port 8080 (#0)
*   Trying 131.159.28.16... connected
* Connected to opentox.informatik.tu-muenchen.de (131.159.28.16) port 8080 (#0)
> GET /OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5 HTTP/1.1
> User-Agent: curl/7.18.2 (x86_64-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g
zlib/1.2.3.3 libidn/1.8 libssh2/0.18
> Host: opentox.informatik.tu-muenchen.de:8080
> Accept:text/uri-list
>
< HTTP/1.1 200 OK
< Date: Sat, 31 Jul 2010 14:47:22 GMT
Date: Sat, 31 Jul 2010 14:47:22 GMT
< Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48
< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
< Accept-Ranges: bytes
< Server: Noelios-Restlet-Engine/1.1.snapshot
< Content-Type: text/uri-list;charset=ISO-8859-1
< Content-Length: 86
<
* Connection #0 to host opentox.informatik.tu-muenchen.de left intact
* Closing connection #0
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48
```

Uniform approach to data processing (e.g. Descriptors calculation)

Read data from a web address - process - write to a web address



http://myhost.com/algorithm/{descriptorX}

http://myhost.com/dataset/trainingset1

http://myhost.com/dataset/results

Uniform approach to models validation and report generation

Read data from a web address - process - write to a web address

http://myhost.com/dataset/trainingset1

http://myhost.com/dataset/predictedresults1

Validation report

http://myhost.com/validation

Validation

GET
POST
PUT
DELETE

+

Dataset

GET
POST
PUT
DELETE

=

Report

GET
POST
PUT
DELETE

Model generating predictions

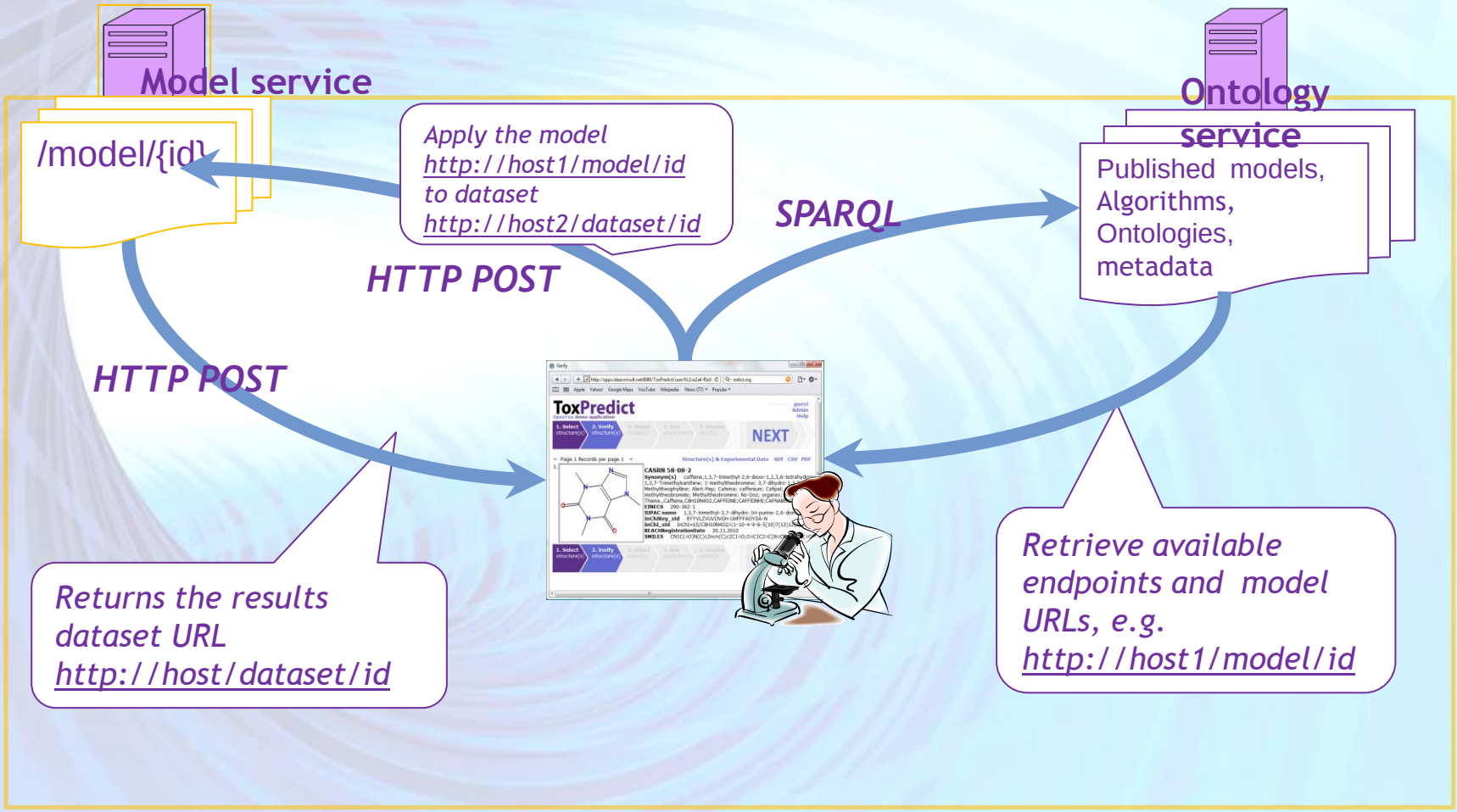
Model

GET
POST
PUT
DELETE

http://myhost.com/report/1

http://myhost.com/model/predictivemodel1

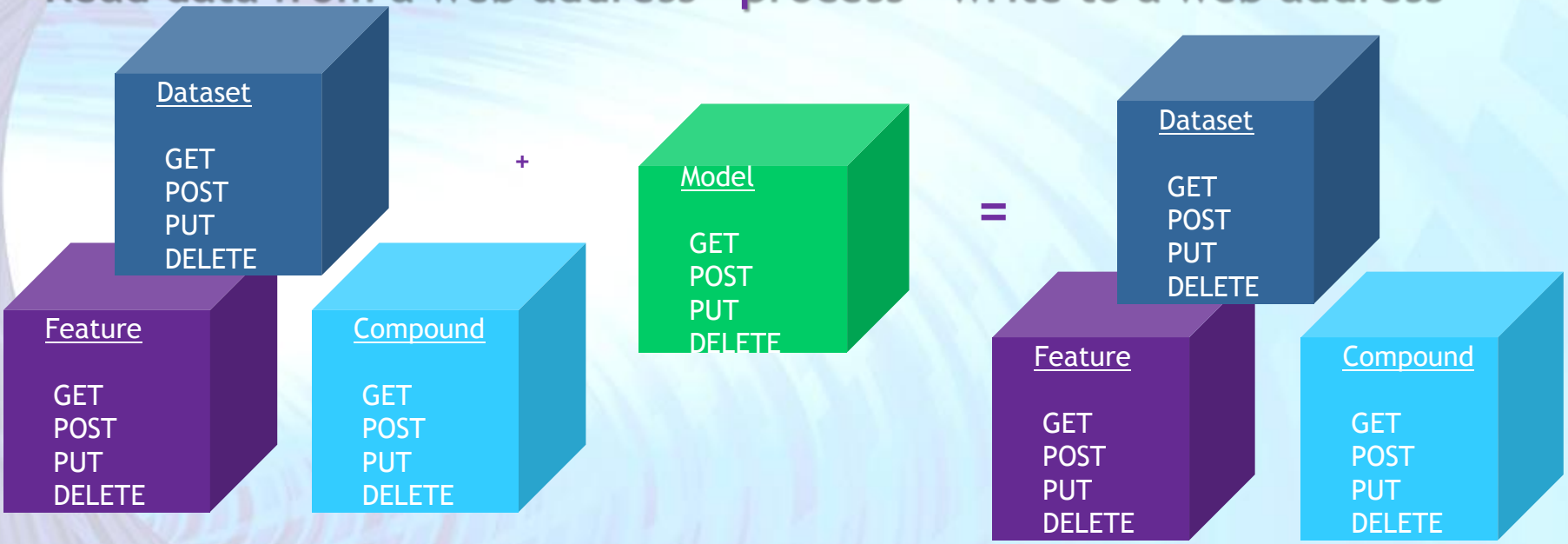
Apply predictive models



OpenTox

Uniform approach to model prediction

Read data from a web address - process - write to a web address



http://myhost.com/model/predictivemodel1

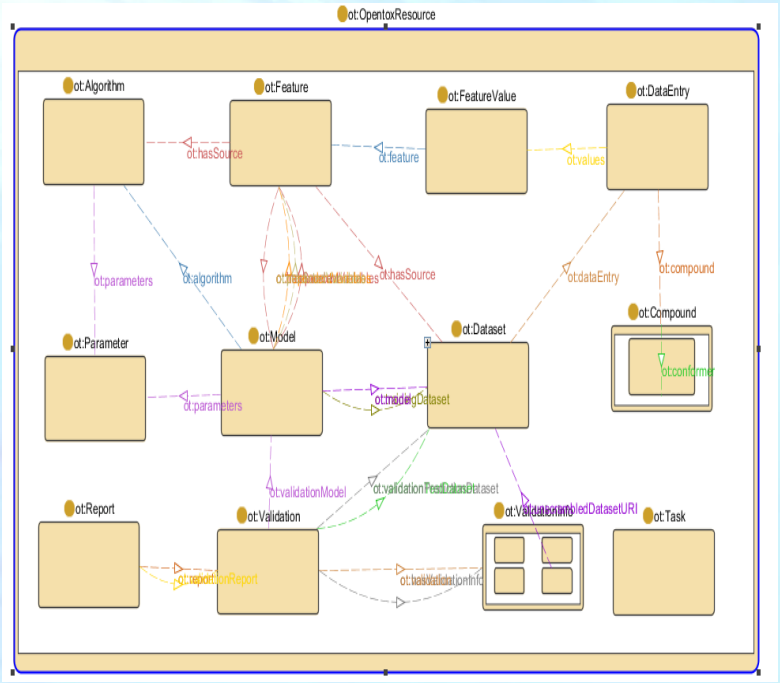
http://myhost.com/dataset/id1

http://myhost.com/dataset/results1

RDF - Resources representation

- The opentox.owl ontology
  - A common OWL data model of all OpenTox resources
  - Describes OpenTox resources
  - Describes relationships between them
  - Generates object's RDF representations.
- RDF/XML representation is mandatory for OpenTox resources.
