LMI Developer Documentation

Set up

To take advantage of further development of the system there are a few prerequisites for developers to have installed in order to work on the system. These can be found in the OpenMRS [http://openmrs.org/wiki/Step-by-Step_Installation_for_Developers Step-by-Step Installation for Developers] Document.

In addition to OpenMRS, we have developed a php library to add rapid development procedures for changing and updating the system if new indicators are needed in country. To make use of this library there are a few prerequisites to a standard linux distribution:

The smarty-gettext library is necessary to allow for multiple languages to be used in the system forms. To that end we made use of the smarty-gettext library which pulls in both the [http://www.smarty.net/ Smarty PHP Library] and [http://php.net/gettext gettext for PHP].

These may be very simple installation on some linux distributions. For example, on Ubuntu linux one may simply type

```
 $ sudo apt-get install smarty-gettext
```

and all of the necessary items should be installed on a trouble-free machine.

`NOTE:`

For Ubuntu 8.04 (Hardy) you may need to issue the following command to fix a small problem with this library.

```
  mv /usr/share/php/smarty/libs/plugins/block.t.php
/usr/share/php/smarty/plugins/block.t.php
```

Extending the Forms

In order to extend the forms, you must add new fields and concepts to the forms in OpenMRS. Correpsonding PHP code has to be written so that the web forms will post informatino into OpenMRS.

For this example, we'll assume you want to add the household insurance indicators from the the second page of the LMI household form.

Extending OpenMRS Forms

Adding fields to an existing form is straight-forward. In [https://launchpad.net/rwanda-pilot the Rwanda Pilot], the <tt>bin/setup-openmrs-for-pilot</tt> script reads in csv files in the <tt>data</tt> directory. (Currently, these CSV files are set up to be editted in [http://orgmode.org Org-Mode] tables, so they must be bar delimited).

For our example, <tt>data/lmi-household-form.csv</tt> contains the following lines before adding the insurance indicators:

```
<pre><nowiki>
| Name             | Description                                        |
Type    | Datatype |
| MOSQUITO NETS    | Per Household Question: "Total # of Mosquito Nets"   |
Concept | Numeric  |
| TREATED NETS     | Per Household Question: "# of Treated Mosquito Nets" |
Concept | Numeric  |
| OLD TREATED NETS | Per Household Question: "Older than 6 months"        |
Concept | Boolean  |
</nowiki></pre>
```

There are two insurance indicators, MEMBERS COVERED and INSURANCE TYPE, that we want to add to the OpenMRS form. We add the following two lines to the file:

```
<pre><nowiki>
| MEMBERS COVERED  | Per Household Question: "# of Members covered"       |
Concept | Numeric  |
| INSURANCE TYPE   | Per Household Question: "Type of Insurance"          |
Concept | Text     |
</nowiki></pre>
```

Any form can be updated without running the whole setup script by using the following PHP snippet:

```
<source lang="php">
require("OpenMRS.php");
$csvfile = "data/lmi-household-form.csv";

$o = new OpenMRS($dbuser, $dbpass, $dbname, $dbserver);
$o->import_form("LMI Household Form", "The form for the household",
$csvfile);
</source>
```

The <code>import_form</code> method will look up the form by name and create it if it doesn't exist. It then reads the header of the file and then reads each line of the file. For each line, it finds or creates an OpenMRS concept with that name and finds or adds that field to the form.

Since the <code>import_form</code> method looks for existing forms, concepts, and fields before creating them, it is safe to run the import repeatedly to add more fields as neccessary to the OpenMRS form.

Extending HTML/PHP forms

Extending the HTML forms is not as simple right now. For our example, we'll need to update the control file for the first page of the household form, create a control file for the the insurance indicators, and create a template file for the insurance indicators. (For more information on how the Rwanda Web forms are set up and why, see [[LMI Webforms]].)

Updating the form's control file

The control file for the the first page of the Household indicators is named <tt>control/malaria_nets.inc.php</tt>. To send the user to the new form page after successfully completing, the <tt>$arg['goto_page']</tt> variable must be changed to point to the new page:

```
<source lang="php">
$arg['goto_page'] = "insurance";
</source>
```

Creating a new form template file

A template file is is named <tt>public_html/templates/''pagename''.tpl</tt> and provides the body of the web page. The LMI forms are called and then placed in a wrapper. All they need to do is provide a title and the body of the page using the Smarty templating engine:

```
<source lang="smarty">
{capture assign="title"}{t}Household Data{/t}{/capture}
<br/>
<form method="post">
  <fieldset>
    <legend>{t}Insurance{/t}</legend>
    <label>{t}Members Covered:{/t} <input type="text"
name="f_MEMBERS_COVERED" size="2"/></label><br/>
    <select name="f_INSURANCE_TYPE">
      <option>{t}Mutuelle{/t}</option>
      <option>{t}Community{/t}</option>
      <option>{t}Other{/t}</option>
      <option>{t}N/A{/t}</option>
    </select>
  </fieldset><br/>
  <input type="submit" name="submit" value="{t}Continue{/t}"/>
</form>
<br/>
</source>
```

Note that the names of the fields we defined in the CSV above are given here with the prefix <code>f_</code> and spaces replaced with an underbar. Also note that translatable strings are wrapped with <code>{t}...{/t}</code> to inform Smarty that a translation may be available.

Creating a new form control file

A control file is is named <tt>control/''pagename''.inc.php</tt> and expected to define a <code>page_handler()</code> function. Since the new page form is just adding observations to an existing encounter, the control page is pretty simple:

```
<source lang="php">
function page_handler($o, $arg) {
    if(array_key_exists('f_MEMBERS_COVERED', $_POST)) {
        $enc = new OpenMRS_Encounter($o,  $arg["this_enc_id"]);

        if($enc) {
            store_enc_values($enc);

            $arg['goto_page'] = 'finish';
        }
    }
    return $arg;
}
</source>
```

After checking that the control file was called with POST data, the current encounter is retrieved and then the <code>store_enc_values()</code> function is called to add observations to the OpenMRS form database.

By following the above example, it should be possible to extend the form and add any needed indicator.

OpenMRS Developer Resources

There is a great deal of information for developers on extending and modifying OpenMRS itself. A full reference to the OpenMRS API can be http://resources.openmrs.org/doc read on their developer resource page and the API can http://resources.openmrs.org/doc.zip also be downloaded as a zip file.

Of particular note is the http://openmrs.org/wiki/Developer_How-To_Guide OpenMRS Developer How-to Guide which will walk a developer step-by-step through setting up their environment to start working on OpenMRS.