JSF 2: Properties Files, Messages, and I18N

JSF 2.2 Version

Originals of slides and source code for examples: http://www.coreservlets.com/JSF-Tutorial/jsf2/
Also see the PrimeFaces tutorial – http://www.coreservlets.com/JSF-Tutorial/primefaces/
and customized JSF2 and PrimeFaces training courses – http://courses.coreservlets.com/jsf-training.html

For live training on JSF 2 and/or PrimeFaces, email hall@coreservlets.com.

Marty is also available for consulting and development support.

Taught by the author of *Core Servlets and JSP*, this tutorial, and JSF 2.2 version of *Core JSF*. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - JSF 2.2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics
  - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Hadoop, Spring, Hibernate/JPA, GWT, RESTful Web Services

Contact hall@coreservlets.com for details

Agenda

- Creating properties files
- Declaring properties files in faces-config.xml
- Simple messages
- Parameterized messages
- Internationalized messages

Simple Messages

Motivation

- **Idea**
  - Store some fixed strings in a simple plain-text file. Load that file and refer to the strings by the names given in the file.
- **Purpose**
  - Reuse same strings in multiple pages.
  - Update in one fell swoop.
- **Notes**
  - Bean properties are for values that change at runtime.
  - Entries in properties files are much simpler strings that are constant for the life of the application, but either appear multiple places or might change some time in the future.

Displaying Fixed Strings (From Top-Level Folder)

1. **Create a .properties file**
   - Contains simple keyName=value pairs
   - Must be deployed to WEB-INF/classes
     - In Eclipse, this means you put it in "src" folder
2. **Declare with resource-bundle in faces-config**
   - base-name gives base file name relative to "src" (classes)
   - var gives scoped variable (Map) that will hold results
     - E.g., for WEB-INF/classes/messages.properties

```
<application>
<resource-bundle>
    <base-name>messages</base-name>
    <var>msgs</var>
</resource-bundle>
</application>
```

3. **Output messages using normal EL**
   - #{msgs.keyName}

Displaying Fixed Strings (Minor Variation: Subfolders)

1. **Create a .properties file**
   - Contains simple keyName=value pairs
   - Deployed to WEB-INF/classes/resources (or other name)
     - In Eclipse, this means you create folder called "resources" under "src" and put the properties file in the src/resources folder
2. **Declare with resource-bundle in faces-config**
   - base-name gives package name, dot, base file name
   - var gives scoped variable (Map) that will hold results
     - E.g., for WEB-INF/classes/resources/messages.properties

```
<application>
<resource-bundle>
    <base-name>resources.messages</base-name>
    <var>msgs</var>
 </resource-bundle>
<application>
```

3. **Output messages using normal EL**
   - #{msgs.keyName}

src/messages1.properties

```
registrationTitle=Registration
registrationText=Please enter your first name,
   last name, and email address.
firstNamePrompt=Enter first name
lastNamePrompt=Enter last name
emailAddressPrompt=Enter email address
buttonLabel=Register Me
successTitle=Success
successText=You registered successfully.
```

This is a single line in actual file. You can break long lines into multiple lines by putting \ at the end of a line.

- At runtime, this will be …/WEB-INF/classes/messages1.properties
- faces-config.xml will load this with (inside "application" element)

```
<resource-bundle>
    <base-name>messages1</base-name>
    <var>msgs1</var>
</resource-bundle>
```

- Facelets page will output messages with
  - #{msgs1.firstNamePrompt}

faces-config.xml

```
<?xml version="1.0"?>
<faces-config …>
 <application>
    <resource-bundle>
      <base-name>messages1</base-name>
      <var>msgs1</var>
    </resource-bundle>
  </application>
</faces-config>
```

simple-messages.xhtml (Top)

```
<!DOCTYPE … >
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head><title>#{msgs1.registrationTitle}</title>
<link href="./css/styles.css"
      rel="stylesheet" type="text/css"/>
</h:head>
<h:body>
<table border="5" align="center">
  <tr>
  <th class="title">#{msgs1.registrationTitle}</th></tr>
</table>
<h3>#{msgs1.registrationText}</h3>
```

simple-messages.xhtml (Bottom)

```
<h:form>
#{msgs1.firstNamePrompt}:
<h:inputText value="#{person1.firstName}"/>
<br/>
#{msgs1.lastNamePrompt}:
<h:inputText value="#{person1.lastName}"/>
<br/>
#{msgs1.emailAddressPrompt}:
<h:inputText value="#{person1.emailAddress}"/>
<br/>
<h:commandButton value="#{msgs1.buttonLabel}"
                 action="#{person1.doRegistration}"/>
</h:form>
</h:body></html>
```

