Galaxia

An open-source Workflow engine for Tiki.

User Manual And Documentation

Garland Foster.
Richard Moore.
Eduardo Polidor.

Please read:
“Galaxia introduction and concepts”
before reading this document.
Thanks!

Index

Modules

The workflow engine contains 3 modules: The Process Manager, The User Interface and the Process Monitor.

- Process Manager
- User interface
- Process Monitor

Behind this modules is the “engine”, a set of PHP classes and scripts that execute the workflow processes.

The Process Manager is used to map a business process to a workflow process. It is used to “design” processes.

The User Interface presents a set of screens and interfaces to let users start a new process, execute process activities, see what activities they have pending and manage their workload. It is used to “execute” processes.

The Process Monitor is used to track what the users are doing, what processes are being executed and the status of each process in execution. It is used to “track” processes.

The Process Manager

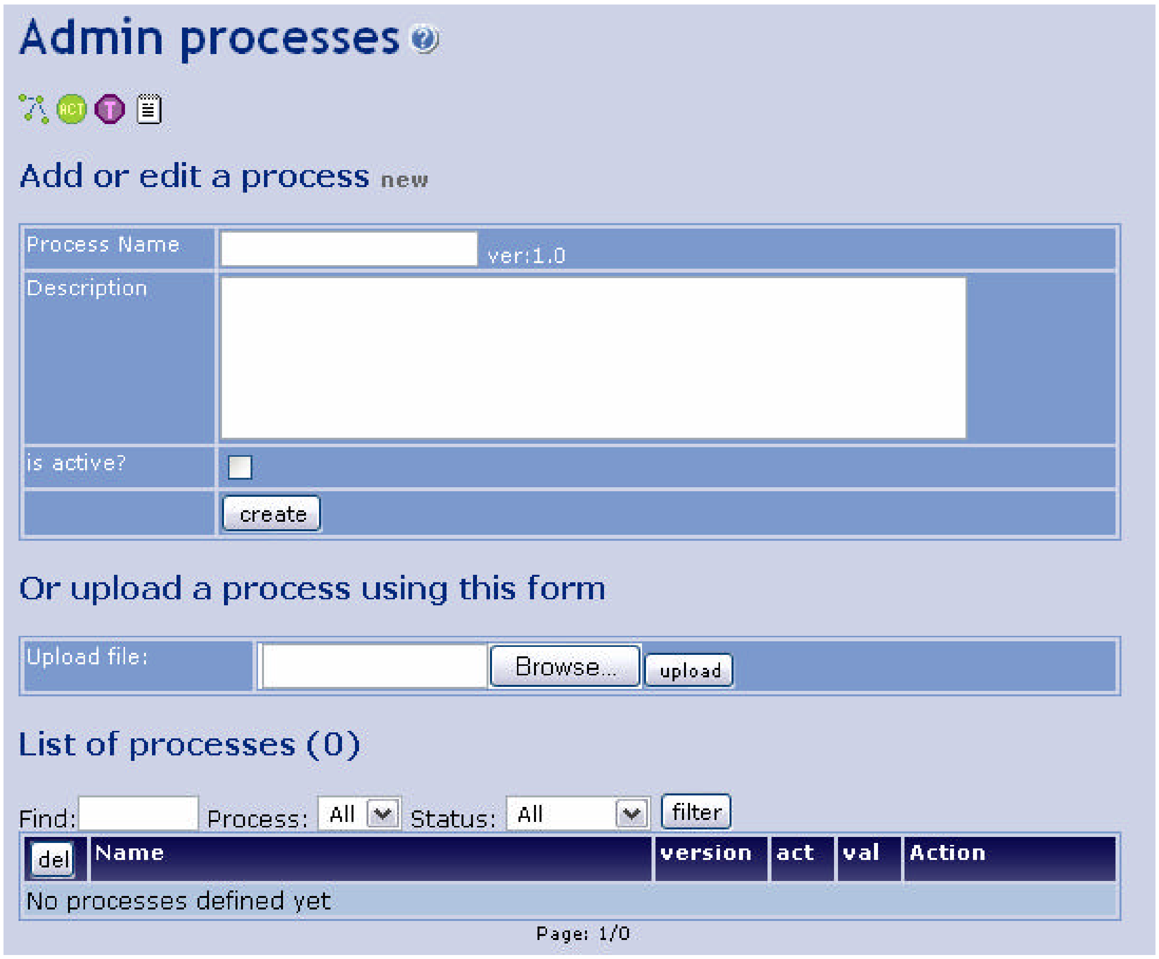
The Process Manager (PM from now on) is used to design processes and make them available to your users. A process can be very simple or very complex depending on the business process being modeled. Processes can be used to extend Tiki adding features particular to your implementation, Processes can be used to map internal process in your company so they are automated using Tiki.

The creation of a process can be divided in the following tasks:

- Creating a new process
- Editing process activities
- Editing and mapping process roles
- Editing process activity source code and templates

Creating a new process

Enter the Process Manager clicking on the “Admin processes” link from the Workflow section of the application menu. You should have the tiki_p_admin_workflow permission to use this feature.



Admin processes

Add or edit a process new

Process Name	ver:1.0
Description	
is active?	☐
	create

Or upload a process using this form

Upload file:	Browse... upload

List of processes (0)

Find: Process: All Status: All filter

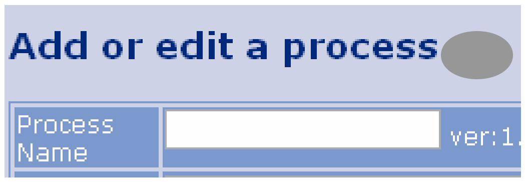
del	Name	version	act	val	Action
No processes defined yet					

Page: 1/0

You can create a new process in 3 different ways:

- Creating a new process from scratch
- Branching an existing process
- Importing a process

Creating an existing process from scratch

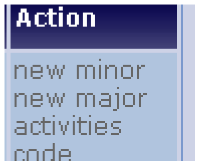


Add or edit a process

Process Name	ver:1.

You can create a new process from scratch clicking the "new" link next to the from to edit a process name and description. Enter a name for the process and a description, a new process will be created with version 1.0. Once the process is created you can edit its activities, roles and so. Note that upon creation the process cannot be set to "active" only valid processes can be "active" and a recently created process is invalid.

Branching an existing process



Once you have a process created you can define new "versions" of the process, this can be useful in many situations: First of all if you want to change a process that is active you may need to create a new version so you don't make unexpected changes to process instances already in execution. You can also "version" a process if you want to create a new process that will be very similar to an existing process.

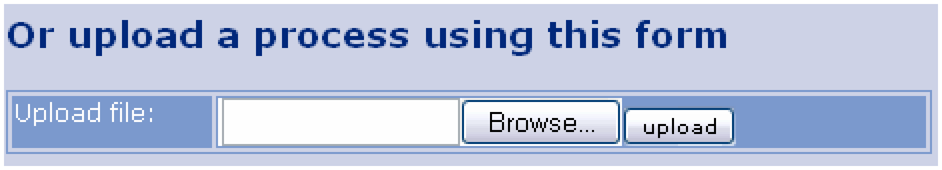
To change an active process:

1. Create a new version of the process.
2. Edit the new version
3. Make the old process "inactive"

You can create a new version defining a new "major" version or "minor" version being the difference how the new process version number is created.

Importing a process

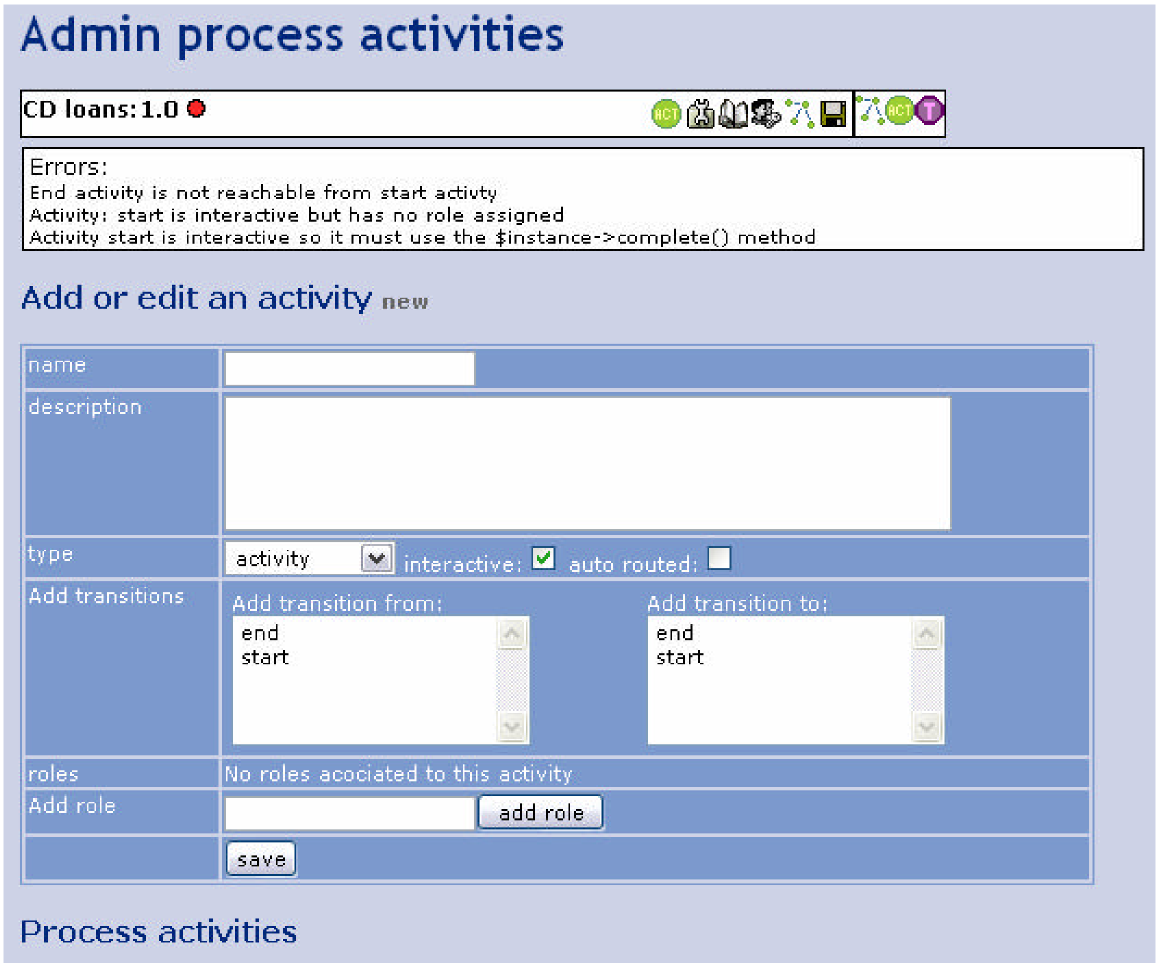
The Workflow engine can be used to "export" a process to an XML file, the XML file will contain all the process activities, transitions, roles and source code. You can then distribute the process in its XML format so other applications can use it. You can upload a process using the import feature in the process manager:



When a process is imported you will have to map the process roles to your site users/groups depending on what users/groups will be able to execute the process activities on your site. Once roles are mapped the process will be ready to use as it was in the installation where it was created.

