# Move a page from core to legacy UI module

As of OpenMRS Platform 2.0, the legacy UI has been moved to the legacy UI module, the module's git repository is at https://github.com/openmrs/openmrs-module-legacyui

JSP files are located in the **webapp/src/main/webapp** folder, it is key to note that some pages are not backed by a controller and that some controllers are configured via xml while others are annotated,  the changes to be made when the controllers are moved differ a little bit because of this. Below are the details of how to move a page from the platform to the module while keeping these factors in mind.



Note that some JSPs and controllers may have already been moved.

- Move the JSP
  - Copy the JSP you wish to move to the **omod/webapp** directory in the module while preserving the folder structure, note that for the JSPs located in the **WEB-INF/view** folder, the folder structure you need to preserve is the one after **WEB-INF/view** e.g a JSP in the platform at **webapp/src/main/webapp/WEB-INF/view/dictionary/conceptForm.jsp** ends up at **omod/webapp/dictionary/conceptForm.jsp** in the module.
  - Delete the JSP from the platform
- Move the controller

  Depending on whether the page has an xml based controller or an annotated one or has no controller at all, follow the appropriate option from the ones below.

  If there is a controller, remember to delete it from the platform after copying it to the module.

  ### A Controller Configured via xml

  It is possible that the entries to be moved have already been moved to the module and possibly commented out, if this is the case,  just un-comment the appropriate entries in webModuleApplicationContext.xml file in the module.

  a. In the platform there is an openmrs-servlet.xml file that contains the mappings between each request URL and the controller that processes it, you need to move the URL mapping entry along with its associated controller bean to the module, the bean with id **urlMapping** has a property named **mappings** which also has a list of props elements, copy the respective prop element for the page you are moving to the webModuleApplicationContext.xml file in the module. For example, if you are moving the concept form, you would need to move the entry shown below,

  ```
  <prop key="dictionary/concept.form">conceptForm</prop>
  ```

  b. As seen from above, the prop entry references the conceptFrom bean shown below, it means we also need to move it to the webModuleApplicationContext.xml file in the module. Note that the formView property has been prefixed with **module/legacyui** because conceptForm.jsp has been moved to the legacy UI module. We don't change the successView property because pretty much all their values are not paths but rather request URLs that are already mapped to some other controllers.

  ```
  <bean id="conceptForm" class="org.openmrs.web.controller.ConceptFormController">
      <property name="commandName"><value>command</value></property>
      <property name="validator">
          <ref bean="conceptFormValidator" />
      </property>
      <property name="formView"><value>/module/legacyui/dictionary/conceptForm</value></property>
      <property name="successView"><value>concept.form</value></property>
  </bean>
  ```

  c. Copy the controller to the **omod/src/main/java** source folder while preserving the original package, in the example above the controller is **org.openmrs.web.controller.ConceptFormController**.
  d. For bean entries like the one above, it would nice to move everything referenced at the same time to have the page working as expected, e.g in the example above you can see that it references the conceptFormValidator bean and also defines the formView and successView which are paths and request URLs too respectively, most likely they will have URL mappings themselves as described in step 'a' and they need to be moved too along with their controllers if any as described. If the page has no controller, you will need to see the section below for how to move a page with no controller.
  e. Be sure to test that the page is still fully functional after moving it.

  ### An annotated controller

  This kind of page will have no URL mapping in the openmrs-servlet.xml file.

a. Copy the annotated controller to the **omod/src/main/java** source folder while preserving the original package name, as you may already know, the methods of an annotated controller can have return types, if the return type is a String, look for all the returned values, if a returned value is path to some JSP i.e it is not a redirect or forward URL, then prefix **module/legacyui** to it. For example if a returned path is like the one below:

```
admin/users/userForm
```

It would become:

```
module/legacyui/admin/users/userForm
```

b. Be sure to test that the page is still fully functional after moving it.

### If a page has no controller

a. Add an annotated controller to the module, and then the controller method is mapped to the original URL and its return value becomes the path to the relevant JSP that was moved to the module. A good example of a controller in the module that does this is the AdminPage Controller which looks like below:

```
@Controller
public class AdminPageController {

   @RequestMapping(value = "admin/index")
   public String displayAdminIndex() {
      return "module/legacyui/admin/index";
   }
```

Note that the main admin JSP page in the platform had no controller, the request URL for it was **/admin/index.htm** and the JSP to it was located in **WEB-INF/view/admin/index.jsp**, so we override it by mapping our new controller method to **admin/index** and then we return the same path but with **module/legacyui** prefixed to it since the page has been moved to the legacy UI module.

b. Be sure to test that the page is still fully functional after moving it.

If you think about it, in the module project structure we don't really have the **module/legacyui** folder but we keep prefixing the formView property value and the return paths from the annotated controllers with **module/legacyui** and things work magically, this is because modules resources like JSPs get copied by the module engine to the module/*myModuleId* folder at run time when the module is installed where *myModuleId* is the module's unique id which in this case is legacyui.

## Move, Update and Run Unit tests

Most of the controllers for the pages have unit tests, move the test classes as well if any and update them accordingly, run the tests to ensure that they still pass before committing your changes.

Remember to update test classes you move to extend BaseModuleWebContextSensitiveTest for those that require a web context or BaseModuleContextSensitiveTest for the ones that don't.