Person.java

```
public abstract class Person {
  private String firstName, lastName, emailAddress;

  public String getFirstName() {
    return(firstName);
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  …  // get/setLastName, get/setEmailAddress

  public abstract String doRegistration();
}
```

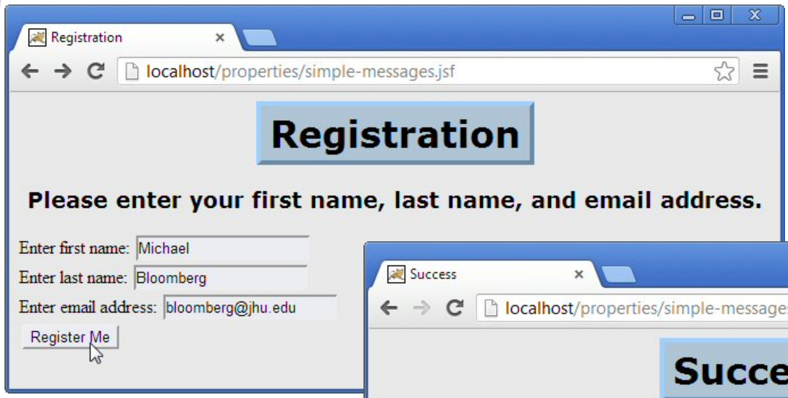
Person1.java

```
@ManagedBean
public class Person1 extends Person {
  public String doRegistration() {
    return("success1");
  }
}
```

success1.xhtml

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head><title>#{msgs1.successTitle}</title>
…
</h:head>
<h:body>
<table border="5" align="center">
  <tr><th class="title">#{msgs1.successTitle}</th></tr>
</table>
<p>#{msgs1.successText}</p>
<ul>
  <li>#{person1.firstName}</li>
  <li>#{person1.lastName}</li>
  <li>#{person1.emailAddress}</li>
</ul>
</h:body></html>
```

Results

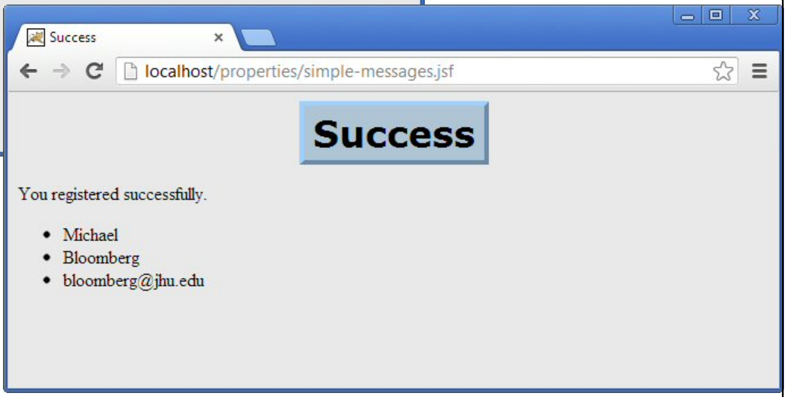

Registration

localhost/properties/simple-messages.jsf

Registration

Please enter your first name, last name, and email address.

Enter first name: Michael
Enter last name: Bloomberg
Enter email address: bloomberg@jhu.edu


Success

localhost/properties/simple-messages.jsf

Success

You registered successfully.

- Michael
- Bloomberg
- bloomberg@jhu.edu

Parameterized Messages

Motivation

- **Idea**
  – Store some strings in a simple plain-text file. Load that file and refer to the strings by the names given in the file. Allow portions of the strings to be replaced.
- **Purpose**
  – Make strings more flexible by having one string refer to another.
  – Allow runtime values to be inserted into strings.
    - Particularly useful for prompts and error messages
- **Notes**
  – These are no longer purely static strings. However, the basic outline of the string is still fixed, so they are different from bean properties, which are very dynamic.

Approach: Parameterizing Strings

1. **Create a .properties file in src**
   – Values contain {0}, {1}, {2}, etc.
   – E.g., someName=blah {0} blah {1}
     - Reminder: “src” in Eclipse becomes WEB-INF/classes when deployed
2. **Declare file with resource-bundle as before**
   – base-name gives base file name
   – var gives scoped variable (Map) that will hold results
3. **Output messages using h:outputFormat**
   – value gives base message
   – Nested f:param gives substitution values. These can be literal strings or runtime values
   – E.g.:

```
<h:outputFormat value="#{msgs.someName}">
    <f:param value="Literal value for 0th entry"/>
    <f:param value="#{someBean.calculatedValForEnty1}"/>
</h:outputFormat>
```

More on f:param

- **You must define f: namespace**
  - Since the tag uses a different prefix (f:), you must define the f: namespace in the <html…> tag at the top
  - We did this in some earlier sections when we used f:selectItems inside of h:selectOneMenu
- **Example**

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head>…</h:head>
<h:body>…</h:body>
</html>
```

messages2.properties

```
registrationTitle=Registration
firstName=First Name
lastName=Last Name
emailAddress=Email Address
registrationText=Please Enter Your {0}, {1}, and {2}.
prompt=Enter {0}
buttonLabel=Register Me
successTitle=Success
successText=You Registered Successfully.
```

faces-config.xml

```
<?xml version="1.0"?>
<faces-config …>
 <application>
    <resource-bundle>
      <base-name>messages1</base-name>
      <var>msgs1</var>
    </resource-bundle>
    <resource-bundle>
      <base-name>messages2</base-name>
      <var>msgs2</var>
    </resource-bundle>
  </application>
</faces-config>
```

parameterized-messages.xhtml (Snippet)

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head><title>#{msgs2.registrationTitle}</title>
…
<h3>
<h:outputFormat value="#{msgs2.registrationText}">
  <f:param value="#{msgs2.firstName}"/>   Replaces {0} in registrationText
  <f:param value="#{msgs2.lastName}"/>    Replaces {1} in registrationText
  <f:param value="#{msgs2.emailAddress}"/> Replaces {2} in registrationText
</h:outputFormat>
…
```

Person2.java

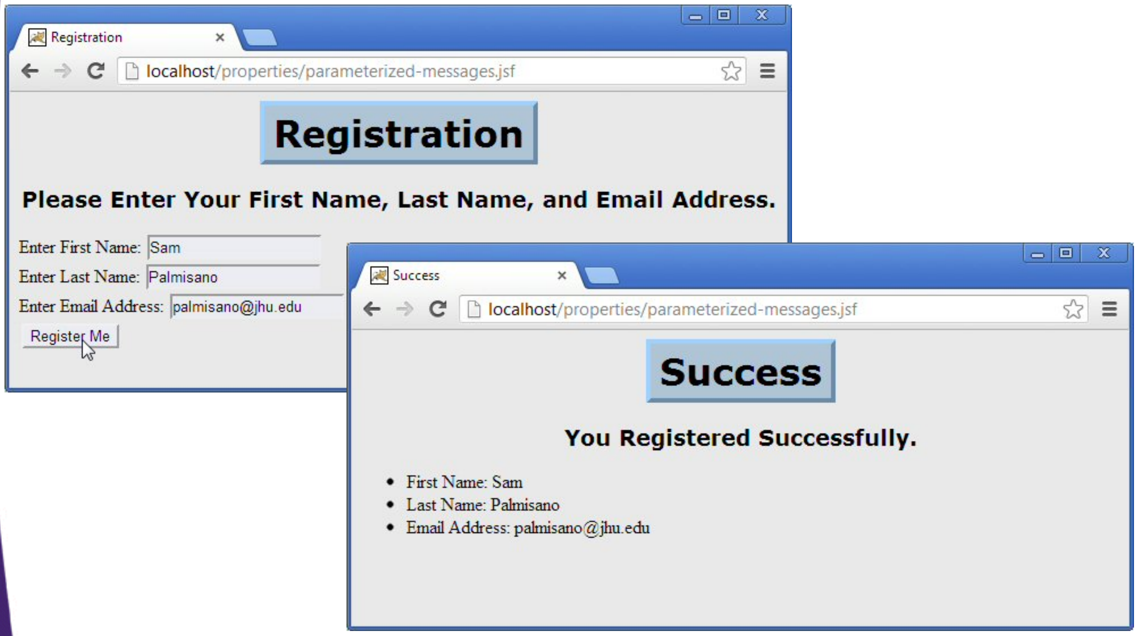
```
@ManagedBean
public class Person2 extends Person {
  public String doRegistration() {
    return("success2");
  }
}
```

success2.xhtml

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head><title>#{msgs2.successTitle}</title>
…
</h:head>
<h:body>
<table border="5" align="center">
  <tr><th class="title">#{msgs2.successTitle}</th></tr>
</table>
<h3>#{msgs2.successText}</h3>
<ul>
  <li>#{msgs2.firstName}: #{person2.firstName}</li>
  <li>#{msgs2.lastName}: #{person2.lastName}</li>
  <li>#{msgs2.emailAddress}: #{person2.emailAddress}</li>
</ul>
</h:body></html>
```

