Presented at
360|Flex 2010, San Jose
cf.Objective 2010, Minneapolis

Building Multi-language (i18n) Flex Applications

Thomas Burleson
Senior Architect / Consultant

March 6, 2010

First some acronyms...

- i18n[Internationalization]

  entire business processes focused on software product support for other languages

  - documentation
  - translation,
  - quality assurance,
  - support,
  - etc.

- l10n [Localization]

  Adaptation of software to 1 or more specific locales and culture contexts.

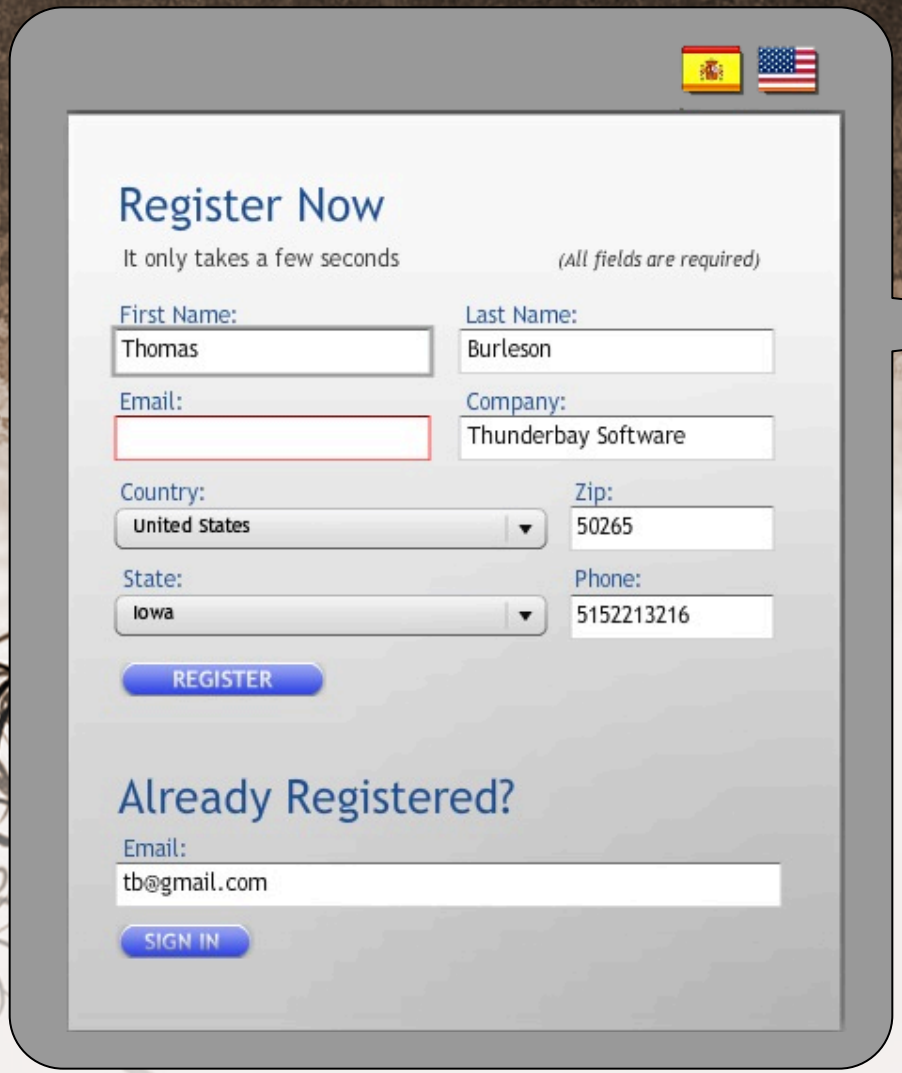
  - Text translation
  - Text Expansion and Flow
  - Changes to images, colors, and sounds
  - Changes to dynamic content and URLs
  - Changes to fonts and styles
  - etc.

English | Spanish | French | Japanese | Chinese

Multi-language?

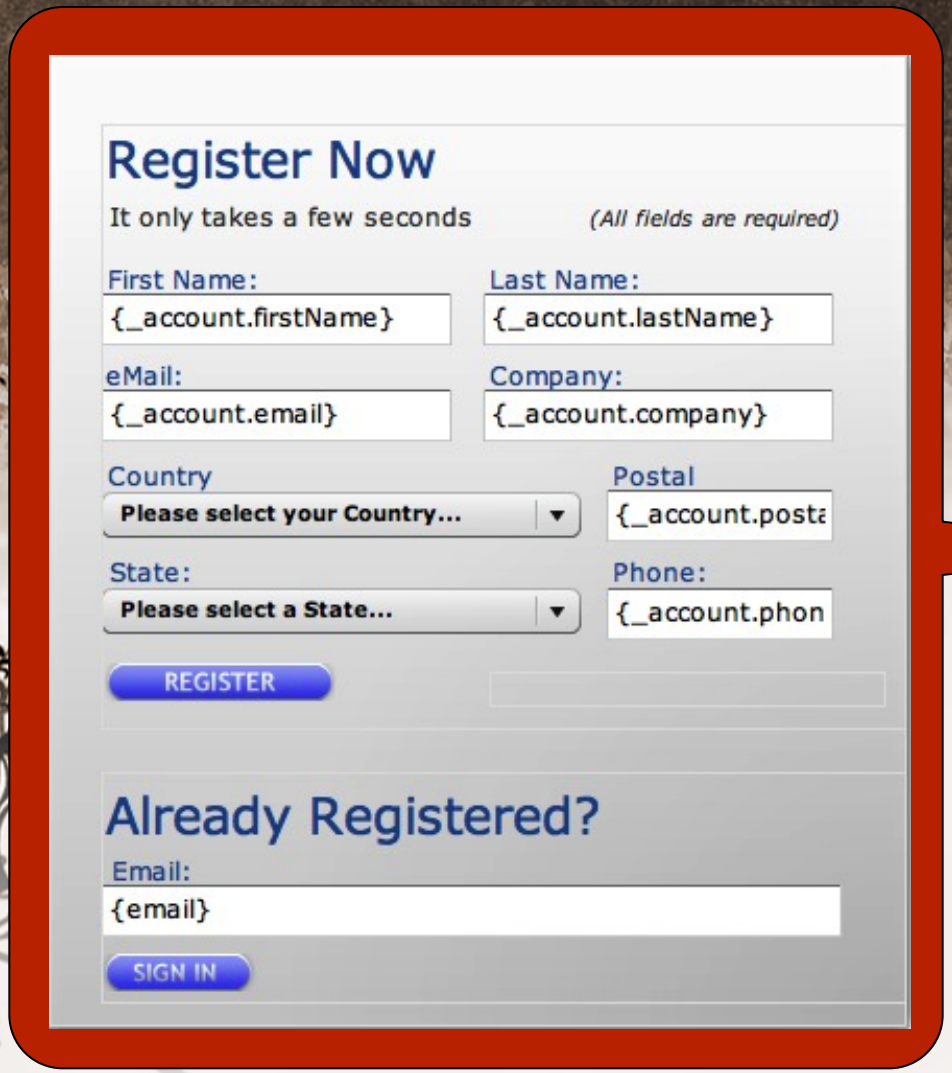
- Want to release your product in different languages?
  - Could make separate build/codebase for each language
  - Could build support for multi-language into your product.
- Could load the localized content from a...
  - text file
  - compiled resource bundle (for text, images, data, etc)
  - remote database
  - remote web service

