The DIRAC Web Portal 2.0

Z Mathe,1 A Casajus Ramo,2 N. Lazovsky,2 F.Stagni1

1 CERN, European Organization for Nuclear Research, Switzerland
2 University of Barcelona, Barcelona, Spain

E-mail: zoltan.mathe@cern.ch

Abstract. For many years the DIRAC interware (Distributed Infrastructure with Remote Agent Control) has had a web interface, allowing the users to monitor DIRAC activities and also interact with the system. Since then many new web technologies have emerged, therefore a redesign and a new implementation of the DIRAC Web portal were necessary, taking into account the lessons learnt using the old portal.

These new technologies allowed to build a more compact, robust and responsive web interface that enables users to have better control over the whole system while keeping a simple interface. The web framework provides a large set of "applications", each of which can be used for interacting with various parts of the system. Communities can also create their own set of personalised web applications, and can easily extend already existing ones with a minimal effort. Each user can configure and personalise the view for each application and save it using the DIRAC User Profile service as RESTful state provider, instead of using cookies.

The owner of a view can share it with other users or within a user community. Compatibility between different browsers is assured, as well as with mobile versions. In this paper, we present the new DIRAC Web framework as well as the LHCb extension of the DIRAC Web portal.

1. Introduction

The DIRAC interware [1, 2] is a software framework which delivers a complete solution for managing distributed heterogeneous computing resources such as Grid, Cloud and DIRAC site (a cluster which has a Batch system such as Portable Batch System (PBS), Torque, etc.) Different experiments are using DIRAC, the biggest one is LHCb. LHCbDIRAC [3] is an extension of DIRAC, which contains LHCb specific code. DIRAC provides a complex web user interface (WUI), which is part of the DIRAC Web framework, in addition to the command line interface (CLI). DIRAC WUI hides the complexity of the DIRAC system. DIRAC Web framework provides the facility to develop, manage and test web applications. Web applications are used to monitor and interact with DIRAC components like services, agents and executors. In addition, these web applications allow monitoring and controlling the underlying computing resources, as well as the DIRAC system. Web applications can be used by experienced and casual users to define and monitor data analysis jobs in a user friendly manner.

In this paper we present the new DIRAC Web framework, which was recently developed by taking into account the lessons learnt using the old portal.

In section 2 we present the technologies used to develop the DIRAC Web framework and its web applications. In section 3 we present the architecture of the system and in section 4 follows a detailed presentation of the DIRAC Web framework. In section 5 we present how to...
extend an existing web application and develop a new one. In section 6 we present a tool used to optimise the JavaScript code in order to reduce the network traffic.

2. Technology
The key technologies used to develop the framework are JavaScript, HTML and CSS for the front-end. The back-end is developed in python.

Front-end:
Nowadays, various JavaScript libraries are available: jQuery[4], AngularJS[5], Dojo[6], ExtJs[7], etc. Taking into account the extensibility and manageability of a web application, we decided to use a JavaScript library which supports the object oriented paradigm. From the list of criteria mentioned above only ExtJs and Dojo have very good object oriented approach compared to other graphical libraries what we evaluated (Qt[8], GTK+[9], Java Swing[10], etc.). We decided to use ExtJs JavaScript library, because we already have good experience using it and in addition it provides rich graphical widgets in a single library compared to others.

Back-end:
We use Tornado[11] non-blocking web server and web application framework written in Python. On top of Tornado we use Nginx[12], which is load balancer, HTTP cache and a web server, in addition it is also a reverse proxy server for HTTP(S).

The web applications use AJAX to communicate with the web server. However, the WebSocket[13] protocol is used to exchange information through a persistent connection in order to keep the consistency of the information being modified in the front-end.

Google Visualization API[14] is used for creating charts in addition to ExtJs chart package[15].

3. Architecture
The architecture of the DIRAC web portal is divided into three layers: presentation, service and data layers. Figure 1 shows a diagram of the chosen architecture. The presentation layer is implemented in ExtJs and provides rich web applications. The UML class diagram shown in Figure 2 describes the structure of the presentation layer.

Ext.dirac.core is the core component, which implements the core functionality of the framework by providing well defined interfaces and classes used to unify the system. Ext.dirac.views is the implementation of the AppView abstract interface, which provides all graphical elements used by the users, and in addition it is responsible for managing the web applications. Currently, it has two implementations, but a new view can be easily implemented and integrated within the framework. Ext.dirac.utils contains predefined widgets, which are extension of the ExtJs widgets and can be used to develop web applications.
applications. These predefined widgets simplify the development of the new web applications as well as providing the save and load mechanism. *StateManagement* manages the state of the web applications by communicating with DIRAC User Profile service, which stores the states of the web applications and desktops. *Stateful* is an interface, which defines the methods used to load and save the state of the web applications and desktops. The *Module* class is the connection between the web application and the *Ext.dirac.core*. It is responsible to load the CSS files of each web application. The web applications are derived from the *Module* overriding the *buildUI* method is used to build the user interface by instantiating ExtJs widgets. The instances are added to the web application in the constructor of the web application. The *buildUI* method always called after when all CSS files are loaded.

*DIRAC* is a package, which contains a set of predefined web applications. *LHCbDIRAC* is the LHCb specific package. We use a global variable called *Global*, which contains the important settings of the framework.

The service layer is implemented in python and it is a wrapper on top of the Tornado web server. The *WebHandler* is an extension of the Tornado *RequestHandler*, which implements the authentication mechanism taking into account the user credential and properties using DIRAC Secure Transport (DISET)\[16\] protocol. The service layer contains the logic of the web application and responds to the client requests by interacting with a dedicated DIRAC service. In order to provide high availability we use Nginx on top of Tornado. The data layer is the source of the information. The DIRAC services are part of the data layer.

While the authentication performs in service layer, the authorization performs in data layer.

---

**Figure 2.** UML class diagram
by a DIRAC service. The users without a valid certificate or without registered in DIRAC have limited access to the system. The data format, which is sent by the server layer from or to the client layer is in JSON format, while to exchange data between the server and data layer the DISET protocol is used.

4. DIRAC web framework

The DIRAC web framework provides all the facilities, which are needed to develop and test robust web applications.

   The following terminologies are used:
   
   • An Application is a standalone web application, which is used to interact with the DIRAC system.
   
   • A Desktop is a container of web applications. Each web application is assigned to a desktop. It is the working environment.

   • A State is the actual status of an application or a desktop.

   • A Theme is a graphical appearance of the WUI. Two themes are provided: Desktop and Tab themes. Both provide the same functionalities. The difference is in the way the applications are managed: the Desktop theme only allows to work with a single desktop while the Tab theme allows to work with multiple desktops at the same time. Figure 3 shows a desktop with three web applications in the Desktop theme. Figure 4 shows the same desktop in the Tab theme with the same web applications.

The users can easily switch between different look and feels (for example: figure 3, 6) of the WUI, switch between different DIRAC systems as well as they can change their properties. The web configuration file is used to configure the DIRAC web portal in addition to the Configuration Manager. The Configuration Manager is a web application used to modify the remote configuration handled by the DIRAC Configuration service (CS) [10] [17]. It is used to configure: the theme of the WUI (the users or Virtual Organizations (VOs) can set the default
theme, list of web application and menu structure, external links and enable/disable the load balancer. Currently, Nginx is supported.

The web applications and desktops can be customized by the users and can be saved by using the User Profile service. Each user has a User Profile, which stores information relative to the users. Because the state of a desktop or web application is centrally managed, the user can open the same desktop or web application in different browsers. The following browsers are supported: Firefox, Safari, Opera, Chrome.

In addition any mobile device can open the web applications. The personalized desktops and web applications can be shared within the same VO by selecting the Make public menu item shown in figure 6 or between the users through a reference string as shown in figure 8.

A shared desktop or web application can be shared and can be loaded using the Public State loader as shown in figure 7. The desktops or web applications, which are shared between the users, can be loaded using the State loader as shown in figure 8. After when a shared desktop or web application is loaded a reference to the original desktop or web application will be saved in the User Profile and can be accessible under the Shared menu as shown in figure 5.

5. Web applications
The DIRAC web framework provides a set of applications, while new applications can be developed, and existing web applications can be extended or changed.
Developing a new web application

Each web application has an associated handler, which is listening on the web server and serving requests coming from the web browser. The new handler has to be derived from WebHandler and has to provide all functionalities required to return the data to the web browser. In the client level the new application has to be derived from Ext.dirac.core.Module and the WUI has
to be implemented inside the buildUI method. The state of the web applications can be saved by overriding the getStateData method. The loadState method can be used to load the saved state of an application or desktop stored in User Profile. The new applications will automatically be available in the WUI after adding them to the web configuration file.

**Extending an existing web application**

The inheritance is used to extend an existing application. Both server and client side the new class has to be derived from the class that need to be extended. New functionalities can be added, removed or replaced easily. The following example illustrates how to add extra functionalities to the LHCbJobMonitor page:

```javascript
Ext.define('LHCbDIRAC.LHCbJobMonitor.classes.LHCbJobMonitor', {
    extend: 'DIRAC.JobMonitor.classes.JobMonitor',
    initComponent: function() {
        var me = this;
        me.callParent();
        me.launcher.title = "LHCb Job Monitor";
    },
    buildUI: function() {
        var me = this;
        me.callParent();
        me.leftPanel.addTextFieldSelector({
            "RunNumbers": {
                name: "Run Number(s)",
                type: "number"
            }
        });
        ....
    }
});
```