Editing process activities

Once a process is created click on the "activities" link in the action menu to edit or create process activities. The activities manager is a module of the process manager used to edit or create actvities. This is the most important and crucial part of the process creation. You may want to design your process in a paper before using this tool to define your process. The activity editor screen should be similar to the following one:



Errors:
End activity is not reachable from start activty
Activity: start is interactive but has no role assigned
Activity start is interactive so it must use the $instance->complete() method

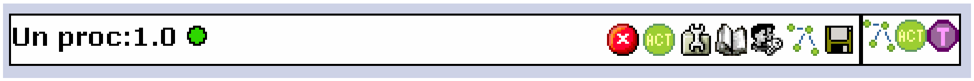
Add or edit an activity new

name	
description	
type	activity interactive: ✓ auto routed: ☐
Add transitions	Add transition from: end, start — Add transition to: end, start
roles	No roles acociated to this activity
Add role	add role
	save

The screen is divided in different sections that will be explained in detail:

- The process toolbar
- The add/edit activity form
- The activity listing
- The transition listing and form

The process toolbar

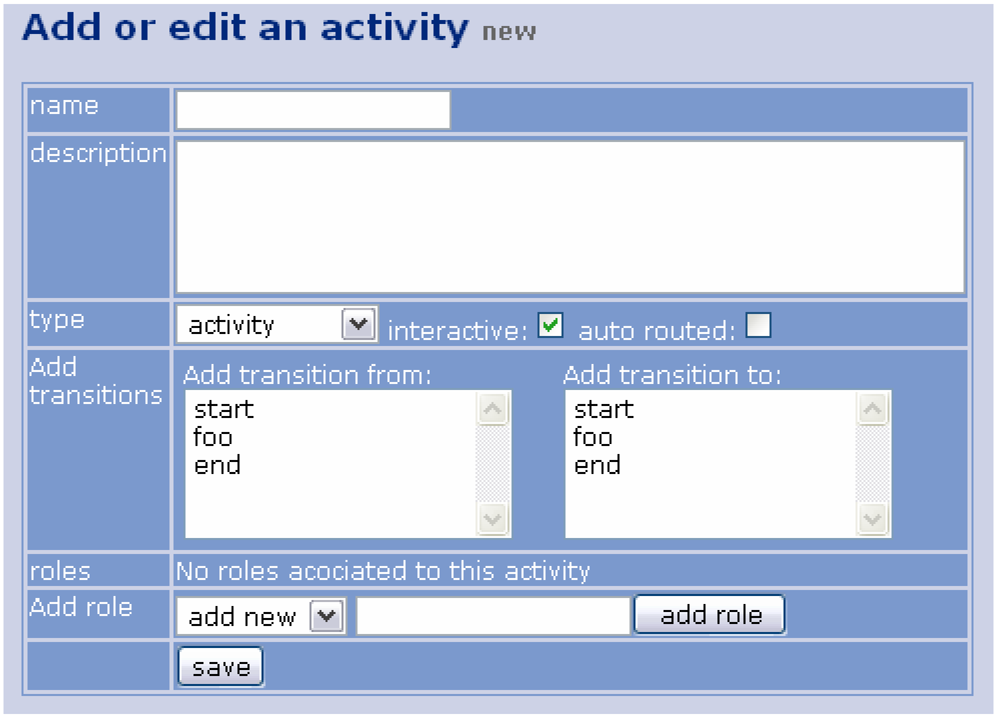


The process toolbar can be used to quickly jump to different modules of the Process Manager, the icons are explained below:

Icon	Meaning
	Configuration and settings for the workflow engine
●	Valid process. This icon indicates that the process is valid.

●	Invalid process. This icon indicates that the process is invalid, a list of errors will be displayed at the top of the screen.
✖	Stop process. If the process is active you can use this icon to "stop" the process. Once stopped no new instances of the process can be created. Already created instances will remain in execution.
⟳	Activate process. If the process is valid but the process is not active this icon will be displayed to activate the process.
●	Configure activities. The activity manager for the process.
⚙	Admin processes. Goes to the admin processes screen.
📖	Process code. Used to edit the code of this process activities.
⚙	Process roles. Used to define roles and map roles for this process.
⁂	Process graph. Used to display a popup with the current graph for the process activities.
💾	Save process. Exports the whole process as an XML file.
⁂	Monitor processes.
●	Monitor activities.
●	Monitor instances.

The add/edit activity form


Add or edit an activity new

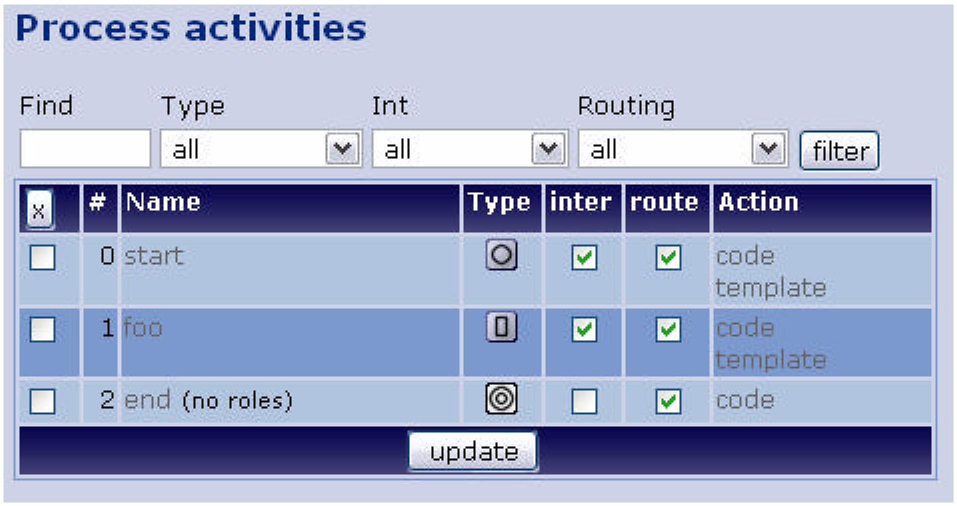
name	
description	
type	activity interactive: ✓ auto routed: ☐
Add transitions	Add transition from: start foo end — Add transition to: start foo end
roles	No roles acociated to this activity
Add role	add new [add role]
	[save]

This form is where you can edit process activities or add new activities to the process. You have to select the activity name and a description for the activity. Then select the activity type, select if the activity is interactive and select if the activity is auto-routed.

Tip: Use the link to the process graph (⁂) frequently to display the graph of your process and check if the activites are correctly linked.

The next boxes can be used to add transitions from existing activities to the activity being edited/created and to create transitions from the activity being edited/created to existint activities.
Then you can define roles for the activity adding already existing roles or creating a new role. Roles will have to be mapped using the admin roles module.

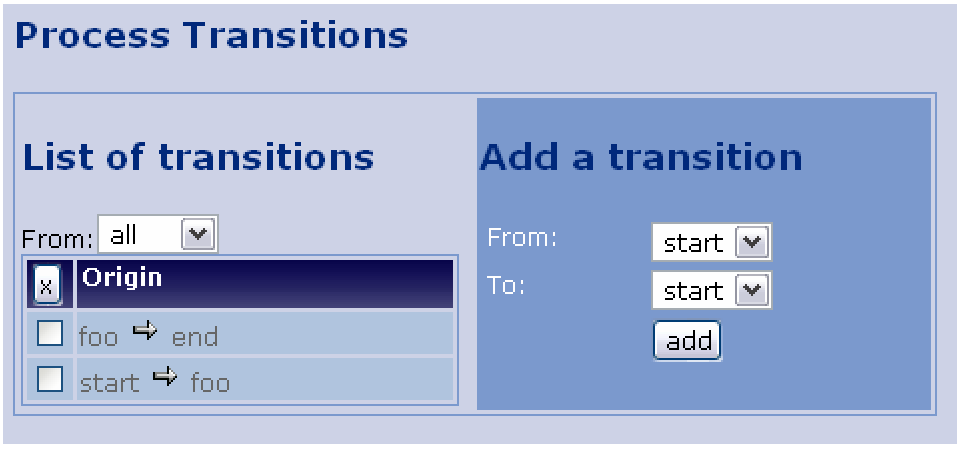
The activity listing



x	#	Name	Type	inter	route	Action
☐	0	start	○	✓	✓	code template
☐	1	foo	▯	✓	✓	code template
☐	2	end (no roles)	◎	☐	✓	code

This section lists your process activities, you can use the filter bar at the top to select the activities that you want to see. Activities are displayed in "flow" order from the start activity to the end activity. You can re-sort by other fields clicking on the column titles. This listing can be used to set if activities are interactive or automatic and if activities are auto-routed or manually routed, just set the checkboxes accordingly and press the update button at the bottom of the listing. Activities can be removed marking the activities with the checkboxes on the left and the pressing the "x" button.

The transition listing and form



Process Transitions

List of transitions

From: all

x	Origin
☐	foo ⇨ end
☐	start ⇨ foo

Add a transition

From: start

To: start

The last section of the activity manager shows the list of transitions for the process, you can remove transitions using the checkboxes and the “x” button. You can also add transitions selecting the From and To activities and clicking the “add” button. As you can see there’s more than one way to add transitions or remove transitions from/to activites to make the user interface more confortable.

Understanding activities

Icon	Type	Description	In	Out
○	Start	The start activity. A process can have many start activities and mut have at least one. Start activities can be executed without the presence of an “instance” of the process because start activities create a new process instance. Normally start activities are interactive requiring some kind of user input to start a new process.	0	1
◎	End	The end activity terminates a process instance. All the processes must have one and only one end activity.The end eactivity should be reachable from the start activyt. Make sure all your process “paths” of execution lead to the end activity or you will have “dangling” instances of your processes.	N	0
▯	Activity	A normal activity. Used to perform some tasks, can be interactive or automatic.	N	1
◇	Switch	A swicth activity is like a normal activity but can be used to send the process instance to different activities depending on some condition. This activities are used to create bifurcations in your process like if “a” then go to activity “a1” else go to activity “z”.	N	N
○	Standalone	A standalone activity can be executed without the presence of a process instance. Standalone activites are used to model actviities independant of the process flow. You can even create some process featuring only standalone activities.	0	0
△	Split	A “split” is used to send an instance to many different actvities in paralell. So it is true that an instance can be in many activities at the same time. If your process has	N	N

		activities that can be done in paralell then you can use a split activity.		
▽	Join	A join activity is used to re-group instances splitted using a split activity. When an instance reaches a join activity the engine checks if the instance is present in other activites if so the instance will wait in the join actvity. Once the instance is only present in the join activity the instance can be routed (manually or automatically) to the next activity.	N	1

Note: When an acivity is interactive the icon is displayed in a blue-background. In the process graph interactive activities are displayed using a blue border. Blue means interactiveness.

Interactiveness

Activities can be either interactive or automatic.

Interactive actvities require some kind of user interaction, normally a form is presented to the user and the user can fill/change values submitting the form as many times as needed until the activity decides that the activity is completed.

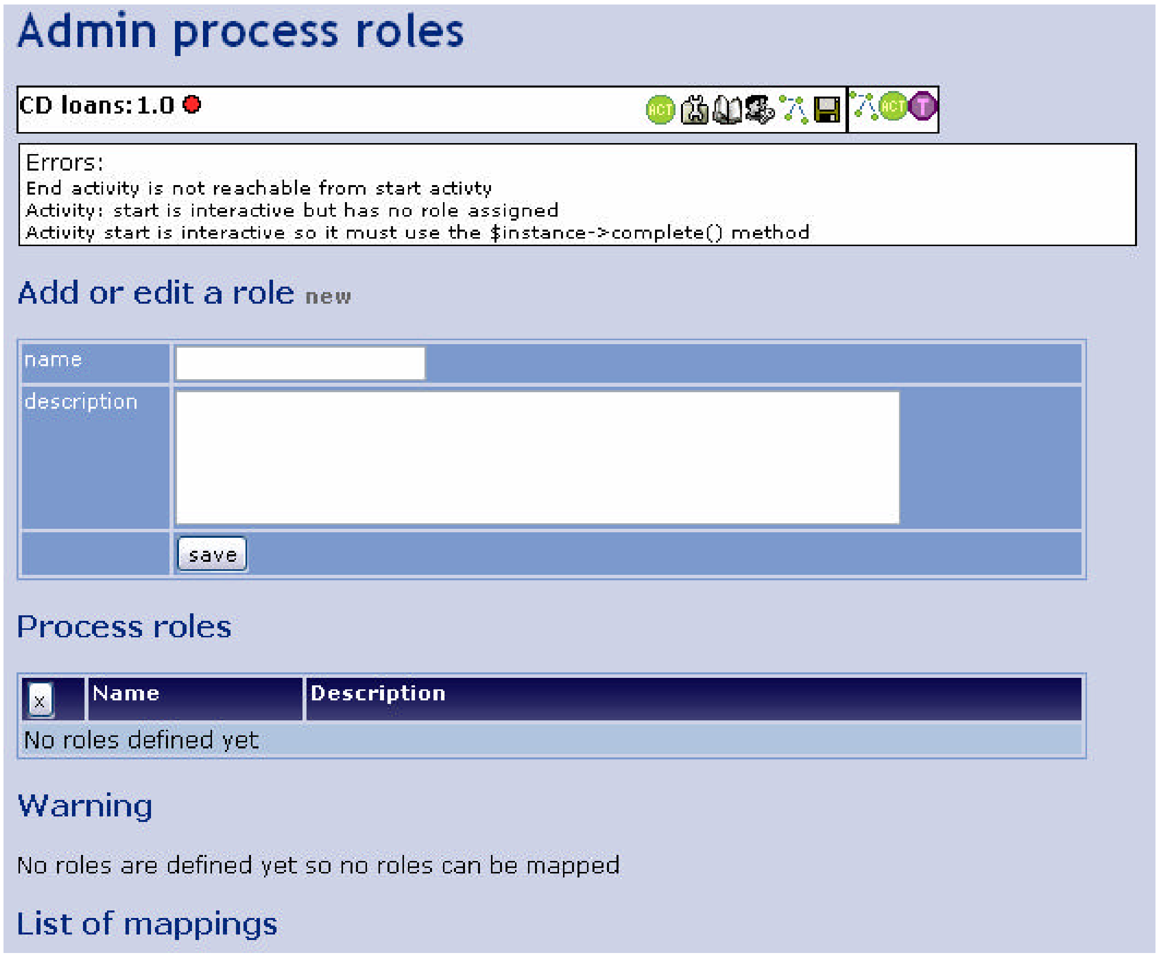
Automatic activites don't require user interaction and are excuted by the workflow engine.

