

Presented at
360|Flex
of Objective 2010, San Jose
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.

❖ **I10n** [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.

[EXPLORAR](#)

[PITTCON NOTICIAS](#)

FEATURED PRODUCTS
[EXPERIENCE THEM NOW ▾](#)



[ALIMENTACIÓN Y AGRICULTURA](#)

[condiciones de uso](#)

[directiva de privacidad](#)

[Request More Information](#)

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

Register Now
It only takes a few seconds
(All fields are required)

First Name: Thomas Last Name: Burleson
Email: Company: Thunderbay Software
Country: United States Zip: 50265
State: Iowa Phone: 5152213216

REGISTER

Already Registered?
Email: tb@gmail.com
SIGN IN

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

Registration Demo I10n

before I10n

Register Now

It only takes a few seconds

(All fields are required)

First Name:

{_account.firstName}

Last Name:

{_account.lastName}

eMail:

{_account.email}

Company:

{_account.company}

Country

Please select your Country...

Postal

{_account.postal}

State:

Please select a State...

Phone:

{_account.phone}

REGISTER

Already Registered?

Email:

{email}

SIGN IN

- ❖ 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.validationError.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 I10n

- ❖ 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 I10n

```
<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 I10n

```
{...istration','register.title')}

{...stration','register.subtitle'...      {...r.allRequired')}

{...','register.firstName')}  {...n','register.lastName'...
{_account.firstName}        {_account.lastName}

{...ation','register.email')}  {...n','register.company')}
{_account.email}           {_account.company}

{...('registration','register.country')}  {...ister.postal'...
                                |    ...
{_account.postal}          {_account.postal}

{...ng('registration','register.state'...  {...ster.phone')}
                                |    ...
{_account.state}           {_account.phone}
```

