How To Create A View

▼ Details

This tutorial explains some of the rendering features of ERP5. In our example, the purpose is to display a list of all Discussion Threads, then below each Discussion Thread, all Discussion Posts which is made from the specific Discussion Thread.

Agenda

- Add State in listbox and State string field in object document
- Organise the view of object document
- Add variables in object document
- Add Relation String Field to link two objects
- Customise your listbox
- Create script to post and reply in the forum

▼ Details

Modify listbox in module: Add State (Change the "State variable name") (Module listbox)

▼ Details

When you set a workflow to an object, the object acquires new variables (like state, date, etc). Now we will learn how to make these variables displayed in the view.

We have created the workflow for our forum module, now we will publish the **State** names of our workflow for each thread, so that the states can be displayed in the Discussion Thread **module listbox view**.

Before modifying the listbox, please go to **/portal_workflow**, then click on the name of the workflow that we are building (discussion_thread_workflow) and in the "**Variables**" tab, change the "**State variable**" property from simulation_state to **validation_state**.

We will use this variable to send the workflow state of your threads to the forum. You can also access this state in python: # print my_workflow.validation_state.

Enable developer mode

▼ Details

We will now need to use **Developer Mode** to facilitate the use of ERP5 user interface and developer interface.

First, go back to your ERP5 instance main page, click on My favourites in your ERP5 instance main page >> Preferences >> Default Site Preference >> User interface. After, check the **Developer Mode** checkbox and save it. Then go to Action tab, and click on **Enable Preferences**.

Now when you open Discussion Threads module, you can already see the **developer button** (the one in blue on the slide.

Modify listbox "Columns" property

▼ Details

Now we will modify the listbox to display the state property of threads. Click on the **Developer Button** on top of the threads listbox in Discussion Threads module, you will be led into the developer page of the listbox directly. Click on **"listbox (Discussion Threads)"** and modify the one of the properties of the listbox **Columns** by adding **"translated_validation_state_title | State"** ("validation_state | State", "translated_" is for the translation of state for your website in various languages).

Be careful of auto-proxifying when a box is checked at the left of a property, it means that the property will be inherited from its parent element: if you configure columns when the checkbox is checked, you will not have the good output (You must uncheck the left checkbox at the columns box and save it for the change to be effective).

Go back to the module main page, the listbox which listed only the titles of the threads will now also list the state of the workflow.

Add State string field in object view (Thread)

▼ Details

After displaying variables from workflow in module listbox, how to make them shown in the object document's view? Now we will learn to add a **String Field** in our **Discussion Thread View** to show the **State** of the workflow.

Go to the main page and click on Discussion Threads, then click on a Thread. That is the **Discussion Thread view**. Now click on the **developer button** as circled in blue in the slide.

On the up right side of the developer page, choose **"StringField"** and add it.

After you define the **ID and Title** of this string field, click on **"Add and edit"**, **uncheck "Editable"** in Validator properties. This will make the **state of thread object not editable**, so that the users' interaction will not break the system's workflow which has been settled.

In order to access the state through **"my_translated_'NAME_OF_MY_STATE'_title"**, define the ID of the string field **"my_translated_validation_state_title"**. You can also verify if the getters are well generated by calling them directly: erp5/"id module"/"id thread"/getValidationState.

If you have problems with this rendering, e.g. nothing showing in State, here are some explanations of what can cause this: It can be your old discussion_thread_module, we made many changes on the portal_type and they did not affect items created before. The best thing to do is to **delete all objects in the module**. You can do it with the delete button, or use manage_main on the module, select all objects in the list field and use "delete" button at the bottom. Another behavior that can be encountered at this step can be caused by the cache system. You can **clean the cache in portal_caches**. The selection of your listbox (a list of what must be listed in it) can also be a problem for you. You can **reset the listbox setting** by using "Show All" button or by going in portal_selections. These two points are not really a problem, they are just an optimization of ERP5.

Organize the view (Thread)

▼ Details

We will now create and handle the "view" of a document in ERP5. We will first organize the objects in the view for the rendering in the page.

Go again to the **developer page of Discussion Thread View** and select the **Order tab**.

The order tab of a view will determine where will be displayed the elements of the View form. To help common users of ERP5 who don't know anything about css, we have developed this tool which splits any page in five big parts. A view is separated into **4 main parts**, three (left, right, center) are delimited by a border and the bottom is at the end of the page, the Hidden part is for the objects that are not shown.

To modify the order of the elements, choose the **Move To** movement and use **Transfer** to move an element between the parts, or **Move Up** and **Move Down** to move an element above or below another one.

If these different parts are not in your view, you have to create them with the **Create** button. This would happen if you created your view from scratch.

Four parts of a ERP5 object view

▼ Details

This is a typical view of a ERP5 object: a Persons page: Person_view. We can see that the four parts of the view are used. This view describes a person, the Left part for the name and the position information, the Right part the address, phone number, the Center part for the description and the Bottom part is a listbox for the coordinates.

You can customise the objects between different parts thanks to the view renderer we just described.

A view example

▼ Details

Go to the Order tab to modify the order of your Discussion Thread View form, and arrange it to obtain this result.

Add variables: Modify object type (Post)

▼ Details

We already know that when you **set a workflow** to an object, the object acquires new variables (like state, date, etc). And we have already learned to make these variables displayed in the view. There are two other ways to add a variable to an object: change its **meta type** or **create a property sheet** for it. We will learn to add new variables which are not related to workflow by changing the object's type, and how to make them displayed in the object's view.

Our example-object **Post** should only contain a **text area** and a **title**. Now the meta type of the Discussion Post is **"ERP5 XML Object"**, that's a basic type with limited possibilities. To increase these possibilities, we will change the meta type of our Post.

Go to the **My Favourites bar >> Configure Portal Types >> Discussion Post** Change the values of the **Class** property and **Property Sheets** to **"TextDocument"**.

The **meta type** is a type already defined in ERP5 and has a defined property sheet. This change affects all Discussion Post objects.

Add Content string field

▼ Details

Same as the steps to show State variable, now we will add and show a new string field-Content.

In your Discussion Thread View, create a new Discussion Post. In the **Discussion Post View**, click on the **developer button** and create a **StringField** with **"my_text_content"** as ID. You must create another Post object to test this modification.

Fill the new inputs generated by (Id and my_text_content) and check if the getters are well calling. The URL:

erp5/"id module"/"id thread"/"id post"/getTextContent. You should only do this after putting some value on the field, otherwise the browser might not display anything.

Add variables: Create PropertySheet (Thread)

▼ Details

Now we will use the third way to add a variable to an object: creating PropertySheet. In our example, we will add a **Rating** value in the **Thread**.

My Favourites >> Configure Property Sheets. Click on **Add Property Sheet in Action menu** to create a new Property Sheet named **DiscussionThread**, then **Add Standard Property** called **"rating_property"** with the Description "the rating of the thread" and for **Property Type** choose **"string"**. Don't forget to save it.

The **Property Sheet** is a python file located in the Property Sheet folder of your instance. It is a simple way to add a special property to an object. Many properties already exist in some other Property Sheets, but if you want a special value like the rating for the forum, you must create yourself a new Property Sheet for your object. Be very careful not to use existing names either from the Property Sheet class itself or of any of its properties.

Use the PropertySheet

▼ Details

To be able to use this new property, go to **erp5/portal_types/manage_main**, select **Discussion Thread** Object and click on **Edit**. Then go to the View tab and select **DiscussionThread** on the **Property Sheets category**.

When you select a property sheet for a type of object, the object acquires all the getters/setters for all the properties of the sheet. In this case we want to get the DiscussionThread property, but if you know what are all the other Property Sheets and want some of them for your object type, you can select many of them for one type.

Now, we will use this property sheet. Go to the **ERP5 Main Page >> Modules tab >> Discussion Threads >> Thread** and then click on the **developer button**, **add new String Field named "my_rating"** . Verify that the getters are well generated directly in your thread (as illustration).

Use Relation string field to link two objects (Post)

▼ Details

Now we will use a relation string field to link a **Discussion Post** object to a **Person** object.

The **Relation String Field** is a special variable which allows linking a type of object with another. You need to get a special category to specify what kind of object to hold here, from your **Property Sheet** or just use the **Base Categories** selection box in the Properties tab of your object's type. We will do this second solution.

Go to **/portal_types/manage_main**, select **Discussion Post** and edit. After that, go to the **View** tab and select the category **"source" in type_base_category_list**, save changes.

Now we need to **create the Relation String Field**, as we have practiced before: go to Main Page >> Modules tab >> Discussion Threads >> Thread and add a new Discussion Post. On the **Discussion Post View**, click on the **developer button** and create a **RelationStringField** named **"my_source_title"**.

Configure your Relation string field

▼ Details

From the screenshot you can see Relation string field is different from the others-it has a wheel icon which is used to link tow objects. So we have to configure it by setting up the main properties of the RelationStringField.

Click on the **developer button** of **the Relation string field "Source"**, you will be led directly to the developer page of **"my_source_title"**.

Fill the **Base Category for the link with first object (here itself, "source")** the **Portal Type with the type you want to link (here a "Person" type)**, and you also need to specify what property will be used to index the objects by filling the **Catalog Index** input.

Test the Relation string field

▼ Details

Now that you've filled the properties of your RelationStringField, you can link it with the object you want.

Here it will be the owner of the Post. To verify if we configured it well, we will choose it manually.

First go to Main Page >> Modules tab >> Discussion Threads >> Thread and add a **new Discussion Post**. On the Discussion Post View click on the **wheel** at the right.

You will be redirected to a listbox which lists all the objects of the type that you have selected (here Person). If there is no Person objects exist, you must create it with Person module in your instance.

In every ERP5 form the action buttons in the bottom are related to the selected CheckBox at the left of the object list, here you just need to **check the box** and to click on the **"Set Relation"** button.

If you have followed this tutorial correctly, you should get the **title of your object** in the **Source field** input of your previous form.

Create your own listbox template (Post)

▼ Details

Now our threads and posts get many values but we can't use a forum by creating manually each post.

The listbox which lists the posts of a thread is very useful for listing but not for reading. So we will create our own template for the listbox, to list all the posts in the thread and show them in a pretty view like any normal forum. We will copy the **basic template of a listbox**, and we will **modify it and assign the new template to the listbox**

First we have to find the script of the basic listbox template. Go to **/portal_skins/manage_main**, use the **Find tab** and fill the **"With ids"** of the element you search for, here **"ListBox_asHTML"** and validate the research.

When you have found the script, **copy** it (go back to the portal skins root folder, check the checkbox at the left and use the Copy button at the bottom of the page) and Paste it in your **skin folder (erp5_forum)**, which is in **/portal_skins/erp5_forum/manage_main**.

Rename it to "ListBox_threadAsHTML".

Modify the code of listbox template-replace comment lines

▼ Details

Now you can edit your listbox template by clicking on**"ListBox_threadAsHTML"**. The comment lines are useless for us, so find the first line of this block **<div class="listbox-container">**, and replace it (including**<div class=""listbox-container"">**) by the code showing in the slide.

Modify the code of listbox template-add post's listbox code

▼ Details

We will add the code for our listbox template.

After the end of the previous code, you can put the block shown on the slide and delete the last line "</tal:block>" (it has been in the code in the slide). This block will generate an iteration on each post that is in our thread.

If you want to use the dynamic of Zope here you should read more about zope and tales, zope have its own template language, learn more about it at:
http://wiki.zope.org/ZPT/TALSpecification14 .

In the next slide you will see a small example of template code to fill this part.

The example template's code

▼ Details

This is an example of template page with a css code, and the full file can be found in ERP5 git repository.

The css are directly in the code due to tutorial restriction, but normally all the css must be placed in a css file in your skin folder and included at the top of the page like all other traditional web pages.