AutoRouting

An activity can be routed automatically (auto-routed) or manually. When an activity is auto-routed instances are automatically sent to the next activity in the process once the activity is completed. If the actvity is not auto-routed a user will have to manually "send" the instance to make the instance flow to the next activity.

Editing and mapping process roles

The Role Manager is the module of the process manager used to admin process roles and map roles to tiki users or groups. The role manager screen is similar to the following one:


Admin process roles

CD loans:1.0

Errors:
End activity is not reachable from start activty
Activity: start is interactive but has no role assigned
Activity start is interactive so it must use the $instance->complete() method

Add or edit a role new

name	
description	
	save

Process roles

x	Name	Description
No roles defined yet		

Warning

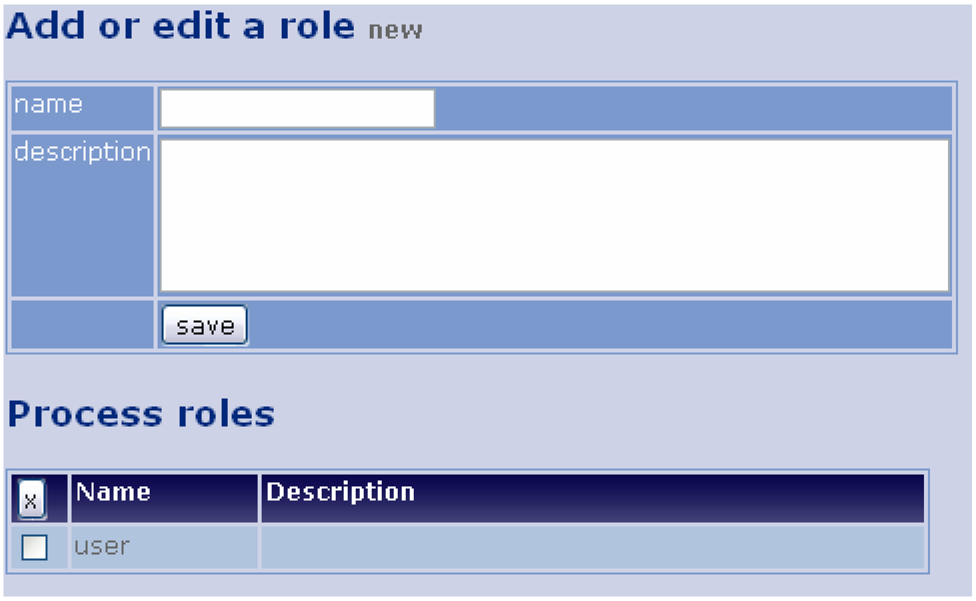
No roles are defined yet so no roles can be mapped

List of mappings

The screen can be divided in sections that will be explained next

- Role editing and listing
- Role mapping and listing

Role editing and listing


Add or edit a role new

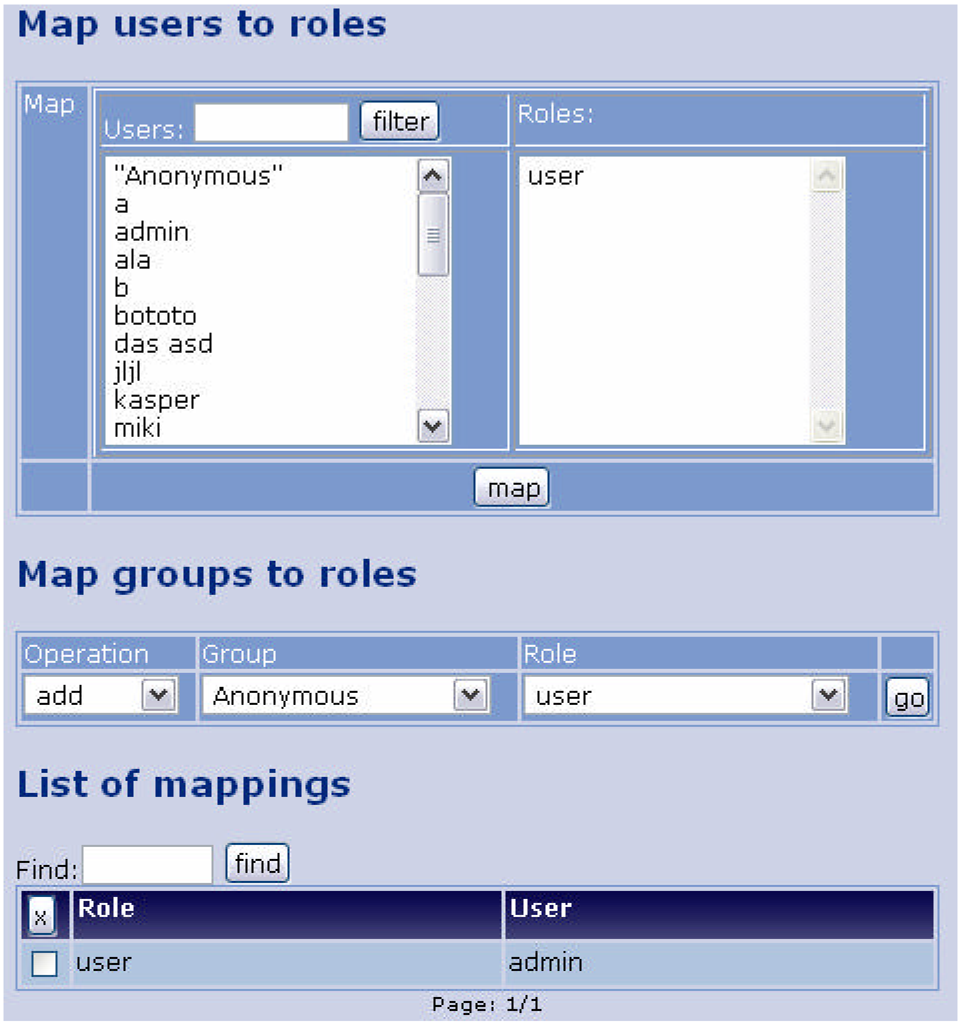
name	
description	
	save

Process roles

x	Name	Description
☐	user	

The first section of the Role Manager module can be used to add new roles to the process, edit existing roles names and descriptions and remove roles as needed. Please note that roles are defined per-process. The “user” role can be absolutely different in ProcessA and ProcessB.

Role mapping and listing


Map users to roles

Map

Users: filter

Roles:

"Anonymous"
a
admin
ala
b
bototo
das asd
jljl
kasper
miki

user

map

Map groups to roles

Operation	Group	Role	
add	Anonymous	user	go

List of mappings

Find: find

x	Role	User
☐	user	admin

Page: 1/1

In the first section you can map user to roles you can select many users and many roles and all the mappings will be executed. So you can use it to map a user to many roles, a role to many users or many roles to many users at the same time. Next you can add or remove all the users in any tiki group to some role, this can be used to “batch” map users in a group to some role. Note that roles are mapped to individual users not groups, this is different to tiki permissions that are handled at group levels.
The last section of the secreen lists all the mappings defined, you can filter the list and remove mappings if you want.

Understanding roles

Roles are used in processes to control who can do what. Roles provide an abstract representation of what users will be able to execute process activities. Since user names and even group names can be very different in different Tiki installations roles are used to asbtract permission to execute process activities so processes can be shared in different Tiki installations by only remapping the process roles.

Roles can also be used to control what each actvity can do, since special variables will be created to indicate if the user executing some activity is mapped or not to some role. You can for example let roles “a” and “b” execute some activity but only users with role “b” can see some information.

Editing process activity source code and templates

Introduction

The workflow engine maps activities to PHP scripts, this is where the power of the engine resides, an activity can do anything that a PHP script can do. Normally you can design processes without a very deep knowledge of PHP creating forms to interact with the user and the setting “instance” properties depending on those values.

Interactive activities and automatic activities

When editing activities code there’s a very importan distinction between automatic activities and interactive actviites. Automatca activities are represented by a PHP script while intreactive activities are represented by a PHP script and a Smarty template file.

When coding interactive activities you shouldn’t output anything from the PHP script, instead assign whatever you want to display to smarty template variables and then display the variables in the template. The template will also contain all the forms, tables and other html elements that you want to use. We’ll see examples in a few paragraphs.

Example: A simple interactive activity and its template

activity php code	**activity template**
`<?php`	`<form method=”post”>`
`if(isset($_REQUEST[‘save’])) {`	`Name : <input type=”text” name=”name” />`
`  $instance->set(‘name’,$_REQUEST[‘name’];`	`<input type=”submit” name=”save” value=”save” />`
`  $instance->complete();`	`</form>`
`}`	
`?>`	

The Instance object

When coding the PHP code for any activity you can use the $instance object to get/set instance properties and other important actvities related to the instance. The most relevant method of the $instance object are listed next and you can read the PHP documentation for the class for a complete description of its usage.

Method	Description
get($property)	Gets the value of an instance property.
set($property,$value)	Sets the value for an instance property.
complete()	(Interactive activities only). Declares the activity as completed, if the activity is auto-routing the engine will route the instance to the next activity automatically.
setStatus($status)	Sets the status of this instance, can be ‘active’, ‘aborted’ or ‘exception’
setNextActivity($act_name)	Indicates the name of the next activity to be executed, this is needed in switch activities.
setNextUser($user)	Indicates the name of a user to assign this instance to that user. The next activity must be performed by the indicated user.
getUserEmail($user)	Returns the email address of the indicated user, this is useful if you want to send emails in some activity using mail functions.

Instance properties

Instances are process in execution, instances have “properties” that can be “getted” and “setted” in process activities, this is how you can pass information from one activity to another one. A typical way to code actvities is to present a form and then set instance properties based on the form values. The following activities can “display” or “get” properties to perform some action.

Switch actvities

In a switch activity the activity must decide the next activity for the instance visiting the switch activity. This is generally decided examinign the instance properties and/or user input if the switch activity is interactive. In order to set the next activity the method setNextActivity($actName) must be used indicating the name of the next activity.

Example:

```
If($_REQUEST[‘amount’] > 1000) {
     $instance->setNextActivity(‘approve loan’);
} else {
     $instance->setNextActivity(‘process loan’);
}
// Because this was interactive we use complete()
$instance->complete();
```

Roles

Roles are used to determine who can execute an activity, when the process is defined one or more roles must be asociated with interactive activities. To make the process valid admin has to map process roles to Tiki users using the role manager tool. When an instance enters an activity the engine checks the roles that are asociated to the activity. If there's only one user that can execute the activity the engine assigns that user as the "activity user" and only he will see the instance in his User Interface. If there's more than one user allowed to execute the activity the engine will show the instance to all the users that can execute the activity, a user can then "grab" an instance so only he can execute the activity, if the user can't complete the activity he can "release" the instance so other user with the proper role can execute the activity. Note that executing the activity automatically "grabs" the instance but the instance can also be grabbed first and the activity executed later.

Setting the instance user for the next activity

The instance object has a method to let an activity determine who will be assigned to execute the next activity.

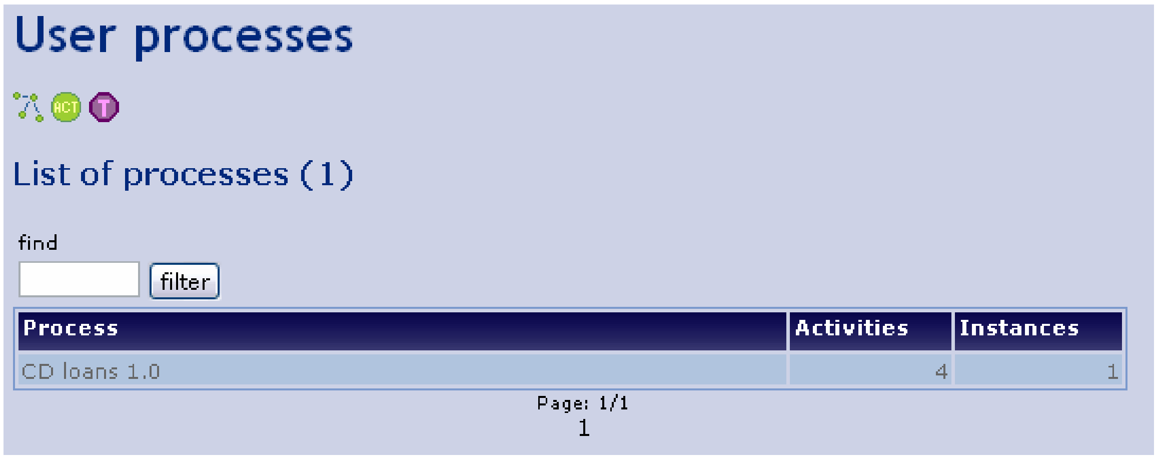
```
$instance->setNextUser('john');
```

In the example we are indicating that the next activity can only be executed by the user 'john'.

The User Interface

The User Interface (UI) is used by the users to launch new processes and execute process activities.

Enter the UI clicking the "User processes" link from the Workflow section of the application menu:



User processes

List of processes (1)

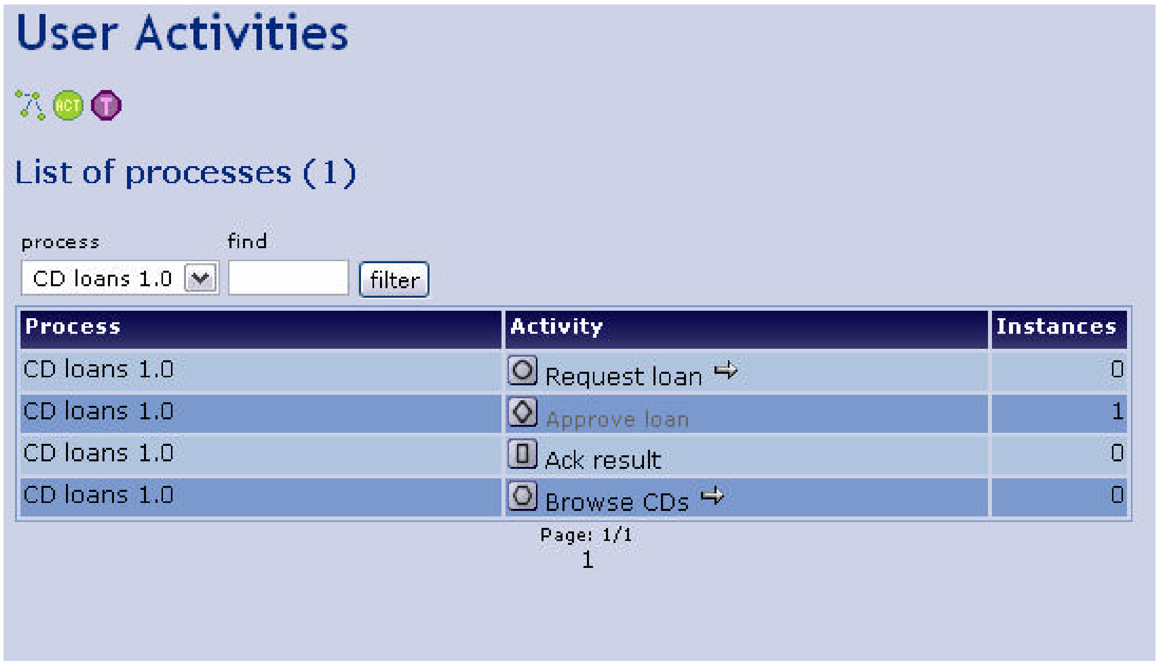
find

filter

Process	Activities	Instances
CD loans 1.0	4	1

Page: 1/1
1

In this screen you will see process that he user can start or where the user has pending activities to execute. In this example the “CD loans” process that will be created in the tutorial is used as an example. Clicking the process name will take you to the “user activities view”



User Activities

List of processes (1)

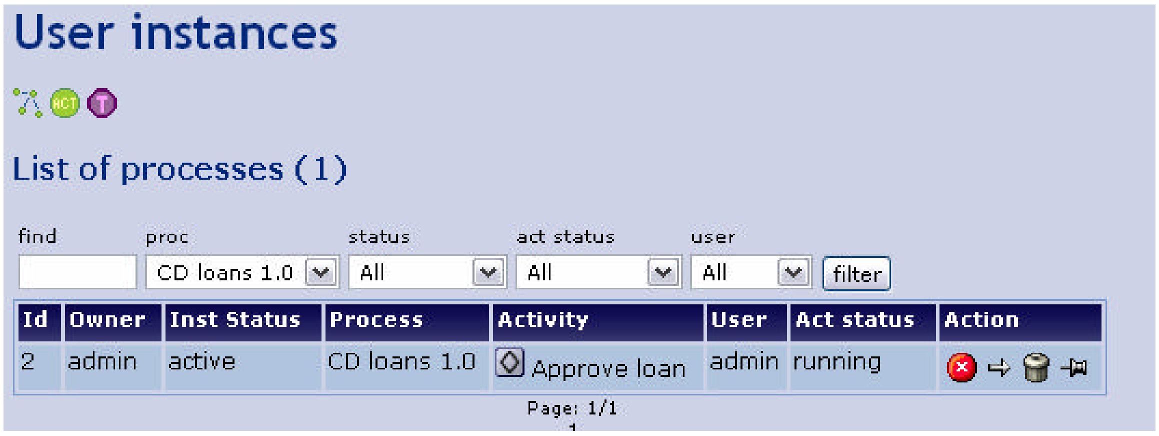
process: CD loans 1.0 find

filter

Process	Activity	Instances
CD loans 1.0	Request loan ⇨	0
CD loans 1.0	Approve loan	1
CD loans 1.0	Ack result	0
CD loans 1.0	Browse CDs ⇨	0

Page: 1/1
1

In this screen you can see activities of the “CD loans” process and the instances present in each activity. The activities with an arrow can be executed without an instance present (the start activity and a standalone activity), the “approve loan” activity can be clicked to display instances in this activity. The “user instances” screen is as follows:



User instances

List of processes (1)

find	proc	status	act status	user
	CD loans 1.0	All	All	All

Id	Owner	Inst Status	Process	Activity	User	Act status	Action
2	admin	active	CD loans 1.0	Approve loan	admin	running	

Page: 1/1

The buttons are explained as follows:

Icon	Meaning
🗑	Abort the instance.
●	"Grab" the instance (assign the execution of this instance in this activity to the user)
📌	"Release" the instance (let other users execute this activity for this instance)
⇨	Execute activity
⊗	Declare the instance in "Exception status"

Tutorial: A process to manage CD loans

Introduction:

"In the *foo* company there's a large collection of CDs, CDs are managed by an admin who has a collection of CDs. Users from any department can request a CD to be loaned for personal usage in the company. A process is requiered to track where each CD is, and authorize CD loans."

Use case 1: "Listing CDs"

Users from any department can list the CD collection, each CD will be listed indicating its title and if the CD is available or not. If not the user having the CD is listed along with some dates (loan date, return date).

Use case 2: "A user requests a CD"

A user can request an "available" CD, when a user requests a CD the user administering the collection has to authorize the loan based on information such as if the CD has to be used soon, etc. If the loan is authorized the soliciting user receives a note and he can pick up the CD from the department.

Use case 3: "A user returns a CD"

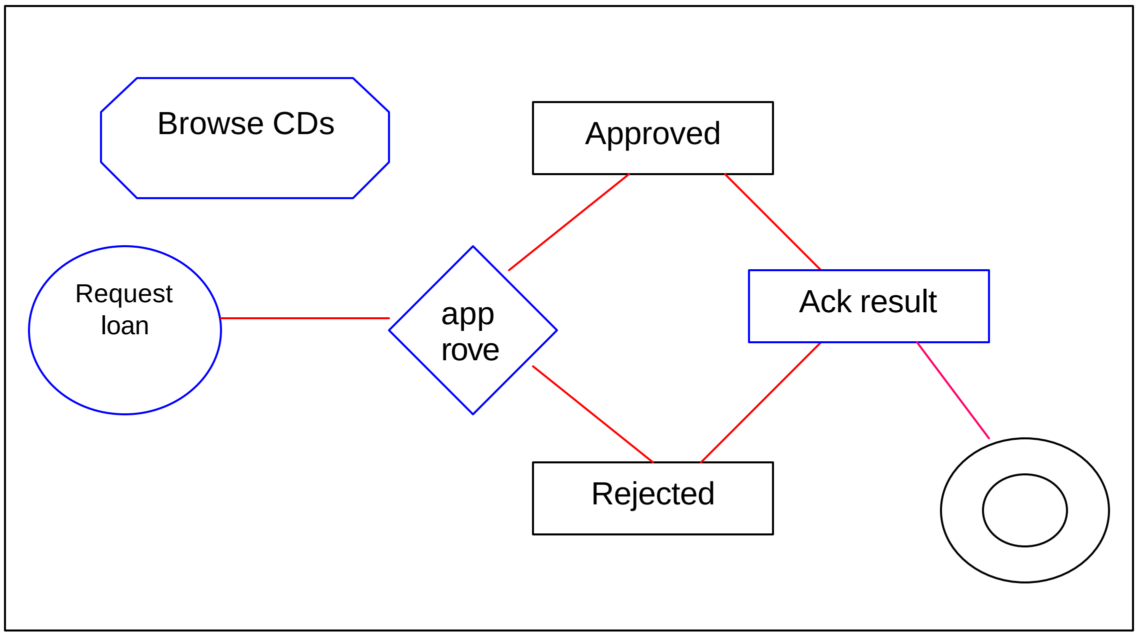
When a CD is returned the user administering the department where the CD belongs marks the CD as "available" in the collection.

Designing the process

The process will have the following activities:

- Browse CDs (A standalone activity where users can browse CDs, if the user is the CD admin he can also mark the CD as "available" or "not available" if he wants.
- Request CD (an interactive activity, theuser will have to enter the CD he wants and the instance will be sent to the CD admin)
- Approve CD (an interactive activity where the CD admin can approve or reject a loan, the user is notified of the result of the approval phase, if the loan is approved the CD status is changed to "not available")

This sketch shows the process



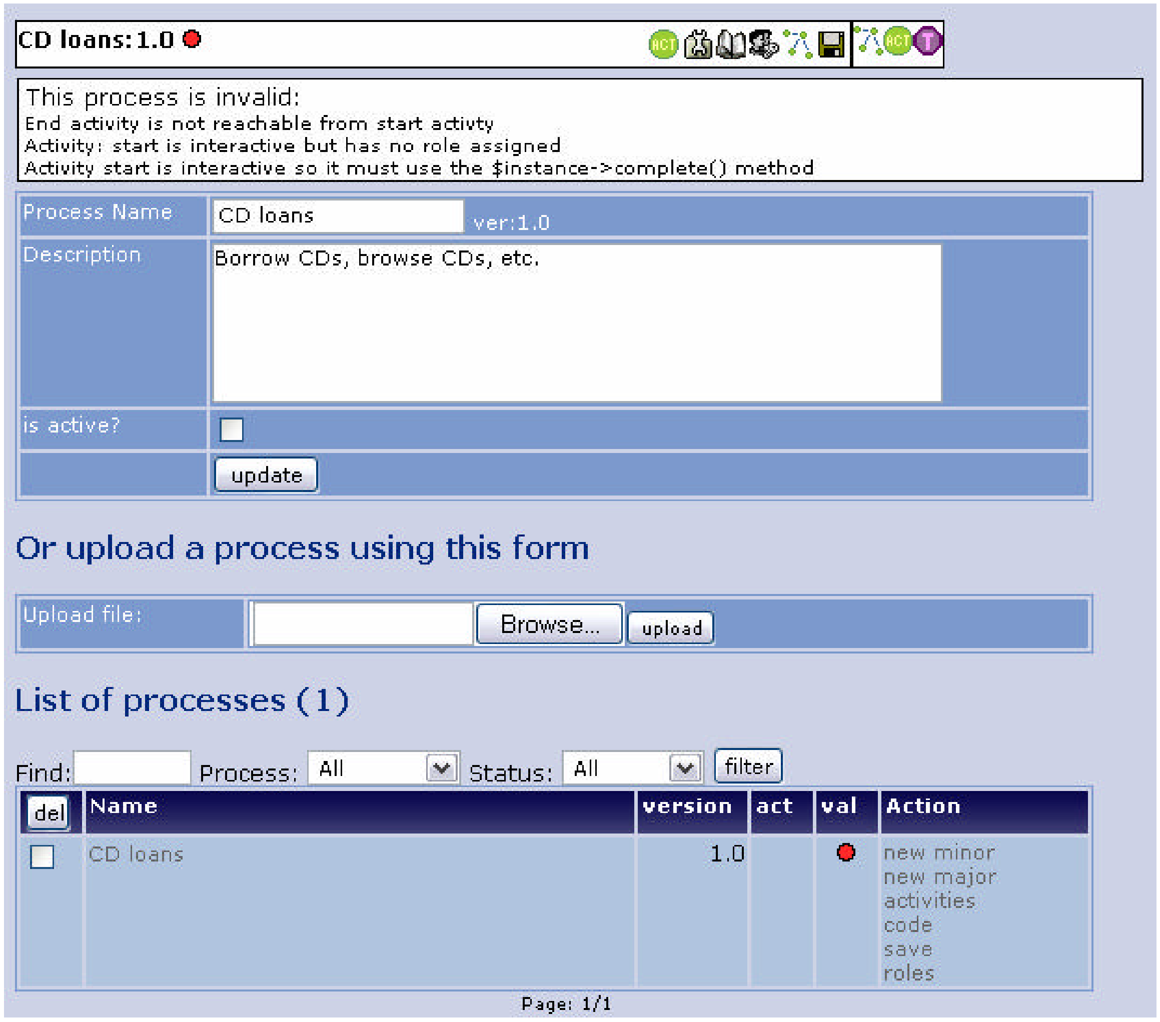
The following table describes each activity in the process

Activity	Type	Description
Browse CDs	Standalone/Interactive	The user lists CDs, he can click in a “loan” button if he wants to start a loan. For each CD it’s name and status are listed.
Request loan	Start/Interactive Auto-routed	A user enters the CD name and the number of days he will be using the CD.
Approve CD	Switch/Interactive Auto-routed	The CD admin sees the username, the cd title and the number of days the user needs a CD, he can either “accept” or “reject” the loan.
Approved	Activity/automatic Auto-routed	The CD is marked as “not available”. The status of the loan process is marked as “Approved”
Rejected	Activity/automatic Auto-Routed	The CD is not marked, the status of the loan is marked as “rejected”
Ack result	Activity/interactive Auto-Routed	The user sees the result of the loan and clicks “accept” to end the process.
End	End/Automatic	The process is terminated

Modeling the process using the Process Manager.