```
{...egistration','signin.title')}

{...urceManager.getString('registration','signin.email')}
{email}
```

- ❖ 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...

The diagram illustrates the impact of a design change on the underlying code. On the left, a "Register Now" form is shown with clear field labels and a "REGISTER" button. On the right, the same form is shown as a series of code snippets, where the labels have been removed, leaving only the field names and their corresponding code representations. A large black arrow points from the clean design on the left to the mangled code on the right.

Register Now
It only takes a few seconds *(All fields are required)*

First Name: Last Name:

eMail: Company:

Country Postal

Please select your Country...

State: Phone:

Please select a State...

REGISTER

Already Registered?

Email:

SIGN IN

Code Transformation:

{...istration','register.title')}
{...stration','register.subtitle'...} {....r.allRequired'})

{...','register.firstName')} {_account.firstName} {...n','register.lastName')} {_account.lastName}

{...ation','register.email')} {_account.email} {...n','register.company')} {_account.company}

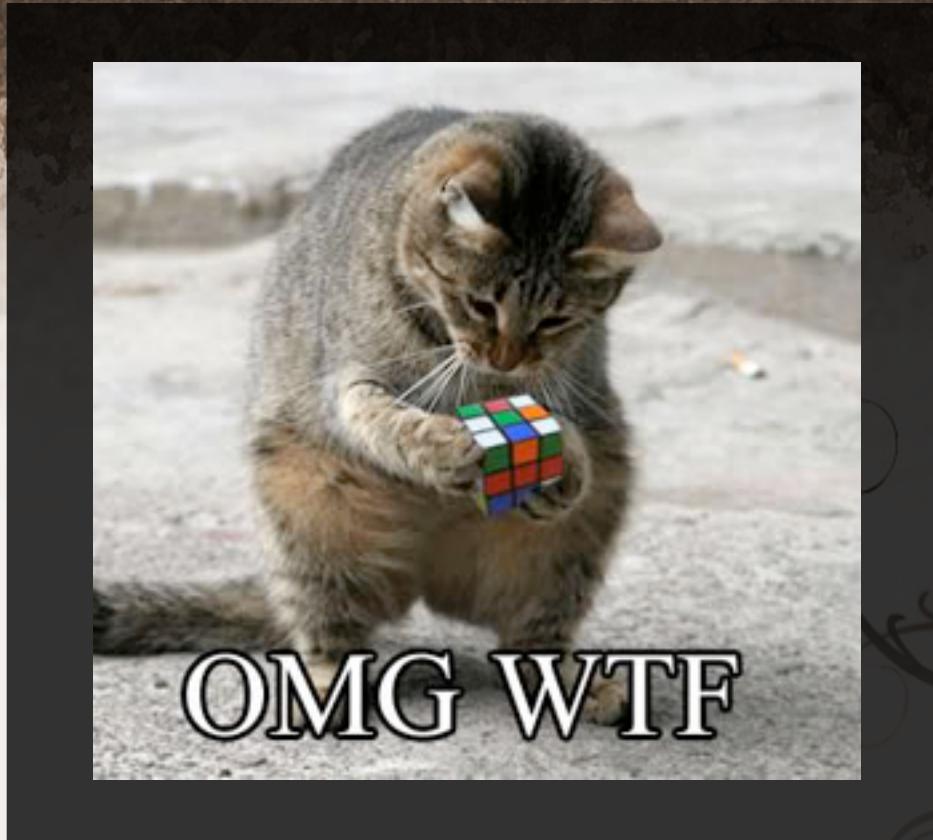