Since “First Name” (etc.) were substituted into the prompts on input page (instead of being hardcoded into “Enter First Name”), then those phrases are reusable in other pages without repetition in properties file.

Results


Registration

localhost/properties/parameterized-messages.jsf

Registration

Please Enter Your First Name, Last Name, and Email Address.

Enter First Name: Sam
Enter Last Name: Palmisano
Enter Email Address: palmisano@jhu.edu

Register Me

Success

localhost/properties/parameterized-messages.jsf

Success

You Registered Successfully.

- First Name: Sam
- Last Name: Palmisano
- Email Address: palmisano@jhu.edu

Internationalization and Messages

Motivation

- **Idea**
  - Store some multiple versions of properties file (msg.properties, msg_es.properties, msg_es_mx.properties, etc.)
    - Base properties file gets loaded (e.g., msg.properties) . Then one most loosely matching current Locale, if any (e.g., msg_es.properties), then one more specifically matching, if any (e.g., msg_es_mx.properties). If same name occurs in more than one file, later file wins.
- **Purpose**
  - Let page be displayed in multiple languages
- **Notes**
  - This example will show Locale selected based on browser language settings. See tutorial section on event handling to see Locale selected based on user choices.

Approach: Localizing Strings

1. **Create multiple similarly named .properties files**
   - blah.properties, blah_es.properties, blah_es_mx.properties
2. **Use f:view and locale attribute**

   `<f:view locale="#{facesContext.externalContext.requestLocale}">`
   - Determines locale from browser language settings
   - Note: can also set the Locale based on user input
     - See event handling section for best approach
3. **Declare file with resource-bundle as before**
   - base-name gives base file name
     - Version matching Locale will be used *automatically*
   - var gives scoped variable (Map) that will hold results
4. **Output using h:outputFormat or normal EL**
   - Same as before

Quick Example: Properties Files

- **messages.properties**
  - company=JsfResort.com
  - feature=Our {0}:
  - pool=swimming pool
- **messages_es.properties**
  - feature=Nuestra {0}:
  - pool=piscina
- **messages_es_mx.properties**
  - pool=alberca

Quick Example: faces-config.xml

```
<?xml version="1.0"?>
<faces-config …>
  <application>
    <resource-bundle>
      <base-name>messages</base-name>
      <var>msgs</var>
    </resource-bundle>
  </application>
</faces-config>
```

