Create Account Page- edit user information

Use cases:

1. **Create My Account button on home screen.**

Implementation Flow:

1. Create navigation component for my account, navigation items (like update-profile, change password, add address) and its respective references through impex as mentioned in next step. (Use electronics store as a reference – Go to electronics store homepage temple (Landing page template 2) where we can see header links content slot which contains MyAccountComponent in it.)
2. Add the impex’s for MyAccountComponent and its references in cms-responsive-content.impex . (Can use the same impex’s which are available in electronics store cms-responsive-content.impex )
3. Add labels for My account and navigation items (like update-profile, change password, add address) in cms-responsive-content\_en.impex file. (These labels are displayed on storefront.)
4. Create CMSLinkComponents for navigation items (like update-profile, change password, add address) in cms-content.impex by adding the impex scripts. . (Use electronics store as a reference – Go to electronics store cms-content.impex and search for CMSLinkComponents .)

Now you should be able to see the My account button and its navigation list (like update-profile, change password, add address) on homepage.



1. **Create update-profile user information page.**

Implementation Flow:

1. Create ContentSlot for Account Update Profile Page.
2. Create ContentSlotForPage for Account Update Profile Page.
3. Create Jsp Include Component for components (AccountLeftNavigationComponent, UpdateProfileComponent) mentioned in ContentSlot for Account Update Profile Page.

Now you should be able to see the Update profile page with a form to update profile and side links to navigate to other pages in My account.



1. **Creating custom variables in update profile form.**

Implementation Flow:

1. Extend the item Type that is used in update profile form and add required attributes in items.xml file.
2. Do an ant all to update the Model classes.
3. Edit the form (accountProfileEditPage.jsp) which is displaying the fields in update profile page by adding your new attributes in that form.
4. Extend the existing bean class (UpdateProfileForm) of the form to add your new attributes.
5. Extend the excisting ItemData bean to accumulate you new attributes in <extensionname>-beans.xml.
6. Go to the controller class (LoginPageController) by which this update profile page is displayed.
7. Edit the update profile get method in that LoginPageController by getting the new attributes from façade layer and setting them to the ItemData bean.
8. Only getting data from façade is not enough even we should also go through it’s spring file to know which converter it used and get the list of populator it uses.
9. Extend the existing populator (CustomerPopulator) and add new attributes in it.
10. Edit the update profile post method in that LoginPageController by getting the new attributes from storefront form and setting them to the ItemData bean. Then send them through façade, service and DAO layer to persist in database.
11. Extend the existing façade (CustomerFacade) to accept the new attributes.
12. Register the façade in <extensionname>-spring.xml
13. Extend the existing service (CustomerService) to accept the new attributes.
14. Register the façade in <extensionname>-spring.xml

Now you would be able to able get the details from user through update profile page and persist them in database.



We should also change the existing form name to new extended form name in AccoyuntPageControllerTest java class. Otherwise the test case will fail and an exception will be given while server start up.