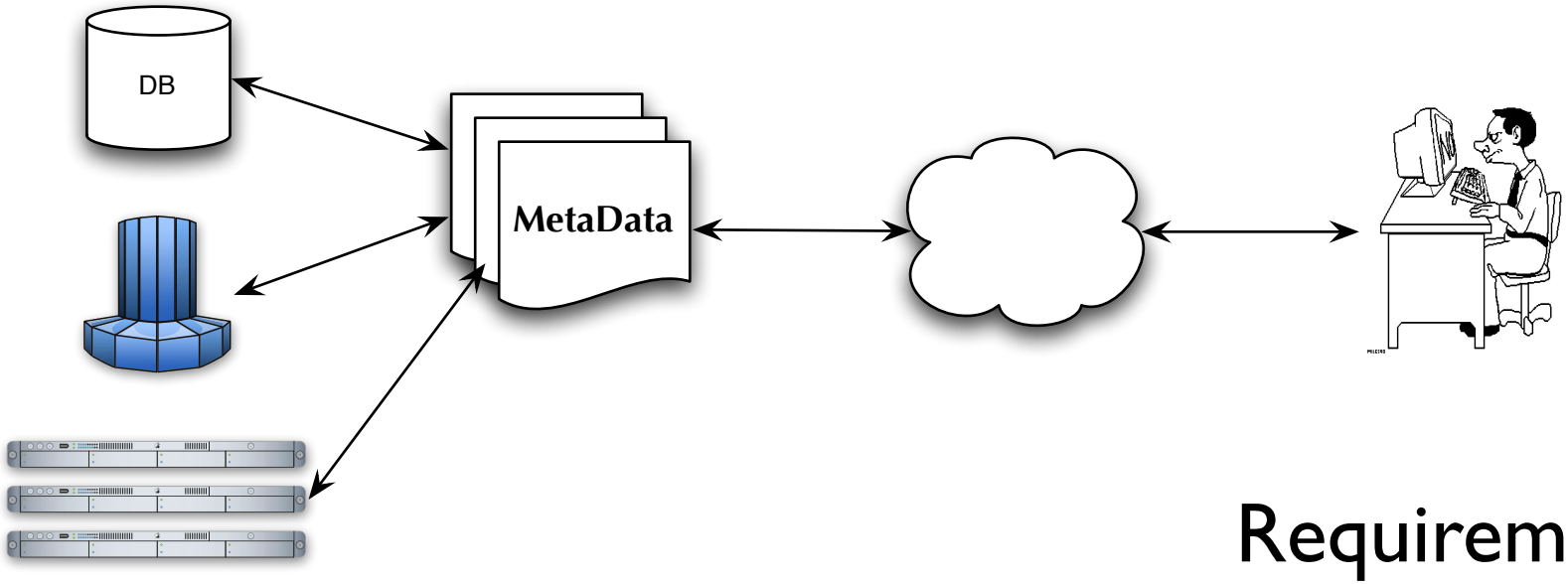
Search for CMS data: rapid web development using python and AJAX