Quick Example: Facelets

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<f:view
   locale="#{facesContext.externalContext.requestLocale}">
…
<h1>#{msgs.company}</h1>
<h2>
<h:outputFormat value="#{msgs.feature}">
  <f:param value="#{msgs.pool}"/>
</h:outputFormat>
</h2>
<img src="fancy-swimming-pool.jpg"…/>
…
</f:view></html>
```

Quick Example: Results

- English (or any language except Spanish)

  JsfResort.com

  Our swimming pool:

  (Picture of pool)

- Non-Mexican Spanish

  JsfResort.com

  Nuestra piscina:

  (Picture of pool)

- Mexican Spanish

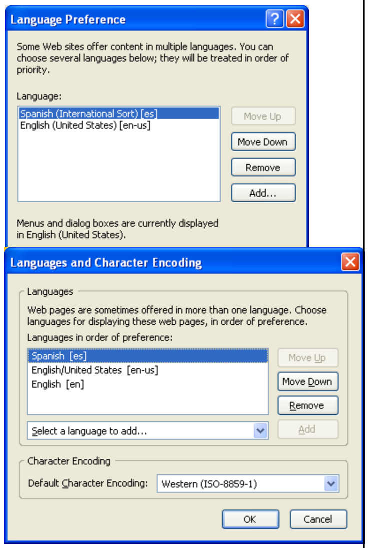
  JsfResort.com

  Nuestra alberca:

  (Picture of pool)

Setting Language Preferences in Browsers

- **Internet Explorer**
  - Tools, Internet Options, General, Languages
  - Click Add, select language, OK
  - Move to top of list using “Move Up”
- **Firefox 3**
  - Enter “about:config” as URL
  - Scroll down to entry named “general.useragent.locale”
  - Double click it and enter Locale by hand (es, es-mx), etc.
- **Firefox 4 through Firefox 6**
  - Tools, Options, Content tab, Languages, click Choose, move to top



Some Web sites offer content in multiple languages. You can choose several languages below; they will be treated in order of priority.

Language:
Spanish (International Sort) [es]
English (United States) [en-us]

Menus and dialog boxes are currently displayed in English (United States).

Web pages are sometimes offered in more than one language. Choose languages for displaying these web pages, in order of preference.

Languages in order of preference:
Spanish [es]
English/United States [en-us]
English [en]

Select a language to add...

Character Encoding
Default Character Encoding: Western (ISO-8859-1)

messages2.properties

```
registrationTitle=Registration
firstName=First Name
lastName=Last Name
emailAddress=Email Address
registrationText=Please Enter Your {0}, {1}, and {2}.
prompt=Enter {0}
buttonLabel=Register Me
successTitle=Success
successText=You Registered Successfully.
```

messages2_es.properties

```
registrationTitle=Registro
firstName=Primer Nombre
lastName=Apellido
emailAddress=Dirección de Email
registrationText=Incorpore Por
                      Favor su {0}, {1}, y {2}.
prompt=Incorpore {0}
buttonLabel=Coloqúeme
successTitle=Éxito
successText=Se Registró con Éxito.
```

This is a single line in actual file.

messages2_fr.properties

```
registrationTitle=Enregistrement
firstName=Prénom
lastName=Nom
emailAddress=Adresse électronique
registrationText=Merci de Entrer
                          Votre {0}, {1}, et {2}.
prompt=Entrez Votre {0}
buttonLabel=Enregistrez Moi
successTitle=Succès
successText=Vous Avez Enregistré Avec Succès.
```

This is a single line in actual file.

faces-config.xml

```
<?xml version="1.0"?>
<faces-config …>
 <application>
    <resource-bundle>
      <base-name>messages1</base-name>
      <var>msgs1</var>
    </resource-bundle>
    <resource-bundle>
      <base-name>messages2</base-name>
      <var>msgs2</var>
    </resource-bundle>
  </application>
</faces-config>
```

internationalized-messages.xhtml (Snippet)

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<f:view locale="#{facesContext.externalContext.requestLocale}">
<h:head><title>#{msgs2.registrationTitle}</title>
…
<h3>
<h:outputFormat value="#{msgs2.registrationText}">
  <f:param value="#{msgs2.firstName}"/>
  <f:param value="#{msgs2.lastName}"/>
  <f:param value="#{msgs2.emailAddress}"/>
</h:outputFormat>
…
</f:view></html>
```

All uses of #{msgs2...} *exactly* the same as in previous example. However, which properties files are loaded depends on the browser settings.

