This page last changed on Jan 10, 2013 by tyler.johnson.

Mobile JSF: Using ICEmobile to rapidly build mobile applications

The goal of this 4 part tutorial is to build an ICEmobile application that will allow users to take a photo using their device's camera, upload that photo to the application, and notify users of the new image upload using ICEmobile cloud push. The application will also provide screen size detection and allow navigation between views.

Part 1: Device Detection and Layout
Part 2: Camera Application
Part 3: Push Notification
Part 4: Page Navigation

Requirements

1. Completed parts 1, 2 and 3
2. ICEmobile Container

Part 4: Page Navigation

Page navigation is an integral part to any JSF application and because ICEmobile/ICEfaces is JSF, all conventional JSF navigation techniques work as expected. However, in the mobile space it is generally desirable to minimize network traffic whenever possible. One technique to minimizing data use is to use the Facelets <ui:include/ > tag handler to dynamically change page content on an action event. Dynamic includes combined with ICEfaces direct-to-DOM technology can significantly reduce the amount of data being send over the air and provide quick page refreshed when bandwidth is limited or comes at a cost.

Building an application with dynamic includes is not always the easiest solution to implement, especially for developers new to JSF. To this end, ICEmobile/ICEfaces has introduced the <mobi:contentStack/> component that takes care of the plumbing behind using dynamic includes. The <mobi:contentStack/> component can have N child <mobi:contentPane /> each of which must have a unique ID attribute. The <mobi:contentStack/> component has a "currentId" attribute which can be use to specify which child <mobi:contentPane/> is to be made visible. The <mobi:contentStack/> component is quite flexible and can be used for general site navigation or more granularly for wizards or workflows.

The <mobi:contentStack/> is optimized to take advantage of the client side render when needed, as well as Facelets tag handlers to minimized the size the JSF component tree.

Page Navigation

The camera application is currently only one page, but in this example we will add a picture viewing page to the application. When a user touches on any of the shared photos, the application will simulate navigation and display the selected photo in a larger format. Touching the large image format will navigate the user back to the camera capture page. To keep things simple we will continue to extend the CameraBean code and use the same picture viewing view for both the large and small view.

The first step to adding the new picture view page is to update the CameraBean model with a new instance variable which will keep track of the current id of the selected contentPane. Add the following instance variable to the CameraBean and the corresponding setters and getters.

```
...
    // currently selected contentPane
    private String currentContentPane = "home";
```

For navigation to take place, we need to add a new action method to the CameraBean. The new method will take as a parameter the ID of the <mobi:contentPane/> that will be displayed. Add the following method to the CameraBean.

```
    public String navigateTo(String contentPaneId){
        // update the currently selected
        currentContentPane = contentPaneId;
        // return to the same view id.
        return null;
    }
```

The CameraBean updates for navigation are complete but we also need to keep track of which picture resource was clicked on so we know which one to assign to the picture view. Add the following instance variable to CameraBean and the corresponding getters and setters.

```
    // selected photo resource.
    private Resource currentCameraResource;
```

All the needed CameraBean updates have been completed. Next we need to update mobile.xhtml with the addition of a <mobi:contentStack/> and two child <mobi:contentPane/> components. All current content in mobile.xhml that is a child of <f:facet name="body"> will be moved into the first <mobi:contentPane/>.

```
    <f:facet name="body">
        <mobi:contentStack currentId="#{cameraBean.currentContentPane}">
            <!-- home page panel -->
            <mobi:contentPane id="home">
                <mobi:viewSelector>
                    <f:facet name="small">
                        <ui:include
                                src="WEB-INF/includes/contents/camera-capture.xhtml"/>
                        <ui:include
                                src="WEB-INF/includes/contents/photo-display.xhtml"/>
                    </f:facet>
                    <f:facet name="large">
                        <div style="float:left;width:30%">
                            <ui:include
                                    src="WEB-INF/includes/contents/photo-list.xhtml"/>
                        </div>
                        <div style="float:left;width:70%">
                            <ui:include
                                    src="WEB-INF/includes/contents/camera-capture.xhtml"/>
                        </div>
                    </f:facet>
                </mobi:viewSelector>
            </mobi:contentPane>
            <mobi:contentPane id="image_viewer">

            </mobi:contentPane>
        </mobi:contentStack>
    </f:facet>
```

Note: The <mobi:contentStack /> had a currentId attribute that has been value bound to the #{cameraBean.currentContentPane}, which will be updated whenever the method navigateTo is called on the CameraBean. The <mobi:contentPane id="image_viewer"/> is currently void of any children, but the intent of this contentPane is to show a larger view of the image that was selected. Add the following XHTML to the "image_viewer" contentPane:

```
    <mobi:contentPane id="image_viewer">
       <h:form>
          <h:commandLink
                action="#{cameraBean.navigateTo('home')}">
             <mobi:graphicImage value="#{cameraBean.currentCameraResource}"
                   style="width:99%"/>
          </h:commandLink>
       </h:form>
    </mobi:contentPane>
```

