Localization of ConfigServer

Introduction

As with many things, only the first time something new is tried it is really difficult. Doing it again becomes a lot easier. That applies to localization. The first two languages we are implementing for sipXconfig are German and Polish. Adding another language is really simple:

1. Find all the *.properties files
   ```
   find . -name *.properties
   ```
2. Add a new set of properties files for your language with a file name _*xx.properties* and translate all the strings

Done!

Many words common to all pages (Name, Description etc.) are localized in the common message repository:

sipXconfig/web/context/WEB-INF/sipXconfig-web.properties

If you see something on the screen and are not sure where to add translation for it, you can use grep to find the specific file. For example to find all the places where 'Active' is translated run this command:

find *path/to/sipXconfig/sources* -name *.properties | xargs grep "=Active"

HTML pages localization

All template (.html) and component/page definition (.page, .jwc) files are kept in:

sipXconfig/web/context/WEB-INF

Template (.html) files

sipXconfig is a Tapestry application. You can read more on localizing Tapestry applications here

All hardcoded string need to be replaced with the references to named properties (keys). For example to replace *Harcoded string* in the markup below you need to replace:

```
<p>Harcoded string</p>
```

with:

```
<p><span key="myKey">this will get replaced by value read from .properties</span></p>
```

and add myKey to the .properties file:

```
myKey=Hardcoded string
```

If you want to instert a localized text that with HTML markup in it use @Insert component instead of span method.

```
<p jwcid="@Insert" value="message:myKey" raw="ognl:true" element="p"/>
```

With this method you can use HTML tags in your properties file

```
myKey=<em>Hardcoded</em> string
```

Specification (.page and .jwc) files

You need to replace "literal:Hardcoded string" binding with "message:myKey" binding and add myKey to properties file.
You have a choice of adding the key to component/page specific properties file (i.e. if Login.page that would be Login.properties) or to global message repository -
sipXconfig-web.properties

For example if you have the following in common/ItemCommon.jwc:

```
<binding name="label" value="literal:Enabled"/>
```

replace it by:

```
<binding name="label" value="message:enabled"/>
```

And add key 'enabled' to common/ItemCommon.properties or to sipXconfig-web.properties.
You can try running:

```
find -name "**.page" -o -name "**.jwc" | xargs grep literal:A-Z
```

to find unlocalized strings. Some of the literal bindings should not be localized: for example internal page names.

Models localization

It works in a very similar way to translating the pages. For every model file (eg. snom/phone.xml) there is a property file (let's say snom/phone.properties). When adding support for a new language, you add a new, localized .properties file (eg. snome/phone_de.properties for German, snom/phone_pl.properties for Polish etc.)

For those using Eclipse your JInto plugin should work.

Full setting name followed either by '.label' or by '.description' is used to idenfity properties for settings labels and descriptions. To continue with SNOM example:

In phone.xml we have:

```
<group name="Basic_Settings">
<setting name="web_language">
....
```

In phone.properties

```
Basic_Settings.web_language.label=Web Language
Basic_Settings.web_language.description=The language for the web interface.
```

The system will default to whatever is in XML file if it does not find a property. However if the property value is empty it'll display empty string.

If you want to start translating model that does not have .properties file, you can run xml_to_properties script to generate skeleton file with English labels skimmed from xml file.

```
meta/xml_to_properties --help
Usage:
meta/xml_to_properties -file phone.xml
Create properties file from label and description information contained in xml file.
If file name is provided properties file is created with the same base name and in the
same directory as xml file.
Options:
-f, --file name of the xml file
```

If started with no options it behaves as filter - reads standard input and writes to standard output.

Enumeration types labels localization

You can optionally localize the labels for enumeration settings. Those settings are displayed in UI as a drop down selection list widget: user selects a value from the small set of options rather than typing in the correct options.
There are options that you probably do not want to localize (such as protocol names: UDP/TFTP?) but in many cases localization would increase usability.

There are couple of ways in which labels can be localized. If the enumeration type has an id, you can use it to construct the keys for the labels. For example:

```
<type id="ringer-pattern">
<enum>
<option>
<value>1</value>
</option>
<option>
<value>other</value>
</option>
<option>
<value/>
</option>
</enum>
</type>
```

You'd need to put the following keys in the .properties file corresponding to the model.

```
type.ringer-pattern.1=Silent Ring
type.ringer-pattern.other=Low Trill
type.ringer-pattern.=-disabled-
```

As you can see you can localize an empty value as well.
It is perfectly valid not to localize some of the options. In such cases do not add the key at all. Adding empty key would make sipXconfig to display empty label instead of default (English) one.

If type does not have to id you can explicitely set the label key prefix ("type.ringer-pattern" in the example above). You need to modify *enum* elemenent in xml file. For example:

```
<type>
<enum labelKeyPrefix="ring_labels">
<option>
<value>1</value>
</option>
<option>
<value>other</value>
</option>
<option>
<value/>
</option>
</enum>
</type>
```

In this case your properties need to look like this:

```
ring_labels.1=Silent Ring
ring_labels.type.ringer-pattern.other=Low Trill
ring_labels.type.ringer-pattern.=-disabled-
```

Finally you can use the setting name as a prefix for the label. This works best if enumeration is used only for a very specific setting.

```
<group name="Basic_Settings">
<setting name="ringer_setting">
<type>
<enum>
<option>
<value>1</value>
</option>
<option>
<value>other</value>
</option>
...
```

In this case your phone.properties will have to include the following keys:

```
Basic_Settings.ringer_setting.label.1=Silent Ring
Basic_Settings.ringer_setting.label.other=Low Trill
```