Translation

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.1/advanced-client-side-development/enabling-translation.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

You can use the IBM MobileFirst Platform Foundation framework to add multilingual translation of Hybrid applications into other languages. Items that can be translated are application strings and system messages. The platform can automatically translate application strings according to a designated file.

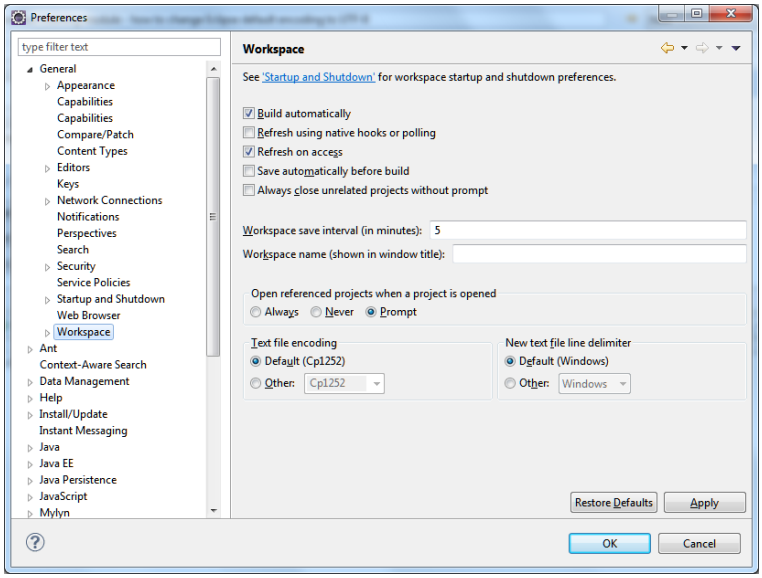
This tutorial covers the following topics:

- Encoding
- Enabling translation of application strings
- Enabling translation of system messages
- Multilanguage translation
- Detecting the device locale and language
- Sample application

Encoding

The default workspace encoding in Eclipse is `Cp1252`.

Before you create the MobileFirst project and start the translation work, you must change the default encoding of the Eclipse workspace.
In Eclipse, navigate to **Window > Preferences > General > Workspace** and change the encoding to `UTF-8`. If you have already created a project, you will need to go over each `.css` and `.js` file and change its encoding property.



Enabling translation of application strings

You can find the `messages.js` file, which is intended for application strings, in the `common\js` folder.

```
Messages = {
   headerText: "Default header",
   actionsLabel: "Default action label"
,
   sampleText: "Default sample text",
};
```

Application messages that are stored in the `messages.js` file can be referenced in two ways:

- As a JavaScript object property. For example: `Messages.headerText`
- As an ID of an HTML element with `class="translate"`.

```
<h1 id="headerText" class="translate"></h1>
```

Enabling translation of system messages

It is also possible to translate the system messages that the application displays, for example "Internet connection is not available" or "Invalid username or password".
System messages are stored in the `WL.ClientMessages` object.
You can find a full list of system messages in the `www\default\worklight\messages\messages.json` file, which is inside the generated projects (iOS, Android, Windows Phone 8, and so on,…).
To enable the translation of a system message, override it in your JavaScript application.

```
WL.ClienMessages.loading = "Application HelloWorld is loading... please wait.";
```

Override system messages at a global JavaScript level because some parts of the code are executed only after the application has successfully initialized.

Multilanguage translation

Using JavaScript, you can implement multilanguage translation for your applications.

1. Set up the default application strings in the `messages.js` file.

```
Messages = {
  headerText: "Default header",
  actionsLabel: "Default action label",
  sampleText: "Default sample text",
  englishLanguage : "English",
  frenchLanguage : "French",
  russianLanguage : "Russian",
  hebrewLanguage : "Hebrew"
};
```

2. Override specific strings when required.

```
function setFrench(){
    Messages.headerText = "Traduction";
    Messages.actionsLabel = "Sélectionnez une langue:";
    Messages.sampleText = "Ceci est un exemple de texte en français."
;
}
```

3. Update the GUI components with the new strings.
   You can perform more tasks, such as setting the text direction for right-to-left languages such as Hebrew or Arabic.
   Each time that an element is updated, it is updated with different strings according to the active language.

```
function languageChanged(lang) {
    if (typeof(lang)!="string")
        lang = $("#languages").val();

    switch (lang){
        case "english":
            setEnglish();
            break;
        case "french":
            setFrench();
            break;
        case "russian":
            setRussian();
            break;
        case "hebrew":
            setHebrew();
            break;
    }

    if ($("#languages").val()=="hebrew") {
        $("#wrapper").css({direction: 'rtl'});
    }
    else {
        $("#wrapper").css({direction: 'ltr'});
    }

    $("#sampleText").html(Messages.sampleText);
    $("#headerText").html(Messages.headerText);
    $("#actionsLabel").html(Messages.actionsLabel)
;
}
```

Detecting the device locale and language

It is possible to detect the locale and the language of the device.
Use the `WL.App.getDeviceLocale()` and `WL.App.getDeviceLanguage()` functions to detect the current locale.

```
var locale = WL.App.getDeviceLocale();
var lang = WL.App.getDeviceLanguage();
WL.Logger.debug(">> Detected locale: " + locale);
WL.Logger.debug(">> Detected language: " + lang);
```

```
05-12 12:27:19.685  D  26294  before: app init onSuccess
05-12 12:27:19.735  D  26294  >> Detected locale: en_US
05-12 12:27:19.745  D  26294  >> Detected language: en
05-12 12:27:19.775  D  26294  after: app init onSuccess
```

Sample application

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/Translation/tree/release71) the MobileFirst project.