Valentin Kuznetsov
Cornell University

Introduction

LHC experiments set a new scale for data management

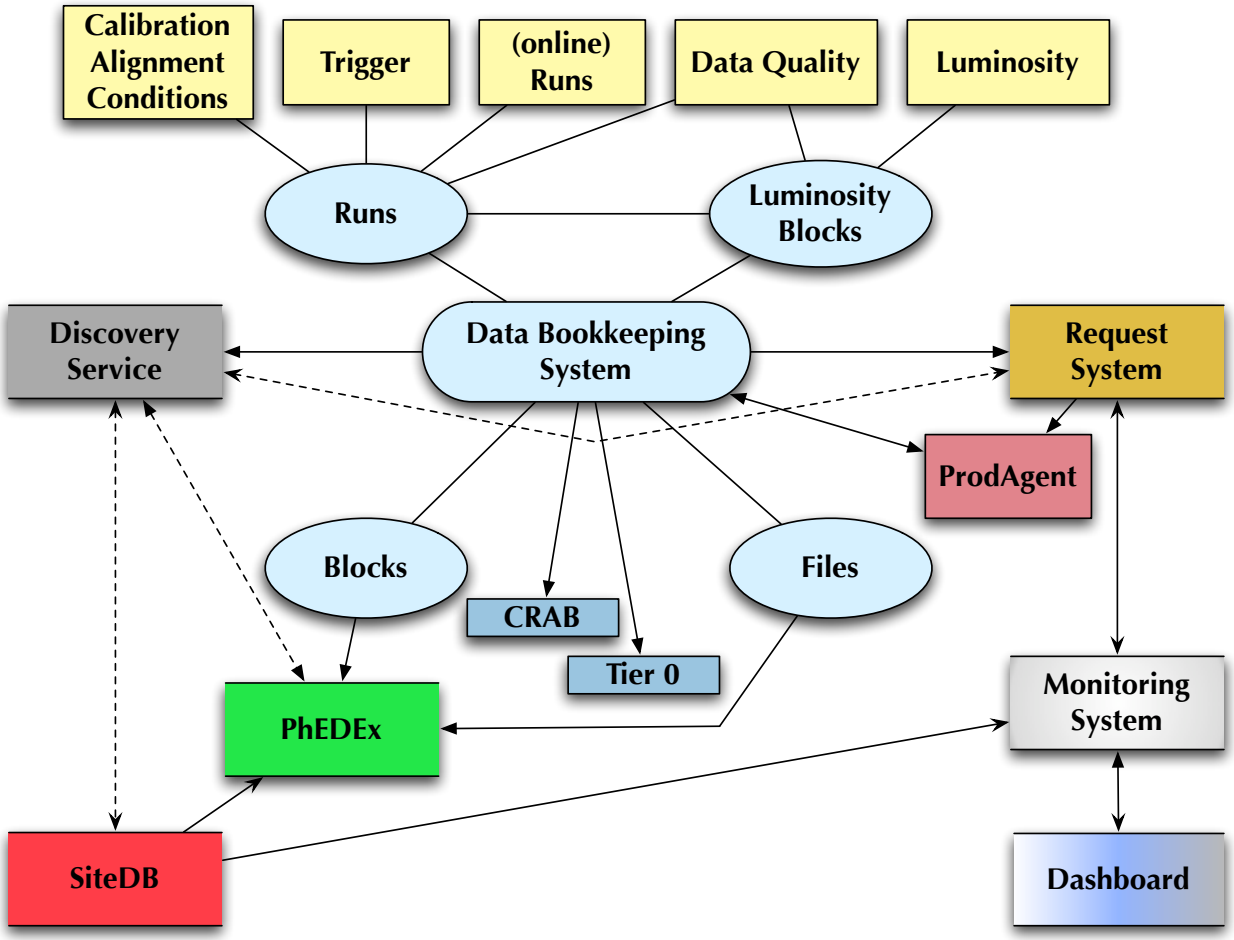


Requirements:

- Easy data access
- Dynamic content and navigation
- intelligent data query
- integration with other applications

DBS within CMS

Data Bookkeeping System (DBS) is the authoritative source of information about CMS event data



CMS data search is done via web interface (users) and DBS API (applications) both written in python and well integrated with CMS data model