- Uniform approach to data representation
  - Calculated and measured properties of chemical compounds are represented in an uniform way
  - Linked to the resource used for data generation
  - Annotated via ontology entries
  - Model representations link to algorithms and data used

All OpenTox components are defined by OWL ontology
http://opentox.org/api/1.1/opentox.owl
All resources are subclasses of ot:OpenToxResource



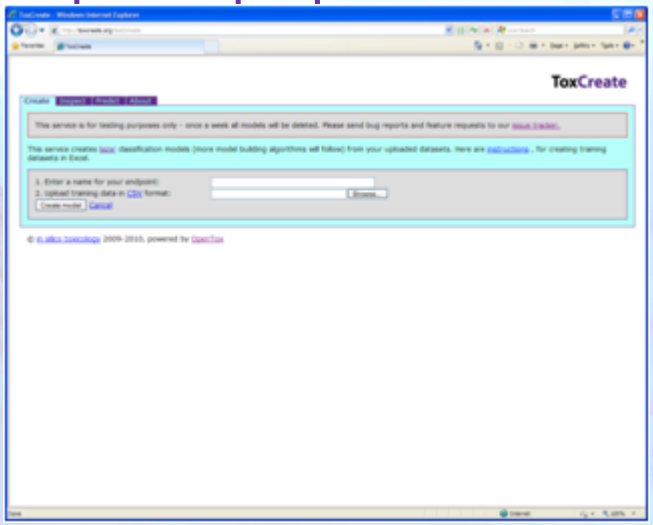
Services implementation by partner and service type

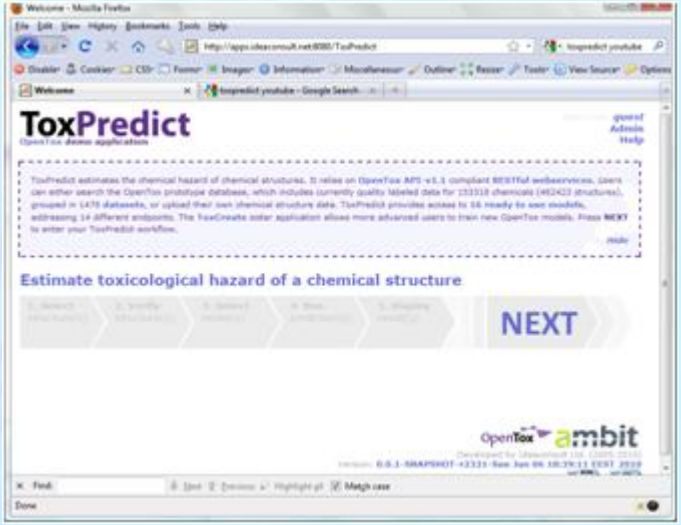
All components are implemented as REST web services.
There could be multiple implementations of same type of components.
(Subset of) services could be hosted by the same provider, or by multiple providers on separate locations.

Partner No. /Service type	Compound	Dataset	Feature	Algorithm (processing)	Algorithm (model)	Model	Validation	Report	Task	Autherntication and Authorisation service	Ontology service
2	Y	Y		Y	Y	Y			Y		
3	Y	Y	Y	Y	Y	Y			Y		Y
5				Y	Y	Y					
6							Y	Y	Y	Y	
7					Y	Y			Y		
10						Y					

Demo applications

- Two end user oriented demo applications, making use of OpenTox webservices, have been developed, deployed and are available for testing – http://toxcreate.org and http://toxpredict.org ;
- ToxCreate creates models from user supplied datasets;
- ToxPredict uses existing OpenTox models to estimate chemical compound properties





ToxPredict: a detailed case study

- ToxPredict estimates the chemical hazard of chemical structures. It relies on **OpenTox API-v1.1** compliant RESTful webservices. Users can either search the OpenTox prototype database, which includes currently quality labelled data for **~150,000 chemicals**, grouped in datasets, or upload their own chemical structure data. ToxPredict provides access to **14 ready to use models**, addressing **13 different endpoints**
- ToxPredict uses the following OpenTox webservices: **Compound**, **Feature**, **Dataset**, **Algorithm**, **Model** , **Task** and **Ontology**;
- more details on its interactions with OpenTox webservices which are taking place behind the scenes and without requiring any end-user intervention;

ToxPredict: Step 1 (Select structure(s))

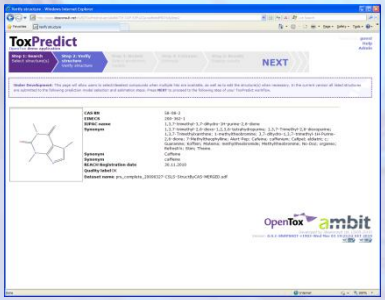
Find structure by name, registry number, SMILES, InChI, structure, substructure, similarity…

ToxPredict Web Application

OT Dataset API ***HTTP GET***

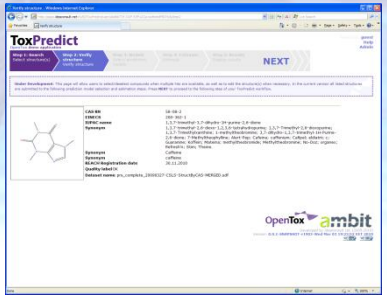
OT Dataset Service

text/uri-list,
application/rdf+xml,
chemical/x-daylight-smiles
chemical/x-mdl-sdfile,…



Here is the list of structures as URI links, RDF, MOL or SMILES.

ToxPredict: Step 2 (Verify structure(s))



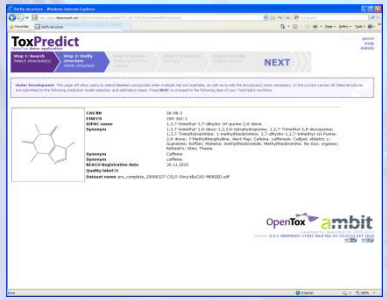
Select and/or edit structure(s)

ToxPredict Web Application

OT Dataset API ***HTTP GET***
(***HTTP POST*** for structure editing)

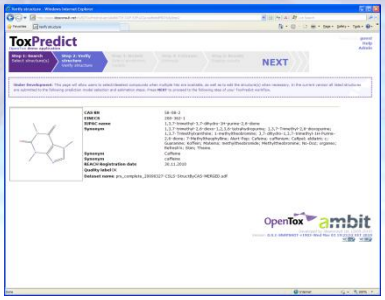
OT Dataset Service

text/uri-list,
application/rdf+xml,
chemical/x-daylight-smiles
chemical/x-mdl-sdfile,
image/png



Here is the list of structures as URI links, RDF, MOL, SMILES or images.