1st step: creating the process

Click in “admin processes” on the Workflow section of the application menu, enter the process name “CD loans” and a description. Click “create” to create the process.



CD loans:1.0 ●

This process is invalid:
End activity is not reachable from start activty
Activity: start is interactive but has no role assigned
Activity start is interactive so it must use the $instance->complete() method

Process Name	CD loans ver:1.0
Description	Borrow CDs, browse CDs, etc.
is active?	☐
	update

Or upload a process using this form

Upload file: Browse... upload

List of processes (1)

Find: Process: All Status: All filter

del	Name	version	act	val	Action
☐	CD loans	1.0		●	new minor new major activities code save roles

Page: 1/1

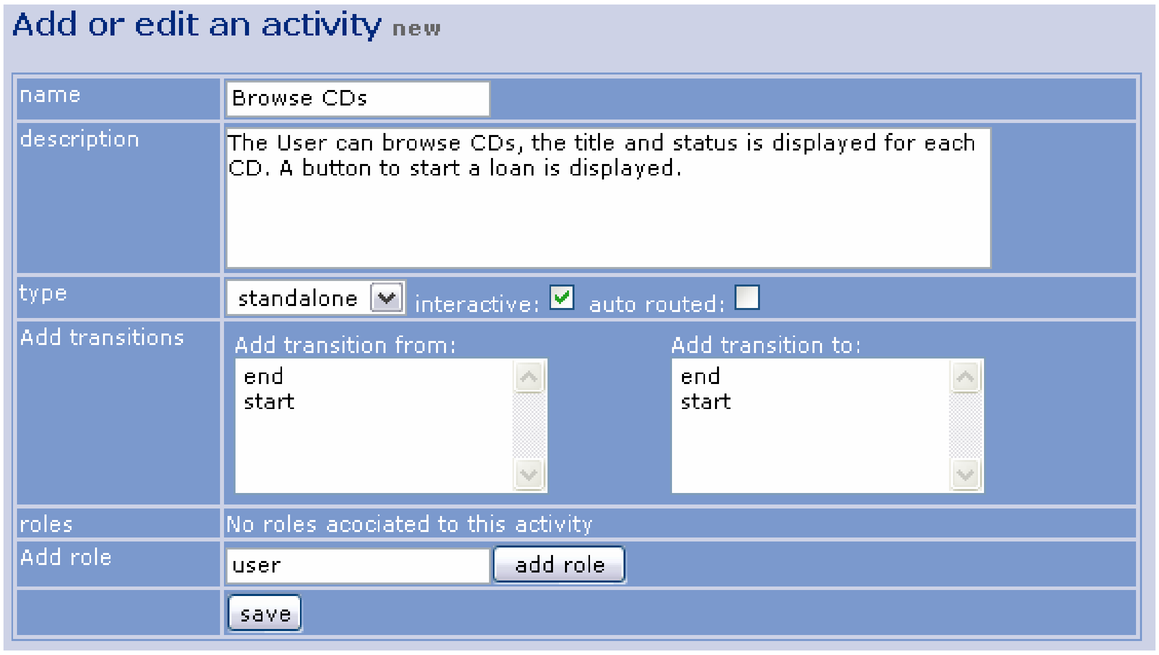
As you can see the process is created but it’s invalid for several reasons. Now click in the “activities” link in the “action” column where the process is listed or click the “act” icon in the process bar. You will be directed to the screen where the process activities can be managed.

2nd step: Entering process activities

In our second step we'll map our design adding the activities and the transitions (connections) between activities.

2.1 Create the "Browse CDs activity"

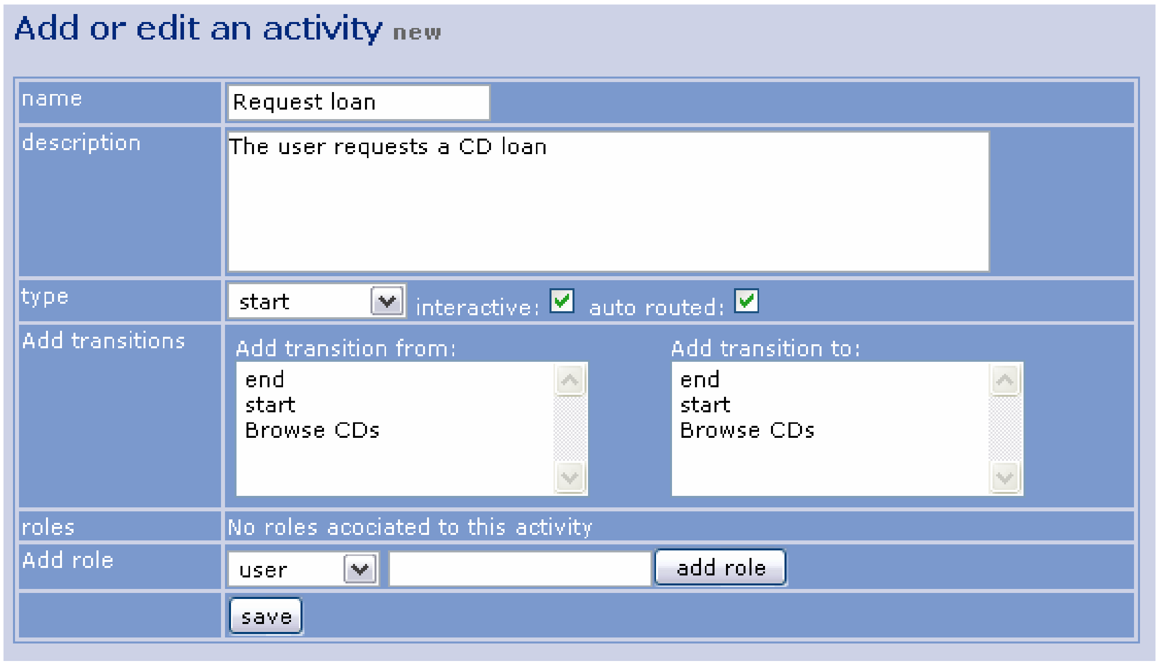
Enter an activity named "browse CDs", set the type as "standalone", set the role as "user" since any "user" of this process can browse CDs.


Add or edit an activity new

name	Browse CDs
description	The User can browse CDs, the title and status is displayed for each CD. A button to start a loan is displayed.
type	standalone interactive: ✓ auto routed: ☐
Add transitions	Add transition from: end, start — Add transition to: end, start
roles	No roles acociated to this activity
Add role	user [add role]
	[save]

2.2 Edit the "start activity"

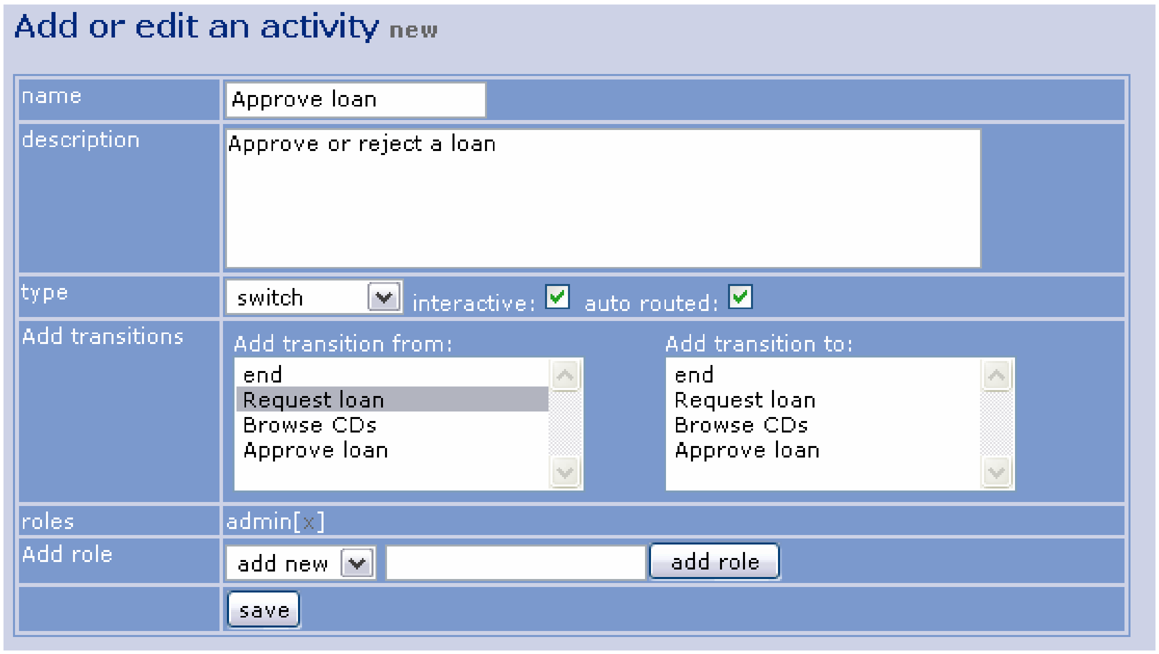
Edit the start activity, set the name as "request loan", set the type as interactive and auto-routed. Set the role as "user"


Add or edit an activity new

name	Request loan
description	The user requests a CD loan
type	start interactive: ✓ auto routed: ✓
Add transitions	Add transition from: end, start, Browse CDs — Add transition to: end, start, Browse CDs
roles	No roles acociated to this activity
Add role	user add role
	save

2.3 Create the “approve loan” activity

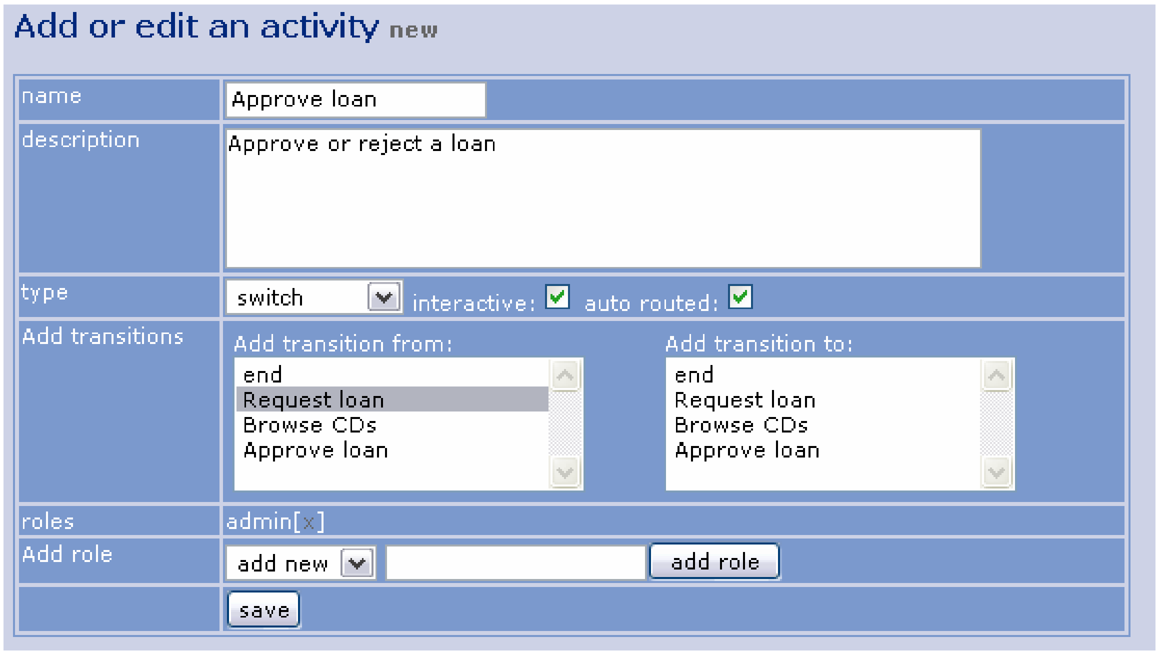
Create the “approve loan” activity clicking the “new” to create a new activity. Set the type to “switch, interactive, auto-routed”. Add a transition from “start” and enter a new role: admin because only the CD administrator can approve loans.