Live Sample



- Registration Demo
- English & Spanish Support
- Startup Locale Detection

Registration Demo
before l10n



- Review App hierarchy
- Review View source code
- Review View design mode

```
#
# !!! This file must be UTF-8 Encoded in order to compile
#     properly as a ResourceBundle
#
#     In Eclipse, use option-enter keys to open properties for this file.
#     Insure the Resource > Text file encoding is set to UTF-8
#
# locale/en_US/registration.properties
# -------------------------------------------------------
register.title                    = Register Now
register.subtitle                 = It only takes a few seconds
register.allRequired              = (All fields are required)
register.firstName                = First Name:
register.lastName                 = Last Name:
register.email                    = Email:
register.company                  = Company:
register.postal                   = Zip:
register.country                  = Country:
register.state                    = State:
register.phone                    = Phone:
register.prompt.country           = Please select your Country...
register.prompt.state             = Please select a State...
register.validationError.firstName = Please enter a valid first name.
register.validationError.lastName  = Please enter a valid last name.
register.validationError.email     = A valid e-mail address is required.
register.validationError.company   = Please enter a valid company name.
register.validationError.postal    = Please enter a valid postal code.
register.validationError.phone     = Please enter a phone number between 10 to 16 digits
register.validationError.country   = Your country of residence is required.
register.validationError.state     = A US state is required.
register.invalidProfile           = Invalid User Registration

signing.invalid                   = Invalid Sign-In
signin.validationError.email      = An e-mail address (previously registered) is required.
signin.title                      = Already Registered?
signin.email                      = Email:

error.profile.notFound            = Cannot find a profile with the email address: '{0}'.
error.profile.notSaved            = A user with the email '{0}' has already registered.

request.thanks.title              = Thank you '{0}'!
request.thanks.message            = A confirmation email will be sent to '{0}'.

button.skin.register              = Embed('assets/images/btn_register.png')
button.skin.signin                = Embed('assets/images/btn_signin.png')
```

Registration Demo
with
Traditional l10n

- Create Properties files
- Compile Properties to Bundles
- Embed or Load Bundles
- Using the ResourceManager

- Review Impacts on Source
- Review Impacts on Design
- Consider Impacts on App

Registration Demo
with
Traditional l10n

```
<mx:Label id="lblTitle"     text="{resourceManager.getString('registration','register.title')}" styleName="siHeader"
<mx:Label id="lblSubTitle"  text="{resourceManager.getString('registration','register.subtitle')}" styleName="siText"
<mx:Label id="lblMsg"       text="{resourceManager.getString('registration','register.allRequired')}" styleName="siText

<!-- Each text input is horizontally aligned on column markers each at 184 pixels wide -->

<!-- 1st Row -->
    <mx:Label       id="lblFirstName"   text="{resourceManager.getString('registration','register.firstName')}"
                    styleName="siFieldLabel"   y="65" x="-2" width="170"/>
    <mx:TextInput   id="txtFirstName"   text="{_account.firstName}"     width="168"
                    focusOut="onCheckValidation(event);"  x="0" y="81" tabIndex="1"/>
    <mx:Label       id="lblLastName"    text="{resourceManager.getString('registration','register.lastName')}" x="181"
                    width="170" y="65"          styleName="siFieldLabel"  />
    <mx:TextInput   id="txtLastName"    text="{_account.lastName}"      width="168" x="183"
                    focusOut="onCheckValidation(event);"  y="81" tabIndex="2"/>

<!-- 2nd Row -->
    <mx:Label       id="lblEmail"       text="{resourceManager.getString('registration','register.email')}"
                    width="171"  x="-3" y="111"     styleName="siFieldLabel"/>
    <mx:TextInput   id="txtEmail"       text="{_account.email}"         width="168" y="127"
                    focusOut="onCheckValidation(event);"  tabIndex="3"/>
    <mx:Label       id="lblCompany"     text="{resourceManager.getString('registration','register.company')}" x="181"
                    width="170" y="111"         styleName="siFieldLabel
    <mx:TextInput   id="txtCompany"     text="{_account.company}"       width="168" y="127" x="183"
                    focusOut="onCheckValidation(event);"   tabIndex="4"/>

<!-- 3rd Row -->
    <mx:Label       id="lblZip"         text="{resourceManager.getString('registration','register.postal')}"
                    width="111" y="159"   x="240"  styleName="siFieldLabel"/>
    <mx:TextInput   id="txtPostal"      text="{_account.postal}"        width="109" x="242"
                    focusOut="onCheckValidation(event);"  y="175" tabIndex="6"/>

    <mx:Label       id="lblCountry"     text="{resourceManager.getString('registration','register.country')}" x="-2"
                    width="234" y="159"         styleName="siFieldLabel"      />
    <mx:ComboBox    id="cmbCountry"     dataProvider="{_countries}"     width="230" x="0" y="176"
                    change="onCountryChanged();"
                    prompt="{resourceManager.getString('registration','register.prompt.country')}"

    <mx:Label       id="lblState"       text="{resourceManager.getString('registration','register.state')}"
                    width="233" x="-1" y="205"  styleName="siFieldLabel"
                    alpha="{(cmbCountry.selectedItem == Countries.USA) ? 1 : .3}"/>
    <mx:ComboBox    id="cmbState"       dataProvider="{_states}"        width="230" x="0" y="222"
                    change="onStateChanged();"              selectedIndex="-1"
                    prompt="{resourceManager.getString('registration','register.prompt.state')}"

    <mx:Label       id="lblPhone"       text="{resourceManager.getString('registration','register.phone')}"
                    width="111" x="240" y="205"     styleName="siFieldLabel"/>
    <mx:TextInput   id="txtPhone"       text="{_account.phone}"         width="109" x="242" y="222"
                    restrict="0-9,.-" focusOut="onCheckValidation(event);"   tabIndex="8"/>
```

- Create Properties files
- Compile Properties to Bundles
- Embed or Load Bundles
- Using the ResourceManager

  ... change all your views!

