Локализация приложений при разработке в XCode 4

Roman greymag Volkov

Введение

Статья даёт общее представление о локализации приложений в XCode, которое я получил в ходе разработки первого приложения под iPhone. Под локализацией здесь я буду в основном подразумевать перевод фраз на разные языки, но следует помнить, что локализация включает в себя кроме этого форматирование дат, цен и т.д. в специфичном для каждой культуры виде, а также что графика тоже требует локализации.
Локализацию приложения в XCode можно условно разделить на две части:

- локализация строк, которые назначаются программно в коде;
- локализация элементов, выстроенных в Interface Builder (.xib, .nib, .storyboard - файлов).

Примечание. В дальнейшем все что будет упоминаться касательно .xib файлов должно быть справедливо и для .nib и .storyboard файлов.

Локализация строк

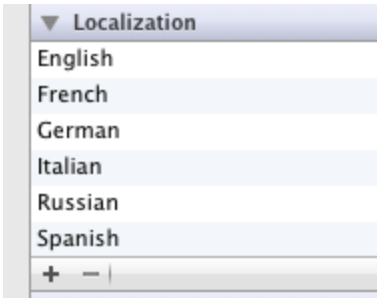
Для локализации строк, которые назначаются программно в Objective-C коде следует создать файл *.string. В этом файле будут перечислены все переводы в формате:

```
“key” = “value”;
```

Например

```
/* Insert Element menu item */
"Insert Element" = "Вставить элемент";
/* Error string used for unknown error types. */
"ErrorString_1" = "Неизвестная ошибка.";
```

В комментариях желательно давать описание с указанием, в каких случаях будет использоваться строка. Комментарии должны облегчить жизнь переводчика, когда вы отдадите эти файлы ему для перевода.
Добавлять разные языки для файла со строками можно как из XCode, воспользовавшись панелью локализации или вручную, создав файл с таким же названием в поддиректории языка .lproj (например ru.lproj).



Обычно приложение имеет как минимум один файл локализации. Имя файла со строками по умолчанию Localizable.strings, но вы можете создавать строковые файлы с любыми именами.
Чтобы использовать в коде эти переводы, вы должны вместо строк использовать макрос NSLocalizedString. Например

```
NSString *btnLabel =  NSLocalizedString(@"button_done", @"заголовок кнопки готово");
```

Первый аргумент это ключ из файла со строками, второй - просто комментарий. Подробнее можно прочитать в документации.

Замечания

Если строка в файле строк содержит несколько переменных аргументов, вы можете изменять порядок аргументов используя символ “$” и номер аргумента

```
/* Сообщение с оповещение что-то пошло не так */"Oh %@! %@ failed!" = "%2$@ бла бла, %1$@ ox!";
```

Также как в C, некоторые символы должны экранироваться обратным слэшем, чтоб быть включёнными в строку. Это двойные кавычки, обратный слэш и возврат каретки. Вы также можете определить возврат каретки с помощью “\n”:

```
"File \"%@\" cannot be opened" = " ... ";"Type \"OK\" when done" = " ... ";
```

Строки могут включать произвольные Unicode символы, задаваемые в виде "\U" и до четырех последующих восьмеричных цифр, определяющих Unicode символ. Например, пробел, который имеет код 20, представляется как “\U0020”. Эта возможность может быть полезна в том случае, если строка должна включать Unicode символ, который не может быть введён по какой-либо причине.
Строковые файлы лучше сохранять в Unicode формате.

Локализация элементов .xib файлов

После создания интерфейса страницы приложения с помощью Interface Builder на основном языке Вам нужно создать версии страницы для других языков, которые поддерживает приложение. Сделать это можно с помощью панели локализации, аналогично тому, как мы делали разные версии файла со строками в первой части.
Далее вы можете с помощью Interface Builder выбрать версию страницы для нужного языка и перевести необходимые элементы на разные языки, при этом скорректировав размеры/положение элементов, если потребуется.
Альтернативным способом перевода является использование консольного инструмента ibtool. С помощью ibtool можно извлечь из .xib файла все строки, требующие перевода

