LOCALIZATION

LOCALIZATION IN DXP

- Liferay makes it easy to support translation of your application into any language.
- The process involves creating `language keys` that correspond to specific translations.
- These key/value pairs belong in language properties files.

WHAT ARE LANGUAGE PROPERTIES FILES?

- Language properties files are documents containing your language keys and translations.
- First, it is necessary to create a default language properties file named `Language.properties`.
- For each language you'd like to support, you will need an additional file that is named `Language_xx.properties`, where `xx` is the language abbreviation.
- For example, if you'd like to support English, French, and Spanish in your application, you would have files named:

```
Language.properties
Language_fr.properties
Language_es.properties
```

DEFAULT LOCALES

- Some locales are available by default in Liferay.
- Look in the `portal.properties` file to find them.

```
locales=ar_SA,eu_ES,bg_BG,ca_AD,ca_ES,zh_CN,zh_TW,hr_HR,cs_CZ,da_DK,nl_NL,
    nl_BE,en_US,en_GB,en_AU,et_EE,fi_FI,fr_FR,fr_CA,gl_ES,de_DE,el_GR,
    iw_IL,hi_IN,hu_HU,in_ID,it_IT,ja_JP,ko_KR,lo_LA,lt_LT,nb_NO,fa_IR,
    pl_PL,pt_BR,pt_PT,ro_RO,ru_RU,sr_RS,sr_RS_latin,sl_SI,sk_SK,es_ES,
    sv_SE,tr_TR,uk_UA,vi_VN
```

LANGUAGE FILES

- In an application with only one module that holds all the views (for example, all its JSPs) and application components, just create an `src/main/resources/content` folder in that module and place all your `Language_xx.properties` there.
- After that, make sure any application components (the `@Component` annotation in your -Portlet classes) in the module include this property:

  `"javax.portlet.resource-bundle=content.Language"`

- Providing translated language properties files and specifying the javax.portlet.resource-bundle property in your application component is all you need to do to have your language keys translated.
- Then, when the locale is changed in Liferay, your application's language keys will be automatically translated.

LANGUAGE FILE FORMAT

- Language files follow the standard properties file format. They should look something like:

```
hello=Hello
welcome-to-liferay=Welcome to Liferay
please-click-here-to-continue=Please click here to continue.
```

AUTOMATICALLY GENERATING LANGUAGE FILES

- Instead of manually creating a language properties file for each locale that's supported by Liferay, you can get them all automatically generated for you with one command.
- The same command also propagates the keys from the default language file to all of the translation files.

MICROSOFT'S TRANSLATOR API

- You can take a few additional steps and get automatic translations using Microsoft's Translator API.
  1. Make sure your module's build includes the `com.liferay.lang.builder plugin` by putting the plugin in build script classpath.
  2. Make sure you have a default Language.properties file in `src/main/content`.
  3. Run the gradle `buildLang` task from your project's root directory to generate default translation files.
- The generated files will contain automatic copies of all the keys and values in your default Language.properties files.
- That way you don't have to worry about manually copying your lanugage keys into all of the files.
- Just run the `buildLang` task each time you change the default language file.

HOW TO AUTO-GENERATE LANGUAGE FILES

- If you'd like to use `Microsoft's Translator API`, you must register your application with Azure DataMarket.
  1. Follow the instructions here: *https://msdn.microsoft.com/en-us/library/hh454950*
  2. Make sure the `buildLang` task knows to use your credentials for translation.
  3. For security reasons, you probably don't want to pass them directly in your application's build script.

```
buildLang {
translateClientId = "my-id"
translateClientSecret = "my-secret"
}
```

- Then, when you run `buildLang`, translations will be automatically generated.

TRANSLATIONS ON THE FRONT-END

- There is a Liferay language object defined at `Liferay.Language`.
- It contains all of the available locales along with a `get` method. When the `get` method is passed, a language key will return the translation in the current locale.

```
Liferay.Language.get('click-here'); //returns "Click Here"
```

Notes: