Internationalization

Suggested Resources

- https://www.tutorialspoint.com/jsf/jsf_internationalization.htm
- https://docs.oracle.com/javaee/7/tutorial/webi18n.htm

Bilingual Web Applications

- Especially important in Government of Canada context
- Best Practice:
  - User is presented with a bilingual splash page (with tags chosen based on the request's stated list of locales that the client accepts)

Internationalization and locales

- **Internationalization:** enabling a web site to provide different versions of content translated into the visitor's language or nationality.
- **Localization:** adding resources to a web site to adapt it to a particular geographical or cultural region for example Hindi translation to a web site.
- **locale:** a particular cultural or geographical region. It is usually referred to as a language symbol followed by a country symbol which are separated by an underscore. For example "en_US" represents english locale for US.

- Adapted from: http://www.tutorialspoint.com/jsp/jsp_internationalization.htm

Getting the list of accepted locales

- **getLocale() or getLocales**
- public java.util.Enumeration **getLocales**()
- Returns an Enumeration of Locale objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header. If the client request doesn't provide an Accept-Language header, this method returns an Enumeration containing one Locale, the default locale for the server.
- **Returns:** an Enumeration of preferred Locale objects for the client

Using a Resource Bundle

```
import java.util.Locale;
import java.util.ResourceBundle;

public class ResourceBundleDemo {
   public static void main(String[] args) {
            // create a new ResourceBundle with default Canadian Locale

            ResourceBundle bundleCA =
                        ResourceBundle.getBundle(“BundleDemo", Locale.CA);

            System.out.println( bundleCA.getString("hello"));             System.out.println( “ : “ +
   bundleCA.getLocale().toString());
   }
 }
```

Preparing the strings for the Resource Bundle

- Put properties files in your CLASSPATH for each supported language
  - BundleDemo_en_CA.properties
  - BundleDemo_fr_CA.properties
  - BundleDemo_en_US.properties
  - Etc
- Files contain lines of the form
  - Key=String Value
  - E.g., hello=Bonjour

Our Resource Bundle Example Output....

- Output

  Bonjour : fr_CA

- Assuming that there was a file on the CLASSPATH
  - BundleDemo_fr_CA.properties

What else changes with locale?

- Dates, times, currency, percentages all have local formats
- Display ordering of text is locale-specific
- See the Internationalization tutorial posted in the course files archive
  - Esp. the tutorial at:
  - http://www.tutorialspoint.com/jsp/jsp_internationalization.htm

What else needs localization?

- We have the user interface "covered" using ResourceBundles and locale-specific layouts using Servlets and/or locale-specific forwarding of requests
  - For example: for a request sent to filename.jsp
    - Could forward the request to either filename_en_CA.jsp or filename_fr_CA.jsp based on the locale
  - OR
    - Could use Java to output locale-specific content within filename.jsp
  - OR
    - Use ResourceBundles to translate the strings

Where to store the translated data?

- Reference Data
  - Naturally in database
- Configuration Data
  - Naturally in database or in resource bundles associated with this application
- Inputs/Generated Outputs
  - Naturally in database;
  - translate at source or near source

How to translate the data?

- Get the user to do it for you
- Crowd-source the translation
  - (get users to do it for you)
- Use a translation web service
  - (see next page)

Web translation tools

- Google translate at https://translate.google.com will provide translated strings for populating the properties file.
- If you need online translation at run-time (e.g., for text strings supplied by users), consider using the following index of translation tools (or pay for Google Translate API).
  - http://www.programmableweb.com/news/63-translation-apis-bing-google-translate-and-google-ajax-language/2013/01/15

Storing structured multilingual data

- Each field that is translated can be replicated to have N fields (where N is the number of languages) in the table
  - A naming convention will be important:
    - E.g., fieldname_en , fieldname_fr
  - Then can adjust queries at run-time to get the required fields.
- OR (based on the Resource Bundle concept)
  - Store keys to strings in each field
  - Use the key to translate the string based on the locale (either using a resource bundle or a resourceString table containing a locale field

The locale field with key lookup

- resourceStringTable – composite key (locale, resourceKey)
  - Locale, ResourceKey, ResourceValue

- Then each table containing a translatable resource would have a ResourceKey stored instead

Best Practice?

- Very application-specific, but….
- Using duplicate fields for bilingual databases is very common → can make move to three languages extremely difficult
- If you expect internationalization, plan for it.