The last step to complete is to add a <h:commandLink /> component to the photo-display.xhtml and photo-list.xhtml views. In both instances a form is added to the view as well as the <f:setPropertyActionListern/> which assigned the currentCameraResource in the CameraBean.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
            xmlns:h="http://java.sun.com/jsf/html"
            xmlns:f="http://java.sun.com/jsf/core"
            xmlns:ui="http://java.sun.com/jsf/facelets"
            xmlns:mobi="http://www.icesoft.com/icefaces/mobile/component">
    <mobi:fieldsetGroup>
     <mobi:fieldsetRow group="true">
        Shared Photo
     </mobi:fieldsetRow>
     <mobi:fieldsetRow>
        <h:form>
          <h:commandLink action="#{cameraBean.navigateTo('image_viewer')}">
              <f:setPropertyActionListener
                    target="#{cameraBean.currentCameraResource}"
                    value="#{cameraBean.uploadedPhoto}"/>
              <mobi:graphicImage
                    value="#{cameraBean.uploadedPhoto}"
                    style="width:150px;"/>
          </h:commandLink>
        </h:form>
     </mobi:fieldsetRow>
   </mobi:fieldsetGroup>
</ui:composition>
```

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
            xmlns:h="http://java.sun.com/jsf/html"
            xmlns:f="http://java.sun.com/jsf/core"
            xmlns:ui="http://java.sun.com/jsf/facelets"
            xmlns:mobi="http://www.icesoft.com/icefaces/mobile/component">
```

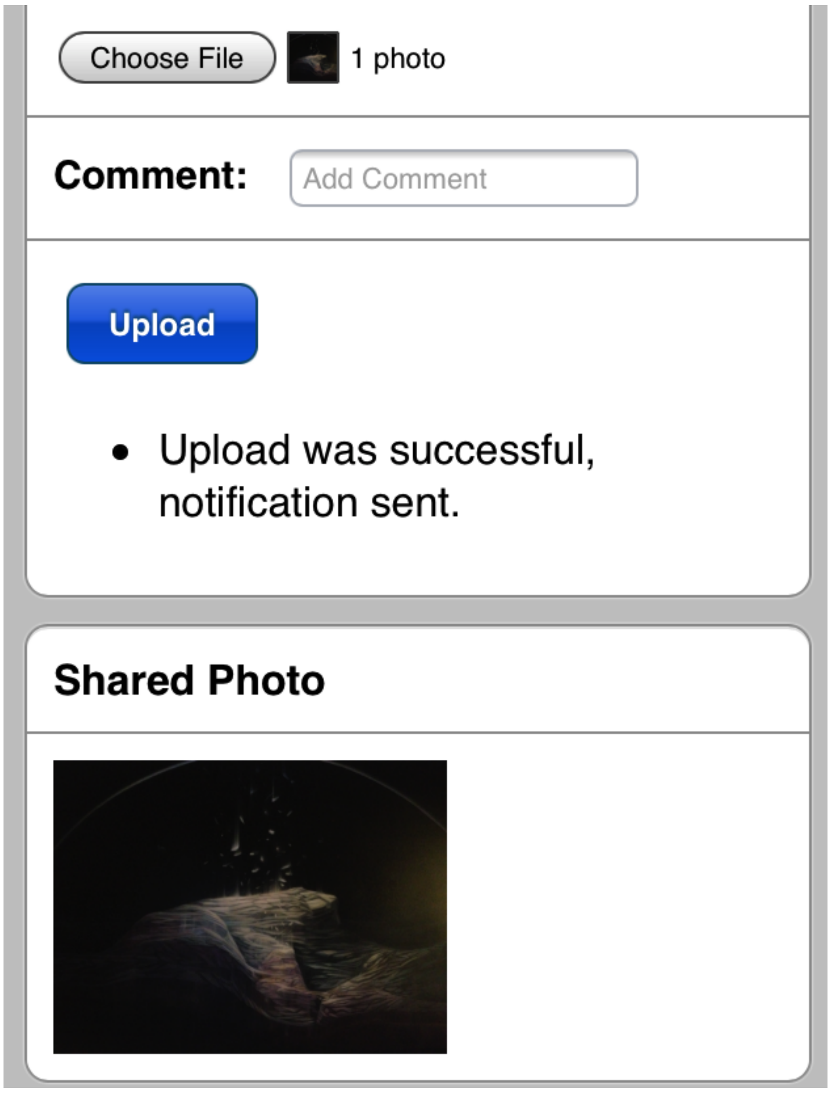
```
    <h:form>
        <mobi:outputList>
            <mobi:outputListItem group="true">
                Recent Photos
            </mobi:outputListItem>
            <mobi:outputListItems value="#{cameraBean.uploadedPhotos}"
                                  var="photos">
                <h:commandLink
                        action="#{cameraBean.navigateTo('image_viewer')}">
                    <f:setPropertyActionListener
                            target="#{cameraBean.currentCameraResource}"
                            value="#{photos}"/>
                    <mobi:graphicImage
                            value="#{photos}"
                            style="width:225px;"/>
                </h:commandLink>
            </mobi:outputListItems>
        </mobi:outputList>
    </h:form>
</ui:composition>
```

Running the Example

Once the camera demo is compiled and redeployed try loading the application once again in a tablet and handheld device. It should be now possible to click on images to view them in a larger context. Clicking on the larger image will initiate navigation back to the "home" panel. Navigation in this fairly simple but it is relatively easy to add a navigation history by simply keeping a stack of the visited contentPanel ID's.