- Review Impacts on Source
- Review Impacts on Design
- Consider Impacts on App

Registration Demo

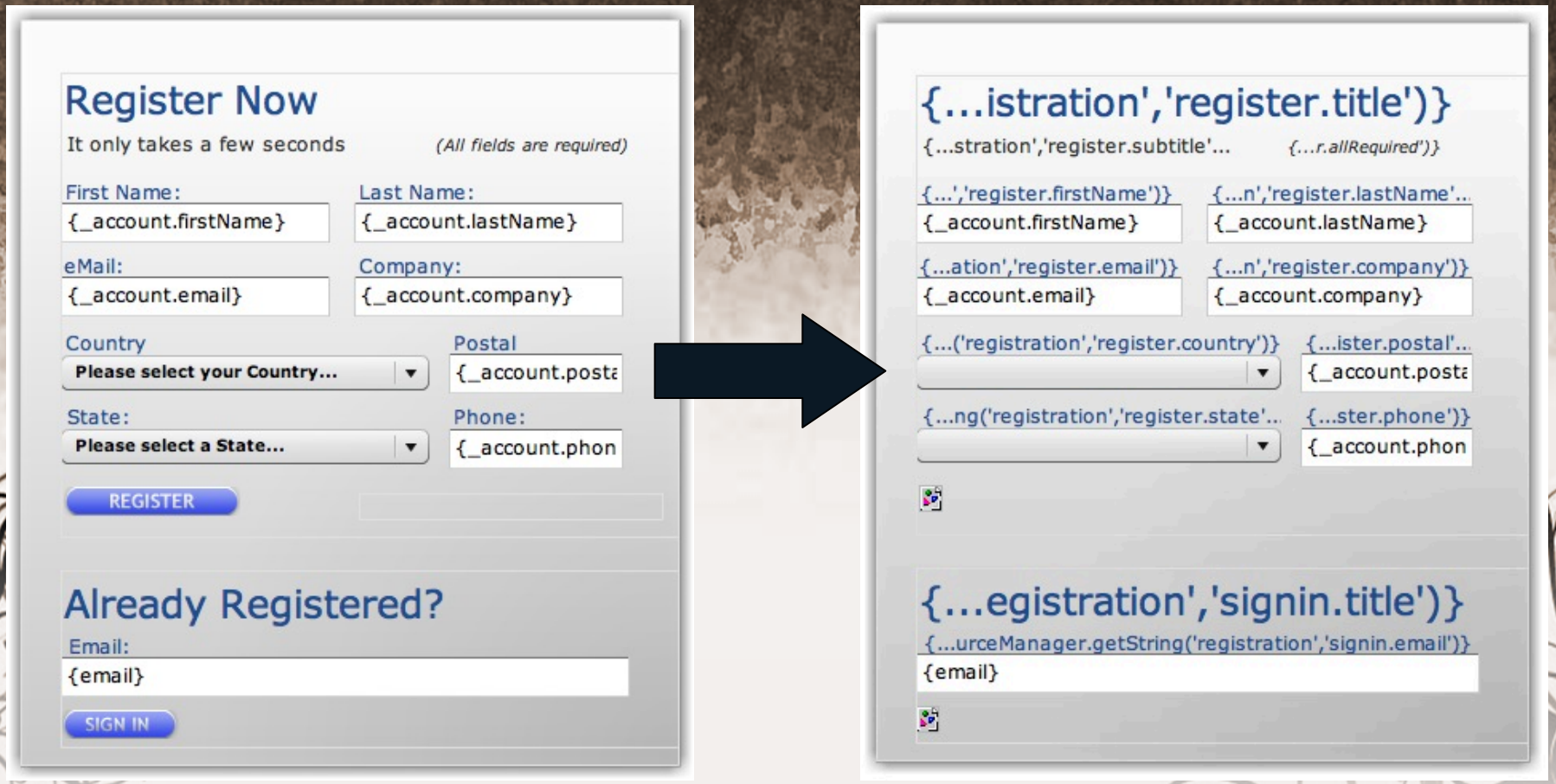
with

Traditional l10n

- Create Properties files
- Compile Properties to Bundles
- Embed or Load Bundles
- Using the ResourceManager