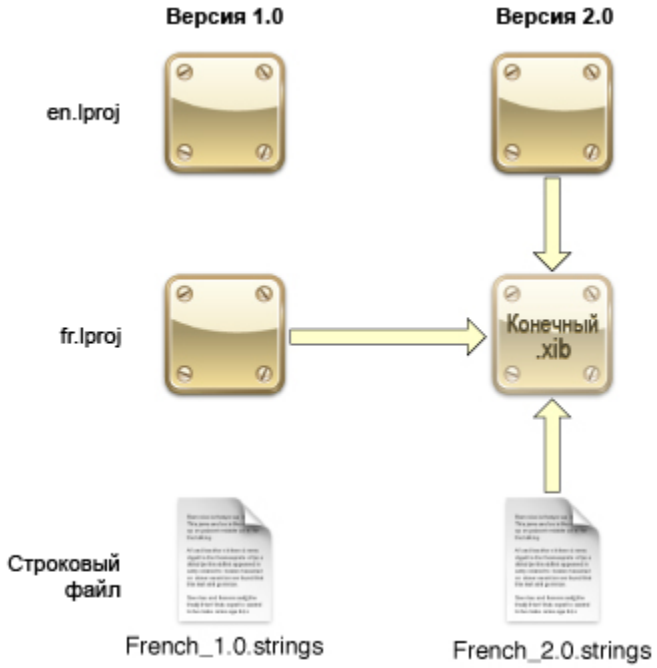
```
ibtool --generate-stringsfile MyPage.strings MyPage.xib
```

При этом получается строковый файл, с которым мы уже знакомы. Его можно отдать переводчику, после чего строки из переведённого файла можно импортировать обратно в .xib файл

```
ibtool --strings-file MyPage.strings --write MyNewPage.xib MyPage.xib
```

Здесь имеется ввиду что MyPage.xib это основной файл, а MyNewPage.xib локализованный на какой-то язык файл. Очевидно что в качестве MyNewPage.xib можно

указать MyPage.xib, чтобы просто обновить переводы в файле MyPage.xib.
Теперь мы имеем несколько файлов, каждый из которых будет отображаться для нужного языка. Но здесь есть проблема - если мы поменяем основной файл, то для остальных языков мы не увидим никаких изменений, т.к. это разные файлы. Конечно можно заново создать файлы страниц для разных языков, используя сохранённые строковые файлы переводов, но при этом мы потеряем все изменения, сделанные в странице для других языков (например размеры/положения элементов).
Для решения этой проблемы можно воспользоваться добавочным обновлением локализации (incremental localization update).
Следующая схема показывает какие файлы участвуют в добавочном обновлении. В этом примере изменения делались в английском (основном) .xib файле в версии 2.0 программы. Обновлённый английский .xib файл также включает некоторое количество новых элементов, строки для которых были переведены на французский и включены в обновлённую версию строкового файла для французского языка.



В процессе объединения ibtool берёт информацию из обоих, оригинального и нового английских .xib файлов, плюс оригинального французского .xib файла и обновлённого строкового файла, чтобы создать новый французский .xib файл. Команда может выглядеть следующим образом:

```
ibtool --previous-file ./old/Resources/en.lproj/MyPage.xib
       --incremental-file ./old/Resources/fr.lproj/MyPage.xib
       --strings-file ./new/Resources/fr.lproj/French_2.0.strings
       --localize-incremental
       --write ./new/Resources/fr.lproj/MyPage.xib
       ./new/Resources/en.lproj/MyPage.xib
```

Параметры --previous-file и --incremental-file определяют основной и переведённый .xib

файлы с предыдущего релиза. В рассматриваемом случае это исходный английский .xib файл и его французская копия. Входной файл для ibtool - это обновлённый основной .xib файл, содержащий новый контент для новой версии приложения.
Параметр --strings-file опциональный и необходим только когда вы ходите включить изменения или новые переводы в объединённый .xib файл. Параметр --write определяет имя файла, который вы хотите создать основываясь на таких входных данных.
Рассмотрим как выполняется слияние. Сначала ibtool копирует новый основной файл и затем переносит изменения из старого переведённого .xib файла, руководствуясь определённым набором правил. Рассмотрим эти правила, используя английский и французский .xib файлы, из примера выше.
1. ibtool копирует версию 2.0 английского .xib файла и использует копию как отправную точку для новой версии 2.0 французского .xib файла.
2. ibtool сравнивает объекты в версиях 1.0 и 2.0 английского .xib файла.

- Если свойства объекта одинаковы в обеих версиях английского .xib файла, ibtool копирует соответствующий объект из версии 1.0 французского .xib файла в версию 2.0 французского .xib файла (заменяя английскую версию этого объекта). Здесь предполагается, что если ничего не менялось между английскими версиями, то объект из старой французской версии .xib файла можно использовать как есть.
- Если версия 2.0 содержит объект, которого не было в версии 1.0, ibtool не делает ничего. (Это означает, что версия 2.0 французского .xib файла содержит новый объект, но с английским содержанием.)
- Если версия 1.0 содержит объект, которого нет в версии 2.0, ibtool не делает ничего. (Объект был удалён, значит его не надо сливать в новый .xib файл.)

3. Если определён строковый файл, ibtool сливает строки в версию 2.0 французского .xib файла.
Когда объединение закончено, вы получите новый .xib файл, который содержит все новые элементы отображения и управления, но также содержит столько переведённого контента, сколько возможно. Любые новые объекты, которые вы добавляете в интерфейс по-прежнему требуют перевода, если вы не обеспечили перевод в строковом файле во время слияния, но, по крайней мере, количество таких объектов снижается.

Предупреждение. Так как ibtool оперирует только объектами, найденными в основном .xib файле, вам следует ограничить изменения в локализованных .xib файлах только изменениями в строках и геометрии. Никогда не следует добавлять или удалять объекты в локализованных .xib файлах. Также не следует изменять типы объектов. такие изменения будут потеряны при добавочном обновлении.

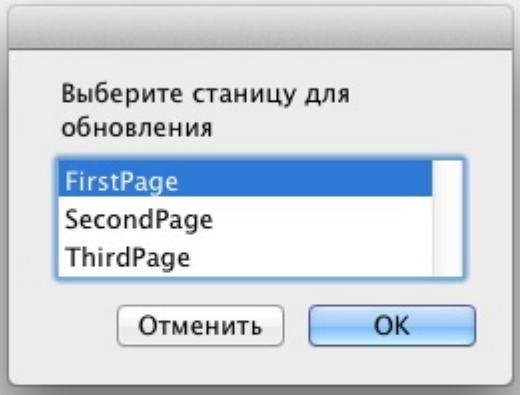
Автоматизация процесса (для Mac OS X)

Каждый раз выполнять эти команды вручную довольно обременительная операция, поэтому я написал для себя небольшой скрипт на AppleScript, автоматизирующий обновления.
Перед первым запуском необходимо отредактировать шапку скрипта, выставив необходимые параметры:

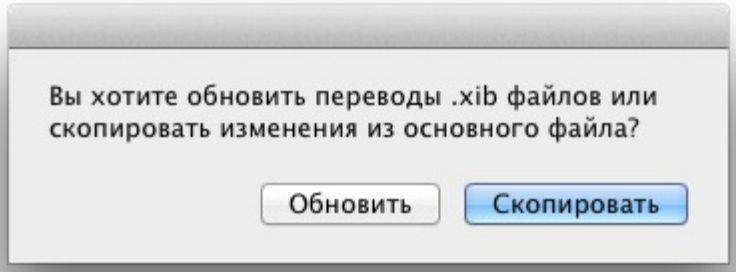
- путь до директории, в которой лежат .xib файлы, требующие локализации;

- список имён .xib файлов, которые требуют локализации (без расширения);
- языки, которые поддерживает приложение;
- основной язык приложения (по умолчанию en - английский);
- расширение файлов (по умолчанию xib).

После этой настройки скрипт готов к запуску. После запуска сначала будет предложено выбрать файл, для которого необходимо производить обновление (по необходимости можно переделать скрипт, чтобы он работал для всех файлов всегда - просто пробежавшись по всем страницам из списка).



Когда страница выбрана, скрипт спросит какую операцию вы хотите выполнить - обновление переводов или копирование изменений.



Если Вы не меняли основной .xib файл, а просто переводили фразы из строковых файлов - выбирайте обновление, оно выполняется быстрее.

После выполнения скрипта, в языковых директориях появятся строковые файлы .string с именами, соответствующими названиям .xib файлов. Вам надо перевести строки в этих файлах на соответствующие языки и выполнить обновление с помощью

скрипта, чтобы записать переводы в локализованные .xib файлы. Также в директории основного языка появится файл с суффиксом “.old”, это версия основного .xib файла на момент последнего обновления, именно она будет использоваться при последующем добавочном обновлении.

Предупреждение. Если .old файл не найден, то будет использован основной файл, а это означает, что все Ваши изменения в локализованных .xib файлах будут утеряны. Вызовите скрипт первый раз до того как сделаете какие-либо изменения в локализованных .xib файлах. Этот запуск создаст строковые файлы для перевода и копию текущей версии основного .xib файла для последующего добавочного обновления.