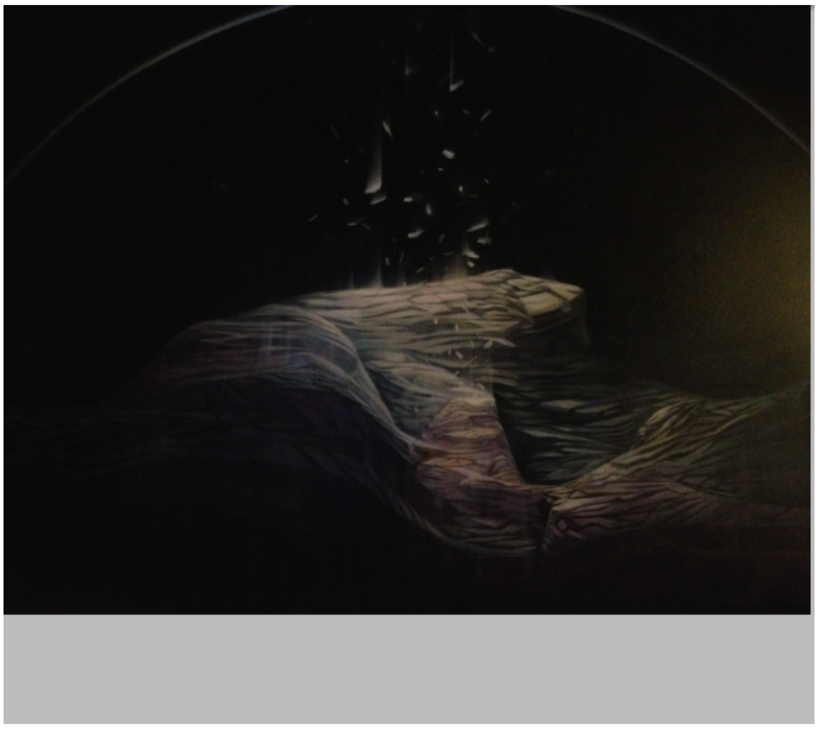
Small View

Using a small view device, upload a photo:

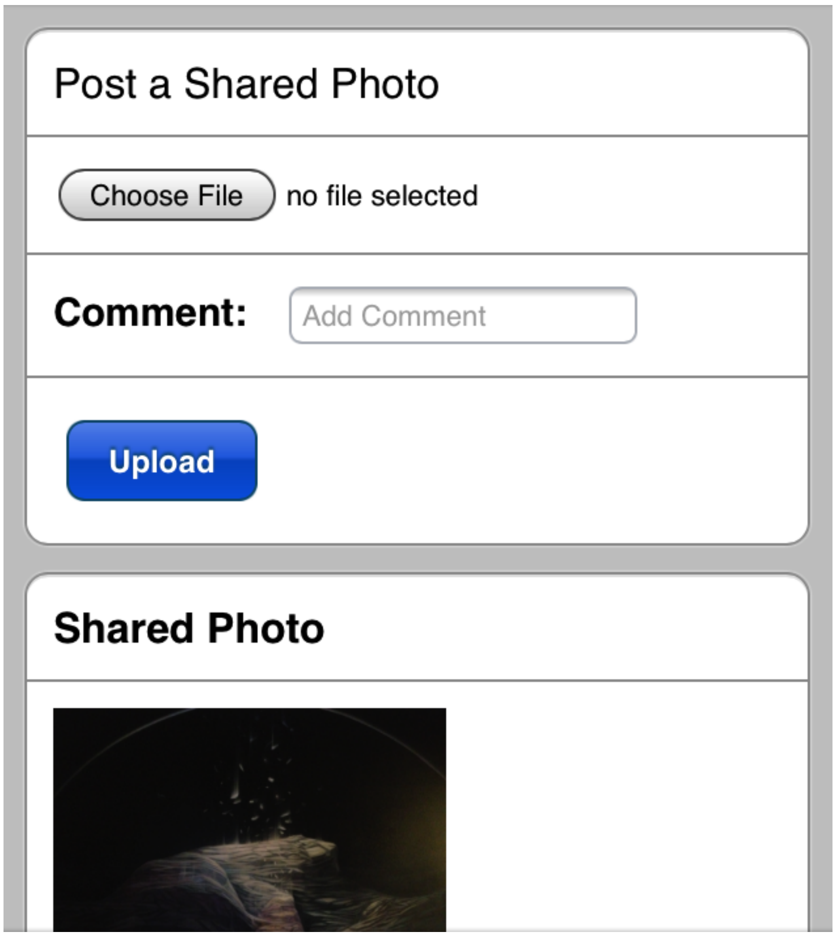


- Upload was successful, notification sent.

After the photo has been uploaded, click on it to navigate to the view image screen:



Now click on the photo to return to the main view:



Source

part4-navigation.zip