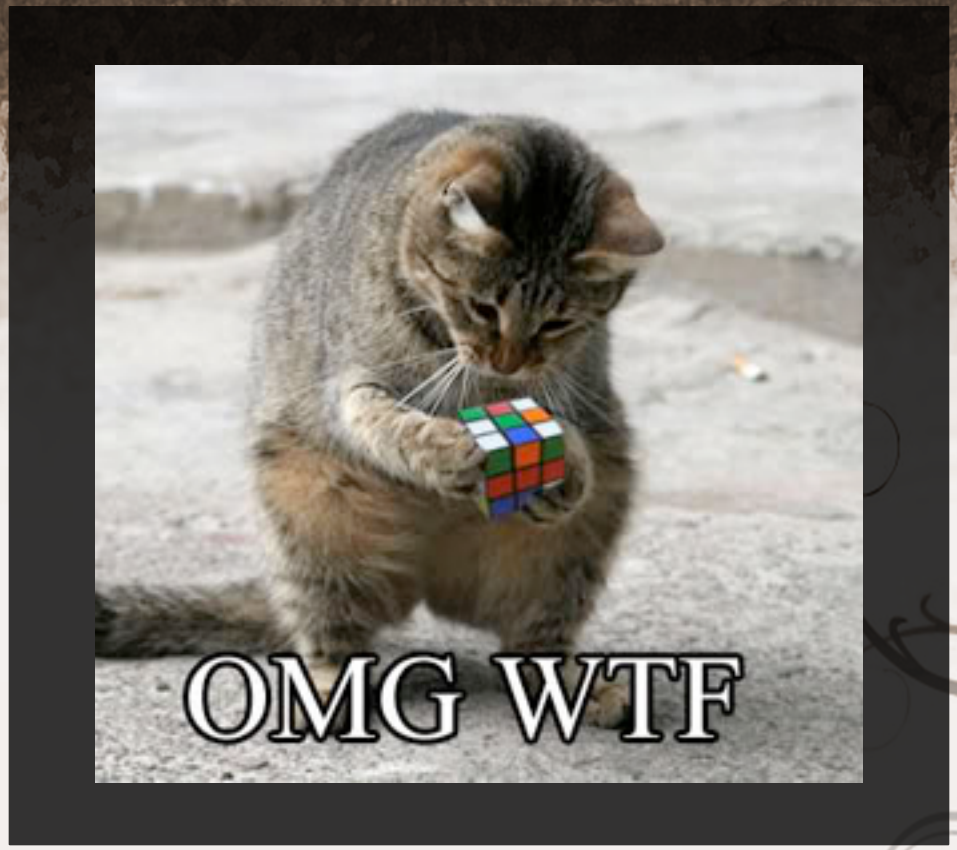
- Review Impacts on Source
- Review Impacts on Design
- Consider Impacts on App

Consider impacts...

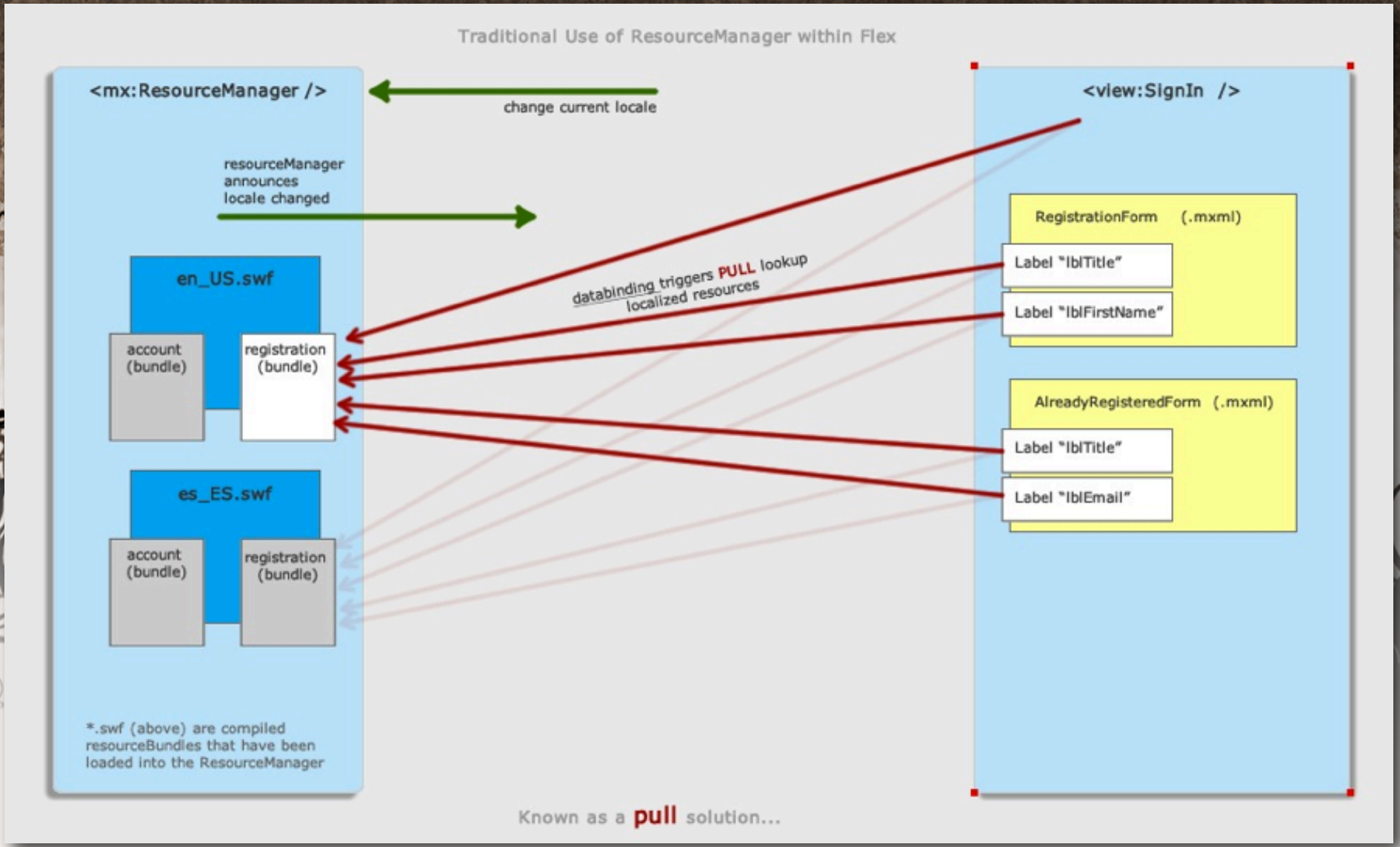


... mangled design views!