6. **Sencha Cmd**

Sencha Cmd [18] is a cross-platform command line tool, which provides a full set of life cycle management features such as Code generation tools, JS Compiler, Web Server, which can be used to build ExtJs, based web applications. We only take advantage of the usage of JS Compiler for making JavaScript code more efficient and for code checking. Sencha Cmd provides three tools used to optimize the JavaScript code:

(i) YUI Compressor [19]
(ii) Google Closure Compiler [20]
(iii) UglifyJS [21]

DIRAC web framework provides a command line interface, implemented on top of the JS Compiler used to compile the JavaScript code. Using the compiled JavaScript code, the numbers of client request are reduced and production quality code is ensured.

7. **Summary**

The new DIRAC web framework has a wide variety of usage. It provides a user friendly WUI, which is used to interact with DIRAC services. The users can customize the web applications
and save the state of the web applications and desktops in their User Profile. Since then, they can be opened in different browsers as well as in different mobile devices. The users can share the web application and desktops in an easy way.

References
[1] Casajus A, Ciba K, Fernandez V, Graciani R, Hamar V, Mendez V, Poss S, Sapunov M, Stagni F, Tsaregorodtsev A and Ubeda M 2014 *Journal of Physics: Conference Series* 396 032107 URL [http://stacks.iop.org/1742-6596/396/i=3/a=032107](http://stacks.iop.org/1742-6596/396/i=3/a=032107)

[2] Tsaregorodtsev A 2014 *Journal of Physics: Conference Series* 513 032096 URL [http://stacks.iop.org/1742-6596/513/i=3/a=032096](http://stacks.iop.org/1742-6596/513/i=3/a=032096)

[3] Stagni F, Charpentier P, Graciani R, Tsaregorodtsev A, Closer J, Mathe Z, Ubeda M, Zhelezov A, Lanciotti E, Romanovskiy V, Ciba K D, Casajus A, Roiser S, Sapunov M, Remenska D, Bernardoff V, Santanna R and Nandakumar R 2012 *Journal of Physics: Conference Series* 396 032104 URL [http://iopscience.iop.org/1742-6596/396/3/032104/pdf/1742-6596_396_3_032104.pdf](http://iopscience.iop.org/1742-6596/396/3/032104/pdf/1742-6596_396_3_032104.pdf)

[4] jquery URL [https://jquery.com/](https://jquery.com/)
[5] Angularjs URL [https://angularjs.org/](https://angularjs.org/)
[6] Dojo URL [http://dojotoolkit.org/](http://dojotoolkit.org/)
[7] Extjs URL [http://www.sencha.com/products/extjs/#overview](http://www.sencha.com/products/extjs/#overview)
[8] Qt URL [http://www.qt.io/developers/](http://www.qt.io/developers/)
[9] Gtk+ URL [http://www.gtk.org/](http://www.gtk.org/)
[10] Tornado URL [http://docs.oracle.com/javase/tutorial/uiswing/](http://docs.oracle.com/javase/tutorial/uiswing/)
[11] Nginx URL [http://nginx.org/](http://nginx.org/)
[12] WebSocket URL [https://www.websocket.org/](https://www.websocket.org/)
[13] Google charts URL [https://developers.google.com/chart/interactive/docs/index](https://developers.google.com/chart/interactive/docs/index)
[14] Extjs chart library URL [http://www.sencha.com/products/extjs/#overview](http://www.sencha.com/products/extjs/#overview)
[15] Diaz R G and Casajus A 2007 *Journal of Physics: Conference Series* URL [https://indico.cern.ch/event/3580/session/21/contribution/178/material/paper/0.pdf](https://indico.cern.ch/event/3580/session/21/contribution/178/material/paper/0.pdf)
[16] Tsaregorodtsev A, Bergiotti M, Brook N, Ramo A C, Castellani G, Charpentier P, Cioffi C, Closer J, Diaz R G, Kuznetsov G, Li Y Y, Nandakumar R, Parerson S, Santinelli R, Smith A C, Miguel M S and Jumenez S G 2008 *Journal of Physics: Conference Series* 119 062048 URL [http://iopscience.iop.org/1742-6596/119/6/062048](http://iopscience.iop.org/1742-6596/119/6/062048)
[17] Sencha cmd URL [https://www.sencha.com/products/sencha-cmd/](https://www.sencha.com/products/sencha-cmd/)
[18] Yui compressor URL [http://yui.github.io/yuicompressor/](http://yui.github.io/yuicompressor/)
[19] Google closure URL [https://developers.google.com/closure/compiler/](https://developers.google.com/closure/compiler/)
[20] Uglifyjs URL [https://github.com/mishoo/UglifyJS](https://github.com/mishoo/UglifyJS)