HowToDefaultPublicationUsefulChanges

Useful changes to the default publication

Most Lenya users use the Default publication as a starting point to set up their publication. There are a number of things that one should either remember or make a conscious decision to ignore when starting a new publication that way. As many Lenya admins don't have to do this very often here is a little checklist of tweaks you may want or not want to inplement before you have your users start the editing:

Remove the Try user "lenya" and password "levi" message on the login screen

Why?

If you did not delete the standard users you probably don't want to tell all the world how to get in. What sense does a lock make if the key is attched next to it? if you changed the password the hint is wrong which is even worse.

How?

Two options:

- Remove it entirely: Edit `lenya/xslt/ac/login.xsl` (either the global one for all publications or your publication specific version at lenya/pubs /<your-publication>/lenya/xslt/ac/login.xsl) and delete this piece:

```
<xsl:template match="login">
  <p>
    <xsl:apply-templates select="authentication_failed" />
    <xsl:apply-templates select="protected_destination" />
    <xsl:apply-templates select="current_username" />
    <xsl:apply-templates select="no_username_yet" />
    <xsl:apply-templates select="authenticator" />
  </p>
  <-- Delete from here ...
  <p>
    <strong>
    <i18n:text>NOTE</i18n:text>: </strong>
    <i18n:translate>
      <i18n:text i18n:key="try-user-lenya" />
      <i18n:param>"lenya"</i18n:param>
      <i18n:param>"levi"</i18n:param>
    </i18n:translate>
  </p>
  ... until here -->
```

- Replace it with your own message either giving users a hint which account to try or telling them where and how to get an account. To achieve this introduce a new key such as "login-account-hint". Replace `try-user-lenya` with the new key. Add you new key in `lenya/resources/i18n /cmsui.xml` as well as in the dictionary files `lenya/resources/i18n/cmsui_XX.xml`. See I18NHowToContribute for details.

It makes sense to introduce a new key and not just change the text for "try-user-lenya" because it actually is a different message and will be much easier to find a diff if you ever want to find out later what you changed against the standard.

Add the Cocoon Clear Cache Actions to the Lenya Admin screen

This needs to be reworked a bit.. Does not work like described because the sitemap entry does not call the usecase. Also, according to Gregor Rothfuss in this message on the mailing list: "this is more of a historic page. back in cocoon 2.1.5 days, the cache sometimes had issues, but with 2.1.7, it is not really necessary to clean out the memory cache."

Why?

Experience shows that it just happens over time (especially when heavily editing) that elements get cached wrong and either never update again correctly or only after a long time. It is not very convenient for content editors if they need to have the Admin stop the webapp, sometimes even delete the persistent cache and start it again.

How?

Edit `lenya/content/admin/sitetree.xml` and add two new nodes:

```
  <node id="ipranges">
    <label><i18n:text>IP Ranges</i18n:text></label>
  </node>

  <!-- start adding here -->

  <node id="clearmemorycache">
    <label><i18n:text>Clear Memory Cache</i18n:text></label>
  </node>

  <node id="clearpersistentcache">
    <label><i18n:text>Clear Persistent Cache</i18n:text></label>
  </node>

  <!-- end adding here -->

 <node id="content">
    <label><i18n:text>Delete Trash</i18n:text></label>
  </node>
```

Edit lenya/admin.xmap and add two new usecases:

```
        <map:match pattern="deleteTrash" type="usecase">

          <map:match pattern="showscreen" type="step">
            <map:generate src="content/admin/content/deleteTrash.xsp" type="serverpages"/>
            <map:transform src="xslt/admin/content/deleteTrash.xsl"/>
            <map:transform src="cocoon://lenya-screen.xsl"/>
            <map:serialize/>
          </map:match>

          <map:match pattern="deleteTrash" type="step">
            <map:act type="task">
              <map:redirect-to session="true" uri="{request:requestURI}"/>
            </map:act>
          </map:match>

        </map:match>

        <!-- start adding here -->

        <map:match pattern="clearMemoryCache" type="usecase">

          <map:match pattern="clearMemoryCache" type="step">
            <map:act type="clear-cache">
              <map:redirect-to session="true" uri="{request:requestURI}"/>
            </map:act>
          </map:match>

        </map:match>

        <map:match pattern="clearPersistentCache" type="usecase">

          <map:match pattern="clearPersistentCache" type="step">
            <map:act type="clear-persistent-store">
              <map:redirect-to session="true" uri="{request:requestURI}"/>
            </map:act>
          </map:match>

        </map:match>

        <!-- end adding here -->
```

Unpack the file clearcache.tar.gz into your lenya webapp directory.