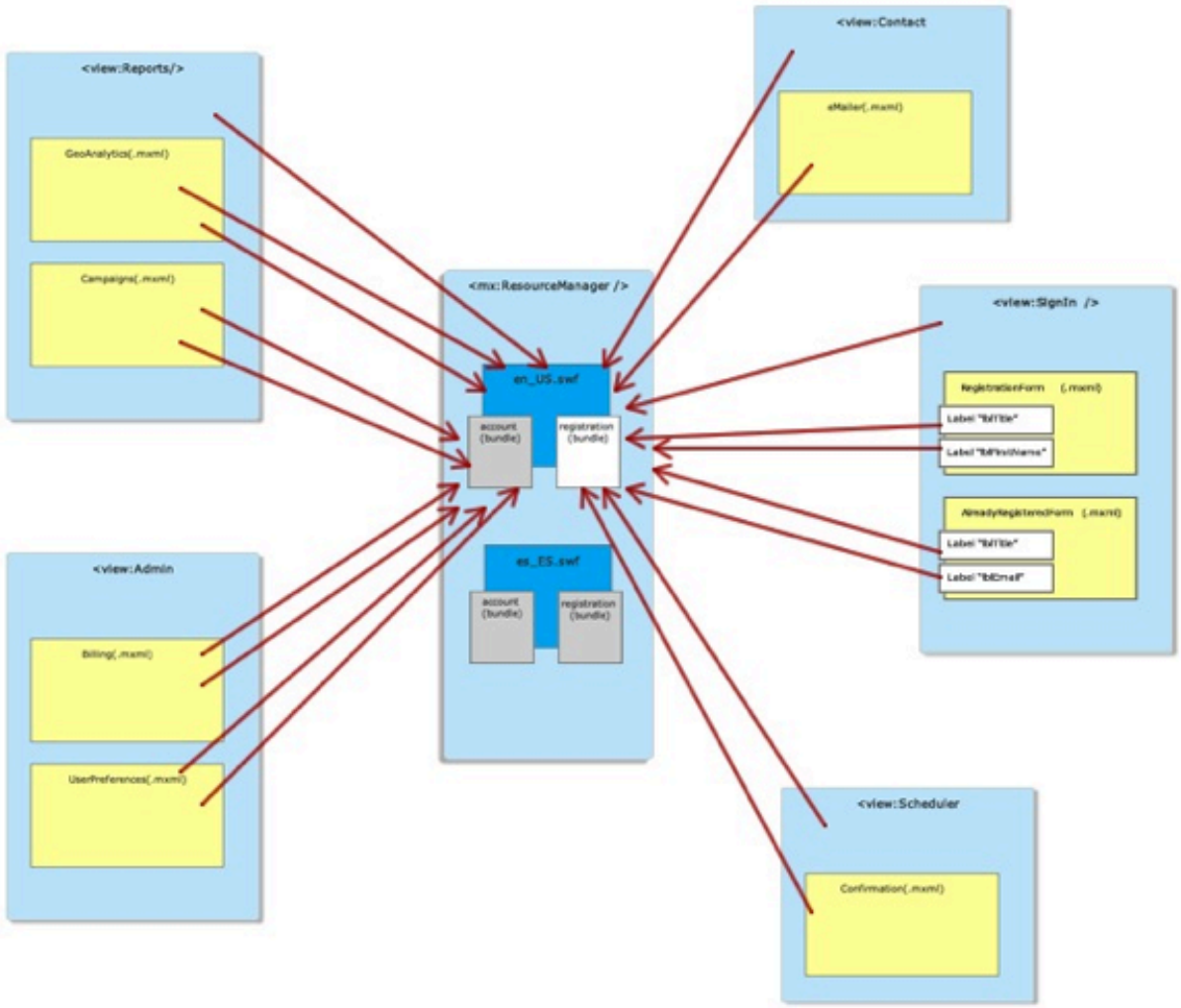
...using ResourceManager



Using ResourceManager



Using ResourceManager



Summary of Traditional Use

Every view uses ResourceManager.

Every view pulls its own localized resource.

Impossible to track which views use which resources

Source code is mangled

Design view is destroyed

How to address isuess of view states that depend upon locales?

Questions / Issues

How do you manage complexity?

How do you insure the "keys" are all correct?

How you insure that all keys are used?

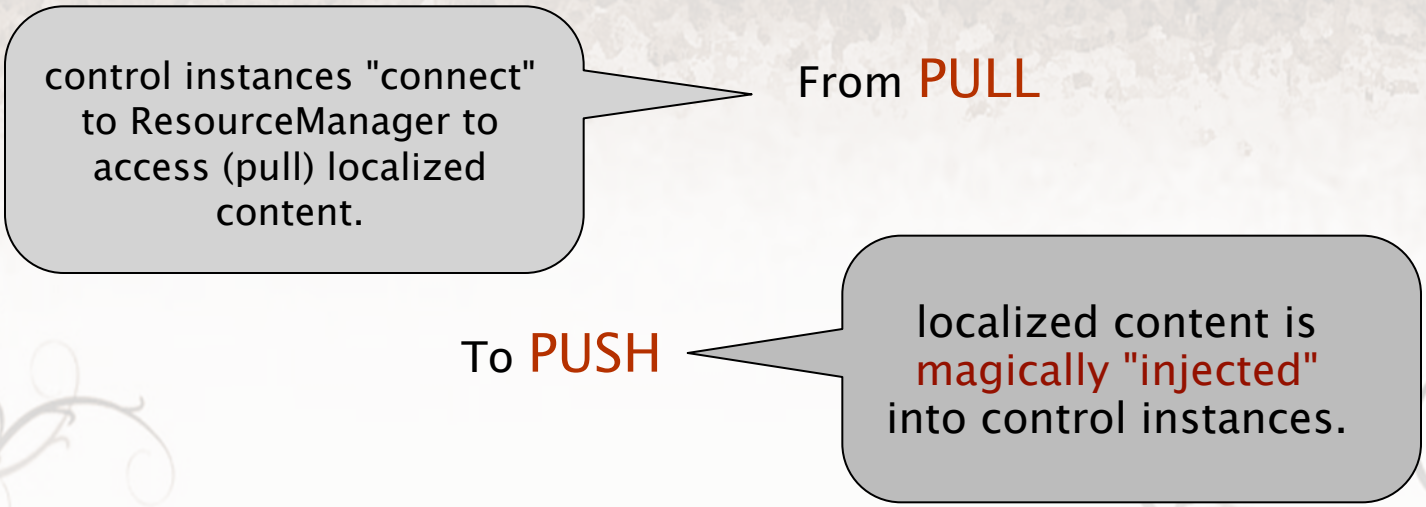
What if a key changes?

What about view states?

What about order dependencies?