Add or edit an activity new

name	Approve loan
description	Approve or reject a loan
type	switch interactive: ✓ auto routed: ✓
Add transitions	Add transition from: end, Request loan, Browse CDs, Approve loan — Add transition to: end, Request loan, Browse CDs, Approve loan
roles	admin[x]
Add role	add new add role
	save

2.4 Create the “approved” activity.

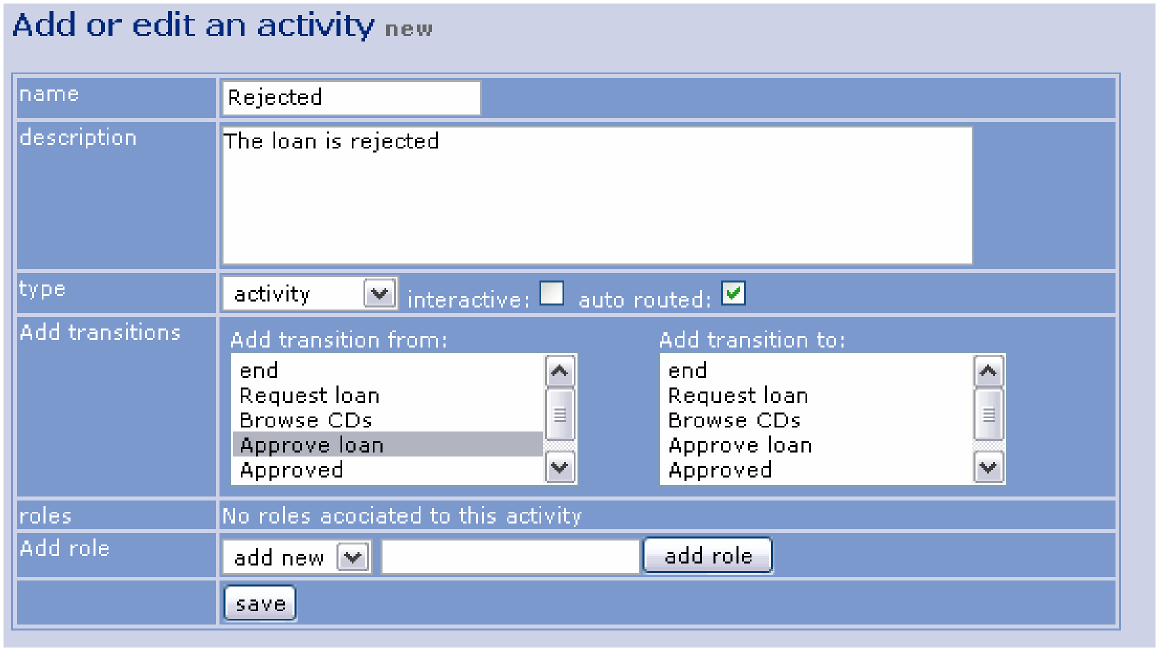
Create the “approved activity” set the type as “activity, automatic, auto-routed” add a transition from “approve loan”



Add or edit an activity new

name	Approve loan
description	Approve or reject a loan
type	switch interactive: ✓ auto routed: ✓
Add transitions	Add transition from: end, [✓ Request loan], Browse CDs, Approve loan — Add transition to: end, Request loan, Browse CDs, Approve loan
roles	admin[x]
Add role	add new, add role
	save

2.5 Create the “rejected activity”

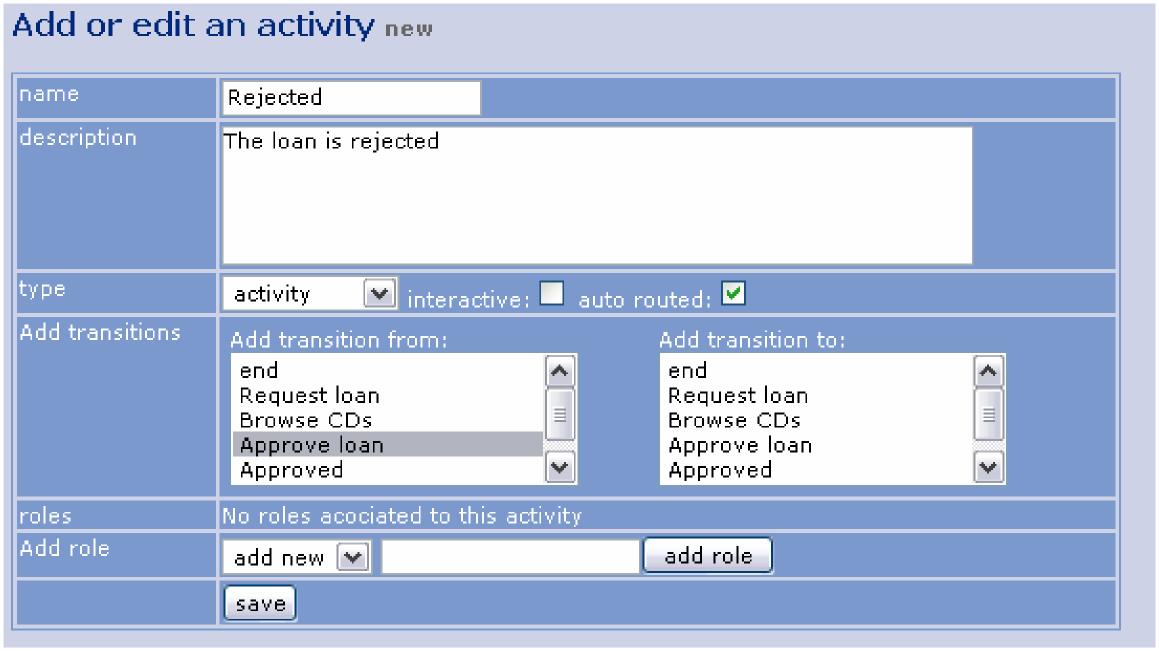


Add or edit an activity new

name	Rejected
description	The loan is rejected
type	activity interactive: ☐ auto routed: ✓
Add transitions	Add transition from: end, Request loan, Browse CDs, [✓ Approve loan], Approved — Add transition to: end, Request loan, Browse CDs, Approve loan, Approved
roles	No roles acociated to this activity
Add role	add new, add role
	save

2.6 Create the “ack result” activity.

Create the “ack result activity” as “activity, interactive, auto-routed”, add transitions from “approved” and “rejected” and add a transition to “end”. Set the role as “user”

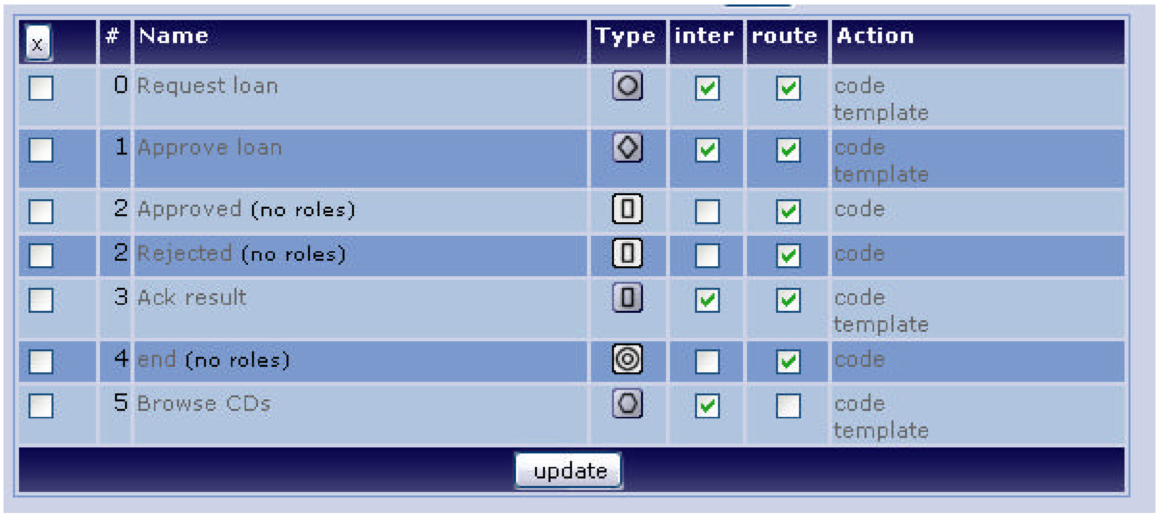


Add or edit an activity new

name	Rejected
description	The loan is rejected
type	activity, interactive: ☐ auto routed: ✓
Add transitions	Add transition from: end, Request loan, Browse CDs, Approve loan, Approved / Add transition to: end, Request loan, Browse CDs, Approve loan, Approved
roles	No roles acociated to this activity
Add role	add new

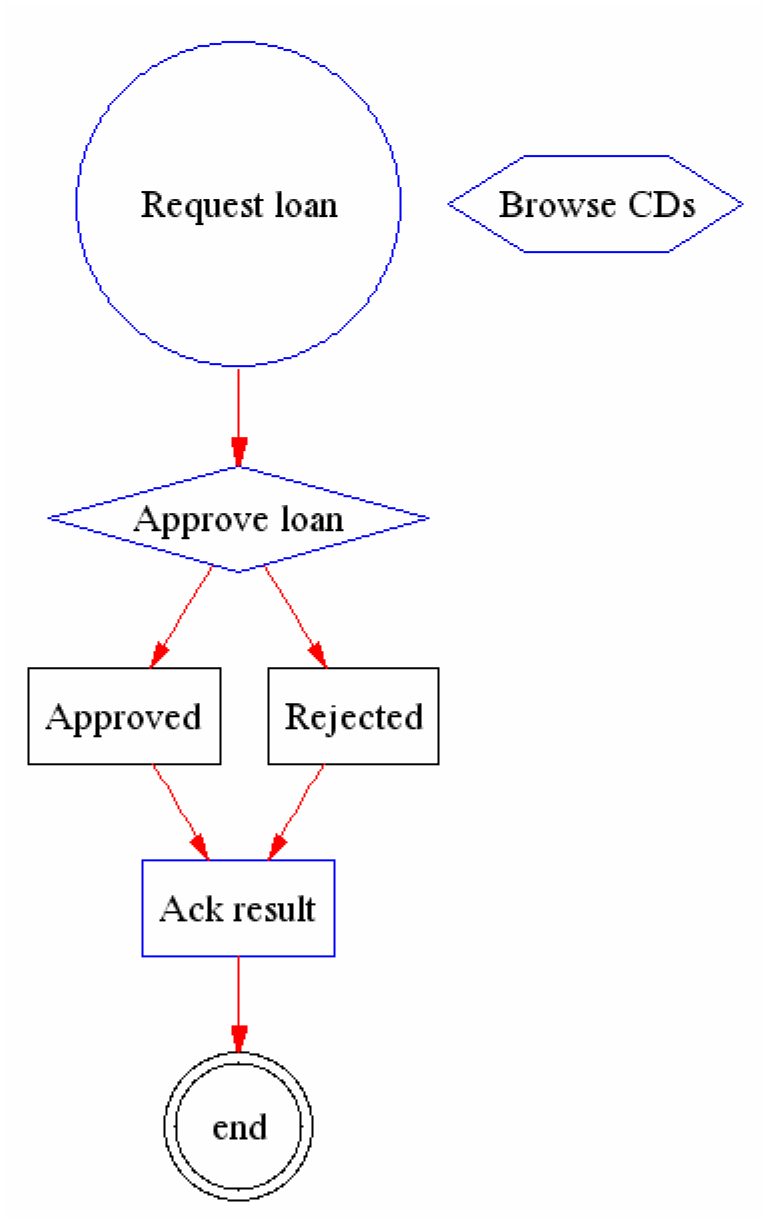
3rd step Verify your process design

The list of activities for your process should be similar to the following one:



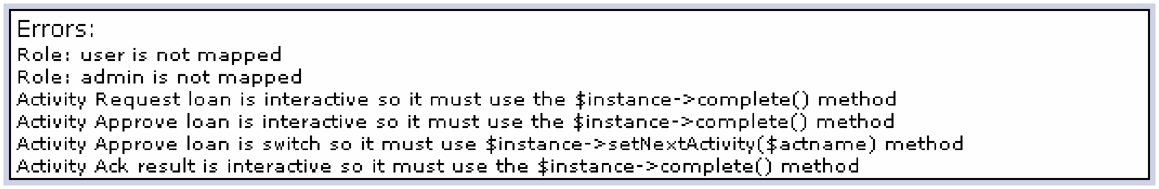
x	#	Name	Type	inter	route	Action
☐	0	Request loan	○	✓	✓	code template
☐	1	Approve loan	◇	✓	✓	code template
☐	2	Approved (no roles)	▯	☐	✓	code
☐	2	Rejected (no roles)	▯	☐	✓	code
☐	3	Ack result	▯	✓	✓	code template
☐	4	end (no roles)	◎	☐	✓	code
☐	5	Browse CDs	○	✓	☐	code template

And you can verify the process graph:



4th step: Analizing errors and solving them.

