OpenNTF.Org

Workflow for XPages

Agenda

- Objectives
- Workflow Types
- Design Goal
- Architecture
- How to use the workflow engine
- API introduction
- How to integrate new workflow engine
- How to setup development environment

Objectives

- Provide a lightweight workflow engine
- Simple workflows build in
- Extensible to connect to other WorkFlow engines
- Easy to add to an XPages application

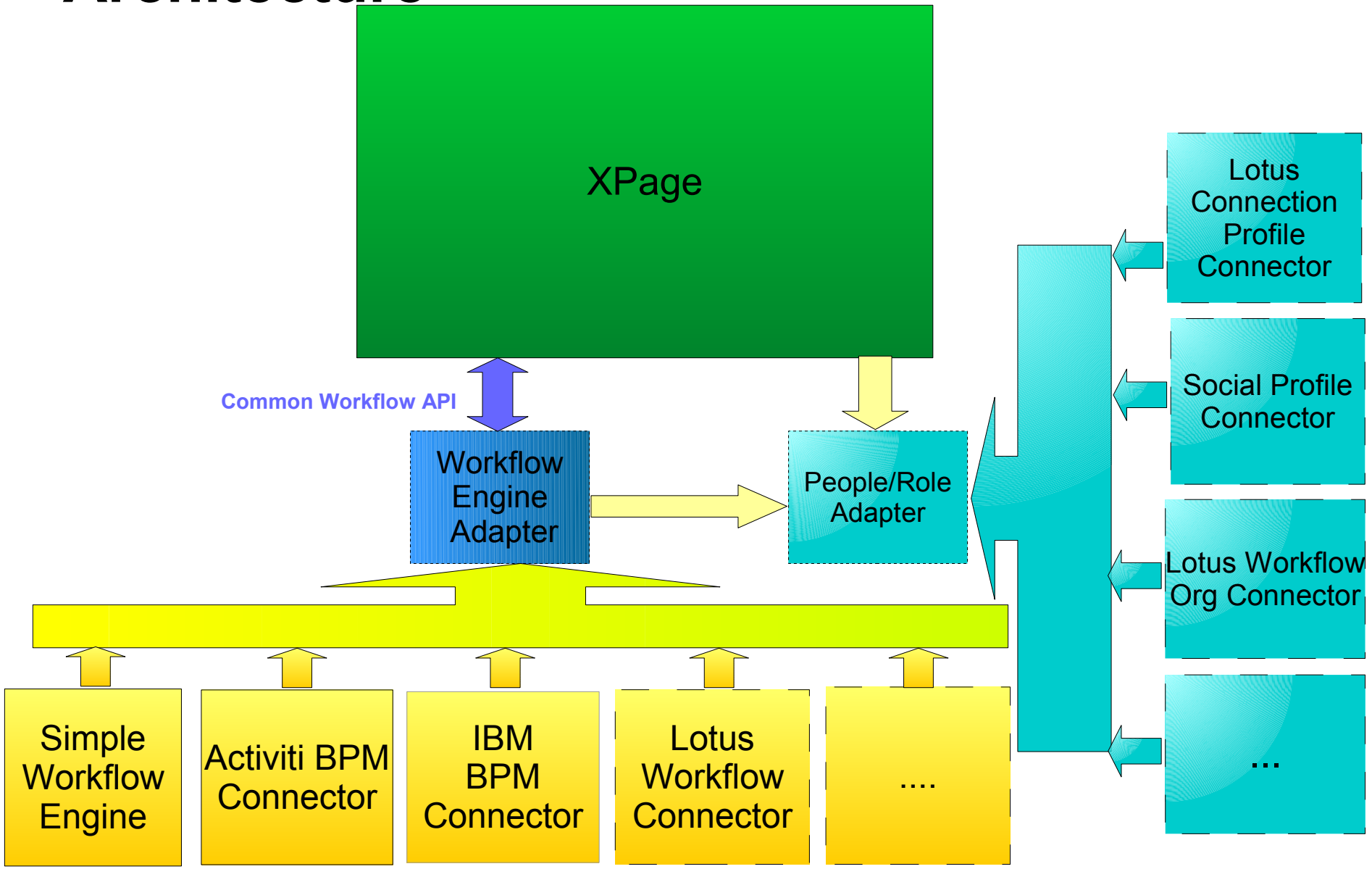
Workflow Types

- Different types of workflows are available
  - Ad hoc workflows, for simple, no administration workflows
    - Defined as part of the forms, using simple components
      - Similar to the Workflow sub form
    - Changing a process means opening Domino Designer and editing an XPage component
    - Actions are happening synchronously
  - Managed Workflows
    - Defined outside of the form, generally is external databases
    - Actions are happening asynchronously, using a background router
- Workflow context abstraction
  - The workflow is accessed the same way, regardless of the underlying implementation
- XPages common controls
  - Comes with a set of ready to use controls (action bar, display history...), connected to the worklow context

Design Goals

- Make the workflow engine easy to use and integrate for the simple cases
  - Predefined Workflow templates
    - Simple approval...
  - Easy to parameterize
  - Connects to the existing infrastructure (directory, ...)
    - Built-in drivers
  - Ready to go set of XPages controls/API
- Make the engine customizable for more complex cases
  - Custom workflow templates
    - Requires some custom Java/XPages custom development
- Developer can use similar API to access workflow data regardless of the workflow type
  - A workflow context object exposes the current state of the workflow and allows actions to be triggered
- Ensure that nothing is “hard coded” but provided through pluggable services
  - From an XPages standpoint, the services will be available through Complex types

