Customization feature in WebStudio (WS)

# Requirements:

* Aim is to provide a facility to customize various aspects of WebStudio UI.
* One should be able change the CSS styles (colours, borders etc.), images, background images, message strings, various titles and tooltips. In short build/configure a new theme for WS that is more in line with his/her corporate theme.
* Customization should be an option (and an easy one at that) and not a compulsion.
* One should always be able to revert back his/her changes and WS should again seamlessly work with defaults.

# Preface:

This document details out support for customization in WebStudio. The functionality is divided into four categories –

1. Company Logo
2. CSS
3. Images
4. Messages/Properties

All customization involve making changes in the war file to provide with a customized view of WS.

# Company Logo

This feature allows the user to show his/her company logo on WebStudio login page and WebStudio banner page.

## Steps:

1. Explode the WebStudio war file. Open “WebStudio.html” in edit mode. Search for “company\_logo\_url”, set its value to the URL of your logo image.
2. The URL can be a hyperlink to the logo image which might be hosted on some web-server or it can point to an image on the file system within WebStudio folder hierarchy itself.

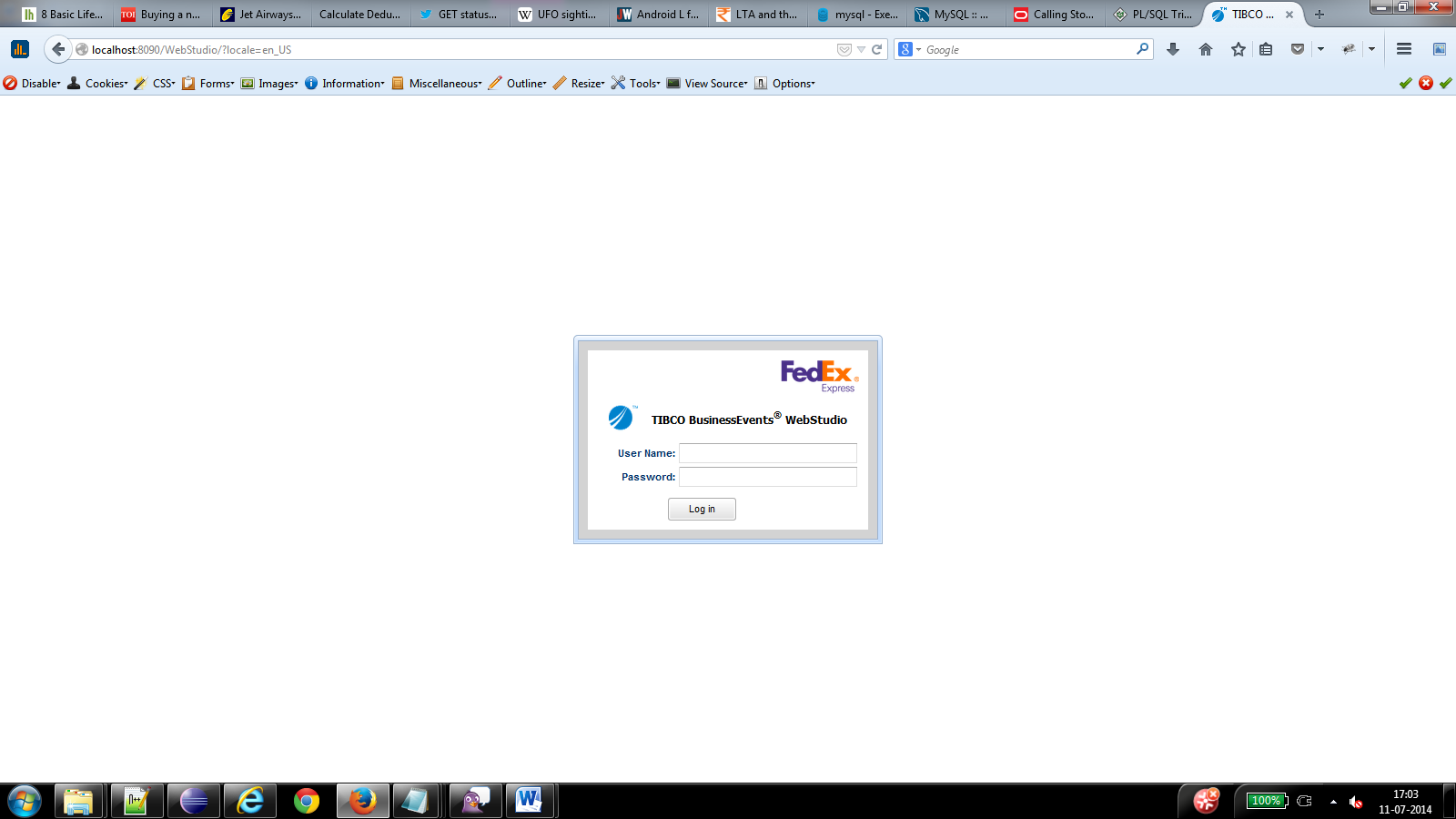
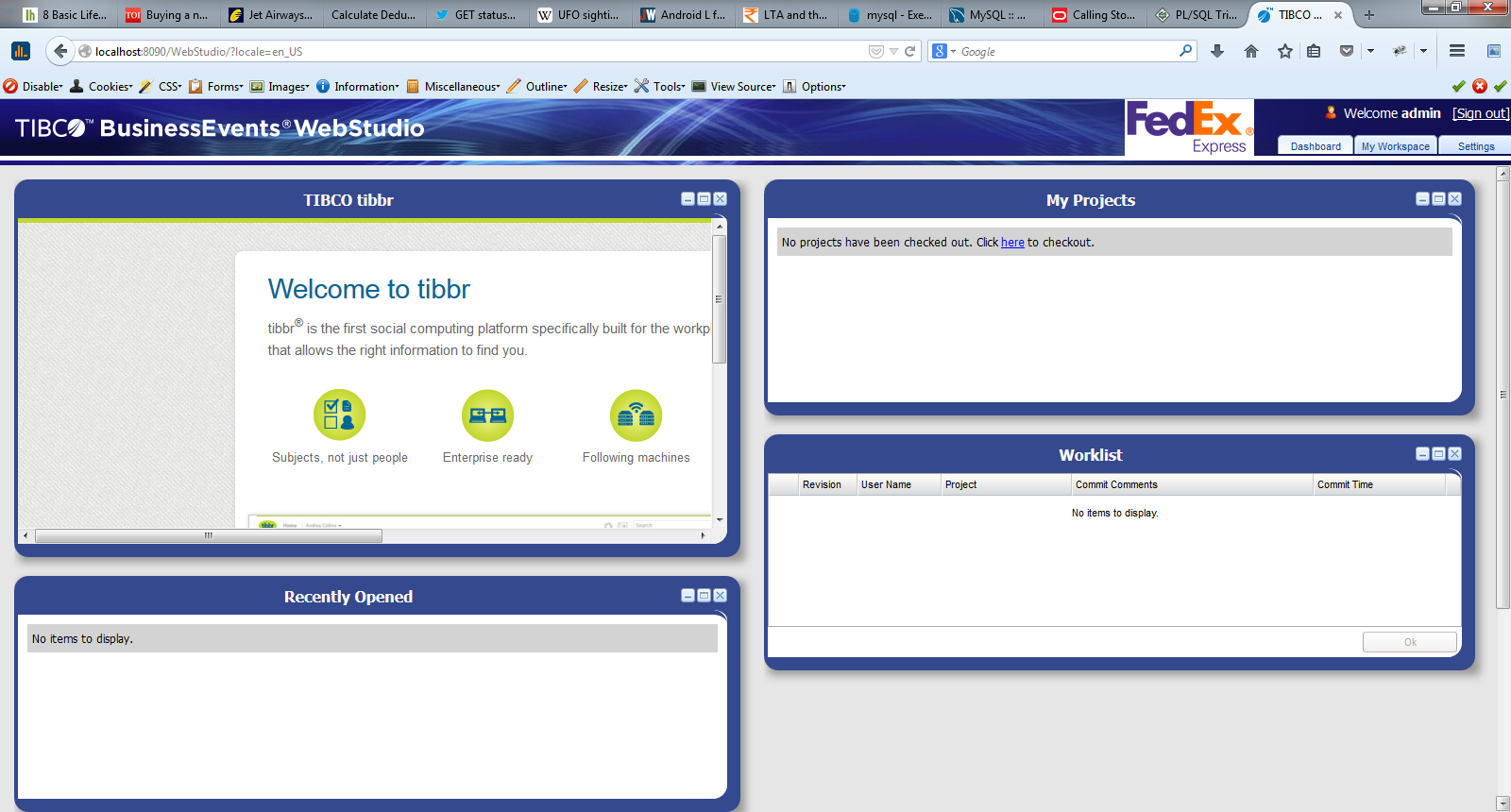
E.g. “http://www.mycompany.com/images/logo.jpg”

Or

“webstudio/images/logo.png” (you’ll need to copy the image to the war at the path you mention)

1. Zip the WebStudio war file once you are done.

**Note:** To remove the company logo simply set value of “company\_logo\_url” to blank string in WebStudio.html.

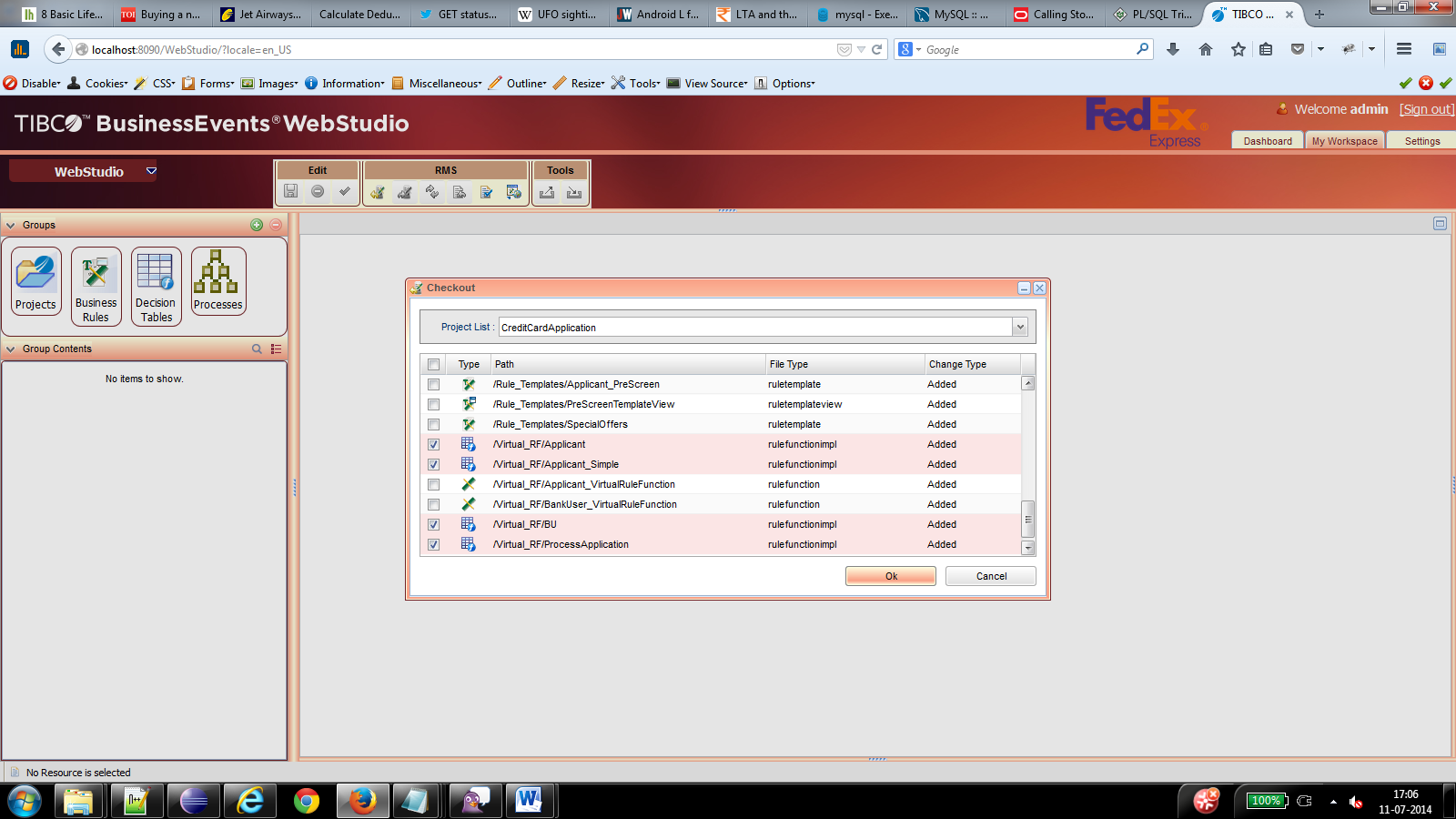
# CSS

This feature allows the user to change the CSS of the WebStudio. This is helpful when one wants to setup/configure a new theme for WS which is more in line with his/her corporate theme.

## Steps:

1. Explode the WebStudio war file.
2. Open “styles/custom\_styles.css” file in edit mode; populate it with CSS styles that you want to change (override). Refer instructions within the file.
3. Open “styles/WebStudio.css” file in edit mode, uncomment (remove #) the first line to import “custom\_styles.css” and save it.
4. Zip the WebStudio war file once you are done.

**Note:** To revert the theme changes simply, edit the “styles/WebStudio.css” and comment the line to import “custom\_styles.css” and save it.



# Images

This feature allows the user to replace the default images with their own customized images. Typical example being a delete/add image/icon being represented very specifically through all our corporate applications and you want WebStudio too should follow the same.

**Steps:**

1. Explode the WebStudio war.
2. Duplicate the “images” folder (find under war/webstudio) at the same level with a name say “my\_images” or any of your choice. Now modify the desired images under “my\_images” hierarchy without changing its file-name.
3. Open “WebStudio.html” and set the value of variable “images\_dir” to point to “my\_images” folder.  
   E.g. var images\_dir = “webstudio/my\_images/”;
4. Zip the WebStudio war file once you are done.

**Note:** To revert back to using WebStudio’s default images set the value of “images\_dir” to “webstudio/images/” in WebStudio.html file.

# Messages/Properties

This feature allows a user to customize messages/labels seen in WebStudio UI. Typical use case being, WS already provides i18n support with a few language packs out of the box. There will be times when certain messages will need to be customized to explain the right meaning/context. Or add a new language support.

**Steps:**

1. All i18n files are located at <BE\_HOME>/rms/lib/locales. Each file corresponding to a particular locale. Format being <locale\_name>.jar.
2. In all there are about 7 properties inside the locale jar corresponding to various areas of the application. The format again is defined as filename\_<locale\_name>.properties.
3. To support a new locale, just make a copy of one of the existing locale jars and replace the values in each of the property file with the corresponding language specific values. Pack the files back into the jar. Maintain the naming convention for the property file as well as the jar file.
4. To edit one of the existing values, extract/unzip the corresponding locale jar, identify the files/properties that you need to change. Make the required change and repack it into the jar again.
5. Once all is done, just refresh the browser to see the changes. If you don’t see the changes, do a browser cache cleanup(at time the values are loaded from the cache) and reload the url again.