Table of Contents

The standard corporation application style has a button area, a menubar and a toolbar. The button and the menubar are always visible and the toolbar will be visible if a sidebar icon was set for a specific screen.

A simple example:

The orange rectangle contains the button area, the yellow area is the menubar and the green area is the toolbar (= sidebar). You have different options to customize the areas. It's possible to remove button from or add buttons to the button area. It's possible to hide the menubar and/or toolbar. You can hide the menubar and/or toolbar always or on-demand.

Let's start with the button area:

We removed the standard buttons and added a custom logout button. To customize the defaults, simply create your own application class:

CustomApplication.java

```
public class CustomApplication extends ProjX
{
    /** the additional button. */
    private UIButton butExit;

    public CustomApplication(UILauncher pLauncher) throws Throwable
    {
        super(pLauncher);

        if (pLauncher.isWebEnvironment())
        {
            Menu menu = getMenu();

            menu.removeItem(Menu.EDIT_RELOAD);
            menu.removeItem(Menu.EDIT_SAVE);
            menu.removeItem(Menu.EDIT_ROLLBACK);
            menu.removeItem(Menu.EDIT_UNDO);
            menu.removeItem(Menu.EDIT_REDO);

            menu.removeItem(WebMenu.OPTION_HOME);
            menu.removeItem(WebMenu.OPTION_EXPAND);
            menu.removeItem(WebMenuCorporation.OPTION_LOGOUT);
            menu.removeItem(WebMenuCorporation.OPTION_USER);

            if (menu instanceof WebMenu)
            {
                butExit = new UIButton();
butExit.setImage(UIImage.getImage(IFontAwesome.SIGN_OUT_LARGE));
                butExit.eventAction().addListener(this, "doLogout");
                Style.addStyleNames(butExit, "topbutton");
```

```
                ((WebMenu)menu).addOption(butExit);
            }
        }
    }

    @Override
    public void updateMenu(Menu pMenu)
    {
        super.updateMenu(pMenu);

        if (butExit != null)
        {
            butExit.setVisible(isConnected());
        }
    }
}
```

The custom application has to be set as main class for your launcher. Simply set the main parameter in your web.xml:

```
<servlet>
  <servlet-name>VaadinUI</servlet-name>
  <servlet-class>com.sibvisions.rad.ui.vaadin.server.VaadinServlet</servlet-
class>

  ...

  <init-param>
    <param-name>main</param-name>
    <param-value>com.sibvisions.apps.CustomApplication</param-value>
  </init-param>

  ...
</servlet>
```

Our CustomApplication does nothing if the runtime environment is not web because it should work in other environments as well. We remove existing buttons and add a new custom button to the button area.

Now we're customizing the menubar. First, let's remove the help menu:

Simply add

```
menu.removeItem(Menu.HELP);
```

to the existing code in our custom application.

Now we're removing the whole menubar:

Only **admin** users will be able to use the menubar.

```
@Override
public void afterLogin()
{
    super.afterLogin();

    if (getLauncher().isWebEnvironment())
    {
        if (isConnected())
        {
            getMenu().setMenuBarVisible(hasRole("admin"));
        }
    }
}
```

And finally, it's also possible to hide the toolbar:

```
@Override
public void afterLogin()
{
    super.afterLogin();

    if (getLauncher().isWebEnvironment())
    {
        if (isConnected())
        {
            Menu menu = getMenu();

            menu.setMenuBarVisible(hasRole("admin"));
            menu.setToolBarVisible(!hasRole("intern"));
        }
    }
}
```

Only external users will be able to use the toolbar.