Filthy Rich Portlets with ICEfaces and Liferay

Neil Griffin
Software Architect
Liferay, Inc.

Overview

- Portals and Portlets
- Liferay Portal
- JSF Portlets
- ICEfaces Portlets
- Standard Inter-Portlet Communication
- Ajax Push Inter-Portlet Communication

LIFERAY
Enterprise. Open Source. For Life.

What is a Portal?

- A **portal** is a framework for creating websites that aggregate different types of content and applications
- Portals are typically referred to as **portlet containers**

LIFERAY
Enterprise. Open Source. For Life.

What is a Portlet?

- A **portlet** is a region of a portal page that contains content and/or application functionality
- With respect to Java EE, a portlet is deployed as Web Application Archive (WAR) and requires a descriptor named **portlet.xml**

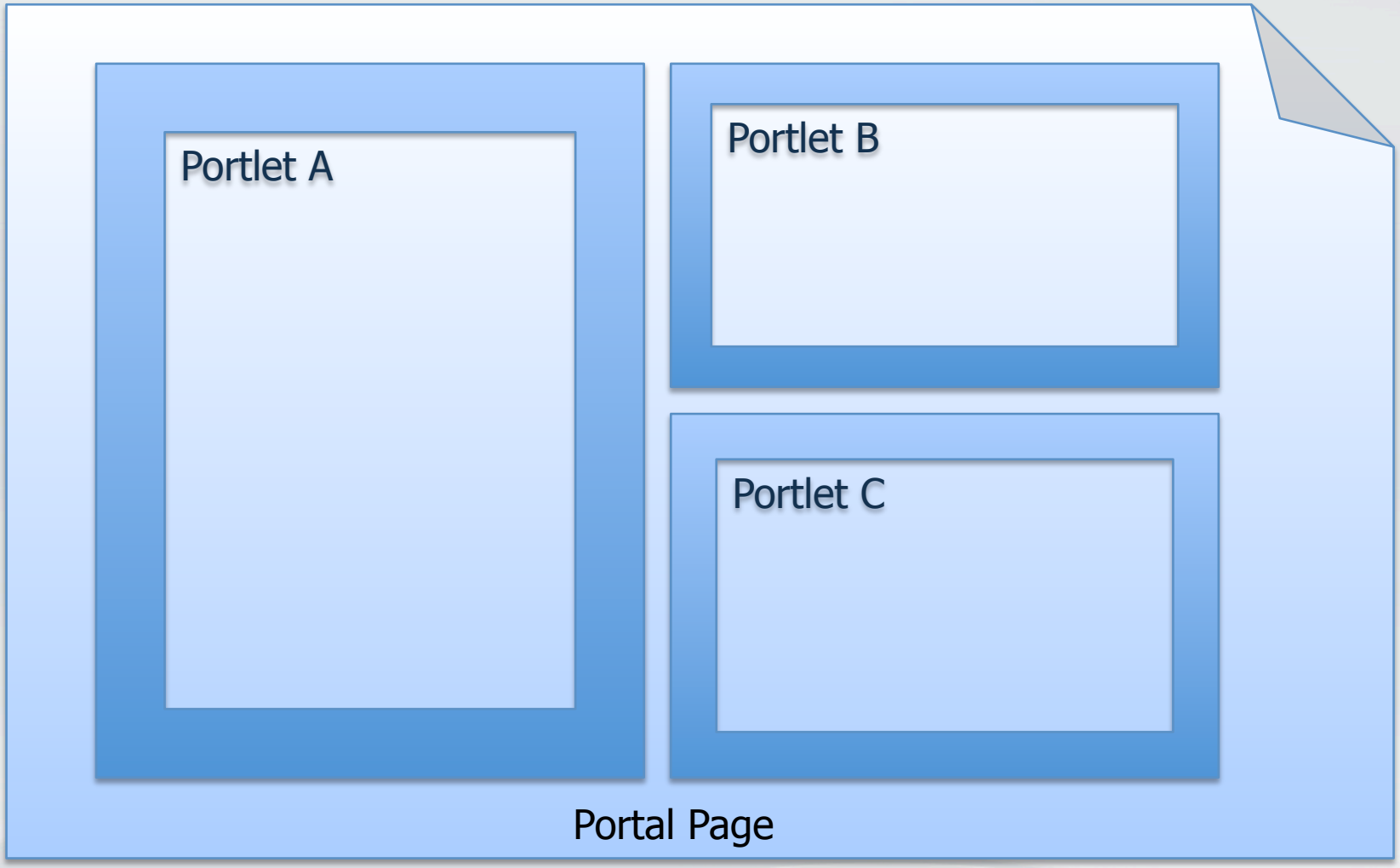
LIFERAY
Enterprise. Open Source. For Life.

Portal Pages

- Like any website, portal sites are made up of pages
- Portal pages contain one or more portlets
- Portlets can be combined on a portal page to create a composite application

LIFERAY
Enterprise. Open Source. For Life.

Portal Page Illustration



LIFERAY
Enterprise. Open Source. For Life.

