Kepler Project Newsletter

May 2009

Web Accessibilty For Kepler

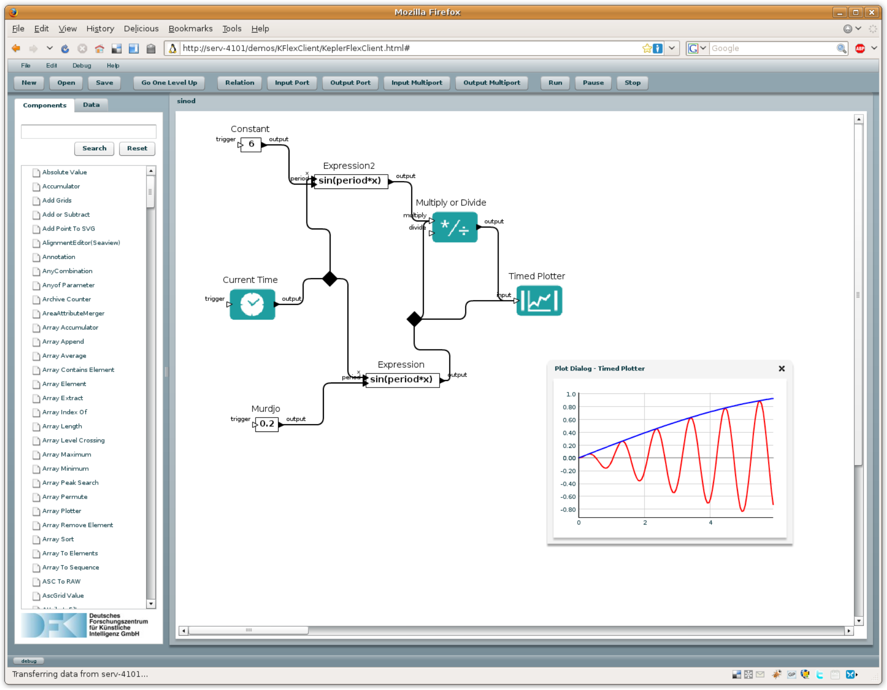
People thinking about adopting Kepler often ask the question, "Can Kepler be used over the web?" The easy answer is "Yes", but a more helpful response might be, "Yes, and in many different ways." A number of projects have used Kepler in a web-based computing environment and others plan to do so. But the differences in the requirements these projects face are as numerous as the similarities, with solutions necessarily spanning a range of deployment scenarios. At one extreme, Kepler can be used as the backend for a web-based application that--from the users' point of view--looks and behaves nothing like Kepler. Kepler is used by the scientists and engineers developing, deploying, and supporting the workflow behind the web application, while the end-user is (or can be) completely unaware of the role of Kepler. The GEON project, among others, has successfully employed Kepler in this way from the earliest days of the Kepler Project.

At the other extreme, the full power and key features of the standard Kepler graphical user interface can be mimicked in a web browser, thus enabling users to compose, run, and share workflows without installing and configuring Kepler on their own computers. Computational and data management resources can be provided to users of such web-enabled installations of Kepler with the added benefit that users can connect and disconnect from running instances of Kepler workflow from wherever they may be. This is the vision of Christopher Tuot's group at the German Research Center for Artificial Intelligence for the KFlex system, and of Paul Allen's group at Cornell who are extending and customizing Kflex to support the SciencePipes project. KFlex and SciencePipes are discussed further below.

Between these two extremes are web portals that enable Kepler workflows to be uploaded to a server, selected by the same or different users, and configured and executed over the web.

The Kepler Web Service

Deploying Kepler as the computational backend to a custom web application is now easier than ever. Jianwu Wang, a member of the Kepler/CORE enginering team, recently developed support for executing Kepler workflows via standard Web service protocols. The Kepler Web Service supports SOAP-based and RESTful invocations, and both synchronous and asynchronous operation. The Web service captures and returns output from display actors that normally would appear within the Kepler GUI at run time. The Kepler Web Service already has been used successfully to decouple back-end execution of workflows from custom browser-based user interfaces. Code and documentation for the Kepler Web Service may be found at https://code.kepler-project.org/code/kepler/trunk/modules/web-service/.



The KFlex web interface to Kepler

Web Accessiblity (continued from p. 1)

Hydrant

The remainder of this article focuses on deployments of Kepler via web-browser interfaces that emulate at least some of the look and functionality of the standard Kepler user interface. In May 2008, Tristan King at James Cook University announced Hydrant, a system for enabling users to view, run, and share their Kepler workflows--and results--via the web. The stated goal of Hydrant was to make Kepler extremely easy to use for those who want to run existing workflows rather than develop new ones. Hydrant's approach depends on some users building and testing new workflows who then upload these workflows to the Hydrant portal. Hydrant allows others to view these workflows; select, configure and run the workflows; and view the outputs of these runs via the web. The system allows the workflow repository to be searched and makes it easy for users to share their workflows. A key feature of Hydrant is its ability to automatically substitute web-enabled versions of actors for those that would otherwise create graphical displays on a local workstation. Hydrant project information and source code can be found at http://www.hpc.jcu.edu.au/projects/kepler/wiki/Hydrant.

KFlex

The KFlex system takes Hydrant's approach one step further by enabling users to compose new workflows and customize existing workflows via the web. The open-source KFlex system consists of two components. KFlex Server is an extended version of the Kepler 1.0 release that provides a Web Service interface for executing workflows and is packaged as a web application deployable to standard web application servers such as Tomcat. The KFlex Client is an Adobe Flex 3 implementation of the Kepler GUI that reproduces the critical features of the standard Kepler user interface and uses a KFlex server for workflow execution, data storage, and collaboration. Christopher Tuot's group at the German Research Center for Artificial Intelligence (http://www.dfki.de) is developing KFlex to serve as the interactive front end to a Spatial Decision Support System (SDSS) that will include a geographical database and a library of models for forecasting the outcomes of various decisions.

SciencePipes

Web access to Kepler supports educational as well as scientific purposes. The SciencePipes web application (http://sciencepipes.org), currently under development by Paul Allen at the Cornell Lab of Ornithology, is meant to enable students, educators, citizens, resource managers, as well as scientists to create and share analyses and visualizations of biodiversity data. SciencePipes builds on the KFlex framework to enable members of the public to create their own workflows from scratch using high-level actors developed specifically for biodiversity analysis. Workflows employing these domain-specific actors are comprehensible to students and non-professionals. SciencePipes makes Kepler accessible to non-scientists, and promotes sharing and discovery of workflows. Because workflows can be copied and used as starting points for new analyses, SciencePipes supports learning by example. Moreover, SciencePipes makes it easy to export the results of workflow runs as web resources that can be incorporated into web sites (e.g. blogs) and automatically updated as workflows are rerun with new data or parameters are changed.

Interested in trying out Kepler on the web either as a user or as a prospective developer of new web-based scientific workflow applications? Create your own workflows on SciencePipes or interact with other Kepler developers via the Kepler Web Interface Interest Group and online forums (https://kepler-project.org/developers/interest-groups/webui).

Recent and Upcoming Events

April 16, 2009 The Eighth Biennial Ptolemy Miniconference was held Thursday, April 16, 2009 at the University of California, Berkeley. Talks and posters described the latest research based on the Ptolemy II system and included a number of contributions from the Kepler community. For the program and links to presentations see http://ptolemy.berkeley.edu/conferences/09/program.htm.

July 20-21, 2009 "Using Kepler for Conservation Training," Kruger National Park, South Africa. For more information please contact Matthew Jones (jones@nceas.ucsb.edu).

Build System Facilitates extension of Kepler

A defining characteristic of the Kepler effort has been the large number of projects extending Kepler for their own needs and applying the system to a broad range of scientific domains. Making it easier to develop, share, and smoothly integrate such independently developed extensions to Kepler is a major objective of the Kepler/CORE project. Additionally, participants at the first Kepler Stakeholders Meeting requested that the existing Kepler code base be modularized and that development of the kernel of code used by all Kepler users be decoupled from development of specific extensions developed by and for particular communities. The Kepler/CORE team has developed a revised build system to address these needs.

Framework for System Modularity

The build system now is based on the notions of modules and suites. A suite is a set of modules meant to be used together. The Kepler base system (the essential core of Kepler) is represented by such a suite of modules, and any number of system extensions can be developed as optional, add-on suites. Modules are represented both in the source code repository and in the Kepler run time system, and the build system decouples building and release of different suites. Developers can select which modules should be included in their development environment, and users ultimately will be able to select which suites are present and active in an installation of Kepler.

The system also allows developers to select which version of the Kepler kernel is to be used when building and testing extensions. This enables developers to package and distribute their suites independently of the Kepler kernel. At the moment, developers can build their code either against the version of Kepler currently under active development (the "trunk" of the Kepler source code repository), or against the 1.0 release of Kepler. The latter option allows projects to develop and release suites that are meant to work with instances of Kepler already installed and in use. In the past, projects developing code based on the Kepler beta releases, or on the 1.0 release, sometimes found it easier to store their code in other repositories. The result was that it was challenging for such projects to share code with the rest of the community and stay informed about parallel efforts.

Although the build system now works with modules stored and versioned anywhere, we encourage projects to store their extensions in the Kepler svn repository (https://code.kepler-project.org/code/kepler/). Each Kepler extension can be tagged and branched independently of the Kepler kernel, allowing extension developers to maintain (and fix bugs) in distinct versions of their extensions meant to run on different versions of Kepler.