ToxPredict: Step 3 (Select model(s))



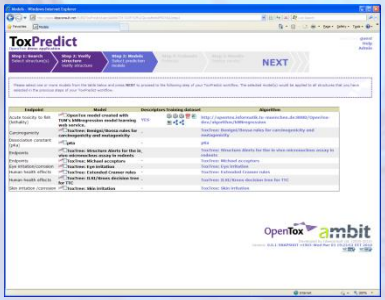
What prediction models are available? Is there a model for endpoint X?

ToxPredict Web Application

HTTP GET SPARQL query

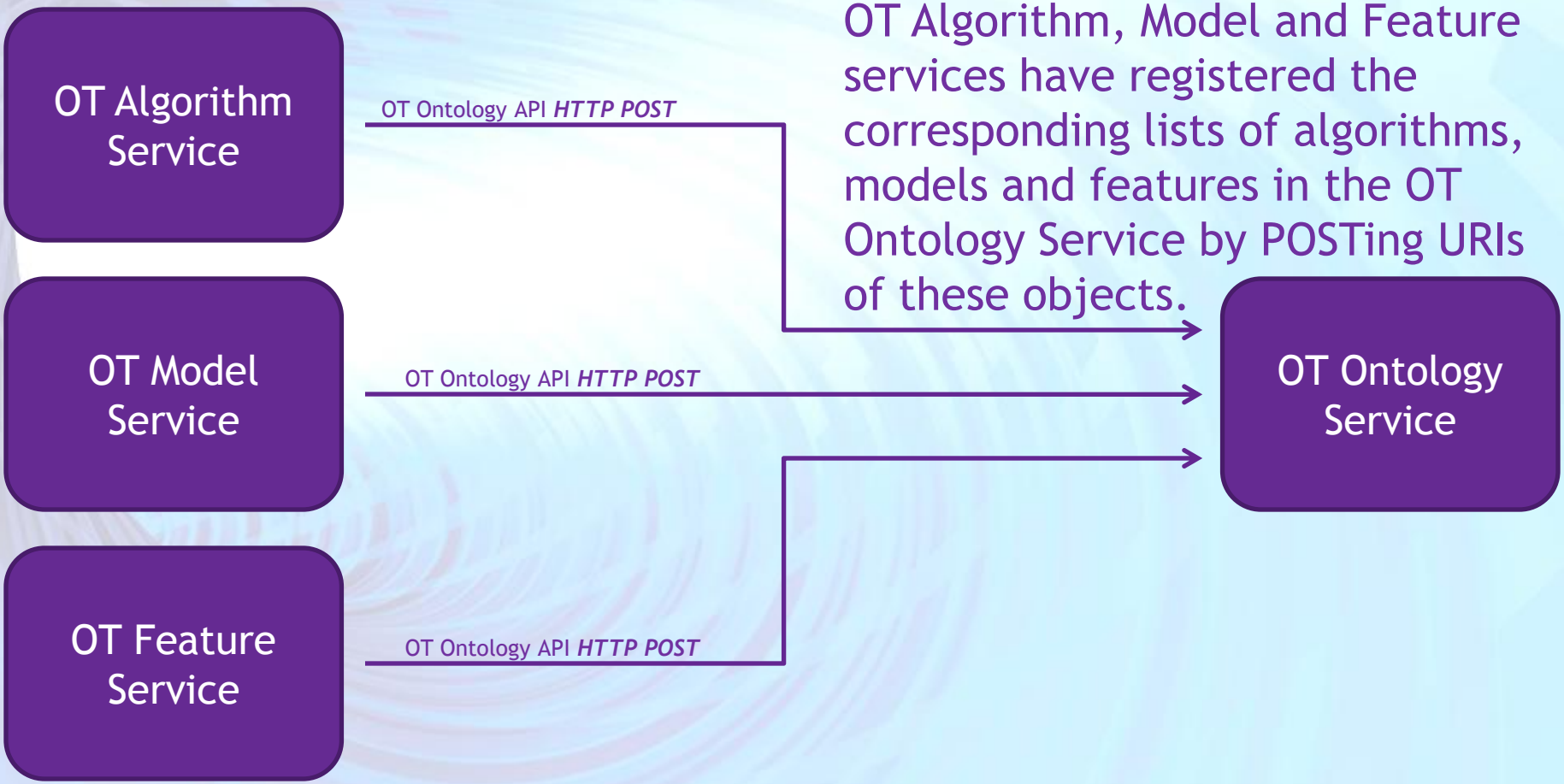
OT Ontology Service

application/sparql-results+xml



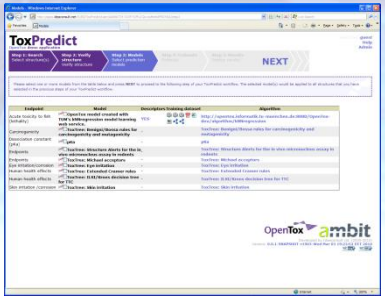
Here is the list of model URIs and related endpoints and algorithms in SPARQL format.

ToxPredict: Step 3 (behind the scenes)



OT Algorithm, Model and Feature services have registered the corresponding lists of algorithms, models and features in the OT Ontology Service by POSTing URIs of these objects.

ToxPredict: Step 4 (Estimate)



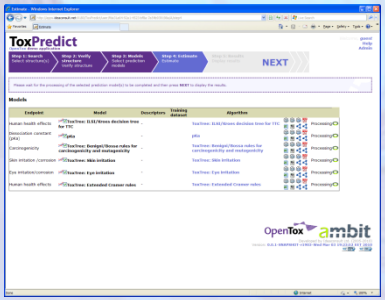
Run the selected models.

