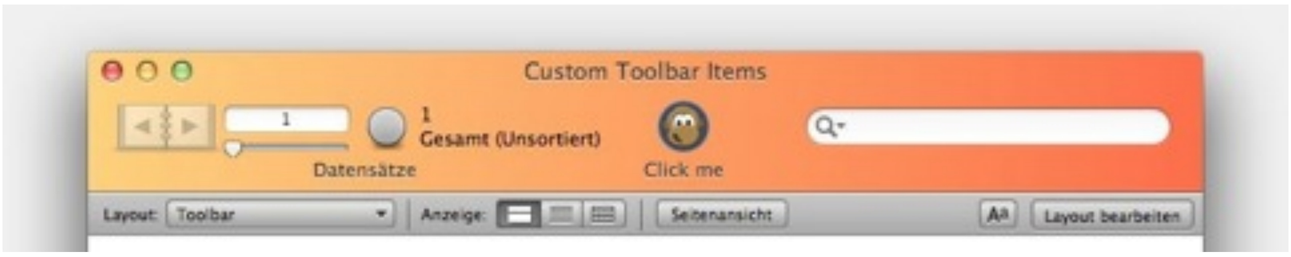
Toolbar functions for FileMaker on Mac



With plugin version 4.3 we introduce a couple of new toolbar functions. First we can install a background image. The Toolbar.InstallBackground function takes a picture from a container and installs the necessary functions to have a background picture. The plugin can draw the picture behind toolbar buttons, so you can customize your application. If you call again with different picture, you can change it or pass no image and clear it.

Configure Toolbar

The toolbar on Mac can be configured by the user. Normally the user will choose the menu command to customize it. Or you call the plugin function Window.RunToolbarCustomizationPalette and show the palette by script.

The configuration settings are saved as XML text. You can query current configuration of a toolbar in a window using Window.GetToolbarConfiguration and set it later with Window.SetToolbarConfiguration. This allows you to hide certain buttons by using special configuration. As the configuration settings are stored as xml, you can edit them yourself.

Your own toolbar buttons

To enable the plugin to add your own toolbar items, please call once Toolbar.Install function. This will instruct the plugin to install the necessary callbacks to intercept some toolbar functions which are called by the OS to find out what toolbar items FileMaker provides. This way plugin can add your custom identifiers to the list.

Now you can register toolbar items using the function Toolbar.Register. Call it once on startup of solution for each button you may want to introduce. You need to define your own unique identifier string for the toolbar button. Be aware that FileMaker already has a few identifiers defined already like NewRecordToolbarItemIdentifier. OS X also defines identifiers for standard items like NSToolbarFlexibleSpaceItem. With the parameters you define which script is called for this button by specifying a filename, a script name and the script parameter. For the button itself you specify an image (best PNG with mask), which is scaled down to the required sizes. You can specify a label for the toolbar and a second label for use in the customization panel. Of course you can also specify a tooltip as a help text.

Now when you registered your toolbar item, you can use the identifier in the configuration xml. In a script triggered by opening the layout, you can setup the right xml to be used for toolbar and enable the buttons you need. Your toolbar may look like the picture above.

Toolbar Configuration details

Normal XML looks like this from standard toolbar:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://
www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
     <key>TB Display Mode</key>
     <integer>1</integer>
     <key>TB Icon Size Mode</key>
     <integer>1</integer>
     <key>TB Is Shown</key>
     <integer>1</integer>
     <key>TB Size Mode</key>
     <integer>1</integer>
</dict>
</plist>
```

This is for a default toolbar and only specifies display mode (1 = Icon and Label), icon size mode (1 = regular), size mode (1 = regular) and that toolbar is visible.

Now if you really customized toolbar, the xml looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://
www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
     <key>TB Default Item Identifiers</key>
     <array>
          <string>RecordNavigationToolbarItemIdentifier</
string>
          <string>ShowAllToolbarItemIdentifier</string>
          <string>NSToolbarSpaceItem</string>
```

```
        <string>NewRecordToolbarItemIdentifier</string>
        <string>DeleteRecordToolbarItemIdentifier</string>
        <string>NSToolbarSpaceItem</string>
        <string>FindDropdownToolbarItemIdentifier</string>
        <string>SortToolbarItemIdentifier</string>
        <string>UploadServerToolbarItemIdentifier</string>
        <string>NSToolbarFlexibleSpaceItem</string>
        <string>QuickFindToolbarItemIdentifier</string>
    </array>
    <key>TB Display Mode</key>
    <integer>1</integer>
    <key>TB Icon Size Mode</key>
    <integer>1</integer>
    <key>TB Is Shown</key>
    <integer>1</integer>
    <key>TB Item Identifiers</key>
    <array>
        <string>RecordNavigationToolbarItemIdentifier</
string>
        <string>ShowAllToolbarItemIdentifier</string>
        <string>NSToolbarSpaceItem</string>
        <string>NewRecordToolbarItemIdentifier</string>
        <string>NSToolbarSpaceItem</string>
        <string>FindDropdownToolbarItemIdentifier</string>
        <string>SortToolbarItemIdentifier</string>
        <string>UploadServerToolbarItemIdentifier</string>
        <string>NSToolbarFlexibleSpaceItem</string>
        <string>QuickFindToolbarItemIdentifier</string>
        <string>TestItem</string>
    </array>
    <key>TB Size Mode</key>
    <integer>1</integer>
</dict>
</plist>
```

As you see this now specifies default item identifiers and also the current item identifiers. And here in the item list on bottom, I already added the TestItem as the identifier for our button. Using this identifier adds now the button when we apply this configuration.

If you have questions, please don't hesitate to ask us.