**Sequence analysis**

**Jflow: a workflow management system for web applications**

Jérôme Mariette¹*, Frédéric Escudié¹, Philippe Bardou², Ibouniyamine Nabihoudine¹, Céline Noirot¹, Marie-Stéphane Trotard¹, Christine Gaspin¹ and Christophe Klopp¹,²

1Plate-forme Bio-informatique Genotoul, INRA, UR875 Mathmatiques et Informatique Appliques Toulouse, Castanet-Tolosan, France and ²Plate-forme SIGENAE, INRA, GenPhyse, Castanet-Tolosan Cedex, France

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on June 30, 2015; revised on October 2, 2015; accepted on October 7, 2015

**Abstract**

**Summary:** Biologists produce large data sets and are in demand of rich and simple web portals in which they can upload and analyze their files. Providing such tools requires to mask the complexity induced by the needed High Performance Computing (HPC) environment. The connection between interface and computing infrastructure is usually specific to each portal. With Jflow, we introduce a Workflow Management System (WMS), composed of jQuery plug-ins which can easily be embedded in any web application and a Python library providing all requested features to setup, run and monitor workflows.

**Availability and implementation:** Jflow is available under the GNU General Public License (GPL) at http://bioinfo.genotoul.fr/jflow. The package is coming with full documentation, quick start and a running test portal.

**Contact:** Jerome.Mariette@toulouse.inra.fr

1 Introduction

Building rich web environments aimed at helping scientists analyze their data is a common trend in bioinformatics. Specialized web portals such as MG-RAST (Meyer et al., 2008), MetaVir (Roux et al., 2011) or NG6 (Mariette et al., 2012) provide multiple services and analysis tools in an integrated manner for specific experiments or data types. These applications require WMS features to manage and execute their computational pipelines.

Generic WMS, such as Galaxy (Goecks et al., 2010), Ergatis (Orvis et al., 2010) or Mobyle (Néron et al., 2009) provide a user friendly graphical interface easing workflow creation and execution. Unfortunately, such environments come with their own interface, complicating their integration within already existing web tools. Other WMS such as weaver (Bui et al., 2012), Snakemake (Koster et al., 2012), Ruffus (Goodstadt, 2010) or Cosmos (Gafni et al., 2014) provide a framework or a domain-specific language to developers wanting to build and run workflows. These software packages offer the flexibility and power of a high-level programming language, but they do not provide a user interface, enable component and workflow definition.

JFlow combines a user friendly interface with an intuitive python API. It is, to our knowledge, the only WMS designed to be embedded in any web application, thanks to its organization as jQuery (http://jquery.com/) plug-ins.

2 Methods

JFlow user interface gathers five jQuery plug-ins providing user oriented views.
Jflow

availablewf lists all runnable workflows accessible to users,
activewf monitors all started, completed, failed, aborted and re-
set workflows,
wfoutputs displays all outputs produced by the workflow organ-
ized per component,
wstatus shows the workflow execution state as a list or an ex-
cution graph. The graph visualization uses the Cytoscape web
JavaScript plug-in (Lopes et al., 2010).

The plug-ins give access to multiple communication methods
and events. They interact with the server through Jflow’s REST
API, running under a cherrypy (http://www.cherrypy.org/) web
server. The included server uses the JSONP communication tech-
nique enabling cross-domain requests.

To be available from the different jQuery plug-ins, the work-
flows have to be implemented using the Jflow API. A Jflow com-
ponent is in charge of an execution step. Adding a component to
the system requires to write a Python Component subclass. In
Jflow, different solutions are available to ease component cre-
ation. To wrap a single command line, the developer can give a
position or a flag for each parameter. Jflow also embeds an XML
parser which allows to run genuine Mobyle (Neron et al., 2009)
components. Finally, to allow developers to integrate com-
ponents from other WMS, Jflow provides a skeleton class. This
class only requires to implement the parsing step. A workflow
chains components. It is represented by a directed acyclic graph
(DAG) where nodes represent jobs and edges links between inputs
and outputs. When paths are disjoint, jobs are run in parallel. A
Jflow workflow is built as a Workflow subclass. Components are
added to the workflow as variables and chained linking outputs
and inputs.