ToxPredict Web Application

OT Model API ***HTTP POST*** with parameter dataset URI from steps 1-2

OT Model Service

HTTP code 202 "Accepted"
Model task URI in HTTP:location header

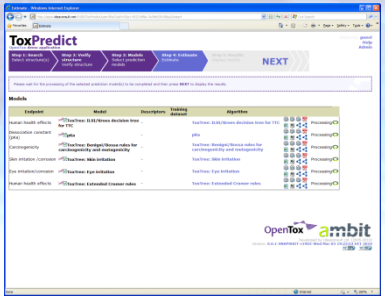


The calculation will take a while; here is a task URI, which can be queried for processing status.

Create a new task.

OT Task Service

ToxPredict: Step 4 (Estimate)



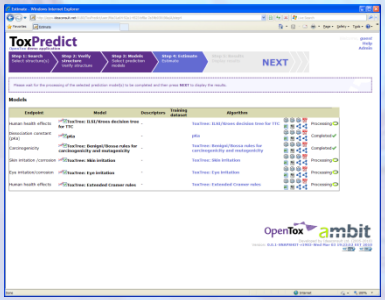
Is the task completed?

ToxPredict Web Application

OT Task API ***HTTP GET*** on task URI

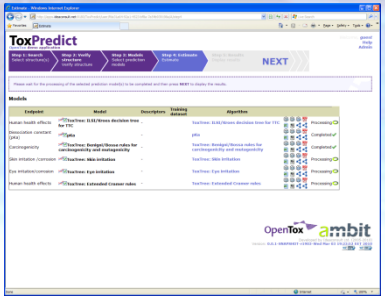