Support for Experimental Overrides

The revised build system also provides advanced features that enable developers to experiment with alternative implementations of classes included in the Kepler kernel without creating a complete branch of the kernel or exposing other developers to such major changes. Much development in the Kepler community centers around not simply developing new actors and workflows for Kepler but significantly enhancing the workflow system itself or customizing the user interface for specific communities. Many projects are funded to enhance Kepler in such major ways. So besides allowing modules to be developed against specific versions or releases of Kepler, the build system allows extensions to experimentally override classes in the Kepler release it is meant to work with. The system automatically orders the class path at run time to respect the module priority order defined in a simple text file named modules.txt. When more than one module defines the same class, the version of the class included in the module listed earlier in modules.txt takes priority at run time. This approach applies the familiar Java classpath approach to Kepler modules, and makes it easy for projects to develop new major features and provide new options in the graphical user interface. Once the usefulness of a new feature has been demonstrated, the Kepler Framework Infrastructure Team (see **Kepler Organization Update** on page 4 of this newsletter) can work with the contributor either to incorporate these changes into the Kepler kernel itself or to add new extension points for supporting optional features demonstrated by the overrides, such that the overrides are no longer necessary.

Resolution of Jar Version Conflicts

Similarly, extensions can employ versions of third-party libraries (i.e., jars) that are different from those used in the Kepler kernel or in other extensions. The build system allows a module optionally to be configured to run in its own Java class loader. A module running in

Build System (continued from p. 3)

its own class loader can employ versions of 3rd-party jars differing from those used by the Kepler kernel. Each module-specific class loader loads only those jars unique to the module or differing in version from those in the main class loader such that jars are not loaded redundantly and so that all modules can continue to interact in the normal Java fashion, except with respect to the classes loaded from alternative versions of the same jars.

Getting Started with the Build System

Finally, the build system now makes it easier for new developers to get started with Kepler and to set up their development environments quickly. It automatically checks out appropriate versions of the Kepler kernel, Ptolemy, and modules based on the suites one wants to work with. It also organizes these files in a uniform way automatically on the developer's machine. Finally, the build system prepares project files for three IDEs (IntelliJ, Eclipse, and NetBeans) in addition to supporting full operation from the command line.

To check out and build the modules included in the kepler base system suite, try the following commands at the command prompt on a MacOSX, Windows, or Linux computer with svn and ant installed:

```
% mkdir kepler
% cd kepler
% svn co https://code.kepler-project.org/code/
          kepler/trunk/modules/build-area
% cd build-area
% ant change-to -Dsuite=kepler
% ant run
```

Alternatively, use these commands to check out at once all of the modules under development at the trunk and to build kepler:

```
% svn co https://code.kepler-project.org/code/
          kepler/trunk/kepler
% cd kepler/modules/build-area
% ant run
```

For more information about how you can use the revised build system to develop your extensions to Kepler and easily share them with other developers via the Kepler repository, please see the complete build system instructions at https://kepler-project.org/developers/teams/build/documentation/build-system-instructions.

Kepler Organization Update

As a multi-institutional effort, the Kepler Project faces numerous challenges in enabling collaboration between the various stakeholders and coordinating contributions to the system. The Kepler/CORE team recently announced new organizational and technological infrastructure to support these efforts.

A new **Leadership Team**, approved by the Kepler project members, has committed to assuring the long-term technical and financial viability of Kepler, making strategic decisions on behalf of the Kepler user community, and representing the interests of the Kepler Project. The Leadership Team aims to guide and facilitate the overall design and development of Kepler through member participation as discussed below. Leadership Team members will serve for terms of 3 years, with renewal for additional terms upon approval of the remainder of the Leadership. The Kepler Leadership Team and others carrying out research and development within the context of Kepler interact via Interest Groups, Development Teams, and Infrastructure Teams. Each of these groups is granted it's own space for web pages and forums for discussion on the Kepler web site.

Interest Groups represent a lightweight mechanism for community members to discuss areas of common interest, and are easy to form. **Development Teams** design, develop, test, and deploy the software and infrastructure necessary for a specific system extension or actor package for Kepler. Each development Team operates under a charter that describes its scope, and a roadmap that identifies its work plan and planned products. Individuals who wish to create a team can request a project space in the Incubation areas of the Kepler web site to assist in the construction and submission of these documents. Finally, **Infrastructure Teams** sustain the development of the kernel of Kepler on which all other subsystems are built, maintain the build and test tools used by the Kepler community, and oversee the process of releasing new distributions of the Kepler base system.

The entire Kepler community is invited to participate in the development of Kepler at each of these levels!

Kepler is based upon work supported by the National Science Foundation under award 0722079 for Kepler/CORE, and by many contributing projects and individual contributors