To define the parameters presented to the final user, Jflow
gives access to different class methods. Each parameter has at least
a name, a user help text and a data type. For file or directory pa-
rameters, it is possible to set required file format, size limitation
and location. Jflow handles server side files with regular expres-
sions, but also URLs and client side files, in which case, it auto-
matically uploads them. Before running the workflow, Jflow
checks data type and sequencing technology.

Workflow integration: (a) The status of the illumina_qc workflow with the id 26 displayed as
an empty div. (b) The jQuery code in charge to build Jflow plug-ins and manage user action. When the select.activewf event is thrown from activewf-
div, a function is called with two parameters: event and workflow. The last parameter stores all the workflow’s information, such as its name and its id,
used in this example to update the modal box title and to build the wfform plug-in. (c) The status of the illumina_qc workflow with the id 26 displayed as
a graph in the NG6 application.

3 Example

Jflow user interface has been designed to allow an easy integration
in mash up web applications. Hereunder, we present its integration
in NG6, which provides a user-friendly interface to process, store
and download high-throughtput sequencing data. The environment
displays sequencing runs as a table. From this view, the user can add
new data by running workflows in charge of loading the data and
checking its quality. Different workflows are available considering
data type and sequencing technology.

Workflow integration: (a) A piece of the NG6 HTML code source in which is
positioned an empty div to build the activewf plug-in and a modal box for the
wfform plug-in. (b) The jQuery code in charge to build Jflow plug-ins and
manage user action. When the select.activewf event is thrown from activewf-
div, a function is called with two parameters: event and workflow. The last
parameter stores all the workflow’s information, such as its name and its id,
used in this example to update the modal box title and to build the wfform
plug-in. (c) The status of the illumina_qc workflow with the id 26 displayed as
a graph in the NG6 application.

To define the parameters presented to the final user, Jflow
gives access to different class methods. Each parameter has at least
a name, a user help text and a data type. For file or directory pa-
rameters, it is possible to set required file format, size limitation
and location. Jflow handles server side files with regular expres-
sions, but also URLs and client side files, in which case, it auto-
matically uploads them. Before running the workflow, Jflow
checks data type and sequencing technology.

Workflow integration: (a) The status of the illumina_qc workflow with the id 26 displayed as
an empty div. (b) The jQuery code in charge to build Jflow plug-ins and manage user action. When the select.activewf event is thrown from activewf-
div, a function is called with two parameters: event and workflow. The last
parameter stores all the workflow’s information, such as its name and its id,
used in this example to update the modal box title and to build the wfform plug-in. (c) The status of the illumina_qc workflow with the id 26 displayed as
a graph in the NG6 application.

To define the parameters presented to the final user, Jflow
gives access to different class methods. Each parameter has at least
a name, a user help text and a data type. For file or directory pa-
rameters, it is possible to set required file format, size limitation
and location. Jflow handles server side files with regular expres-
sions, but also URLs and client side files, in which case, it auto-
matically uploads them. Before running the workflow, Jflow
checks data type and sequencing technology.

Workflow integration: (a) A piece of the NG6 HTML code source in which is
positioned an empty div to build the activewf plug-in and a modal box for the
wfform plug-in. (b) The jQuery code in charge to build Jflow plug-ins and
manage user action. When the select.activewf event is thrown from activewf-
div, a function is called with two parameters: event and workflow. The last
parameter stores all the workflow’s information, such as its name and its id,
used in this example to update the modal box title and to build the wfform
plug-in. (c) The status of the illumina_qc workflow with the id 26 displayed as
a graph in the NG6 application.

4 Conclusion

Jflow is a simple and efficient solution to embed WMS features within a
web application. It is, to our knowledge, the only WMS designed with
this purpose. It is already embedded in RNAbrowse (Mariette et al., 2010) WMS, which had a separate user interface. With Jflow, all
actions are now available from the same application, which makes it user friendly.

Conflict of Interest: none declared.

References

computing on clusters, clouds, and grids. SWEET at ACM SIGMOD, 20.