{...('registration','register.country'))} {_...ister.postal'} {_account.postal}

{...ng('registration','register.state'...)} {_...ster.phone')} {_account.phone}

{...egistration','signin.title')}
{...urceManager.getString('registration','signin.email')}{email}

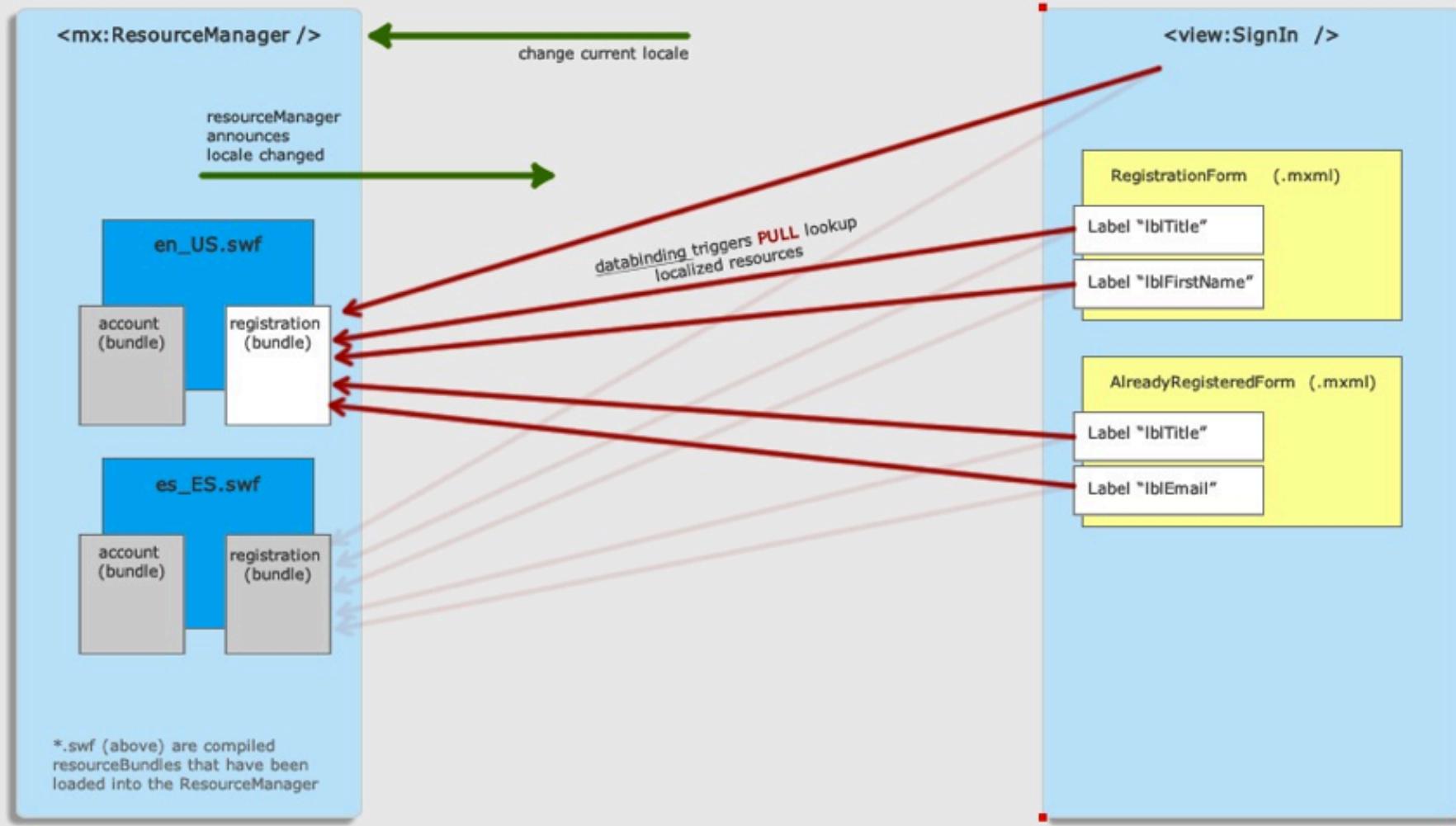
... mangled design views!

...using ResourceManager

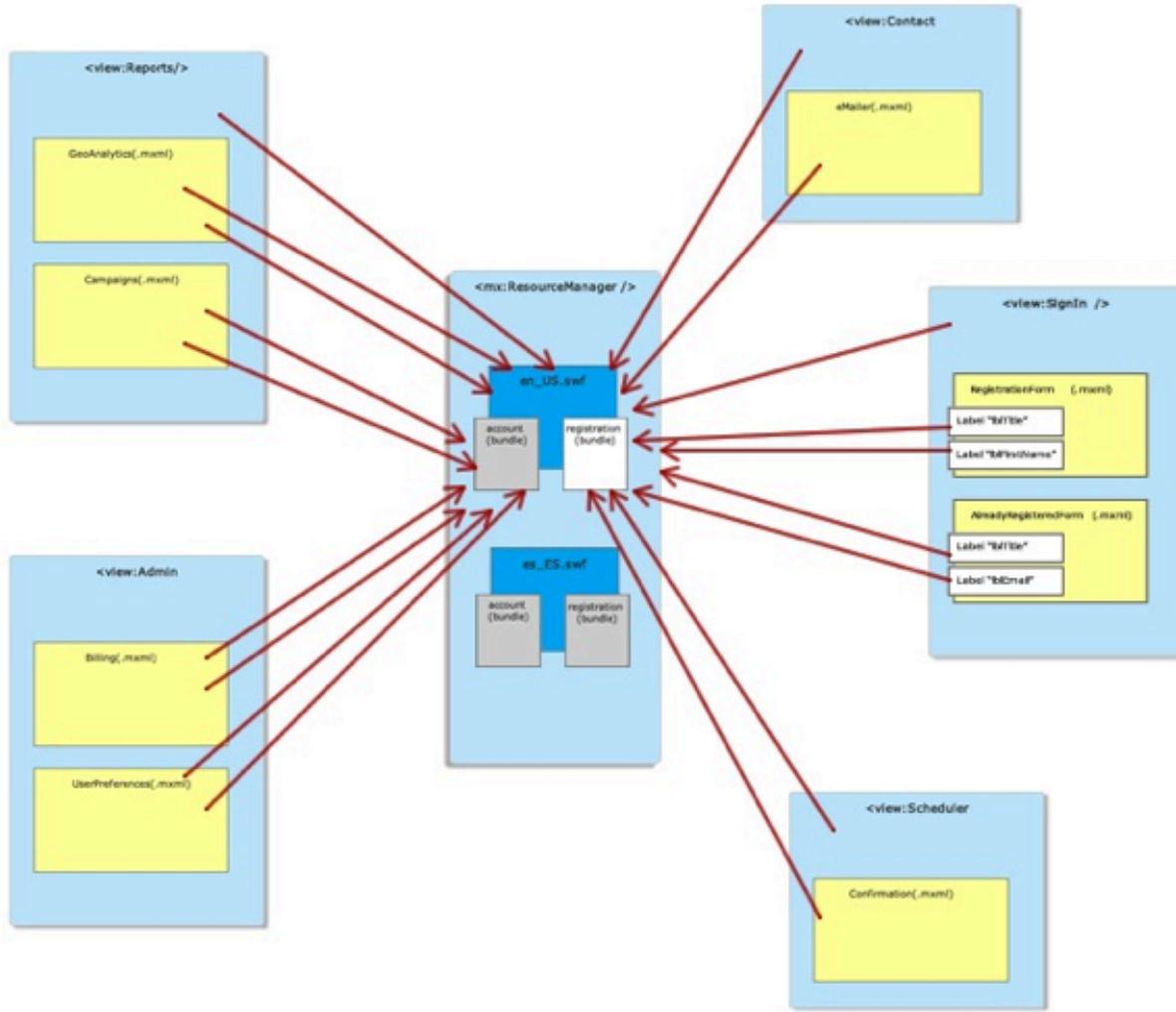


Using ResourceManager

Traditional Use of ResourceManager within Flex



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 issues 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)

control instances "connect" to ResourceManager to access (pull) localized content.

From **PULL**

To **PUSH**

localized content is magically "**injected**" into control instances.

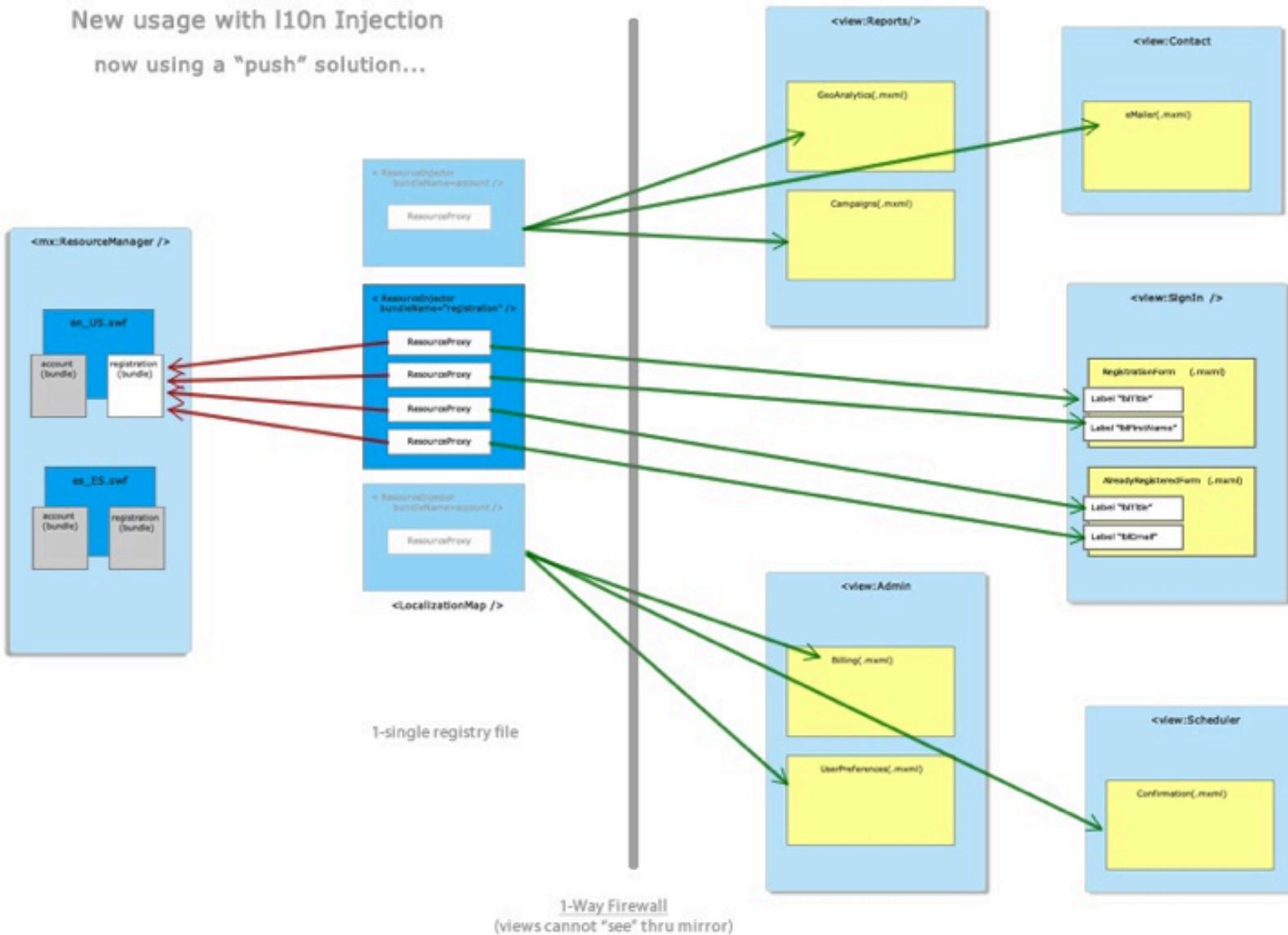
- ❖ With IOC, we can

- Centralize all injections of localized content
- Centralize all mappings from properties

IOC with I10n Injection

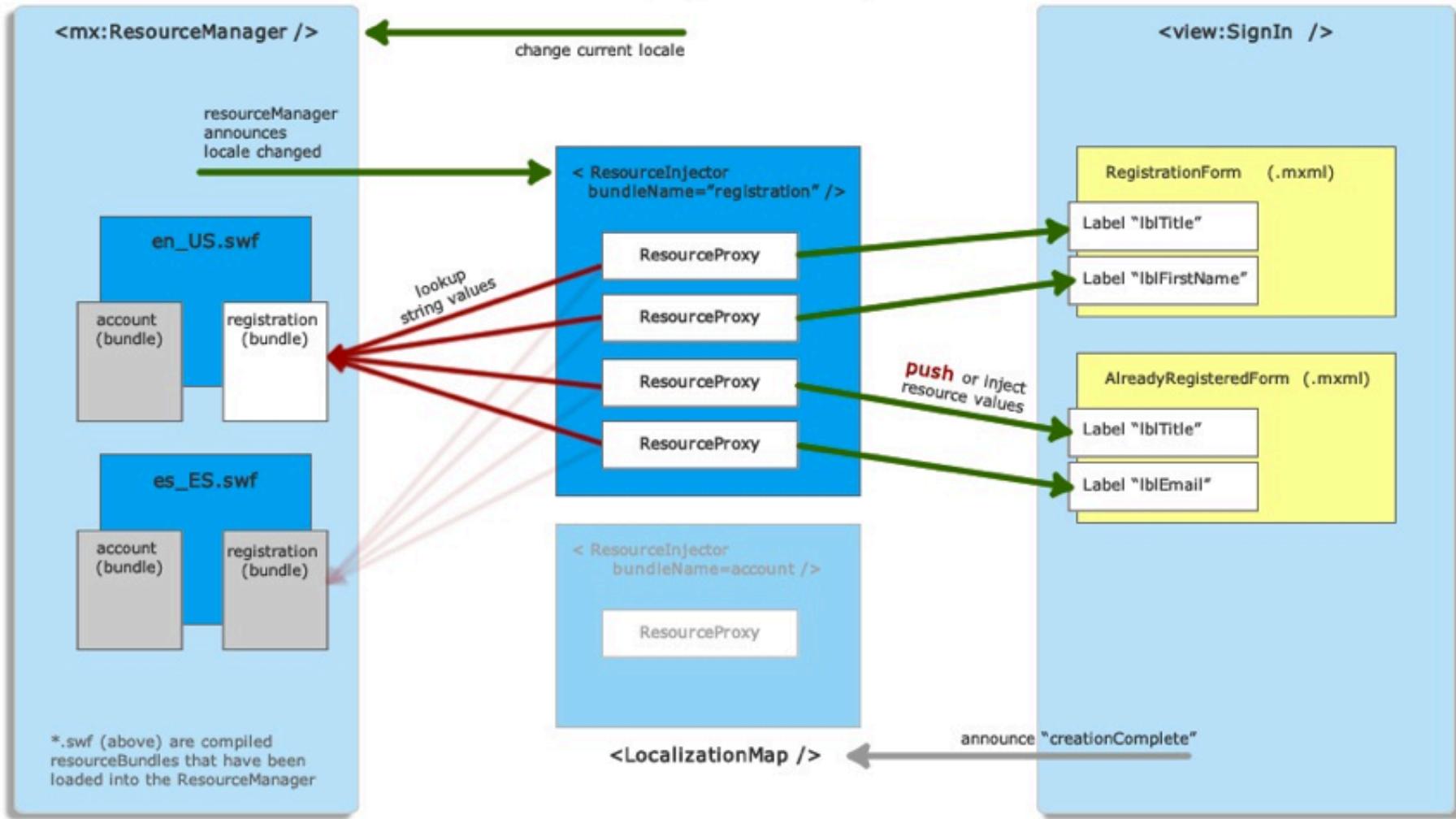
New usage with I10n Injection

now using a "push" solution...



Injection with LocalizationMaps

Flex Multi-language with I10n Injection



I10n 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 I10n Injection

```
<?xml version="1.0" encoding="utf-8"?>
<map:LocaleMap xmlns="com.asfusion.mate.l10n.injectors."
    xmlns:map="com.asfusion.mate.l10n.maps."
    xmlns:mxml="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" property="requiredFieldError" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmSignIn.validator}" property="alertTitle" property="requiredFieldError" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmSignIn.lblTitle}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmSignIn.lblEmail}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[0]}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[1]}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[2]}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[3]}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[4]}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[5]}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[6]}" property="lowerThanMinError" property="lowerThanMinError" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.registrationValidators[7]}" property="lowerThanMinError" property="lowerThanMinError" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblTitle}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblSubTitle}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblMsg}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblFirstName}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblLastName}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblEmail}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblPhone}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblCompany}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblZip}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblState}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.cmbState}" property="prompt" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.lblCountry}" property="text" property="text" property="text" property="text"/>
    <ResourceProxy target="{signInPage.frmRegister.cmbCountry}" property="text" property="text" property="text" property="text"/>
</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 I10n Injection

Register Now

It only takes a few seconds *(All fields are required)*

First Name: Last Name:

eMail: Company:

Country Postal

State: Phone:

REGISTER

Already Registered?

Email:

SIGN IN

- ❖ 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 application -->  
<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

1. Source code: http://github.com/ThomasBurleson/I10n_Injection
2. Sample apps: http://github.com/ThomasBurleson/I10nInjection_Samples
3. Personal blog: <http://www.gridlinked.info/>
4. Intro to i18n: <http://www.tinyurl.com/ykru3yp>

The screenshot shows a blog layout with a dark background and light-colored text. At the top, there are 'HOME' and 'ABOUT' buttons. Below them is a large article thumbnail for 'Flex I10n Injection: Video Tutorial', which includes a diagram of a localization architecture and a 5-star rating. Below this is another article thumbnail for 'Building Multi-language Flex Apps', featuring a screenshot of a registration form and a 5-star rating. To the right, a sidebar has a red header with 'GRIDLINKED' and 'PATTERNS & SOLUTIONS'. The 'ABOUT ME' section contains a photo of Thomas Burleson, his name, and a detailed biography about his experience in software development and product management.

Flex I10n Injection: Video Tutorial
Posted by thomasb in Cairngorm, Flex, Localization, Mate, I18n | Dec 17th, 2009 | 24 Comments

In a previous article, I blogged about Flex I18n and an "Inversion of Control" (IOC) approach to radically simplify localization issues within your Flex solutions. That article discussed the issues with the current, traditional localization implementations [used in Flex applications] and then presented the new LocalizationMap solution. In this Part 2 on Flex I10n solutions, I offer a video tutorial, a demo of a Registration application localized to support Spanish and English, and an architecture diagram of LocalizationMap usages. Also included with the live Flex demo is source...

Building Multi-language Flex Apps
Posted by thomasb in Flex, Localization, Mate, I18n | Nov 13th, 2009 | 33 Comments

At some point, Flash and Flex developers encounter the issues of modifying a Flex/Flash rich internet application (RIA) to support another language and accommodate software changes for specific cultural preferences. This article focuses on localization of Flash solutions implemented with Flex 3 or Flex 4. Both Flash CS4 and the Flex frameworks provide localization tools and programmatic support features for Flash applications and widgets. Unfortunately the support for localization differs significantly between the toolsets and frameworks offered. Internationalization (I18n) is the process of generalizing...

GRIDLINKED
PATTERNS &
SOLUTIONS

ABOUT ME
Thomas Burleson has been building consumer and e-Commerce software solutions for more than 22 years. During the last 6 years, Thomas has provided product development skills and senior leadership for Universal Mind, RoundArch, and Straker Ltd. In June 2009, Thomas formed Thunderbay Software and is working on a secret multi-language, SaaS product that will leverage Flex and Railo technologies. Thomas has significant, real-world architecture-level experience in Flex, Flash, Java, Groovy, and ColdFusion. Product demos and details are available upon request.
[More...](#)

- President, Thunderbay Software
- VP Development, Social e-Commerce
- Principal Architect, Flex, Java, ColdFusion
- Certified Adobe Flex Instructor
- Adobe Author for Cairngorm