After that you should have two new entries in the Admin Page which are "Clear Memory Cache" and "Clear Persistent Cache". Feel free to add translations for your language to the appropriate language files. Otherwise they will show up in English no matter in what language you run the rest of the Lenya GUI.

Note: If you call the "Cocoon and Server Status" right after having cleared the Cache you will still find entries in the cache. To be more accurate: You will **ag ain** find entries in the cache because in order to serve the page items got cached again already. So you will probably never see zero items in the cache in the Status page.

Hand the Navigation Title through to the browser

It's just a detail, but by default pages delivered by Lenya don't show the navigation title in the browsers window title. There have been a number of different approaches discussed on the lenya-users mailing list. Two are presented here:

Method 1: Use the Navigation Title from the Metadata

Edit xhtml2xhtml.xsl to wrap the navigation title into a div tag:

```
<xsl:template match="/xhtml:html">

  <!-- BEGIN change -->
  <div id="title">
    <xsl:value-of select="lenya:meta/dc:title" />
  </div>
  <!-- END change -->

  <div id="body">
    <xsl:if test="$rendertype = 'edit'">
      <xsl:attribute name="bxe_xpath">/xhtml:html/xhtml:body</xsl:attribute>
    </xsl:if>
    <xsl:apply-templates select="xhtml:body/node()"/>
  </div>
</xsl:template>
```

Then edit page2xhtml.xsl to extract this div tag and put it into the <head> section of the final page that gets delivered to the browser:

```
<xsl:template match="cmsbody">
  <html>
    <head>
      <!-- BEGIN change -->
      <title><xsl:value-of select="xhtml:div[@id = 'title']"/></title>
      <!-- END change -->
      <link rel="stylesheet" href="{$root}/css/page.css" type="text/css"/>
    </head>
    <body>
      <div id="page">

    ...
```

If you want to use anything else except the navigation title from `lenya:meta/dc:title` you can change the line

```
    <xsl:value-of select="lenya:meta/dc:title" />
```

in xhtml2xhtml.xsl with whatever you would like to see in the title.

Note: If you use custom document types make sure you change the <yourdoctype>2xhtml.xsl stylesheets to generate the `<div id="title">` tag.

Method 2: Hand the Navigation Title through to the browser

This is a useful way of making the menu bar in the browser display the same text as found on the link in the menu, i.e. the navigation title NOT from the metadata. This means that your content editors need only maintain the document name in one place.

In publication_sitemap.xmap, add a new line

```
    <map:parameter name="document-label" value="{page-envelope:document-label}"/>
```

within the clause <map:transform src="xslt/page2xhtml-{4}.xsl">. This makes the navigation title (document-label) available to your xslt. Then in your xslt (such as page2xhtml-xhtml.xsl) make the top of the template something like this:

```
<xsl:param name="document-label"/>

<xsl:template match="cmsbody">

<html>
  <head>
    <title><xsl:value-of select="$document-label"/></title>
    <link href="css/styles.css" rel="stylesheet" media="screen, print" type="text/css"/>
  </head>
```

It's worth noting that if you play with the navigation title, make a mistake, and end up returning a blank title, lenya tends to create HTML for the title in the form <head><title /></head>. Internet Explorer can't cope with this and will return a blank page. If you are getting blank pages in IE, check for this problem.

Allow users to override checkouts by other users

Why?

Lenya users often begin to edit a document (implicitly checking it out) but fail to exit the editor normally – by using the browser's Back button, or by exiting the browser, or by letting the session time out. This leaves the document in a "checked out" state, and only that same user can subsequently re-enter the document in the editor. To others, the page will be locked and uneditable.