ListBox Template

▼ Details

Your code should be similar to the one in the slide.

Click on **Save Changes** button in the end.

Add ListBox

▼ Details

Now we need to create a Listbox to display the posts inside a thread.

For that, go to **Main Page >> Modules tab >> Discussion Threads >> Thread** On the Discussion Thread View click on the **developer button** and add a **Listbox** named **"my_listbox" and with the title "Threads"**.

After that, click on the Listbox to see the**Edit tab** and fill the fields like the illustration. Be careful with the property**Page Template** at the bottom, put**"ListBox_threadAsHTML"**.

The new listbox

▼ Details

Now if you go to main page and click on a Thread in Discuss will see the listbox we created, which is applied to our template.

It's possible that you must create a new thread, because we have modified an object type and as we have already said it is possible that the previous objects have not been modified.

Modify listbox in module: Add Dates and Rating (Module listbox)

▼ Details

After creating the post listbox in the Thread document, we will now modify another listbox - the thread listbox in Discussion Threads module to make it more complete.

For that, go to **Main Page >> Modules tab >> Discussion Threads**. On the Discussion Threads click on the **developer button** and click on **listbox**.

In the Discussion Thread listbox form, you can add the **columns Rating** (added with the propertySheet) and the **creation_date** (generated by the workflow).

The **Default Sort** field allows to sort the contain of the listbox with the property input, be careful after "|" is the style "asc" (ascending) or "desc" (descending for the sort).

Remember to uncheck the box in left of the fields to fill in new information.

Modify listbox in module: Add Count (Use TALES to override default values) (Module listbox)

▼ Details

We will now use a **TALES "override" expression** to count the replies to a thread in the Discussion Threads listbox. A TALES override allows you to calculate dynamically a property of a form field in context instead of hardcoding its value. After the TALES override is **defined for a property of the form field**, this property will be surrounded by "[ ]" in the **Edit tab** of the form field. The **cell** is the element iterated by the listbox, **objectIds** return a list of all the id of the object's sub-elements.

We want the total of posts less one because you want to display the number of replies. Go to **Main Page >> Modules >> Discussion Threads** and then click on the **developer button**. Add a **StringField named listbox_count**, edit it and set the **Default property in the TALES tab** by this little script: **python: len(cell.objectIds())** (do not put it in the edit tab).

Use the TALES value

▼ Details

We will now add a column in the Discussion Thread listbox for display the count of replies.

Change the DiscussionThread listbox view to add the **columns "Reply" with the attribute "Count"** (which call the TALES override of the default property of the listbox_count). You would then have this rendered.

Hide a StringField

▼ Details

You can see in the module view page the previously default value (listbox_count) defined in the listbox. We will hide this StringField.

This tip will make the StringField not editable and disable it in the main page, the best way to hide it is to transfer it in the hidden part of the page, in the **Order tab** of the **Discussion Threads View**. If you don't see any hidden emplacement, you can create it in the Order tab, just by giving the name "hidden" to a section and when an element will be in it, it will not be rendered.

Create scripts to post threads (Module)

▼ Details

We have now created a sweet listbox for the posts in threads. In order to make our forum system more intuitive, we will create an **user interface** mechanism to create a **thread and the first post inside it at the same time**

We will need a Python Script to post a thread. The first to authenticate the current user who's posting the thread and the second to manage the new posting.

In the **/portal_skins/manage_main**, go to your forum folder (here **erp5_forum**) and create a **Script (Python)**, name it **"DiscussionThreadModule_addThread"**.

Edit the script

▼ Details

Create (or edit) the Python (Script) object DiscussionThreadModule_addThread. Replace the default body of that script with the text in the box on this slide, which can be found in ERP5 repo (remove batch_mode lines).

This script **creates and publishes a new thread with its embedded first post** It calls the script ERP5Site_getAuthenticatedMemberPersonValue, which can be found in /portal_skins >> erp5_base. This erp5_base is a Business Template very important for ERP5 system.