At the top of the admin activities screen you will see the list of errors for your process should be similar to this one:

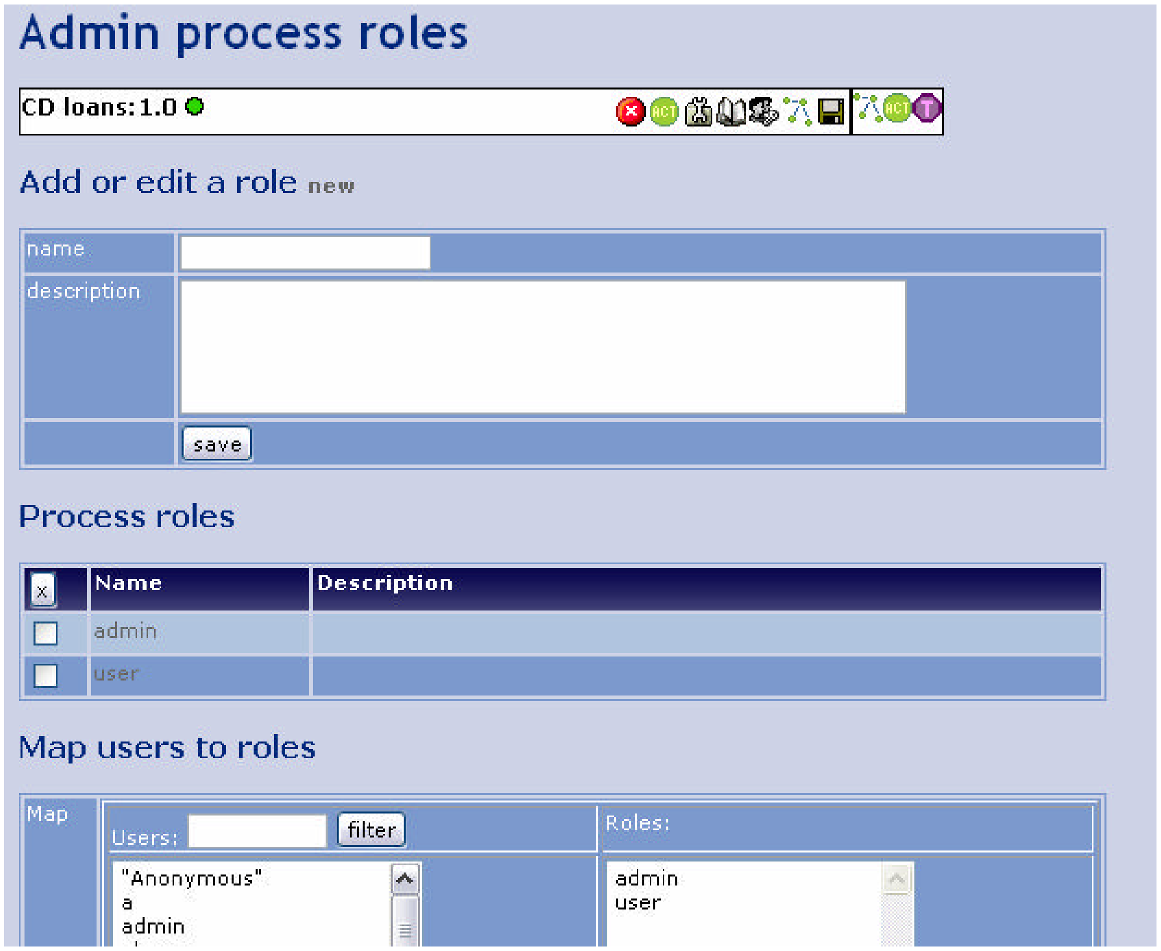


```
Errors:
Role: user is not mapped
Role: admin is not mapped
Activity Request loan is interactive so it must use the $instance->complete() method
Activity Approve loan is interactive so it must use the $instance->complete() method
Activity Approve loan is switch so it must use $instance->setNextActivity($actname) method
Activity Ack result is interactive so it must use the $instance->complete() method
```

The first to errors say that we have roles that are not mapped to any users in our Tiki system so no one will be able to execute those activities. The next 4 errors are related to the “code” for each activity since we have yet no code we have some errors.

5th step: Mapping users to roles.

Enter the role manager and map the “admin” and “user” roles to Tiki users in your systems, you can map entire groups to roles or individual users. In our simple example we mapped “admin” to the role “admin” and “test” to the role “user”. Once the roles are mapped the two errors related to roles should disappear from the process status.



Admin process roles

CD loans:1.0

Add or edit a role new

name	
description	
	save

Process roles

x	Name	Description
☐	admin	
☐	user	

Map users to roles

Map	Users: filter	Roles:
	"Anonymous" a admin	admin user

6th step: Coding activities.

Now the process is “designed” and the “coding” phase begins, PHP code and templates should be entered for each activity, as you will see you don’t need advanced PHP skills to code the activities, just some basics. We’ll design very simple (ugly) templates and code for this tutorial is up to you to improve the interfaces for the process screens, remember that an activity can do and look as any PHP program so the limit is your imagination/skills.

6.1 Preparing the code.

Before coding the activities we’ll need a table where the CD information will be stored, adding and removing CDs is not covered by this simple tutorial, you can easily add the features to the “browse CDs” activity with just some extra forms and PHP code.

The table will be very simple:

```
          drop table if exists cdcollection;
          create table cdcollection(
               cdId integer(12) not null auto_increment,
               title varchar(200),
               status varchar(40),
               user varchar(200),
               primary key(cdId)
          );
```

And insert some CDs to be used as test items.

```
insert         into         cdcollection(title,status,user)
values(‘cd1’,’available’,’’);
insert         into         cdcollection(title,status,user)
values(‘cd2’,’available’,’’);
insert         into         cdcollection(title,status,user)
values(‘cd3’,’available’,’’);
```

6.2 Coding the “browse CDs activity”

From the activity manager click on “code” next to any activity to enter the code editor where you can edit the code and templates for your process activities. If you use a PHP debugger or IDE it is a good idea to code your activities in your IDE and then simply paste the code into the Workflow code editor.

The browse CDs activity will have PHP code to list the CDs and a template to show them. The PHP code is:

```
<?php

  if(isset($_REQUEST['status'])                              &&
$_REQUEST['status']=='available') {
    $tikilib->query("update  cdcollection  set  status='not
available' where cdId=".$_REQUEST['cdId']);
  }

  if(isset($_REQUEST['status'])  &&  $_REQUEST['status']  ==
'not available') {
    $tikilib->query("update         cdcollection         set
status='available' where cdId=".$_REQUEST['cdId']);
  }


  // Get the CDs
  $query = "select * from cdcollection";
  $result = $tikilib->query($query);

  $ret = Array();
  while($res = $result->fetchRow(DB_FETCHMODE_ASSOC)) {
      $ret[] = $res;
  }
  $smarty->assign('cds',$ret);


?>
```

And the template is:

```
{*Smarty template*}
<h3>Listing CDs</h3>
<table class="normal">
<tr>
  <td class="heading">Title</td>
  <td class="heading">Status</td>
  <td class="heading">Action</td>
</tr>
{section name=ix loop=$cds}
<tr>
  <td class="odd">{$cds[ix].title}</td>
  <td class="odd">{$cds[ix].status}</td>
  <td class="odd">
    <a                                          class="link"
href="?activityId=3&amp;cdId={$cds[ix].cdId}&amp;status={$c
ds[ix].status}" >
      change status
    </a>
 </td>
</tr>
{/section}
</table>
```

In a more advanced version a link to the “request loan” start activity can be set from each CD so the user can directly start a request from the listing. (Hint: link to tiki-g-run_activity.php?activityId=XXX&cdId=XXX)

6.3 Coding the “request loan” activity.

This activity will show a dropdown where the user can select a CD and request its loan, only “available” CDs will be shown.

The PHP code:

```
<?php
$query    =    "select    *    from    cdcollection    where
status='available'";
$result = $tikilib->query($query);

$ret = Array();
while($res = $result->fetchRow(DB_FETCHMODE_ASSOC)) {
    $ret[] = $res;
}
$smarty->assign('cds',$ret);

if(isset($_REQUEST['request'])) {
  $instance->set('cdId',$_REQUEST['cd']);
  $instance->complete();
}
?>
```

The template

```
{*Smarty template*}
<h3>Request a CD</h3>
<form method="post">
```

```
Cd to request:
<select name="cd">
{section name=ix loop=$cds}
  <option  value="{$cds[ix].cdId}"  {if  $cds[ix].cdId  eq
$smarty.request.cdId}selected="selected"{/if}>{$cds[ix].tit
le}</option>
{/section}
</select>
<input type="submit" name="request" value="request" />
</form>
```

6.4 Coding the “approve loan” activity.

The information for the request will be displayed with “accept” and “request” buttons. Since this is a switch activity the setNextActivity method will be used to decide which activity will be executed next to this one.

PHP code:

```
<?php
  // Get the CD that is requested and assign CD info to
info
  $cdId = $instance->get('cdId');
  $query = "select * from cdcollection where cdId = $cdId";
  $result = $tikilib->query($query);
  $res = $result->fetchRow(DB_FETCHMODE_ASSOC);
  $smarty->assign('info',$res);

  $instance->set('title',$res['title']);

  if(isset($_REQUEST['approve'])) {
    $instance->setNextActivity('approved');

    $instance->complete();
  }
  if(isset($_REQUEST['reject'])) {
    $instance->setNextActivity('rejected');

    $instance->complete();
  }
?>
```

Template

```
{*Smarty template*}
```

```
<h3>Approve or reject CD loan</h3>
CD: {$info.title} <br/>
Status: {$info.status} <br/>
<form method="post">
<input type="submit" name="reject" value="reject" />
<input type="submit" name="approve" value="approve" />
</form>
```

6.5 Coding the “approved” activity.

This is just a dummy automatic activity setting up the status of the request.

PHP code

```
<?php
   $instance->set('status','approved');
   $cdId = $instance->get('cdId');
   $query = "update cdcollection set status='not available'
where cdId=$cdId";
   $tikilib->query($query);
?>
```

6.6 Coding the “rejected” activity

```
<?php
  $instance->set('status','rejected');
?>
```

6.7 Coding the “ack result” activity

PHP code:

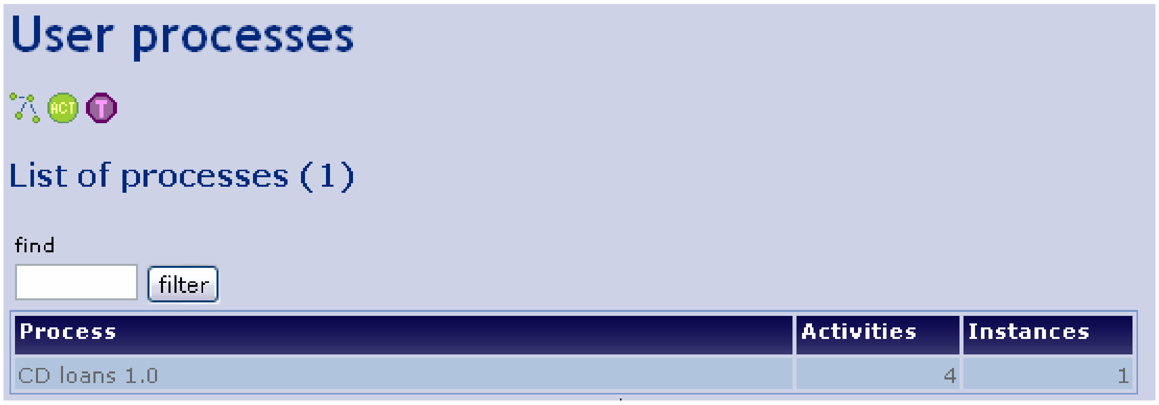
```
<?php
if(isset($_REQUEST['ack'])) {
  $instance->complete();
}
$smarty->assign('status', $instance->get('status'));
$smarty->assign('title',$instance->get('title'));
?>
```

Template

```
{*Smarty template*}
<h3>Loan result</h3>
Your request for {$title} was {$status}
<form method="post">
<input type="submit" name="ack" value="accept" />
</form>
```

7. Testing your process

Enter the UI clicking the “User processes” link from the Workflow section of the application menu:



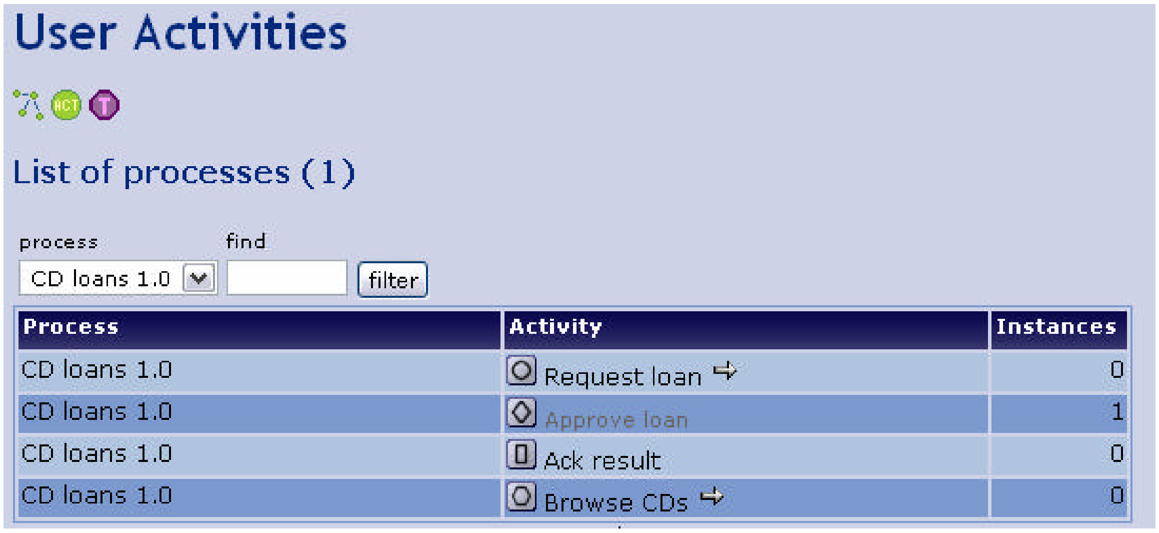
User processes

List of processes (1)

find

Process	Activities	Instances
CD loans 1.0	4	1

Click the “CD loans” title to view the process activities:



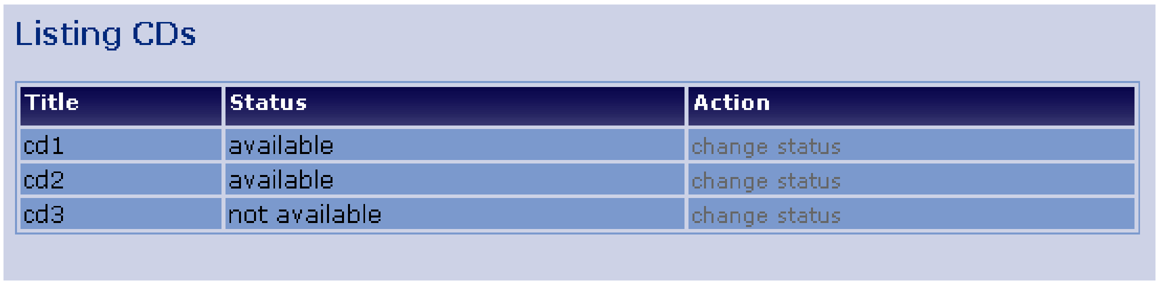
User Activities

List of processes (1)

process: CD loans 1.0

Process	Activity	Instances
CD loans 1.0	Request loan	0
CD loans 1.0	Approve loan	1
CD loans 1.0	Ack result	0
CD loans 1.0	Browse CDs	0

Click the arrow next to the "Browse CDs" activity to execute the "Browse CDs" activity.

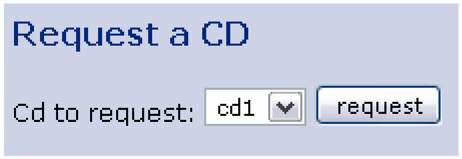


Listing CDs

Title	Status	Action
cd1	available	change status
cd2	available	change status
cd3	not available	change status

This screen is generated by the "Browse CDs" activity that you coded!, you can see the CDs in the "cdcollection table and click the "change status" button to change the CD status from "available" to "not available".

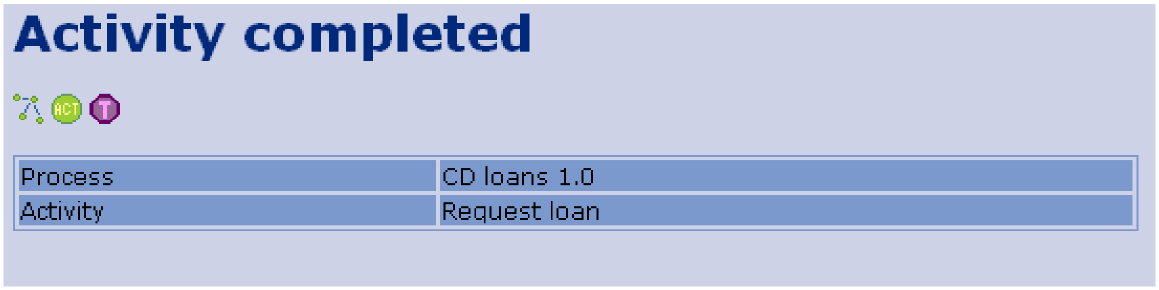
Now go back to the activities for the "CD loans" process and execute the "request loan" activity:



Request a CD

Cd to request: cd1

Note that only "available" CDs are present in the dropdown, select a CD and click "request"

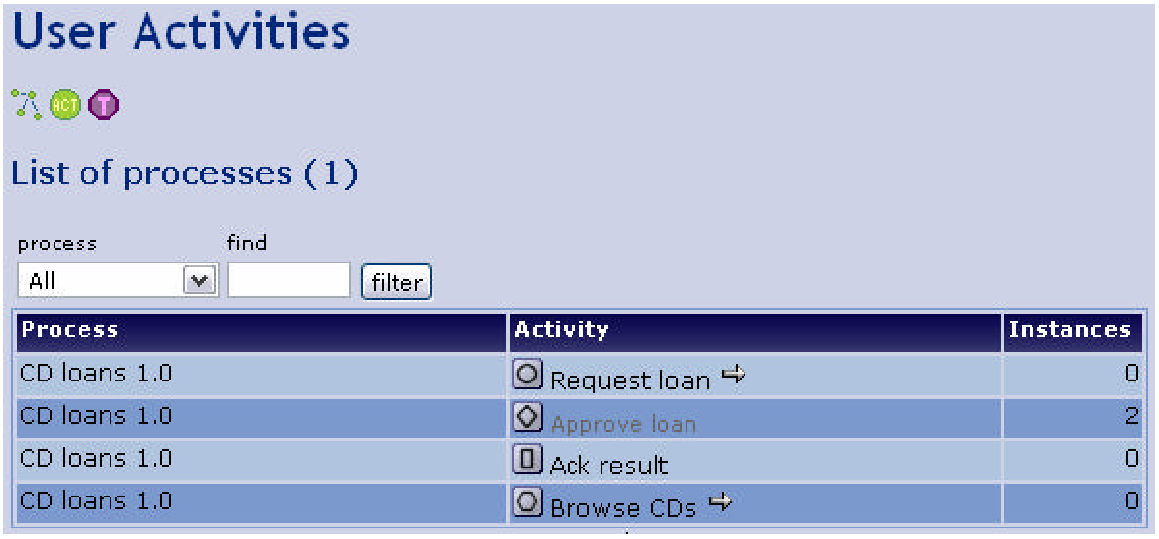


Activity completed

Process	CD loans 1.0
Activity	Request loan

The activity completed screen is displayed. This shows that the activity "Request loan" of the "CD loans 1.0" process was completed.

Now back to the activities:



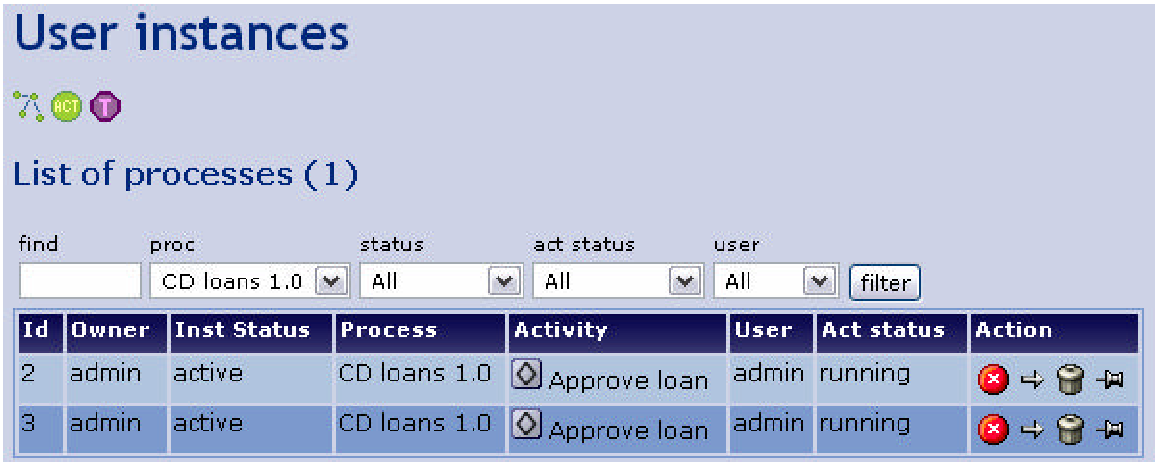
User Activities

List of processes (1)

process	find	
All		filter

Process	Activity	Instances
CD loans 1.0	Request loan	0
CD loans 1.0	Approve loan	2
CD loans 1.0	Ack result	0
CD loans 1.0	Browse CDs	0

As admin you will see now that the "Approve loan" activity has one or two instances present mening that there's work to do, click the "approve loan" icon to see the instances:



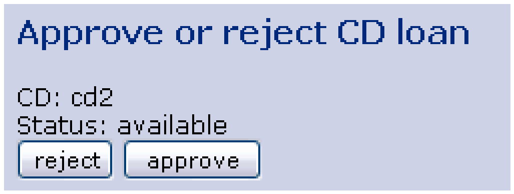
User instances

List of processes (1)

find	proc	status	act status	user	
	CD loans 1.0	All	All	All	filter

Id	Owner	Inst Status	Process	Activity	User	Act status	Action
2	admin	active	CD loans 1.0	Approve loan	admin	running	
3	admin	active	CD loans 1.0	Approve loan	admin	running	

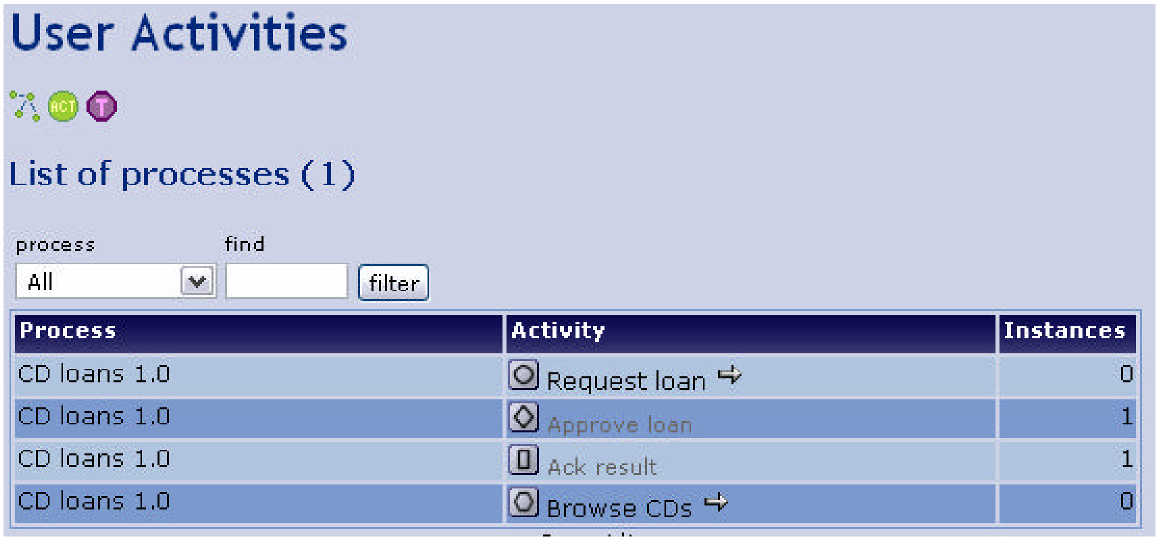
Note that the instances are "assigned" to admin because in our example only "admin" has the "admin" role for this process. If more than one user could execute the activity a "*" will be displayed in the user column and the user could be able to "grab" the instances he wants to execute. Click the "arrow" to execute the approve request activity for each instance.



Approve or reject CD loan

CD: cd2
Status: available

You can now "approve" or "reject" the loan. Choose one.



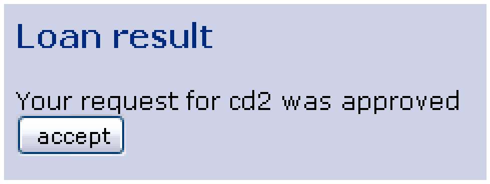
User Activities

List of processes (1)

process: All | find: | filter

Process	Activity	Instances
CD loans 1.0	Request loan	0
CD loans 1.0	Approve loan	1
CD loans 1.0	Ack result	1
CD loans 1.0	Browse CDs	0

Now an instance disappeared from “approve loan” and appeared a new instance in “ack result” as a user click the “ack result” link to view the instances and execute the activity.



Loan result

Your request for cd2 was approved

accept

Once the user acks the result of the request the process is terminated, if you accepted the loan use the “Browse CDs” activity to verify that the CD is now “not available”

Note: The “approve” or “reject” activity should set the user for the ackActivity as the user requesting the loan so only the user who requested the loan can ack the result. Create a “user” property for the instance in the “request loan” and set it to $user, then use setNextUser in “approve” and “reject” to set the user for the next activity to the value of the “user” property of the instance.