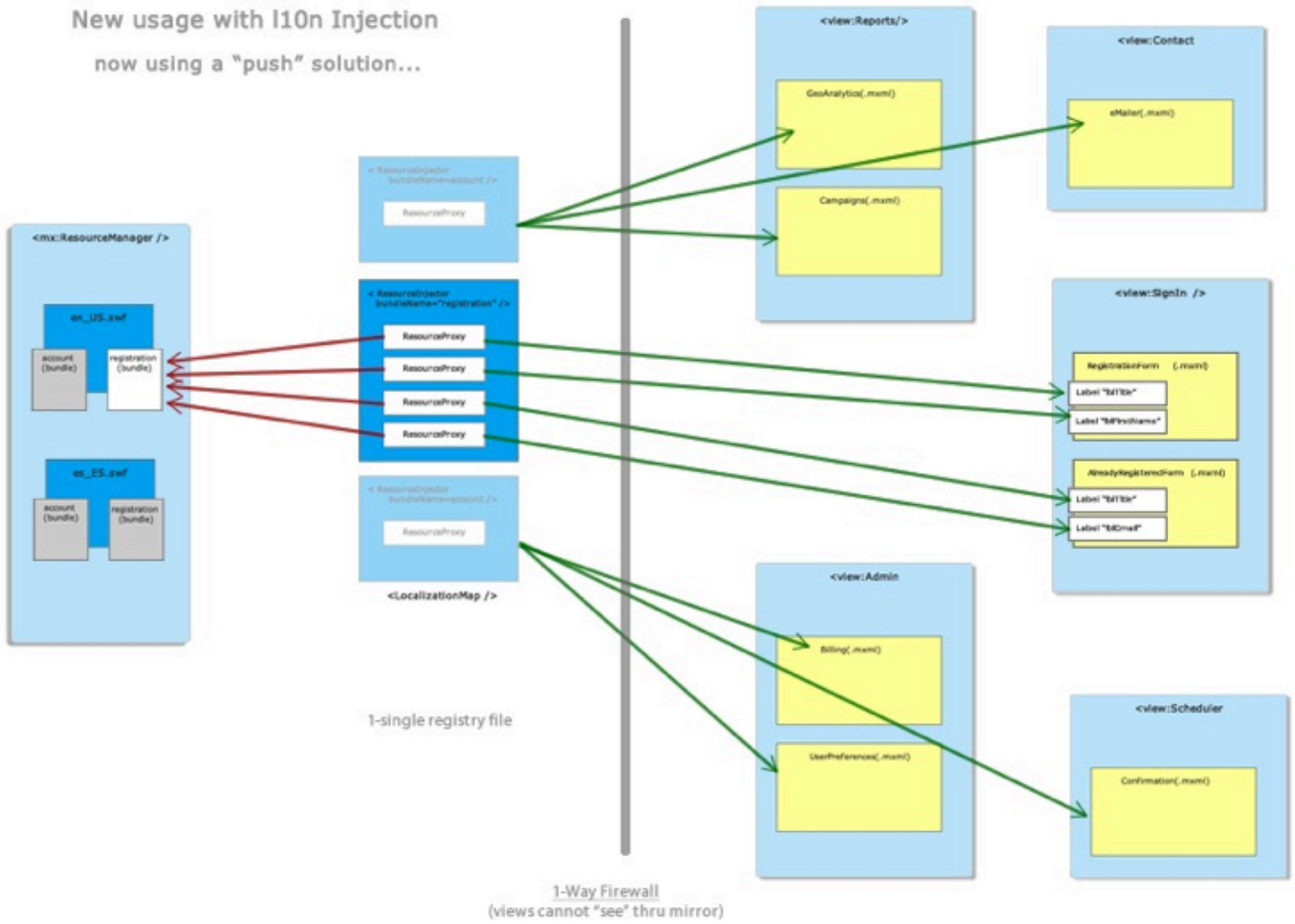
Consider IOC...

- Use Inversion of Control (IOC)

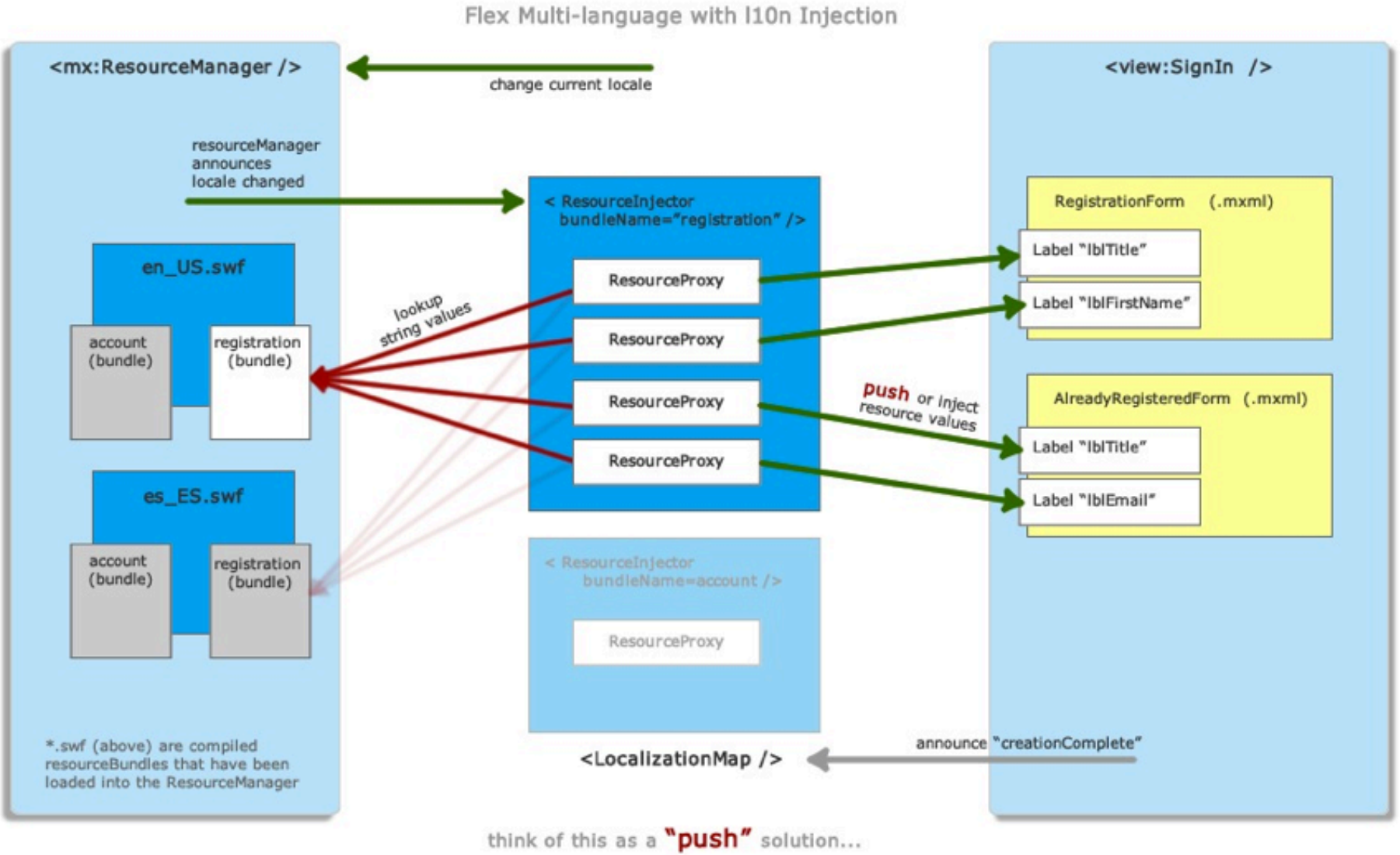


- With IOC, we can
  - Centralize all injections of localized content
  - Centralize all mappings from properties

IOC with l10n Injection



Injection with LocalizationMaps



l10n Injection

- Understanding concepts of
  - LocalizationMap
  - Resource Injectors
  - Resource Proxies
  - Smart ResourceInjectors
- Create your custom LocalizationMap

1. Inject text to registered instances
2. Inject parameterized Values
3. Inject to ALL instances
4. Inject localized bitmaps
5. Inject skins

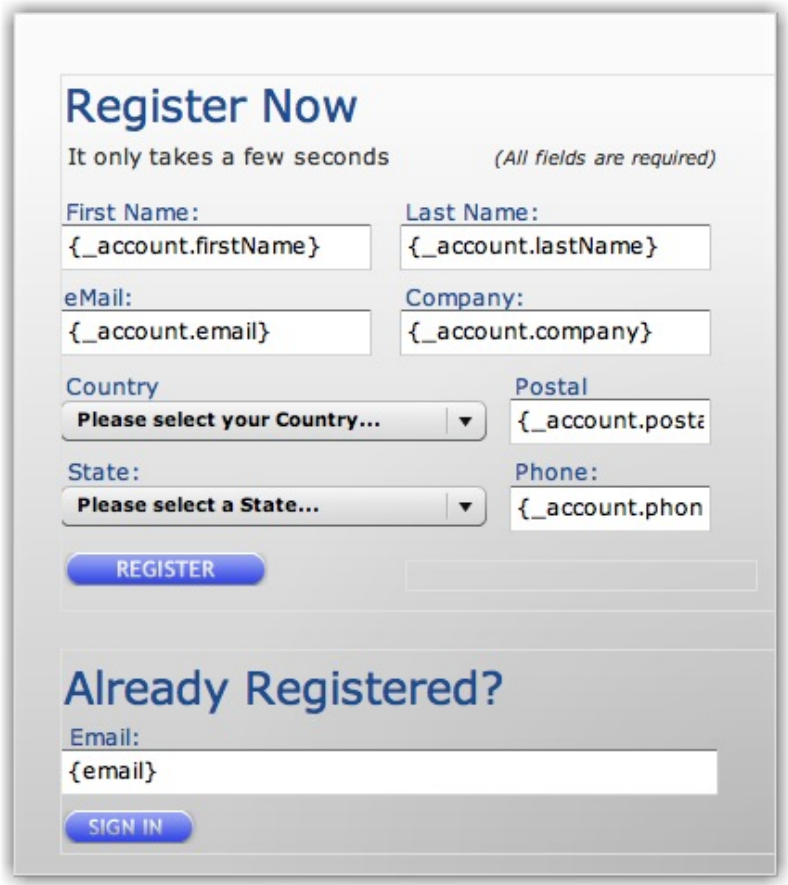
Registration Demo with l10n Injection

```
<?xml version="1.0" encoding="utf-8"?>
<map:LocaleMap  xmlns="com.asfusion.mate.l10n.injectors.*"
                xmlns:map="com.asfusion.mate.l10n.maps.*"
                xmlns:mx="http://www.adobe.com/2006/mxml"
                xmlns:injectors="http://com.asfusion.mate.l10n/"
                xmlns:mate="http://mate.asfusion.com/">

    <ResourceInjector bundleName="registration" localeChange="clearValidationErrors()">
        <ResourceProxy target="{signInPage.frmSignin}"                          property="alertTitle"
        <ResourceProxy target="{signInPage.frmSignin.validator}"                property="requiredFieldError"
        <ResourceProxy target="{signInPage.frmSignin.lblTitle}"                 property="text"
        <ResourceProxy target="{signInPage.frmSignin.lblEmail}"                 property="text"

        <ResourceProxy target="{signInPage.frmRegister}"                        property="alertTitle"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[0]}"  property="requiredFieldError"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[1]}"  property="requiredFieldError"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[2]}"  property="requiredFieldError"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[3]}"  property="requiredFieldError"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[4]}"  property="requiredFieldError"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[5]}"  property="requiredFieldError"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[6]}"  property="lowerThanMinError"
        <ResourceProxy target="{signInPage.frmRegister.registrationValidators[7]}"  property="lowerThanMinError"

        <ResourceProxy target="{signInPage.frmRegister.lblTitle}"               property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblSubTitle}"            property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblMsg}"                 property="text"

        <ResourceProxy target="{signInPage.frmRegister.lblFirstName}"           property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblLastName}"            property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblEmail}"               property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblPhone}"               property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblCompany}"             property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblZip}"                 property="text"
        <ResourceProxy target="{signInPage.frmRegister.lblState}"               property="text"
        <ResourceProxy target="{signInPage.frmRegister.cmbState}"               property="prompt"
        <ResourceProxy target="{signInPage.frmRegister.lblCountry}"             property="text"
        <ResourceProxy target="{signInPage.frmRegister.cmbCountry}"             property="prompt"
    </ResourceInjector>


    <!-- ********************************************************************************************** -->
    <!-- SmartInjectors for targeted class instantiations; where the class will have multiple instances created -->
    <!--       Here we are injecting BitmapAssets (pngs) into the <Image instance>.source properties           -->
    <!-- ********************************************************************************************** -->

    <SmartResourceInjector bundleName="registration" target="{Image}"   >
        <ResourceProxy targetID="btnRegister"   property="source"       key="button.skin.register"  type="class" />
        <ResourceProxy targetID="btnSignIn"     property="source"       key="button.skin.signin"    type="class" />
    </SmartResourceInjector>

</map:LocaleMap>
```

- Create Properties files
- Compile Properties to Bundles
- Embed or Load Bundles
- ~~Using the ResourceManager~~
- Create LocalizationMap registry

- ~~Review Impacts on View Source~~
- Review Impacts on Design
- Consider Impacts on App

Registration Demo with l10n Injection



- Create Properties files
- Compile Properties to Bundles
- Embed or Load Bundles
- ~~Using the ResourceManager~~
- Create LocalizationMap registry

- ~~Review Impacts on View Source~~
- Review Impacts on Design
- Consider Impacts on App

Registration Demo with l10n Injection

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application creationComplete="signIn.visible = true;"
               pageTitle="i18n Demo with LocalizationMaps"
               backgroundColor="#777777"
               backgroundGradientColors="[#777777,#BABABA]"
               layout="absolute"
               horizontalScrollPolicy="off" verticalScrollPolicy="off"
               xmlns:views="com.thunderbay.i18nDemo.views.*"
               xmlns:mx="http://www.adobe.com/2006/mxml" >

    <mx:Style source="locale/en_US/assets/css/styles.css" />

    <!-- Classic Cairngorm MVC structures -->
    <mvc:i18nController         />
    <model:Model id="_model"    />

    <!-- Login/Registration Form -->
    <views:SignInPage  id="signIn"      visitor="{_model.account}"
                       visible="false"  width="854" height="526"
                       horizontalCenter="0" verticalCenter="0" />

    <!-- Single tag used to provide l10n features for the entire applicat
    <locale:LocalizationMap  account="{_model.account}"
                             signInPage="{signIn}"  />

</mx:Application>
```

- Create Properties files
- Compile Properties to Bundles
- Embed or Load Bundles
- ~~Using the ResourceManager~~

- ~~Review Impacts on View Source~~
- Review Impacts on Design
- Consider Impacts on App

LocalizationMaps

- Learned how to us LocalizationMaps in your App

- When does the injection occur?

    1. When ResourceManager announces locale changes.
    2. When instances are created (if...)
    3. When parameterized values change (if...)
    4. When view states change (if...)

LocalizationMaps

- Extra considerations

  1. Dynamically loading Bundles
  2. Font Dependencies and Unicode
  3. Compiling with Unicode
  4. Dynamically loading Fonts

- Issues still remaining...

  1. Text Expansion
  2. Text Translation & Online Tools
  3. Databases vs Bundles

❖ Use ColdFusion as remote localization server:

1. RESTful Service for dynamic xml as virtual bundles
2. Performed at the business services layer
3. Completely transparent to LocalizationMaps and your app.
4. Can load dynamically compiled fonts

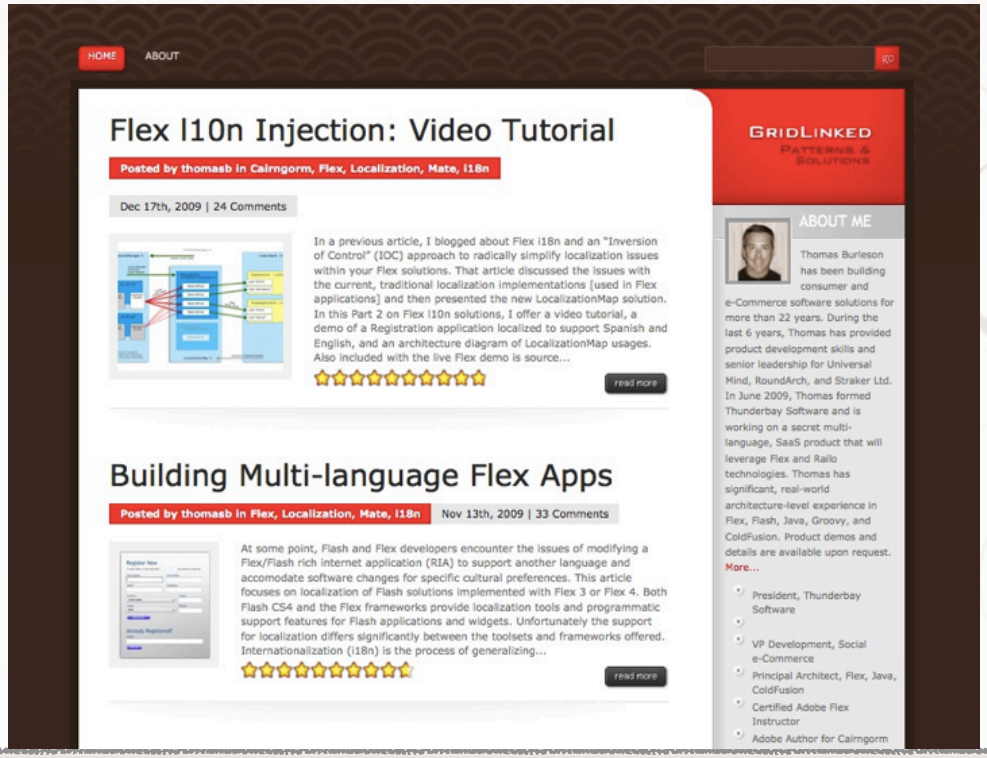
❖ Use ColdFusion for Translation services:

1. Save/load translations of properties files (key/value pairs).
2. Duplicate for other locales
3. Upload content for localized bitmaps and other assets

Open Source Framework
Adobe Flex 3 & Flex 4

Thomas Burleson
Senior Architect / Consultant

1. Source code: http://github.com/ThomasBurleson/l10n_Injection
2. Sample apps: http://github.com/ThomasBurleson/l10nInjection_Samples
3. Personal blog: http://www.gridlinked.info/
4. Intro to i18n: http://www.tinyurl.com/ykru3yp



ThomasBurleson@gmail.com