Workflow

Configuration

Main workflow configuration

As of DSpace 7, the workflow.xml configuration file has been migrated to use Spring Bean syntax (instead of a custom XML format). The structure of this XML has changed. If you need help migrating your old workflow.xml file (which started with a `<ref-config>` tag) to the new format (using `<bean>` tags), an XSLT script is available: workflow-migration.xsl

The workflow main configuration can be found in the workflow.xml file, located in `[dspace]/config/spring/api/workflow.xml`. An example of this workflow configuration file can be found below.

```
<beans>
  <bean class="org.dspace.xmlworkflow.XmlWorkflowFactoryImpl">
    <property name="workflowMapping">
      <util:map>
        <entry key="defaultWorkflow" value-ref="defaultWorkflow"/>
        <entry key="123456789/4" value-ref="selectSingleReviewer"/>
        <entry key="123456789/5" value-ref="scoreReview"/>
      </util:map>
    </property>
  </bean>

  <!--Standard DSpace workflow-->
  <bean name="defaultWorkflow" class="org.dspace.xmlworkflow.state.Workflow">
    <property name="firstStep" ref="reviewstep"/>
    <property name="steps">
      <util:list>
        <ref bean="reviewstep"/>
        <ref bean="editstep"/>
        <ref bean="finaleditstep"/>
      </util:list>
    </property>
  </bean>

  <!-- Another workflow configuration-->  
  <bean id="{workflow.id}" class="org.dspace.xmlworkflow.state.Workflow"> 
    <property name="firstStep" ref="reviewstep"/>
    <property name="steps">
      <util:list>
        <ref bean="reviewstep"/>
        <ref bean="editstep"/>
        <ref bean="finaleditstep"/>
      </util:list>
    </property>
  </bean>
</beans>
```

`workflowFactory bean` (org.dspace.xmlworkflow.XmlWorkflowFactoryImpl)

The workflow map contains a mapping between collections in DSpace and a workflow configuration, and is defined by the `workflowMapping` property of the workflow factory. Similar to the configuration of the submission process, the mapping can be done based on the handle of the collection. The mapping with "defaultWorkflow" as the value for the collection mapping, will be used for the collections not occurring in other mapping tags. Each mapping is defined by a "entry" element with two attributes:

- key: can either be a collection handle or "defaultWorkflow"
- value-ref: the value of this attribute points to one of the workflow configurations defined by the "Workflow" beans

`workflow beans` (org.dspace.xmlworkflow.state.Workflow)

The workflow bean is a repeatable XML element and represents one workflow process. It requires the following:

- "name" attribute: a unique name used for the identification of the workflow and used in the workflow to collection mapping
- "firstStep" property: the identifier of the first step of the workflow. This step will be the entry point of this workflow-process. When a new item has been committed to a collection that uses this workflow, the step configured in the "firstStep" property will he the first step the item will go through.
- "steps" property: a list of all steps within this workflow (in the order they will be processed).

`role beans` (org.dspace.xmlworkflow.Role)

Each workflow step has defined "role" property. A role represents one or more existing DSpace EPersons or Groups and can be used to assign them to one or more steps in the workflow process. One role is represented by one "role" bean and has the following:

- "id" attribute: a unique identifier (in one workflow process) for the role
The step element represents one step in the workflow process. A step represents a number of actions that must be executed by one specified role. In case no role attribute is specified, the workflow framework assumes that the DSpace system is responsible for the execution of the step and that no user interface will be available for each of the actions in this step. The step element has the following in order to further configure it:

- "name" attribute: The name attribute specifies a unique identifier for the step. This identifier will be used when configuring other steps in order to point to this step. This identifier can also be used when configuring the start step of the workflow item.
- "userSelectionMethod" property: This attribute defines the UserSelectionAction that will be used to determine how to attach users to this step for a workflow-item. The value of this attribute must refer to the identifier of an action bean in the workflow-actions.xml. Examples of the user attachment to a step are the currently used system of a task pool or as an alternative directly assigning a user to a task.
- "role" property: optional attribute that must point to the id attribute of a role element specified for the workflow. This role will be used to define the epersons and groups used by the userSelectionMethod.
- RequiredUsers

Each step contains a number of actions that the workflow item will go through. In case the action has a user interface, the users responsible for the execution of this step will have to execute these actions before the workflow item can proceed to the next action or the end of the step.

There is also an optional subsection that can be defined for a step part called "outcomes". This can be used to define outcomes for the step that differ from the one specified in the nextStep attribute. Each action returns an integer depending on the result of the action. The default value is '0' and will make the workflow item proceed to the next action or to the end of the step. In case an action returns a different outcome than the default '0', the alternative outcomes will be used to lookup the next step. The "outcomes" element contains a number of steps, each having a status attribute. This status attribute defines the return value of an action. The value of the element will be used to lookup the next step the workflow item will go through in case an action returns that specified status.

Workflow actions configuration

API configuration

The workflow actions configuration is located in the [dspace]/config/spring/api/ directory and is named "workflow-actions.xml". This configuration file describes the different Action Java classes that are used by the workflow framework. Because the workflow framework uses Spring framework for loading these action classes, this configuration file contains Spring configuration.

This file contains the beans for the actions and user selection methods referred to in the workflow.xml. In order for the workflow framework to work properly, each of the required actions must be part of this configuration.
Two types of actions are configured in this Spring configuration file:

- User selection action: This type of action is always the first action of a step and is responsible for the user selection process of that step. In case a step has no role attached, no user will be selected and the `NoUserSelectionAction` is used.
- Processing action: This type of action is used for the actual processing of a step. Processing actions contain the logic required to execute the required operations in each step. Multiple processing actions can be defined in one step. These user and the workflow item will go through these actions in the order they are specified in the workflow configuration unless an alternative outcome is returned by one of them.

