CHEP – Sofia, July 9-13th 2018

Broadcasting dynamic metadata content to external web pages using AMI embeddable components

LPSC Grenoble – Laboratoire de Physique Subatomique et de Cosmologie
CNRS IN2P3 Les deux infinis
CERN

The ATLAS Metadata Interface[1] (AMI) is a mature ecosystem more than 15 years old. As part of AMI Web Framework (AWF), we describe a cross-domain system of embeddable HTML5 controls, based on the AJAX and CORS technologies, to share up-to-date metadata with other content services such as wikis.

What is AMI?

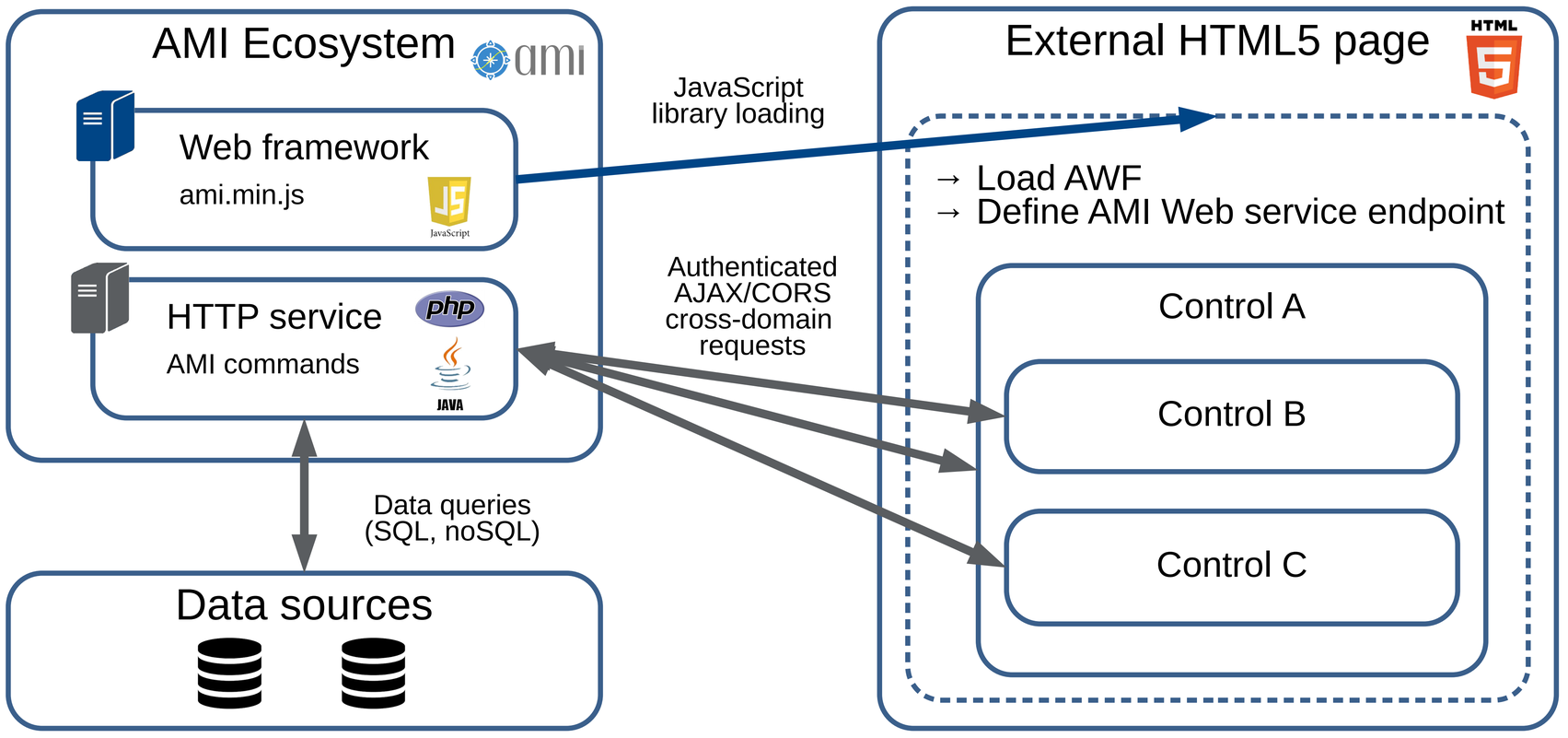
AMI is a generic ecosystem, **used by ATLAS and other scientific experiments**, to quickly design **metadata-oriented** applications. It provides facilities for aggregating and searching data with easy-to-use Web interfaces and lightweight clients.

AMI Web Framework (AWF)

AWF is a **JavaScript framework**, part of AMI ecosystem, based on modern technologies such as JQuery, TWIG, Twitter Bootstrap, … It provides ready-to-use controls for Web applications and a tool to generate custom applications and control skeletons.

Embeddable control mechanism

- The AMI ecosystem has a unique HTTP service endpoint to access the data.
  - It is configured server-side to allow cross-domain HTTP requests using the CORS technology. Consequently, data can be accessed from anywhere.
  - It can simultaneously connect to several database systems, making possible to aggregate data from various sources.
- The AMI Web Framework (AWF) is another part of the ecosystem. It is written in JavaScript 6.
  - It is a library totally independent from the HTTP service and can be hosted on a separate server.
  - It can be loaded and used by any modern Web browser.
- AWF offers features to easily embed controls in external pages.
  - Controls interact with the HTTP service using AJAX asynchronous requests. Thus user interfaces are very dynamic and responsive.
  - AWF has a set of standard controls. They can be combined to create composite custom controls, fitting user needs.



How to embed an AMI controls in a web page?

```
<head>
    <script type="text/javascript" src="https://ami.in2p3.fr/app/js/jquery.min.js"></script>
    <script type="text/javascript" src="https://ami.in2p3.fr/app/js/ami.min.js?embedded"></script>

    <script type="text/javascript">
        var myTable;

        amiWebApp.onRefresh = function(isAuth) {

            if(isAuth)
            {
                var result = new $.Deferred();

                myTable.render('#myDiv',{parameter1: paramvalue1, parameter2: paramValue2,...}).always(function(){
                    result.resolve();

                });

                return result;
            }
            else
            {
                $('#myDiv').empty();
            }
        };

        amiWebApp.onReady = function() {

            var result = new $.Deferred();

            amiWebApp.loadResources(['ctrl:table']).done(function(controls){
                myTable = new controls[0];
                result.resolve();

            }).fail(function(){
                result.reject();
            });

            return result;
        };

        amiWebApp.start({
            endpoint_url: 'https://ami.in2p3.fr/AMI/servlet/net.hep.atlas.Database.Bookkeeping.AMI.Servlet.FrontEnd'
        });

    </script>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <ul class="navbar-nav ml-auto" id="ami_login_content"></ul>
    </nav>

    <div id="ami_alert_content"></div>
    <div id="myDiv"></div>
</body>
```

Importing JQuery and AWF JavaScript libraries.

When a user authenticate using the “Sign In” control (with credentials or certificate), the web framework triggers the “onRefresh” event where the loaded controls render their HTML contents.

When the Web framework is loaded and ready, it triggers the “onReady” event where the resources are loaded (controls, css, twig files, ...).

Initializing and starting the Web framework. The endpoint URL is specified there.

HTML body where are defined:
- the “ami_login_content” HTML “div” to display the authentication control.
- the “ami_alert_content” HTML “div” to display warning or error messages from the Web Framework.
- the HTML “div” to display the AMI control generated content.

Develop your own AMI controls

Getting started with AWF

```
curl https://rawgit.com/ami-team/AMIWebFramework/master/tools/build.xml > build.xml
ant setup-prod

ant make-home-page

ant make-control
```

It sets-up:
- the AWF environment
- a Web App skeleton
- a new control skeleton

Control pattern structure

AWF API

AWF permits developing applications and controls with an **MVC** pattern.

The **MODEL** is based on the CORS cross-domain AMI Web Service (AMI commands). It makes it possible to embed AMI controls anywhere.

The **VIEW** is based on the AMI-Twig template engine and the Twitter Bootstrap 4 CSS framework. It dynamically generates HTML5 fragments.

The **CONTROLLER** uses standard JavaScript technologies such as AJAX, JQuery, JSPath, …

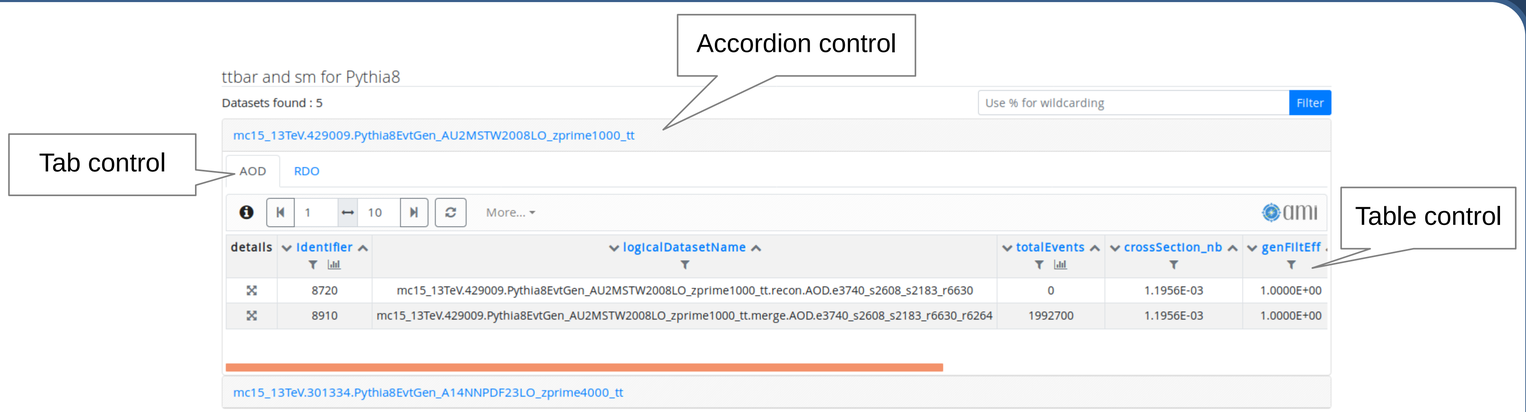
VIEW fragment example: AMI-Twig and JSPath in action!

```
{% for row in linkedElementRowset %}
    <tr>
        <td class="text-center">
            {{row|json_jspath('..field{.@name==="entity"}.$')[0]|e}}
        </td>
        <td class="text-center">
            {{row|json_jspath('..field{.@name==="count"}.$')[0]|e}} records
        </td>
    </tr>
{% endfor %}
```

ATLAS AMI controls in action!

details	AMITag	tagType	tagNumber	SWReleaseCache	transformationName
	r10529	r	10529	Athena_21.0.67	Reco_tf.py
	e6736	e	6736	MCProd_19.2.5.30.1	Generate_tf.py
	c1240	c	1240	Athena_21.0.71	NTUPMerge_tf.py
	r10528	r	10528	Athena_21.0.68	Trig_reco_tf.py
	e6735	e	6735	MCProd_20.7.9.9.9	Generate_tf.py
	r10527	r	10527	AthenaP1_21.1.30-21.1-2018-05-22T2138	Trig_reco_tf.py
	c1239	c	1239	Athena_21.0.71	LArNoiseBursts_tf.py
	c1238	c	1238	Athena_21.0.71	NTUPMerge_tf.py
	c1237	c	1237	Athena_21.0.71	/afs/cern.ch/atlas/tzero/software/trfs/LAr/LArMerg
	c1236	c	1236	Athena_21.0.71	HISTMerge_tf.py

A standard AWF control: the “Table” control.



A composite control: the “PMG view” control for ATLAS collaboration.
It gives ATLAS physicists an easy view to information on MC datasets from their own documentation.
It is made from base controls like accordion, tab and table controls.

J Fulachier, F Lambert, J Odier (LPSC – Grenoble),
on behalf of the ATLAS Collaboration.

Refs.: [1] J Odier and O Aidel and S Albrand and J Fulachier and F Lambert.
Evolution of the architecture of the ATLAS Metadata Interface (AMI).
J. Phys.: Conf. Ser., vol. 664, num. 4, year 2015.
http://stacks.iop.org/1742-6596/664/i=4/a=042040

ATLAS EXPERIMENT