Architecture

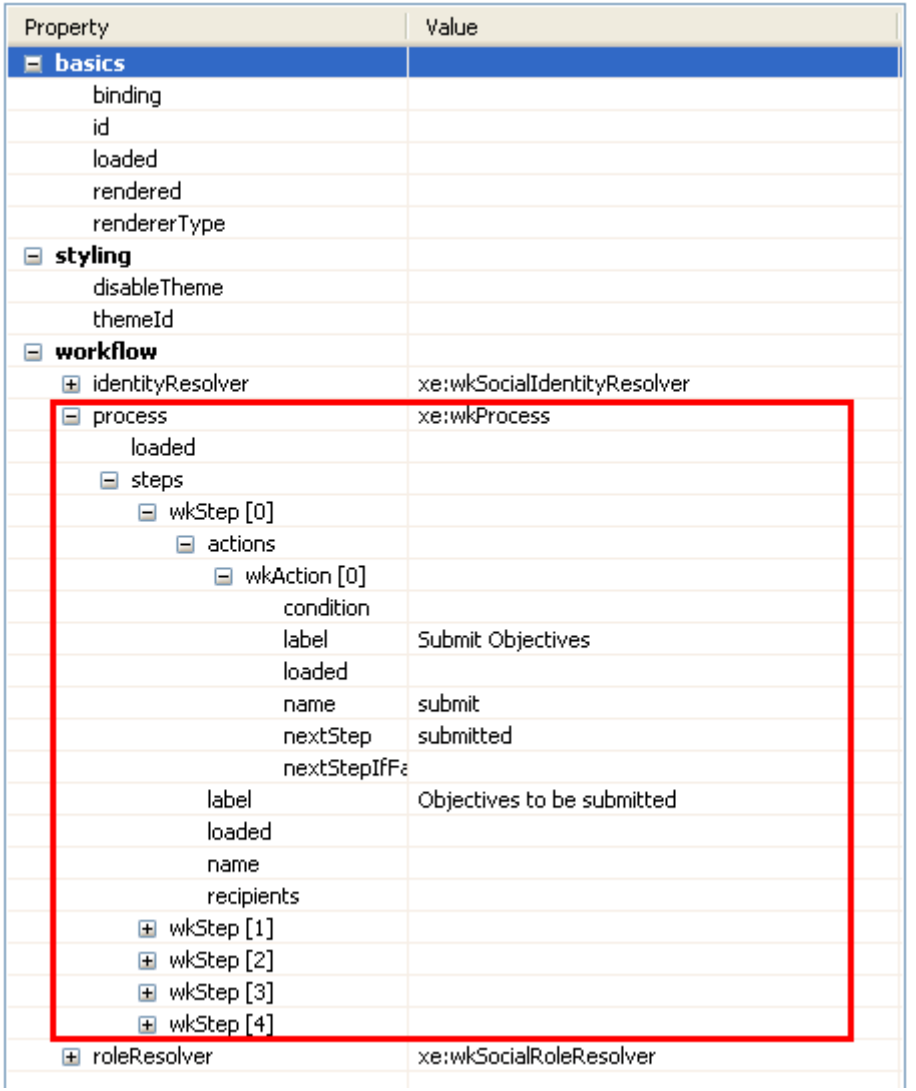


How to Use the Workflow Engine:

- For simple ad hoc workflow applications
  - You can use the workflow engine directly in this distribution(called SimpleWorkflow)
  - In your XPage, you can use workflowContext to access the workflow engine data
- For complex workflow applications
  - We have implemented a custom control called CommonWorkflow and a Activiti workflow engine adaptor. You can use this workflow control and adaptor to access Activiti workflow server. Or you can implement you own adaptor to connect to different workflow engine.
  - In your XPage, you can use workflowContext to access the workflow engine data

Use Simple Workflow Engine

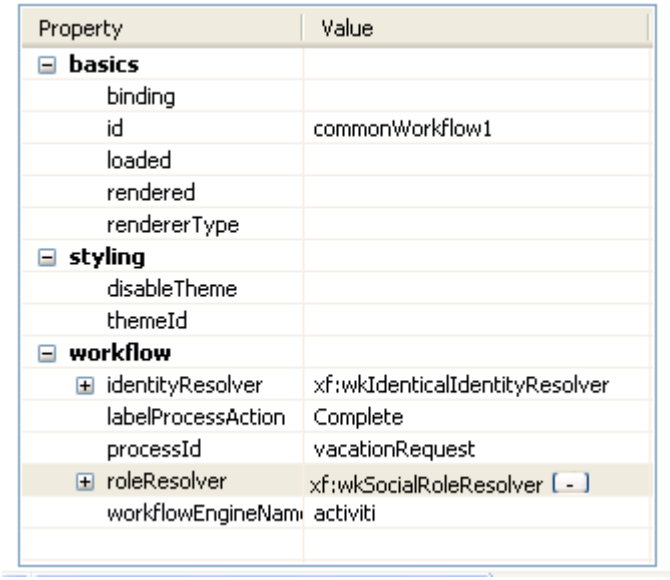
- For simple ad hoc workflow application, you can use the workflow engine in this distribution
  - In Application Properties → Advanced tab, add required libraries: com.ibm.xsp.extlib.library and com.ibm.xsp.xflow.library
  - Drag the SimpleWorkflow control to your Xpage
  - Define your workflow process in SimpleWorkflow's “Process” property
  - Design your form in the XPage as usual
  - Access workflow data through data bean workflowContext in Xpage
  - Use custom control wkWorkflowActions to show workflow action bar



Use CommonWorkflow control

- Sample usage for Activiti server
  - Add following library dependency: com.ibm.xsp.extlib.library and com.ibm.xsp.xflow.library
  - Create a workflow engine definition file in WebContent/WEB-INF/workflow/*.workflow, file name is the engine name. File content likes right. We can also define that in a profile document called: XFLOW_WORKFLOW_PROFILE
  - Drag CommonWorkflow Control into XPage
  - In properties of Common Workflow Control, define process(see right snapshot)
    - processId: process id defined Activiti server
    - workflowEngineName: workflow engine name defined by above step
  - Develop form in XPage
  - In XPage, we can use bean “workflowContext" to access workflow information
  - We can also add wkWorkflowActions control to XPage which will show an action bar with workflow actions on it
  - (Option)Define identityResolver which can map between workflow user id and Notes id;
  - (Option)Define roleResolver which can resolve role name to workflow user id;

```
<workflow>
<server>yourFullServerName</server>
   <port>80</port>
   <url>http://yourFullServerName/activiti-rest/service</url>
<workflowengine>com.ibm.xsp.xflow.activiti.ActivitiWorkflowContextFactory</workflowengine>
</workflow>
```



