Advanced different-space publishing

Overview

In this example, we'll look at some advanced markup including customized approvals, queued actions, custom events, and error handling.

Before you start

If you haven't already done so, please read Different-space publishing and complete the required configuration for the local instance of Confluence.

Basic Markup

Here is the basic markup from the Different-space publishing page.

```
{workflow:name=Different-space Publishing}
   {state:Editing|submit=Review}
   {state}
   {state:Review|approved=Published|rejected=Editing}
      {approval:Review|assignable=true}
   {state}
   {state:Published|final=true|updated=Editing}
   {state}
   {trigger:statechanged|state=Published}
      {publish-page}
   {trigger}
{workflow}
```

While the above example will enable the different-space publishing, there are some potential issues that should be considered:

- how long does the content take to publish, particularly in cases where several pages might be published at the same time?
- what happens if an error occurs? The publishing app might be mid-upgrade, or the configuration could be invalid, etc.

To accommodate these potential issues, we'll make the following updates to the workflow markup

1. Add a new **Synchronise** state which will trigger the publishing process
2. Add some custom events to handle success and failure scenarios
3. Add an **Error** state to decide what happens if synchronization fails

Adding the states

Here is the markup from above, with the added states **Synchronise** and **Error**.

We have also

- updated the trigger to be activated by the **Synchronise** state change rather than the **Published** state
- also, the **Review** state will now transition to **Synchronise** instead of **Published** when approved

```
{workflow:name=Different-space Publishing}
   {state:Editing|submit=Review}
   {state}
   {state:Review|approved=Synchronise|rejected=Editing}
      {approval:Review|assignable=true}
   {state}
   {state:Synchronise}
   {state}
   {state:Error}
   {error}
   {state:Published|final=true|updated=Editing}
   {state}
   {trigger:statechanged|state=Synchronise}
      {publish-page}
   {trigger}
{workflow}
```

Improving the states

Currently, the **Synchronise** state will show the default state selection drop-down, which isn't what we want.

Let's hide that using the **`hideselection`** parameter, and we'll also give the state a **`description`** to explain what's going on to anyone who opens the workflow popup during synchronization.

```
   {state:Synchronise|hideselection=true|description=Content synchronization in progress, please wait...}
   {state}
```

We should probably show an on-screen message during synchronization too, using a **`{set-message}`** action in the trigger.

```
   {trigger:statechanged|state=Synchronise}
      {set-message}
         Content synchronization in progress...
      {set-message}
      {publish-page}
   {trigger}
```

Our **Error** state also needs some work. Let's

- add buttons to **Retry** or **Ignore** – we can achieve that using a customized **`{approval}`**
- hide the **Error** state from the Progress tracker using the **`hidefrompath`** parameter
- set the color for the state to be red

```
   {state:
Error|colour=#ff0000|approved=Synchronise|rejected=Published|hidefrompath=true|description=Synchronization
failed, what do you want to do?}
      {approval:Error|approvelabel=Retry|rejectlabel=Ignore}
   {state}
```

Queued and custom triggers

Complex content might take a few moments to synchronize, especially if an entire page hierarchy is being published at the same time.

So we'll **queue** the actions in the trigger.

Here's the updated trigger set on the state change event to the **Synchronise** state.

```
{trigger:statechanged|state=Synchronise|queue=true}
   {set-message}
      Synchronising content...
   {set-message}
   {publish-page}
{trigger}
```

If the server is busy, it may take a moment before the queued actions are processed.

We want the ***synchronizing*** message to be displayed as quickly as possible, so let's split that out into a separate trigger that uses the **`pageapproved`** event instead.

```
{trigger:pageapproved|approval=Review}
   {set-message}
      Synchronizing content...
   {set-message}
{trigger}
{trigger:statechanged|state=Synchronise|queue=true}
   {publish-page}
{trigger}
```

The **`pageapproved`** event is sent before the state changes, and thus before the **`statechanged`** event.

We're almost finished.

The last thing we need to do is handle the outcome of the **`{publish-page}`** action – to achieve that we can use the **`newevent`** parameter.

- the **`newevent`** parameter creates a new custom event after the actions in the trigger have been completed
- we'll call this event **`AfterSync`**

In addition to creating the event, the **`newevent`** does two other things

- it sends a **`success`** flag to any trigger or triggers listening to the event – this indicates whether the actions in the original trigger were successful or not
- if errors occur whilst processing the actions, the **`@errormessage@`** value reference will also be set in triggers listening to the new event

It's perfect for our needs, almost as if **`newevent`** was created precisely for this purpose.

```
{trigger:statechanged|state=Synchronise|queue=true|newevent=AfterSync}
   {publish-page}
{trigger}
{trigger:AfterSync|success=true}
   {set-message:style=success|duration=PT1M}
      Synchronisation complete, content is published!
   {set-message}
   {set-state:Published|comment=Synchronization completed successfully}
{trigger}
{trigger:AfterSync|success=false}
   {set-message:style=error}
      Synchronization failed.
      @errormessage@
   {set-message}
   {set-state:Error|comment=Synchronization failed}
{trigger}
```

Putting it all together

Here's the finished workflow.

```
{workflow:name=Different-space Publishing}
   {state:Editing|submit=Review}
   {state}
   {state:Review|approved=Synchronise|rejected=Editing}
      {approval:Review|assignable=true}
   {state}
   {state:Synchronise|hideselection=true|description=Content synchronisation in progress, please wait...}
   {state}
   {state:
Error|colour=#ff0000|approved=Synchronise|rejected=Published|hidefrompath=true|description=Synchronization
failed, what do you want to do?}
      {approval:Error|approvelabel=Retry|rejectlabel=Ignore}
   {state}
   {state:Published|final=true|updated=Editing}
   {state}
   {trigger:pageapproved|approval=Review}
      {set-message}
         Synchronising content...
      {set-message}
   {trigger}
   {trigger:statechanged|state=Synchronise|queue=true|newevent=AfterSync}
      {publish-page}
   {trigger}
   {trigger:AfterSync|success=true}
      {set-message:style=success|duration=PT1M}
         Synchronisation complete, content is published!
      {set-message}
      {set-state:Published|comment=Synchronization completed successfully}
   {trigger}
   {trigger:AfterSync|success=false}
      {set-message:style=error}
         Synchronization failed.
         @errormessage@
      {set-message}
      {set-state:Error|comment=Synchronization failed}
   {trigger}
{workflow}
```