Overriding Requests to Pages and Fragments

This tutorial illustrates how to use RequestMappers to override pages and fragments provided by other modules within your custom module.

This functionality requires the following modules:
1. App Framework 2.8
2. EMR API 1.16
3. UI Framework 3.6

This demo requires the following additional modules which has the overridden pages:

1. Reference Application 2.3

2. App UI 1.5.1

The overrides illustrated in this tutorial can be found in this github repo https://github.com/ssmusoke/openmrs-module-mapperoverridedemo.git

Overriding a Fragment

1. Create the fragment and associated controller following the UI framework guidelines at https://wiki.openmrs.org/display/docs/Using+the+UI+Framework+in+Your+Module#UsingtheUIFrameworkinYourModule-FoldersandPackages usually best to copy the existing code from the fragment being overridden
2. Create a fragment request mapper class following the same convention as the fragment controller. The fragment for this tutorial is called demoHeader.gsp hence the class name is DemoHeaderFragmentRequestMapper with sample code below noting the following:

    - The @Component annotation is required so that the class can be loaded
    - The class implements the FragmentRequestMapper interface that provides the mapRequest(FragmentRequest) method
    - The example shows an override of both the providerName, the module providing the new fragment, and the fragmentId which is the name of the fragment. Changing the name of the fragmentId is optional

Sample Fragment Request Mapper

```
@Component
public class DemoHeaderFragmentRequestMapper implements FragmentRequestMapper {

   protected final Log log = LogFactory.getLog(getClass());

   /**
    * Implementations should call {@link FragmentRequest#setProviderNameOverride(String)} and
    * {@link FragmentRequest#setFragmentIdOverride(String)}, and return true if they want to remap a request,
    * or return false if they didn't remap it.
    *
    * @param request may have its providerNameOverride and pageNameOverride set
    * @return true if this page was mapped (by overriding the provider and/or page), false otherwise
    */
   public boolean mapRequest(FragmentRequest request) {
      log.info(request.toString());
      if (request.getProviderName().equals("appui")) {
         if (request.getFragmentId().equals("header")) {
            // change to the custom fragment provided by the module
            request.setProviderNameOverride("mapperoverridedemo");
            request.setFragmentIdOverride("demoHeader"); // no need to specify this if the name of the fragment
is the same as the one being over
            log.info(request.toString());
                              return true;
         }
      }
          // no override happened
      return false;
   }

}
```

Overriding a Page

1. Create the page and associated controller following the UI framework guidelines at https://wiki.openmrs.org/display/docs/Using+the+UI+Framework+in+Your+Module#UsingtheUIFrameworkinYourModule-FoldersandPackages usually best to copy the existing code from the page being overridden
2. Create a page request mapper class following the same convention as the page controller. The fragment for this tutorial is called demoLogin.gsp hence the class name is DemoLoginPageRequestMapper with sample code below noting the following:

    - The @Component annotation is required so that the class can be loaded
    - The class implements the PageRequestMapper interface that provides the mapRequest(PageRequest) method
    - The example shows an override of both the providerName, the module providing the new fragment, and the pageName which is the name of the page. Changing the name of the pageName is optional

Sample Page Request Mapper

```
@Component
public class DemoLoginPageRequestMapper implements PageRequestMapper {

   protected final Log log = LogFactory.getLog(getClass());

   /**
    * Implementations should call {@link PageRequest#setProviderNameOverride(String)} and
    * {@link PageRequest#setPageNameOverride(String)}, and return true if they want to remap a request,
    * or return false if they didn't remap it.
    *
    * @param request may have its providerNameOverride and pageNameOverride set
    * @return true if this page was mapped (by overriding the provider and/or page), false otherwise
    */
   public boolean mapRequest(PageRequest request) {
      if (request.getProviderName().equals("referenceapplication")) {
         if (request.getPageName().equals("login")) {
            // change to the custom login provided by the module
            request.setProviderNameOverride("mapperoverridedemo");
            request.setPageNameOverride("demoLogin");

            log.info(request.toString());
                              return true;
         }
      }
          // no override happened
      return false;
   }

}
```

Troubleshooting Tips

1. Request Mappers working in a random order, with some requests not being overridden - check to ensure that the classes only return true when an override has occurred not all the time