OT Task Service

HTTP code 202 "Accepted for processing"
Returns Task URI



Not yet, but processing has finished and the results have been posted to the OT Dataset service; here is a task URI for the dataset import.

ToxPredict: Step 4 (Estimate)



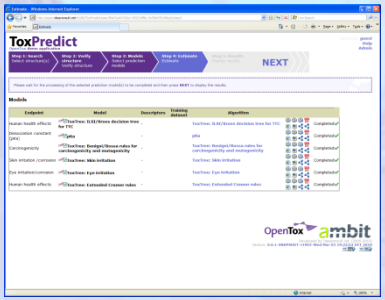
Is the task completed?

ToxPredict Web Application

OT Task API ***HTTP GET*** on task URI

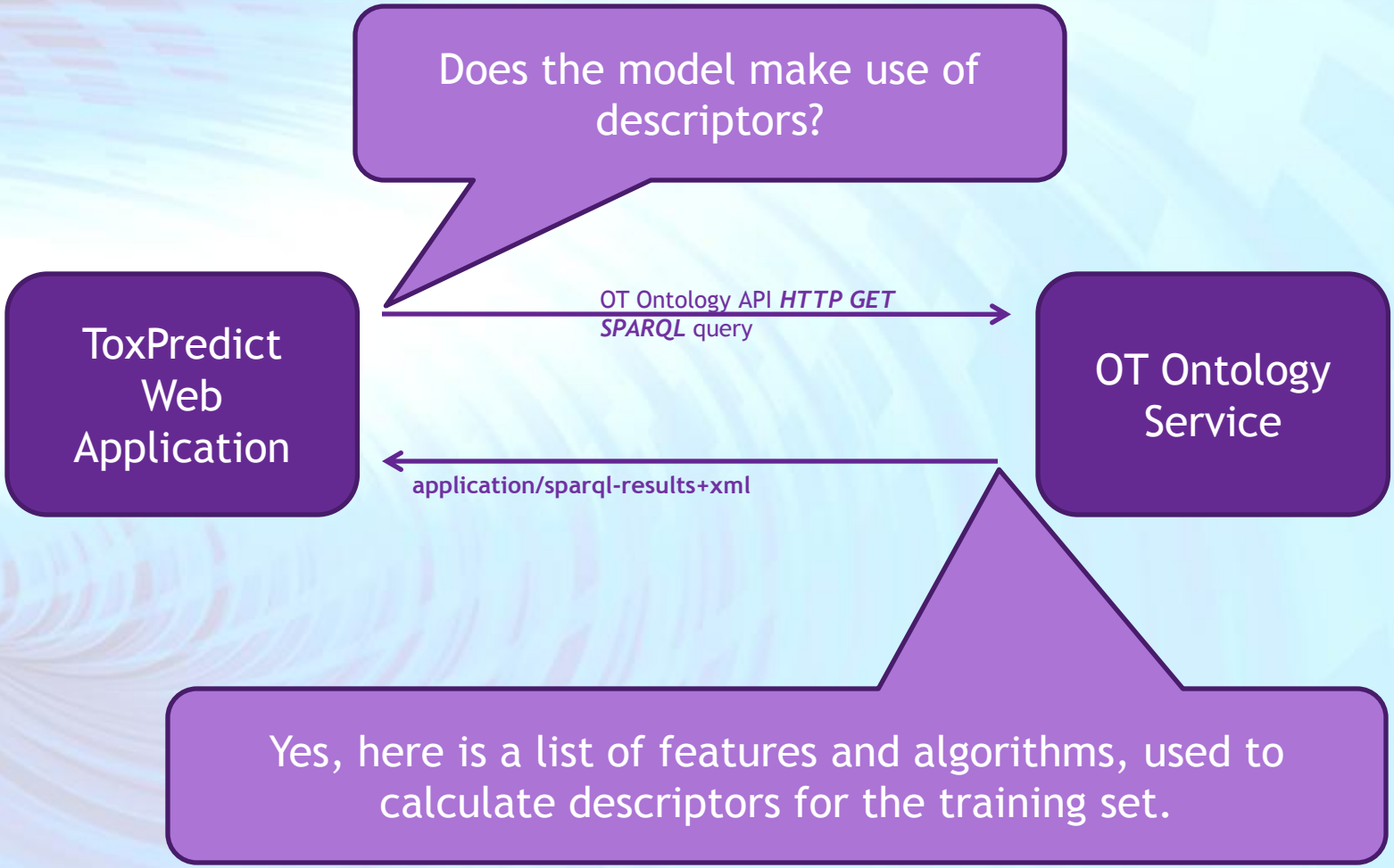
OT Task Service

HTTP code 200 "OK"
Returns Dataset URI

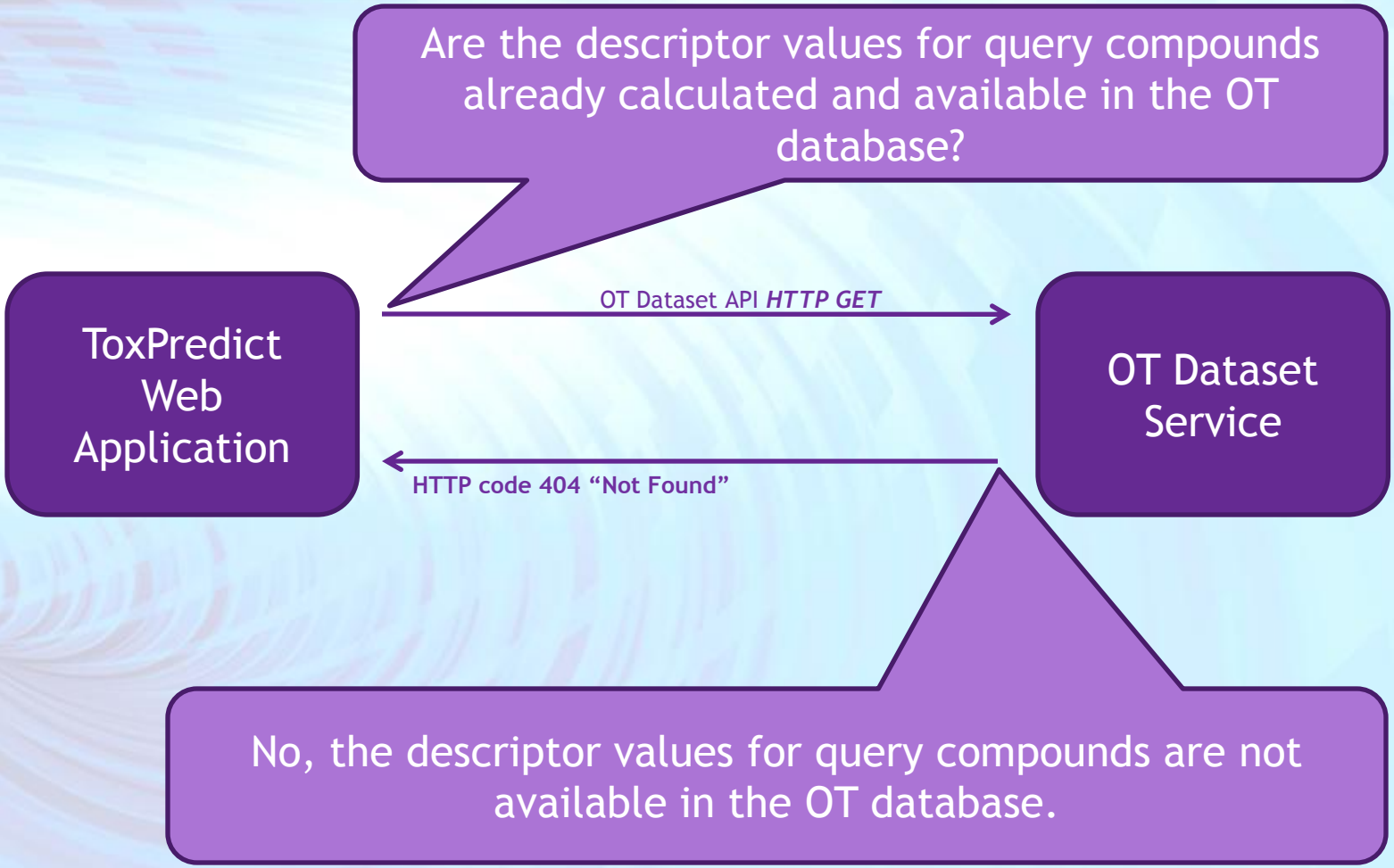


Yes, here is the dataset URI of the results.

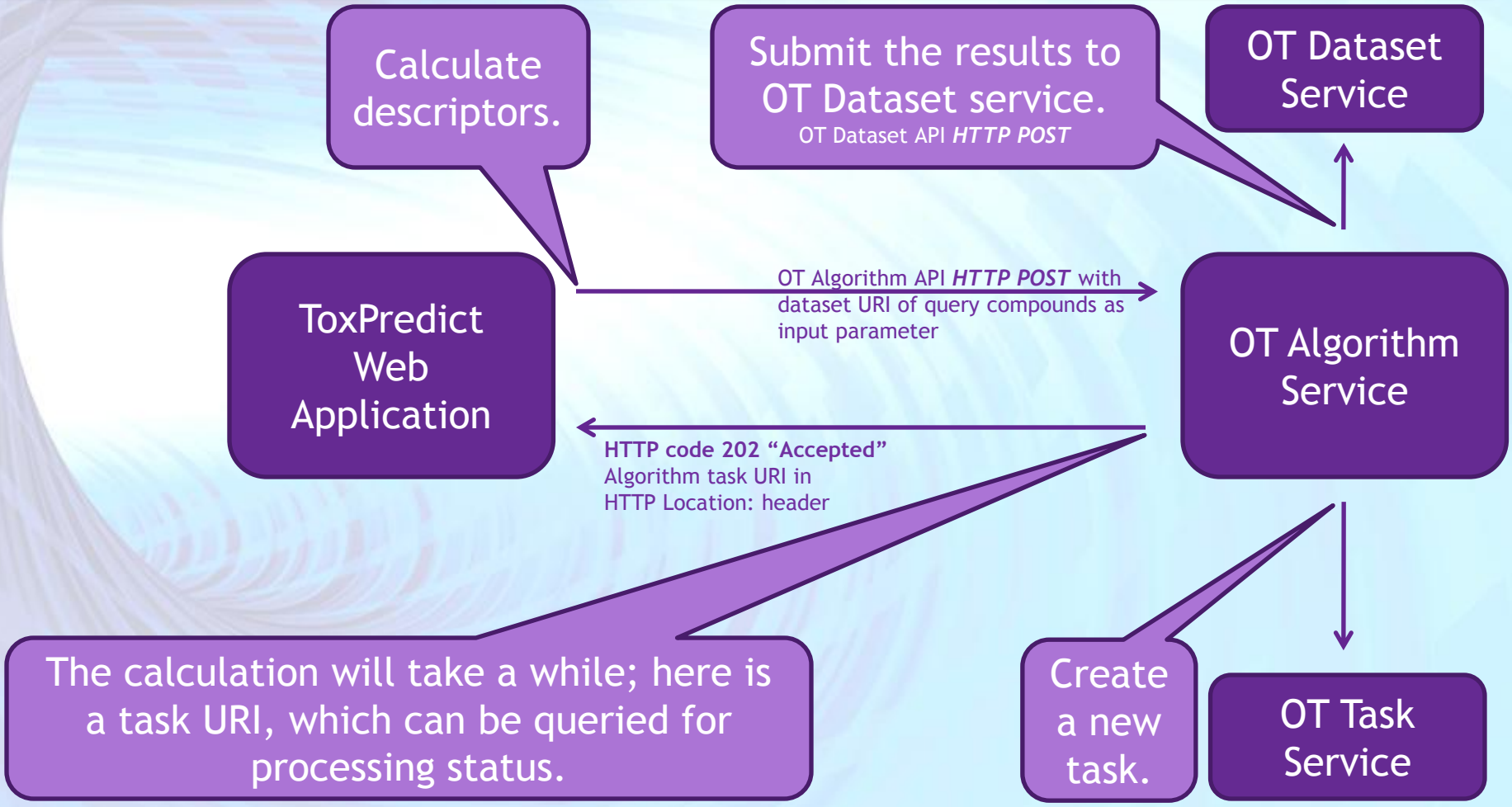
ToxPredict: Step 4 (behind the scenes)



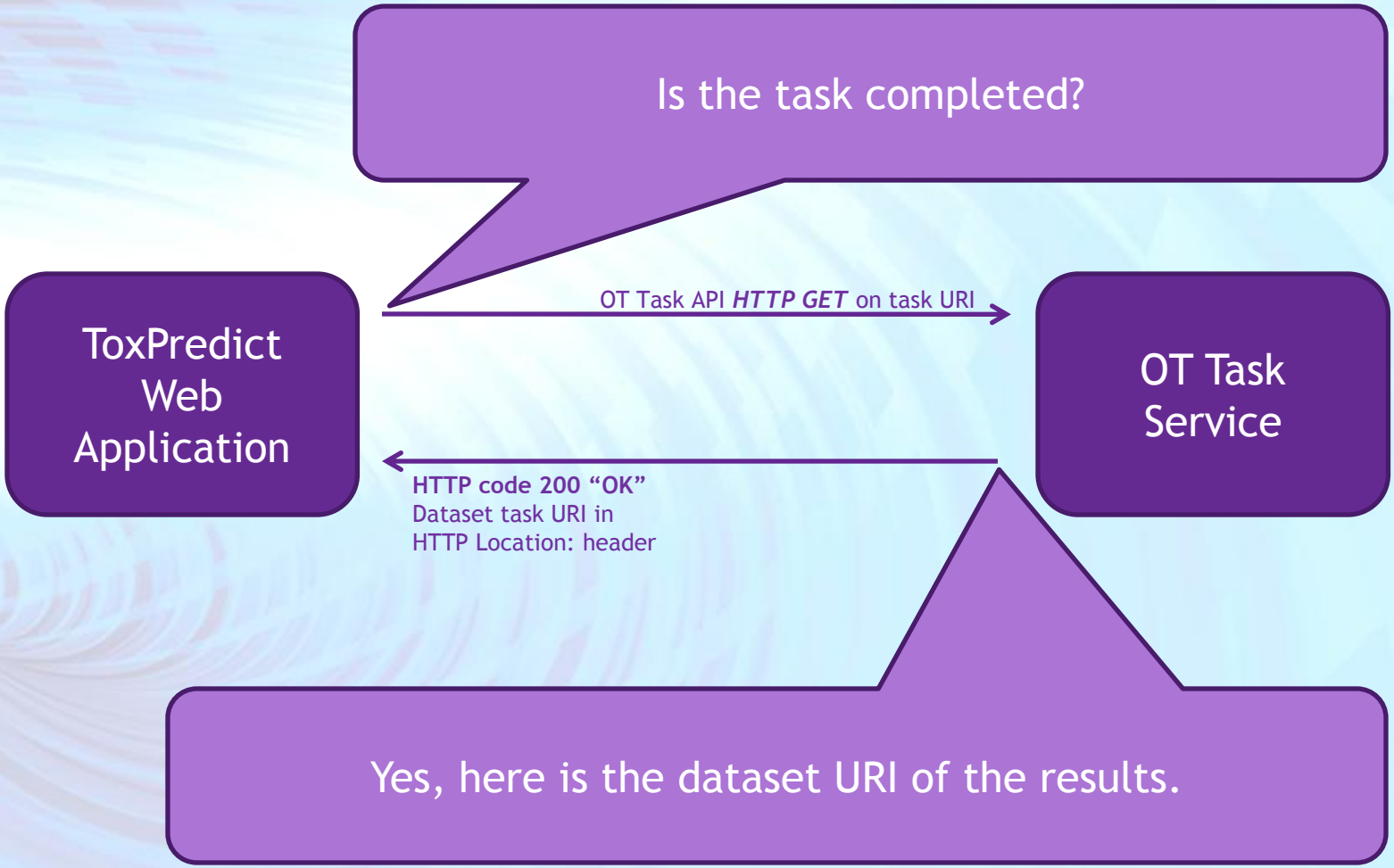
ToxPredict: Step 4 (behind the scenes)



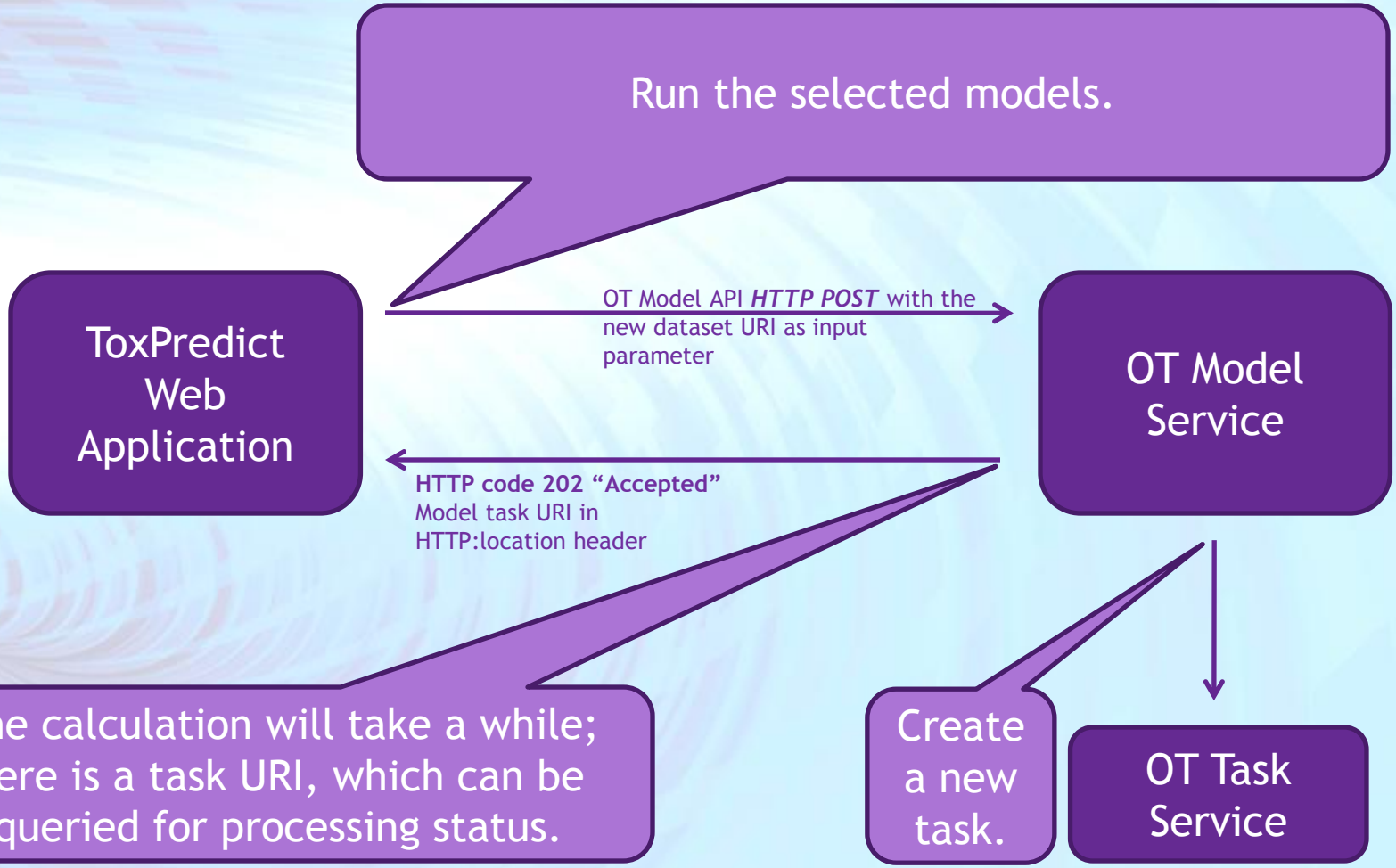
ToxPredict: Step 4 (behind the scenes)



ToxPredict: Step 4 (behind the scenes)

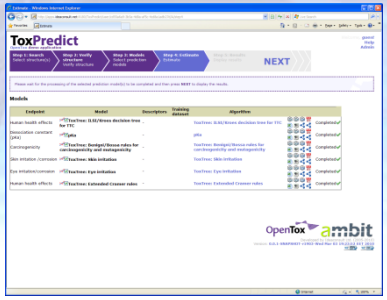


ToxPredict: Step 4 (behind the scenes)



… the remaining part of the Estimation step has been described

ToxPredict: Step 5 (Display results)



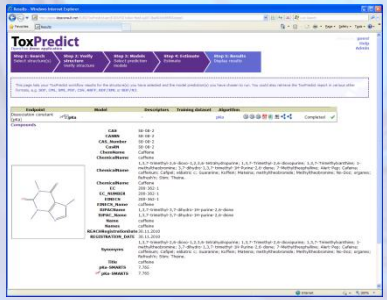
Retrieve calculation results from the final dataset URI, obtained in Step 4 (Estimation).

ToxPredict Web Application

OT Dataset API ***HTTP GET***

OT Dataset Service

application/rdf+xml



Here is the dataset content in RDF format, according to OpenTox.owl and containing estimation results, as well as compound identifiers and experimental data.

Thank you!