Листинг скрипта

```
-- Скрипт, который автоматизирует процесс локализации .xib (.nib, .storyboard) файлов
-- (c) greymag, 2012

global pagesPath
global fileExt

-- --
-- Здесь вам необходимо выставить нужные параметры
-- --

-- доступные языки для текущей локализации
-- основной
set mainLang to "en" -- укажите необходимый язык
-- остальные языки
-- для примера - немецкий, испанский, французский, итальянский и русский
set langs to {"de", "es", "fr", "it", "ru"} -- перечислите необходимые языки
--set langs to {"de"}

-- перечисляем файлы, которые будут локализованы
-- пример: set files4Localization to {"FirstPage", "SecondPage", "ThirdPage"}
set files4Localization to {} -- перечислите необходимые файлы (без расширения)

-- путь до директории с файлами для локализации
set pagesPath to "path_to_dir_with_xib_files" -- укажите необходимый путь

-- расширение локализуемых файлов
-- одно из трех значений: xib, nib, storyboard
set fileExt to "xib"

-- --
-- Дальше скрипт, ничего менять не надо
-- --

-- функция организации имени файла
```

```
on getFileName(pageName, lang)
	set filename to pagesPath & lang & ".lproj/" & pageName & "." & fileExt
end getFileName

-- функция проверки существования файла
on fileExist(filePath)
	tell application "Finder" to if exists filePath as POSIX file then return true
	return false
end fileExist

-- даём выбрать какой файл будет локализован
set pagesChooser to choose from list files4Localization with prompt "Выберите
станицу для обновления"
if pagesChooser = false then return

--выбираем название страницы из выборки
set curPage to item 1 of pagesChooser

-- берём путь до основного файла
set curPagePath to getFileName(curPage, mainLang)

-- имя файла основного языка без расширения
set mainLangName to mainLang & ".lproj/" & curPage
-- имя файла основного языка с расширением
set mainLangFileOld to mainLangName & ".old." & fileExt
-- имя файла основного языка с расширением
set mainLangFile to mainLangName & "." & fileExt

-- переходим в папку страниц проекта
set shellScript to "cd \"" & pagesPath & "\"; "

-- спрашиваем что делать - локализовать файл или создать новую версию
set action to display dialog "Вы хотите обновить переводы ." & fileExt & " файлов или
скопировать изменения из основного файла?" buttons {"Обновить", "Скопировать"}
default button 2

set doCopy to (button returned of action = "Скопировать")

-- если такой файл существует - пытаемся сделать всё как надо
if fileExist(curPagePath) then
  -- если есть "старый" файл основного языка - то будем пытаться сливать
  -- иначе - просто копируем все как есть
  set mergeFile to fileExist(getFileName(curPage & ".old", mainLang))

  repeat with i from 1 to number of items in langs
	set tmpLang to item i of langs

	-- имя файла текущего языка без расширения
	set tmpLangName to tmpLang & ".lproj/" & curPage
	-- имя файла для локализации
	set tmlLangFile to tmpLangName & "." & fileExt
```

```
		-- проверим - если нет локализованного файла - пытаемся создать его из
				текущей версии файла
		if fileExist(pagesPath & tmlLangFile) = false then
			set shellScript to shellScript & "ibtool --write " & tmlLangFile & " " &
					mainLangFile & "; "
		end if

		-- проверим - если нет файла со строками, то делаем его из текущей версии
				файла
		if fileExist(pagesPath & tmpLangName & ".strings") = false then
			set shellScript to shellScript & "ibtool --generate-strings-file " &
					tmpLangName & ".strings " & tmlLangFile & "; "
		end if

		-- если выбрано копирование - делаем объединение
		if doCopy = true then
			-- импортируем строки и сливаем изменения
			if mergeFile then
				set shellScript to shellScript & "ibtool --previous-file " &
						mainLangFileOld & " --incremental-
						file " & tmlLangFile & " --strings-
						file " & tmpLangName & ".strings
						--localize-incremental --write " &
						tmlLangFile & " " & mainLangFile
						& "; "
			else
				set shellScript to shellScript & "ibtool --strings-file " &
						tmpLangName & ".strings --write "
						& tmlLangFile & " " & mainLangFile
						& "; "
			end if

			-- обновляем файл со строками
			set shellScript to shellScript & "ibtool --generate-strings-file " &
					tmpLangName & ".strings " & tmlLangFile & "; "
		else
			-- иначе - просто импортируем строки
			set shellScript to shellScript & "ibtool --strings-file " & tmpLangName
					& ".strings --write " & tmlLangFile & " " &
					tmlLangFile & "; "
		end if
	end repeat

	if doCopy = true then
		-- сохраняем "старую" версию основного файла
		set shellScript to shellScript & "ibtool --strings-file " & mainLangName & ".strings
				--write " & mainLangFileOld & " " & mainLangFile & "; "
	end if

	-- запуск выполнения скрипта
```

```
  if (do shell script shellScript) = "" then
          if doCopy = true then
                  set message to "Файл " & curPage & " успешно обновлён"
          else
                  set message to "Обновление переводов файла " & curPage & "
                                          успешно завершено"
          end if
  else
          set message to "Ошибка при выполнении команды"
  end if
  display dialog message buttons {"OK"} default button 1
else
  -- если не существует - выводим сообщение об ошибке и все
  display dialog "Указанный файл не найден" buttons {"OK"} default button 1
end if
```

Использованы материалы из документации Apple http://developer.apple.com/