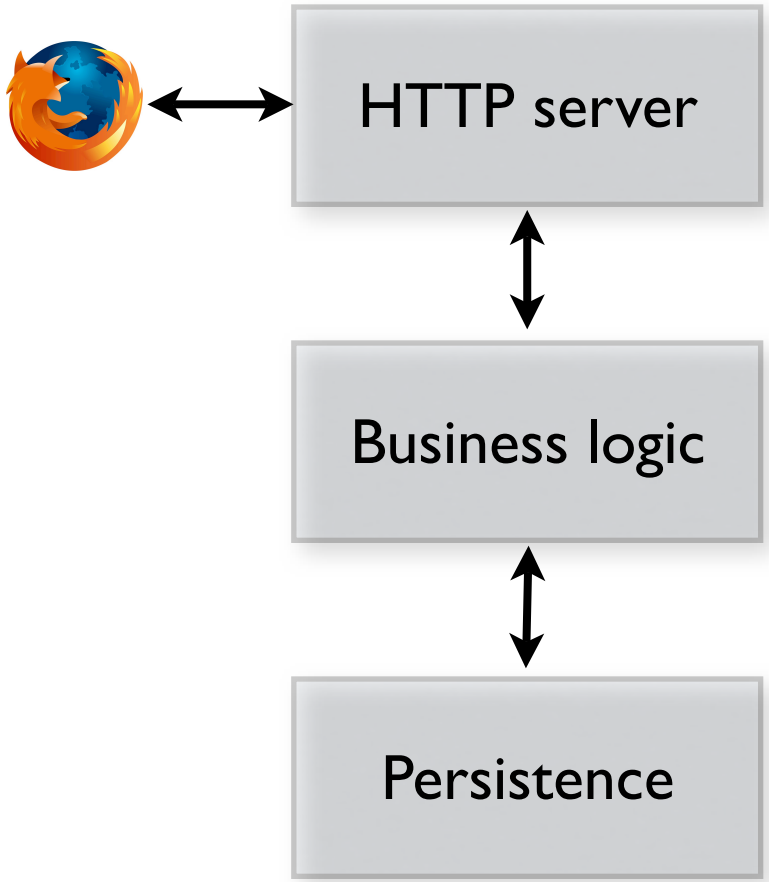
CMS data model

- Distributed & file centric
- LFN/PFN logical/physical file names
- File Block holds group of files and used for transfer
- A dataset defines a single sample produced with one set of configuration files
- Users' interests:
  - Physicists in datasets
  - production managers in datasets, file blocks
  - site admins in LFN/PFN at a given site

Building web applications

- Tons of tools: J2EE, .NET, PHP, Ruby on Rails, Python frameworks, etc.
- Requirements:
  - rapid turn around
  - easy to use (convention over configuration)
  - easy to develop and maintain
  - SQL generation
  - validation, security, etc.

n-tier architecture



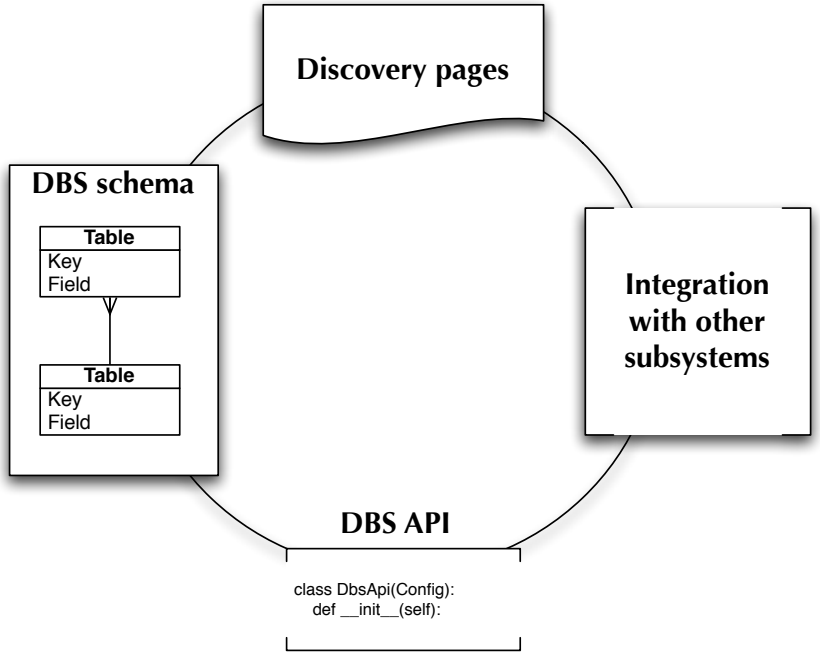
http://oodt.jpl.nasa.gov/better-web-app.mov