User Selection Action

Each user selection action that is used in the workflow configuration refers to a bean definition in the `workflow-actions.xml` file. In order to define a new user selection action, the following XML code is used:

```xml
<bean id="{action.api.id}2" class="org.dspace.xmlworkflow.state.actions.UserSelectionActionConfig" scope="prototype">
    <constructor-arg type="java.lang.String" value="{action.api.id}2" />
    <property name="processingAction" ref="{user.selection.bean.id}" />
    <property name="requiresUI" value="{true/false}" />
</bean>
```

This bean defines a new UserSelectionActionConfig and the following child tags:

- `constructor-arg`: This is a constructor argument containing the ID of the task. This is the same as the `id` attribute of the bean and is used by the workflow configuration to refer to this action.
- `property processingAction`: This tag refers to the ID of the API bean, responsible for the implementation of the API side of this action. This bean should also be configured in this XML.
- `property requiresUI`: In case this property is true, the workflow framework will expect a user interface for the action. Otherwise the framework will automatically execute the action and proceed to the next one.

Processing Action

Processing actions are configured similarly to the user selection actions. The only difference is that these processing action beans are implementations of the `WorkflowActionConfig` class instead of the `UserSelectionActionConfig` class.

Authorizations

Currently, the authorizations are always granted and revoked based on the tasks that are available for certain users and groups. The types of authorization policies that is granted for each of these is always the same:
Database

The workflow uses a separate metadata schema named workflow. The fields this schema contains can be found in the [dspace]/config/registries directory and in the file workflow-types.xml. This schema is only used when using the score reviewing system at the moment, but one could always use this schema if metadata is required for custom workflow steps.

The following tables have been added to the DSpace database. All tables are prefixed with ‘cwf_’ to avoid any confusion with the existing workflow related database tables:

**cwf_workflowitem**

The cwf_workflowitem table contains the different workflowitems in the workflow. This table has the following columns:

- workflowitem_id: The identifier of the workflowitem and primary key of this table
- item_id: The identifier of the DSpace item to which this workflowitem refers.
- collection_id: The collection to which this workflowitem is submitted.
- multiple_titles: Specifies whether the submission has multiple titles (important for submission steps)
- published_before: Specifies whether the submission has been published before (important for submission steps)
- multiple_files: Specifies whether the submission has multiple files attached (important for submission steps)

**cwf_collectionrole**

The cwf_collectionrole table represents a workflow role for one collection. This type of role is the same as the roles that existed in the original workflow meaning that for each collection a separate group is defined to described the role. The cwf_collectionrole table has the following columns:

- collectionrol_id: The identifier of the collectionrole and the primary key of this table
- role_id: The identifier/name used by the workflow configuration to refer to the collectionrole
- collection_id: The collection identifier for which this collectionrole has been defined
- group_id: The group identifier of the group that defines the collection role

**cwf_workflowitemrole**

The cwf_workflowitemrole table represents roles that are defined at the level of an item. These roles are temporary roles and only exist during the execution of the workflow for that specific item. Once the item is archived, the workflowitemrole is deleted. Multiple rows can exist for one workflowitem with e.g. one row containing a group and a few containing epersons. All these rows together make up the workflowitemrole. The cwf_workflowitemrole table has the following columns:

- workflowitemrole_id: The identifier of the workflowitemrole and the primary key of this table
- role_id: The identifier/name used by the workflow configuration to refer to the workflowitemrole
- workflowitem_id: The identifier of the workflowitem for which this workflowitemrole has been defined
- group_id: The group identifier of the group that defines the workflowitemrole role
- eperson_id: The eperson identifier of the eperson that defines the workflowitemrole role

**cwf_pooltask**

The cwf_pooltask table represents the different task pools that exist for a workflowitem. These task pools can be available at the beginning of a step and contain all the users that are allowed to claim a task in this step. Multiple rows can exist for one task pool containing multiple groups and epersons. The cwf_pooltask table has the following columns:

- pooltask_id: The identifier of the pooltask and the primary key of this table
- workflowitem_id: The identifier of the workflowitem for which this taskpool exists
- step_id: The identifier of the step for which this taskpool was created
- action_id: The identifier of the action that needs to be displayed/executed when the user selects the task from the task pool
- eperson_id: The identifier of an eperson that is part of the task pool
- group_id: The identifier of a group that is part of the task pool

**cwf_claimtask**

The cwf_claimtask table represents a task that has been claimed by a user. Claimed tasks can be assigned to users or can be the result of a claim from the task pool. Because a step can contain multiple actions, the claimed task defines the action at which the user has arrived in a particular step. This makes it possible to stop working halfway the step and continue later. The cwf_claimtask table contains the following columns:

- claimtask_id: The identifier of the claimtask and the primary key of this table
- workflowitem_id: The identifier of the workflowitem for which this task exists
- step_id: The step that is currently processing the workflowitem
- action_id: The action that should be executed by the owner of this claimtask
- owner_id: References the eperson that is responsible for the execution of this task
cwf_in_progress_user

The cwf_in_progress_user table keeps track of the different users that are performing a certain step. This table is used because some steps might require multiple users to perform the step before the workflowitem can proceed. The cwf_in_progress_user table contains the following columns:

- in_progress_user_id: The identifier of the in progress user and the primary key of this table
- workflowitem_id: The identifier of the workflowitem for which the user is performing or has performed the step.
- user_id: The identifier of the eperson that is performing or has performed the task
- finished: Keeps track of the fact that the user has finished the step or is still in progress of the execution