Person3.java

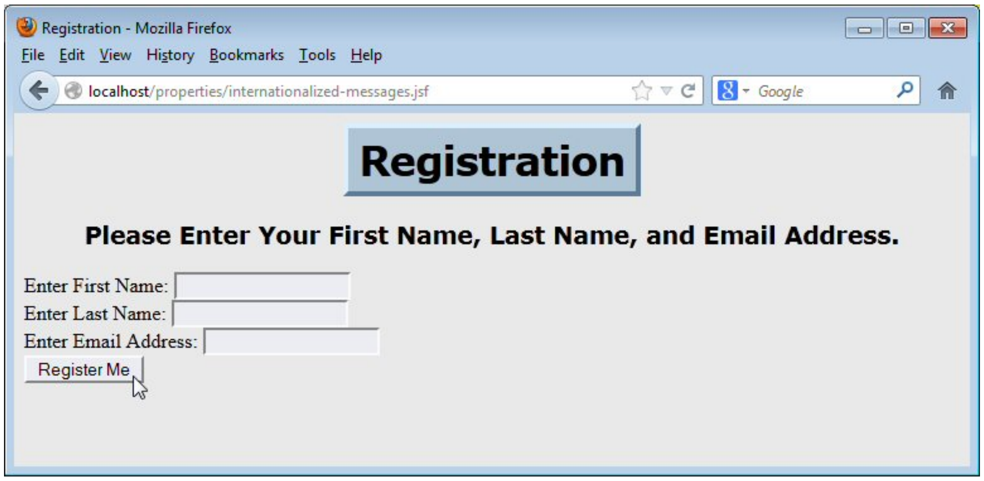
```
@ManagedBean
public class Person3 extends Person {
  public String doRegistration() {
    return("success3");
  }
}
```