API Introduce - IWorkflowContext

- IWorkflowContext: main interface between XPages and workflow engine, in XPage, we can use this API to get/set information from/to workflow engine for process instance attached to current XPage
  - public String getCurrentStep(): return current step name
  - public String getCurrentStepLabel(): return label of current step
  - public String getRequester(): get requester id
  - public String[] getActions(): get current step action names
  - public String[] getActionLabels(): get labels of current step actions
  - public String getLocale(): return locale
  - public boolean isInitiated(): whether current process instance is initialized
  - public boolean isRunning(): whether current process instance is running
  - public boolean isCompleted(): whether current process instance is completed
  - public boolean isReadonly():
  - public void setWorkflowData(HashMap data);
  - public HashMap getWorkflowData();
  - public void executeAction(String name) throws WorkflowException;

API Introduce - IWorkflowContextFactory

- IWorkflowContextFactory: Workflow engine adapter need to implement this interface to supply workflow context
  - public IWorkflowContext createWorkflowContext(final FacesContext context, final AbstractWorkflow workflow, final DataSource dataSource, final Object data) : This method will be called when a workflow context is needed in a XPage.
  - public void setWorkflowInfo(String server, String port, String url, String endpoint, IIdentityResolver idResolver, IRoleResolver roleResolver, String actionHandler) : This method will be called when the WorkflowContextFactory created. All the information set here come from xml files under WebContent/WEB-INF/workflow folder or from a profile document XFLOW_WORKFLOW_PROFILE.

How to integrate new workflow engine

- First implement interface IWorkflowContext to create your own workflow context class. Your class must inherit com.ibm.xsp.xflow.domino.CommonWorkflowContext. You can also refer to implementation in class com.ibm.xsp.xflow.activiti.ActivitiWorkflowContext.
- Then, implement interface IWorkflowContextFactory. You can refer to code in class com.ibm.xsp.xflow.activiti.ActivitiWorkflowContextFactory.
- In your plugin.xml file, contribute following extension point:
  - <extension point="**com.ibm.commons.Extension**">
  - <service class="**YourWorkflowContextFactoryName**"
  - type="**com.ibm.xsp.xflow.workflowFactory**">
  - </service>
  - </extension>
- You can use your workflow engine in same way of using CommonWorkflow control

How to Setup Development Environment

1. Install Domino server and Eclipse 3.6 or 3.4 above
2. Download XPage Extension Library
3. Install XPage Extension Library to Domino server: unzip the XPageExtensionLibrary.zip and get the updatesite.zip; Unzip the updatesite.zip, copy the unzipped folder into \Domino\data\domino\workspace\applications\eclipse
4. Follow this guide to setup debug env: http://www.openntf.org/Projects/pmt.nsf/6EA1D2D26DF8D5BC862577D00055BB21/%24file/Lotus%20Domino%20Debug%20Plugin.pdf(Use the JVM installed in dominon directory <Domino Install Directory>\Domino\jvm)
5. Import source code to Eclipse workspace, set the target platform to Domino\osgi\rcp\eclipse, Domino\osgi\shared\eclipse, Domino\data\domino\workspace\applications\eclipse
6. Copy the sample NSF(EmployeeReview.nsf) to your Domino data directory
7. In Domino Designer there maybe errors, you also need install updatesite.zip to domino designer. You also need to install update site "com.ibm.xsp.xflow.updatesite.zip" into Designer.