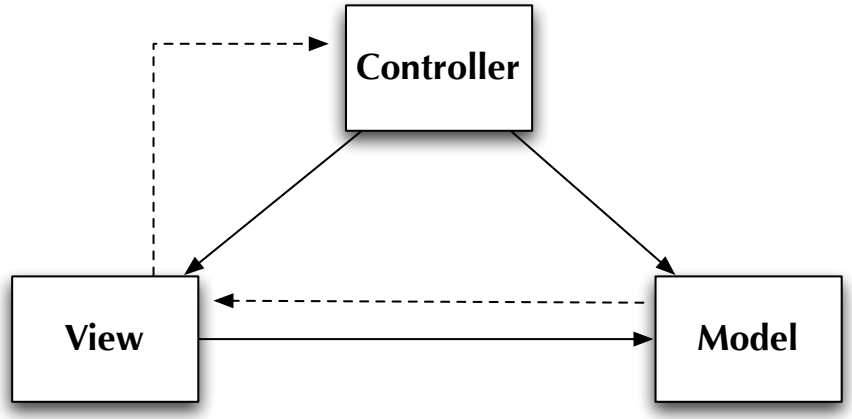
Choice

- DBS Core (data management) done using Java and Python
  - Java Servlets under Tomcat (server), python API (client), see Lee Lueking talk "The CMS DBS", this session
- Web Service (data discovery) done using Python and AJAX
  - Python is widely adopted, used almost everywhere in CMS
    - all tools for web development were freely available and well tested
    - we did not used particular web framework, instead we used individual components and model-view-controller architecture
  - AJAX (Asynchronous JavaScript and XML) technology helps funnel a large set of information into manageable sizes on a web pages

Development model

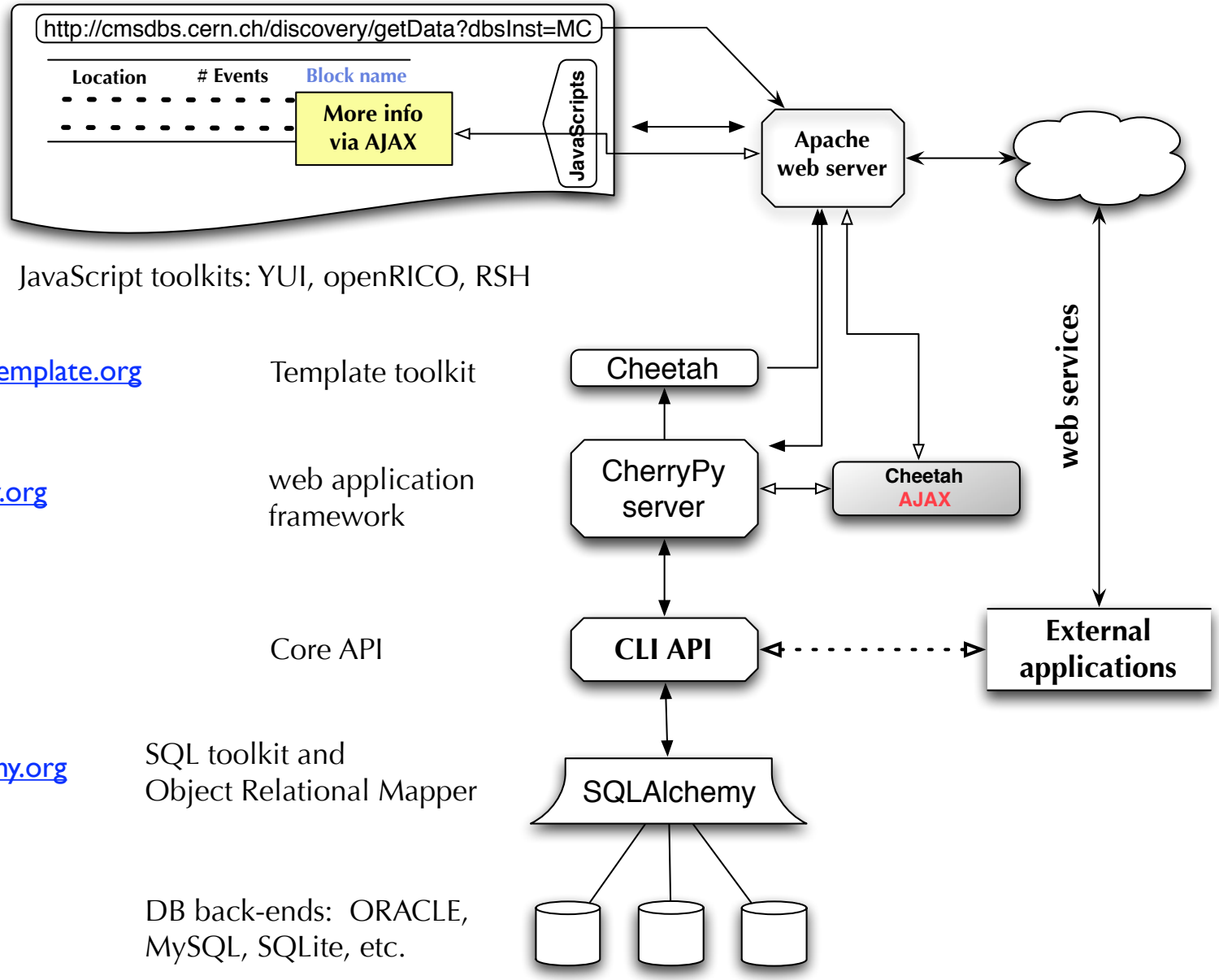
Stackable approach separates underlying data and logic from the actual presentation of the content (model-view-controller architecture)





DBS API, schema and discovery tools were developed together based on detailed use case analysis and using Python as a common "glue" for integration

Tools & workflow



JavaScript toolkits: YUI, openRICO, RSH

www.cheetagtemplate.org — Template toolkit

www.cherrypy.org — web application framework

Core API

www.sqlalchemy.org — SQL toolkit and Object Relational Mapper

DB back-ends: ORACLE, MySQL, SQLite, etc.

Advantages of using python/AJAX

- All components were ready to use
  - web server, SQL layer, template engine, frameworks
- Fast turn around with requirements, schema changes, use cases, views and scopes
  - modify user interface via templates and AJAX
  - changes to database implementation done via SQL layer
  - user interaction affects only web server (CherryPy)
- Easy to write and simplicity to use
- Concentrate on developing of search interface instead of debugging the code

Example

AJAX callback (JavaScript using RICO framework)

```
function getData(data) {
   ajaxEngine.sendRequest('getData',"data="+data);
}
function registerAjaxCalls() {
ajaxEngine.registerRequest('getData','ajaxResponse');
ajaxEngine.registerAjaxElement('dataHolder');
}
```

Web server (using CherryPy)

```
import cherrypy, MyTemplate
from   Cheetah.Template import Template

class Main(object):
    def ajaxResponse(self,data,**kwargs):
        cherrypy.response.headerMap['Content-Type']="text/xml"
        return """<ajax-response>
                  <response type="element" id="dataHolder">
                  <div>We got data: '%s'</div>
                  </response></ajax-response>"""%data
    ajaxResponse.exposed=True
    def index(self):
        t = Template(MyTemplate,searchList=[{'iList':[1,2]}])
        return str(t)
    index.exposed = True

cherrypy.quickstart(Main(),'/')
```

MyTemplate (using cheetah)

```
<div id="dataHolder"></div>
#for item in $iList
<a href="#" onclick="getData('$item')">Ajax call $item</a>
#end for
```

Data discovery

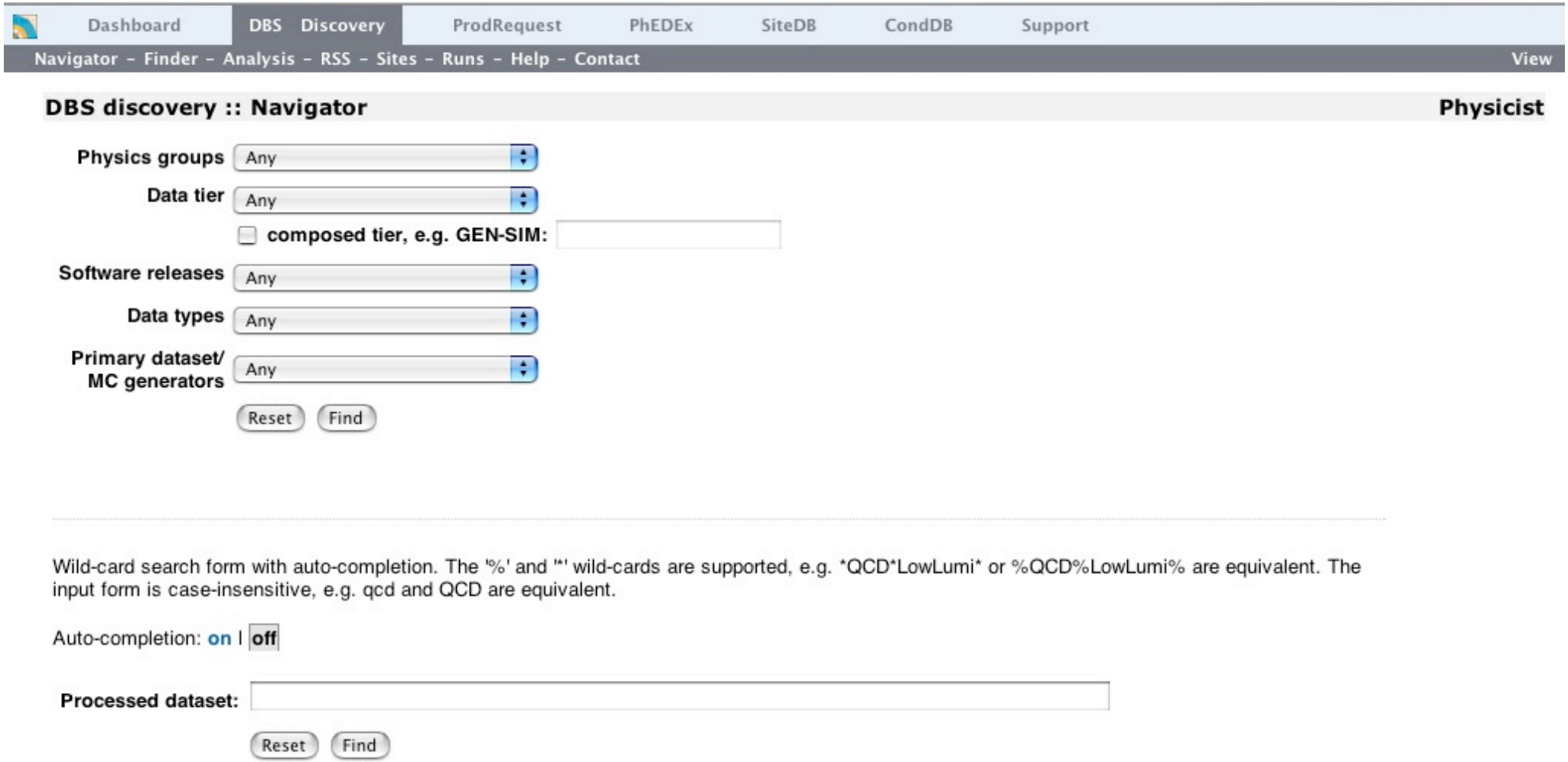
Read-only web service to find out CMS data

http://cmsdbs.cern.ch/DBS2_discovery/

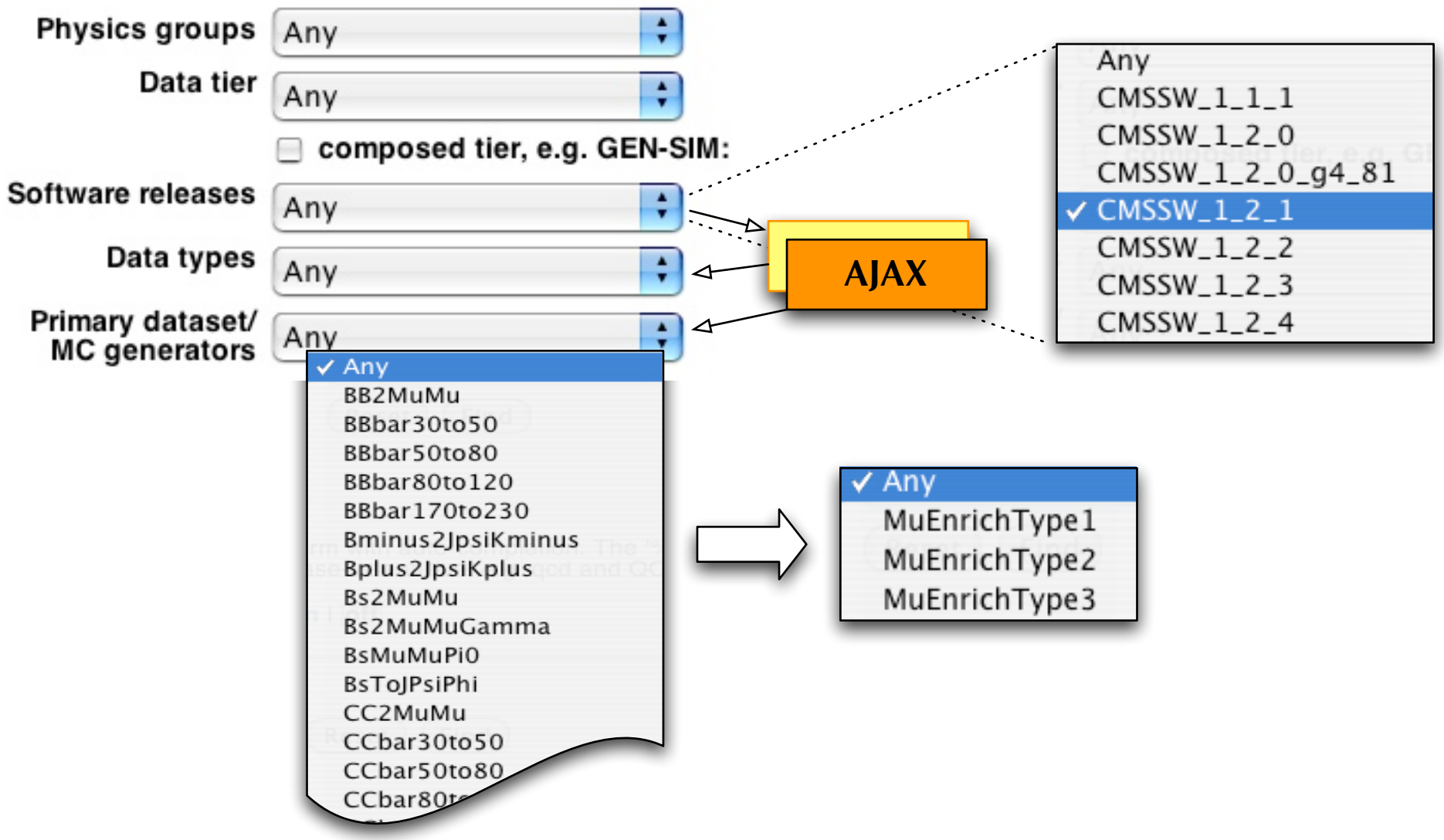
- Scopes define visibility of data within collaboration
- Views provide a different level of details over your data lookup
- Specific search options:
  - **Navigator** is a menu-driven approch
  - **Finder** constructs your own queries
  - Site/Run search and analysis dataset lookup

Regardless which way you go your data is just one click away

Navigator interface



Navigator & AJAX



Finder interface (advanced search)

Dashboard | DBS Discovery | ProdRequest | PhEDEx | SiteDB | CondDB | Support

Navigator - Finder - Analysis - RSS - Sites - Runs - Help - Contact | View

DBS Discovery :: Finder - Builder | My queries | Demo **Physicist**

DBS treeview (expand)

- Algorithm
- Datasets
  - PrimaryDataset
    - lastmodificationdate
    - creationdate
    - name
    - annotation
    - description
    - startdate
    - enddate
    - type
    - createdby
    - lastmodifiedby
  - ProcessedDataset
  - AnalysisDataset
- Description
- Files

Apply conditions: HELP

(PrimaryDataset.name <operator> '%Higgs%')

Query limit: 20 Reset Search

save query as: Save

To use Finder follow those three steps

- Use DBS tables treeview on your left to choose a columns you wish to look at
- Click on column name to add it to **Apply condition** input area
- Fill your condition by placing appropriate operator and value

Condition has a form:

Table.Column <operator> '<value>'

Leave spaces between Table.Column operator value

Supported operators: >, <, >=, <=, *like*

To match a pattern use %,
e.g. *TEST%* will match TEST with everything else

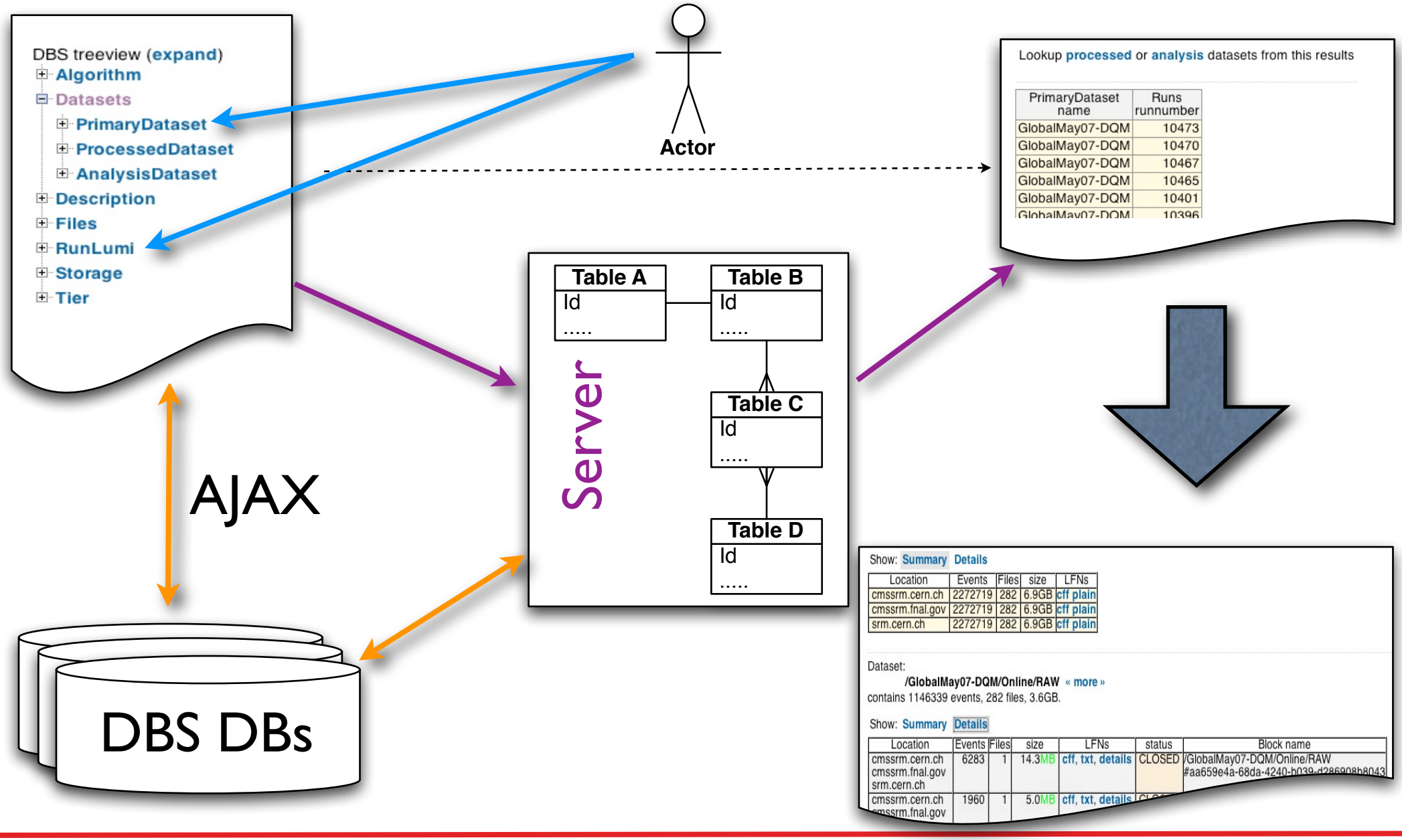
Several conditions can be grouped together using *AND, OR and brackets*
e.g., condition1 AND (condition2 OR condition3)

Explore MetaData

Place your condition

Query builds dynamically

Finder under the hood



Status

- Data Discovery service is in production for almost a year
  - web and CLI interfaces
- No significant problems with performance
- Can be deployed at any site with local DBS via RPM
- Configurable:
  - can be run behind apache or in stand-along mode
- Expandable:
  - external information, e.g. PhEDEx or run summary info added via AJAX mechanism
  - works with different DB back-ends, ORACLE/MySQL/SQLite

Summary

- Web development using python & AJAX is a simple and powerful tandem to accommodate a new scale in data retrieval in HEP community
- The correct choice of architecture and tools allowed us to concentrate on the presentation layer and achieve flexibility in data search
- The approach was tested during CSA06 and CSA07 data challenges and adopted by other web tools in CMS, with full integration among them into common framework