success3.xhtml

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<f:view locale="#{facesContext.externalContext.requestLocale}">
<h:head><title>#{msgs2.successTitle}</title>
…
</h:head>
<h:body>
<table border="5" align="center">
  <tr><th class="title">#{msgs2.successTitle}</th></tr>
</table>
<h3>#{msgs2.successText}</h3>
<ul>
  <li>#{msgs2.firstName}: #{person3.firstName}</li>
  <li>#{msgs2.lastName}: #{person3.lastName}</li>
  <li>#{msgs2.emailAddress}: #{person3.emailAddress}</li>
</ul>
</h:body></f:view>
```

All uses of #{msgs2...} *exactly* the same as in previous example. However, which properties files are loaded depends on the browser settings.

Result (Browser Language English)


Registration - Mozilla Firefox

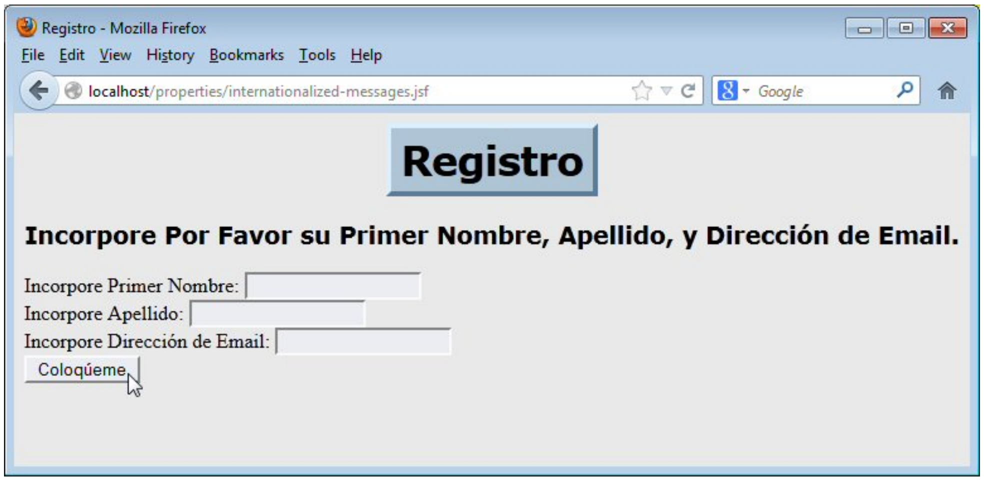
localhost/properties/internationalized-messages.jsf

Registration

Please Enter Your First Name, Last Name, and Email Address.

Enter First Name:
Enter Last Name:
Enter Email Address:

Result (Browser Language Spanish)


Registro - Mozilla Firefox

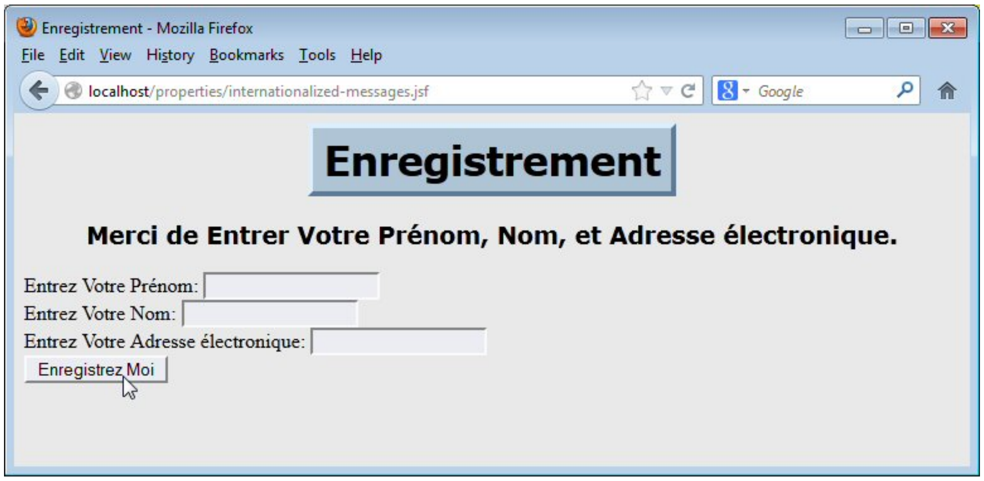
localhost/properties/internationalized-messages.jsf

Registro

Incorpore Por Favor su Primer Nombre, Apellido, y Dirección de Email.

Incorpore Primer Nombre:
Incorpore Apellido:
Incorpore Dirección de Email:

Result (Browser Language French)


Enregistrement - Mozilla Firefox

localhost/properties/internationalized-messages.jsf

Enregistrement

Merci de Entrer Votre Prénom, Nom, et Adresse électronique.

Entrez Votre Prénom:

Entrez Votre Nom:

Entrez Votre Adresse électronique:

Enregistrez Moi

Getting Properties from Database

- **Idea**
  - Instead of using properties file, it is also possible to use Java code to get properties (often from a database)
- **faces-config.xml**
  - <message-bundle>
        somepackage.SomeCustomBundle
    </message-bundle>
- **Java**
  - Extend ResourceBundle
  - Override handleGetObject and getKeys
- **Details**
  - http://stackoverflow.com/questions/9080474/messages-properties-taken-from-db

Wrap-Up

Summary

- **Deploy one or more .properties files**
  - In Eclipse, you put .properties file in "src" folder, and it gets deployed to WEB-INF/classes automatically
- **Declare with resource-bundle in faces-config**

```
<application>
  <resource-bundle>
     <base-name>someFile</base-name>
     <var>someVar</var>
  </resource-bundle> …
</application>
```

- **Output values using normal EL**
  - #{someVar.someNameFromFile} for simple values
  - h:outputFormat for parameterized values
- **Set view's locale if I18N needed**
  - Extract it from browser setting or user setting
    - We'll cover user settings in section on event handling
  - Automatically loads locale-specific resource bundle

Questions?

More info:
http://www.coreservlets.com/JSF-Tutorial/jsf2/ – JSF 2.2 tutorial
http://www.coreservlets.com/JSF-Tutorial/primefaces/ – PrimeFaces tutorial
http://courses.coreservlets.com/jsf-training.html – Customized JSF and PrimeFaces training courses
http://coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training