You also need to specify the parameters accepted by this script in the **Parameter List (title, text_content, form_id, **kw)**, which are necessary for it to be called as the action script of a form.

Be careful with the " " in the script. If some error appears, try to substitute them.

Create the post form

▼ Details

Continue in the **erp5_forum in Portal Skins folder** and add a new **ERP5 Form** named **"DiscussionThreadModule_viewAddThreadDialog"**, then create in this form two new **StringField: "your_text_content" and "your_title"**. The your_ prefix is a naming convention for forms that are evaluated in the context of one object, but are actually collecting objects for another object. In this case, the form is rendered in the context of the **thread** object but the form fields are collecting information for a **post** object.

Change the type of the form

▼ Details

This form needs to call the script defined previously, so navigate to the **Setting tab** of the **DiscussionThreadModule_viewAddThreadDialog** and change the contents of the **Form action** property to **"DiscussionThreadModule_addThread"**, the name of the script. Also please change the **Page Template** property to **"form_dialog"**, as we don't want the usual tab elements to be displayed along with this form.

Associate a post action with the script

▼ Details

Before testing the creation for a posting thread, we will associate the script to our module with a new action.

In **portal_types**, edit the **Discussion Thread Module** type, then in **Action tab** you select **Add Action Information** and fill these properties:

- **Title** "New Thread", **Reference** "new_thread",
- **Action** "string:${object_url}/DiscussionThreadModule_viewAddThreadDialog",
- **Permission** "Add portal content"
- **Category** "object_action".

Afterwards, go to the listbox in the **developer page of the Discussion Thread View**, and check if the **"List method"** category is **"contentValues"** (be careful not to misspell it, as it will fail silently if you do). The original searchFolder method uses the catalog, which is updated asynchronously and won't list the embedded post immediately after you created it.

Test the script to add threads in your module

▼ Details

We will test the creation of a new thread from forum module with the scripts.

In your module choose the **"Action..." >> "New Thread", fill both inputs and post the thread. Update the view.**

With our forum, we can add a posting thread which looks like traditional forum with some posts.

If after the test you get an Attribute Error, go to Portal Types >> Discussion Thread and check the Init Script ID field. Make sure that there is nothing written there.

Create the reply script (Thread)

▼ Details

We have some threads with some posts now, but we can't reply to a post yet. So we will create the reply process. It is similar to the post process, but we will associate the reply script to a thread, instead of a module.

For that, go to **portal_skins/manage_main>>erp5_forum** and add a new **Script (python)** named **"DiscussionThreadModule_addReply"**. Change the default code by the code in the slide, which is also on ERP5 repository (remove batch_mode-related lines), with **"title, text_content, form_id, **kw" in Parameter list** You can see that this script uses the previous script of authentication and the content of a post.

Be careful with the " ", cause this may cause you some problems.

Create the reply form

▼ Details

Here we create a form for the reply. You do exactly the same thing as for the previous form.

Be careful, you must have one **ERP5 form** with **two StringField** as in the illustration.

Change the type of the form

▼ Details

In the **Settings tab**, we edit the **Form action** field to point to the last script we created: **DiscussionThreadModule_addReply**.

Don't forget to modify the **Page Template input: set form_dialog**

Associate a reply action with the script

▼ Details

Before testing the creation for a replying to a post, we will associate the script to our module with a new action.

In **portal_types**

, click the Discussion Thread type, then in Action tab you select Add Action Information and fill these properties:
- **Title "New Reply", reference "new_reply",**
- **Action "string:${object_url}/DiscussionThreadModule_viewAddReplyDialog",**
- **Permission "Add portal content"**
- **Category "object_action".**

Now, you can test the reply.

Test to reply to a thread

▼ Details

To test the reply, you click on **"Reply" in a post which is listed in a thread**, and fill both inputs, you can see that the new post is well created.

To fix the date you **assign a workflow (here edit_workflow)** in the **Discussion Post** type which is in **Workflows tab** of the **portal_workflow**.

You can now create a thread with a post and reply.