Of course, overriding someone else's checkout partially subverts the purpose of the checkout, and the original editor may loose any unsaved changes. But in most environments the ability to occasionally override another's checkout is important to normal web site operations.

How?

Step 1: Add an "override-checkout" usecase in `lenya/usecases.xmap`:

```
        <!-- Rename a Label -->
        <map:match pattern="rename-label" type="usecase">

        ...

        </map:match>

        <!-- start inserting here -->

        <!-- Override a checkout -->
        <map:match pattern="override-checkout" type="usecase">

          <map:match pattern="showscreen" type="step">
            <map:generate src="content/info/override-checkout.xsp" type="serverpages"/>
            <map:transform src="xslt/info/override-checkout.xsl">
              <map:parameter name="use-request-parameters" value="true"/>
            </map:transform>
            <map:call resource="style-cms-page"/>
          </map:match>

          <map:match pattern="override-checkout" type="step">
            <map:act type="forced-checkin">
              <map:generate src="content/rc/{exception}.xsp" type="serverpages">
                <map:parameter name="user" value="{user}"/>
                <map:parameter name="filename" value="{filename}"/>
                <map:parameter name="checkType" value="{checkType}"/>
                <map:parameter name="date" value="{date}"/>
                <map:parameter name="message" value="{message}"/>
              </map:generate>
              <map:transform src="xslt/rc/rco-exception.xsl"/>
              <map:serialize/>
            </map:act>
            <map:redirect-to session="true" uri="{request:requestURI}"/>
          </map:match>

        </map:match>
          <!-- end inserting here -->
```

To be added: Configure the new usecase in the authorization framework so it will not be grayed out!

Step 2: Add a `forced-checkin` action to `sitemap.xmap`:

```
   <map:action name="reserved-checkin" src="org.apache.lenya.cms.cocoon.acting.ReservedCheckinAction" logger="
sitemap.action.reserved-checkin"/>
   <!-- start inserting here -->
   <map:action name="forced-checkin" src="org.apache.lenya.cms.cocoon.acting.ForcedCheckinAction" logger="
sitemap.action.forced-checkin"/>
   <!-- end inserting here -->
   <map:action name="rollback" src="org.apache.lenya.cms.cocoon.acting.RollbackAction"/>
```

Step 3: If you're working with the binary copy of Lenya, copy the ForcedCheckinAction.class file into your build/lenya/classes/org/apache/lenya/cms/cocoon /acting/ folder. Or, if you're working with the Lenya source code, instead copy ForcedCheckinAction.java into your existing src/java/org/apache/lenya/cms /cocoon/acting/ folder and rebuild Lenya.

- override-checkout2.tar.gz
- ForcedCheckinAction.java
- ForcedCheckinAction.class

Step 4: Decide what "unlock" interface suits your needs. You can either 1) add an "Unlock" button on the error page that tells you that the page is locked by another editor, or 2) add a menu entry in the Site tab. Decide which interface to use base on how difficult you want to make it for editors to unlock pages. The first approach is the easiest and most natural for the users, but some believe that an unlock action should be difficult to use so as to discourage its use.

Unlock Interface 1: Button on Error Page

This will add an "Unlock Document" button on the error page users get when they try to edit a document that is locked by someone else. This approach includes a warning message that appears on the error page as well as a javascript confirmation pop-up that repeats the warning.

First, in webapp/lenya/resources/i18n/cmsui.xml, add these lines:

```
<message key="lenya.rc.checkedoutalready.overridewarning">
    WARNING - To avoid loss of unsaved changes, only unlock a document checked out
    by another user if you require immediate access and that user is unavailable.
</message>
```

Add a similar line in any other desired languages to cmsui_LL.xml, where LL is the language code.

Next, modify lenya/lenya/xslt/rc/rco-exception.xsl as follows:

1. After this line:

<i18n:text>lenya.rc.checkedoutalready</i18n:text>

add this line:

<i18n:text>lenya.rc.checkedoutalready.overridewarning</i18n:text>

b. Replace the first occurance of these lines (that is, within the "rc:file-reserved-checkout-exception" section):

```
<form>
<input type="button" value="OK" onClick="history.go(-1)"/>
</form>
```

with these:

```
<xsl:param name="lenya.event" select="'delete'"/>
<xsl:output version="1.0" indent="yes" encoding="UTF-8"/>
<xsl:variable name="document-id"><xsl:value-of select="/page/info/document-id"/></xsl:variable>
<xsl:variable name="area"><xsl:value-of select="/page/info/area"/></xsl:variable>
<xsl:variable name="task-id"><xsl:value-of select="/page/info/task-id"/></xsl:variable>
<xsl:variable name="request-uri"><xsl:value-of select="/page/info/request-uri"/></xsl:variable>

<xsl:variable name="nbsp" select="' '"/>
<xsl:value-of select="$document-id"/>
<xsl:value-of select="$area"/>
<xsl:value-of select="$task-id"/>
<xsl:value-of select="$request-uri"/>
<script language="JavaScript"  type="text/javascript">
function confirmation() {
    var strMsg = "<i18n:text>lenya.rc.checkedoutalready.overridewarning</i18n:text>";
    if (confirm(strMsg)) { return true;}
    return false;
}
</script>
<form>
    <input type="hidden" name="lenya.usecase" value="override-checkout" />
    <input type="hidden" name="lenya.step" value="override-checkout" />
    <input type="hidden" name="task-id" value="{$task-id}" />
    <input type="hidden" name="properties.node.firstdocumentid" value="{$document-id}" />
    <input type="hidden" name="properties.firstarea" value="{$area}" />
    <input type="hidden" name="properties.secarea" value="trash" />
    <input type="hidden" name="parenturl" value="{parent-url}" />
    <input type="button" value="Cancel" onClick="history.go(-1)" />
    <input type="submit" value="Unlock Document" onClick="return confirmation();" />
</form>
```

Unlock Interface 2: Menu Item on Site Tab

This adds a menu item on the drop-down menu available within the "Site" tab. By design, this makes it somewhat difficult for users to unlock documents, because they have to go over to the Site tab and find the menu item.

Note that such a menu item is already included in the lenya-1.2.x branch. You will find it under the Edit-Menu called CheckIn. This usecase informs the user that the document is already checked out by the user xy and on what date. The user then has the option of checking in the document or cancelling.

First, insert a new block into the file `lenya/pubs/<your-pub>/config/menus/generic.xsp`:

```
      <block authoring="false">
        <xsp:logic>
        {
          if (isDocument) {
            <item uc:usecase="override-checkout" uc:step="showscreen" href="?"><i18n:text>Unlock Document</i18n:
text></item>
          }
          else {
            <item><i18n:text>Unlock Document</i18n:text></item>
          }
        }
        </xsp:logic>
      </block>

      <!-- end inserting here -->

      </menu>

      <menu i18n:attr="name" name="Workflow" label="Help">
          <xsp:logic>
```

This solution is based on the fact that the RevisionControler bypasses all checks if the user is "System". It might be better to introduce a parameter in the ReservedCheckinAction class that would enable setting the identity or to set an override flag.

Only allow admins to change AC Auth values for pages

By default any user can change permissions to pages. This allows them to upgrade their capabilities on any page to either an editor or reviewer. This solution is from the user list:

- Assign admin group to the role of admin for your site tree
- In lenya/pubs/<your-pub>/config/ac/usecase-policies.xml add the following inside of the usecases tags: <usecase id="info-ac-authoring"><role id="admin"/></usecase>

Now, whenever a none admin tries to use the AC Auth page they will be prevented from doing so.

Allow users with both "edit" and "review" roles to publish a page in one step

Often, users do not need the 2-step workflow between editor and reviewer. You can assign both roles to a user, and then allow her/him to publish in one step, going from state "authoring" directly to state "live". To do this, add the following snippet to <yourpub>/config/workflow/workflow.xml:

```
<transition source="authoring" destination="live">
  <event id="publish"/>
  <condition class="org.apache.lenya.cms.workflow.RoleCondition">edit</condition>
  <condition class="org.apache.lenya.cms.workflow.RoleCondition">review</condition>
  <assign variable="is_live" value="true"/>
</transition>
```