Portlet Standards

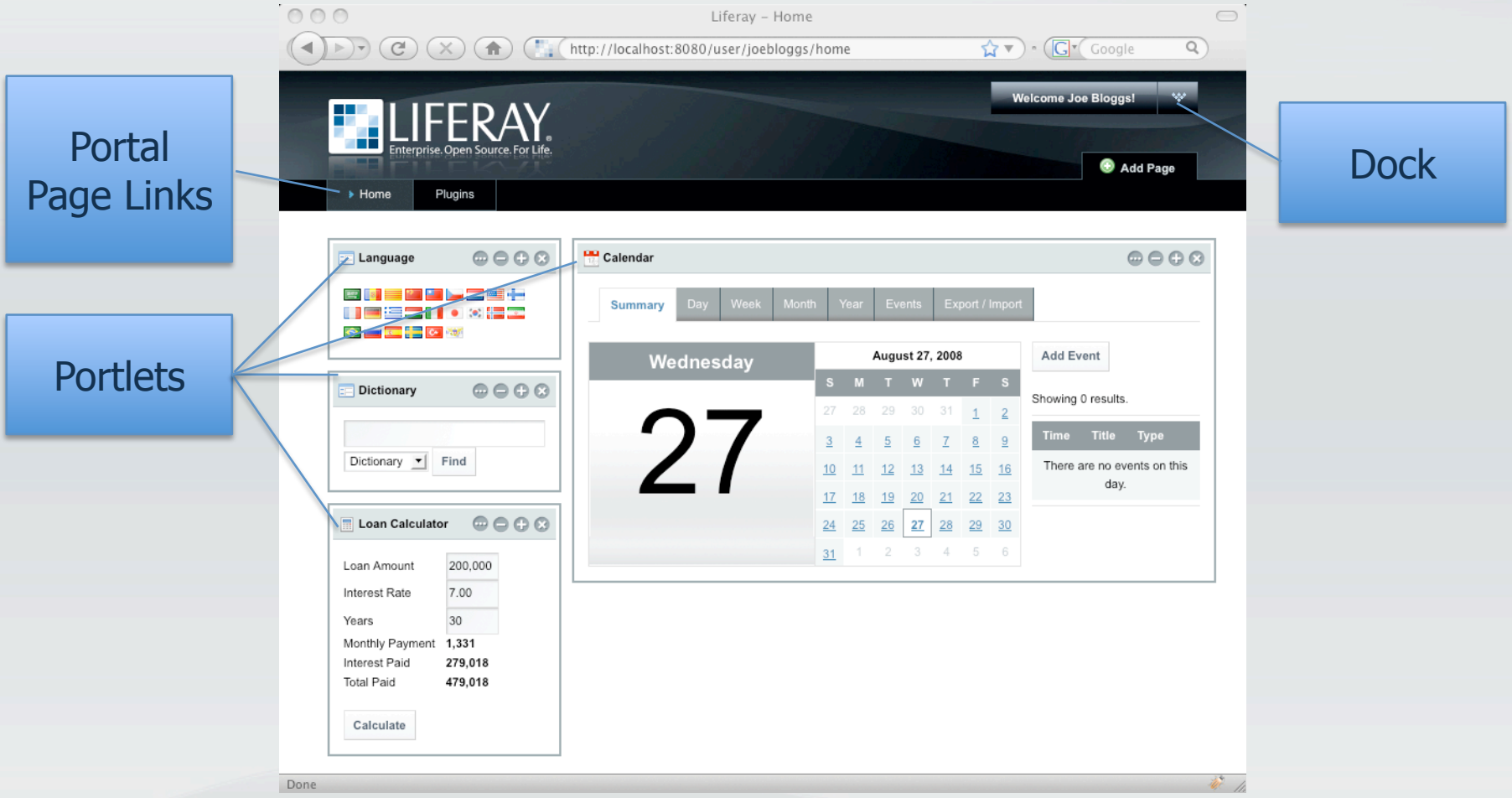
- JSR-168 (Portlet 1.0)
  - Released on 10/27/2003
- JSR-286 (Portlet 2.0)
  - Released on 6/12/2008

LIFERAY
Enterprise. Open Source. For Life.

Liferay Portal

- Liferay Portal is an **open source portlet container** built with Java technology
- Liferay **portlets** can be built with a variety of technologies, including:
  - Java
  - JSP
  - JSF
  - Struts
  - Tapestry
  - Javascript
  - PHP
  - Python
  - Ruby

Liferay Portal Page Screenshot



LIFERAY
Enterprise. Open Source. For Life.

Liferay Portal Features

- Standards compliant portlet container
- Ships with 60+ out-of-the-box portlets
- Built-in Content Management System (CMS)
- Built-in social networking portlets:
  - Friends, Message Forums, Shared Calendar, Wiki, Blogs
- Extensible with custom portlets

LIFERAY
Enterprise. Open Source. For Life.

JSF Portlets

- JSR-127 (JSF 1.1) specification was designed with JSR-168 (Portlet 1.0) in mind
- Because of this, JSF web applications can typically run as portlets with little to no modification
- Liferay was one of the first portal vendors to provide support for JSF portlets back in May, 2005

LIFERAY
Enterprise. Open Source. For Life.

JSF Portlet Bridge

- JSF webapps require a bridge in order to be deployed as portlets
- Liferay currently supports two bridges:
  - Sun OpenPortal JSF-Portlet Bridge: **jsf-portlet.jar**
    - Sun RI JSF 1.1 and 1.2
  - MyFacesGenericPortlet: **myfaces-impl.jar**
    - MyFaces RI 1.1 only
- JSR-301 is defining a standard portlet bridge API for JSF portlets

JSF Portlet Bridge (Cont.)

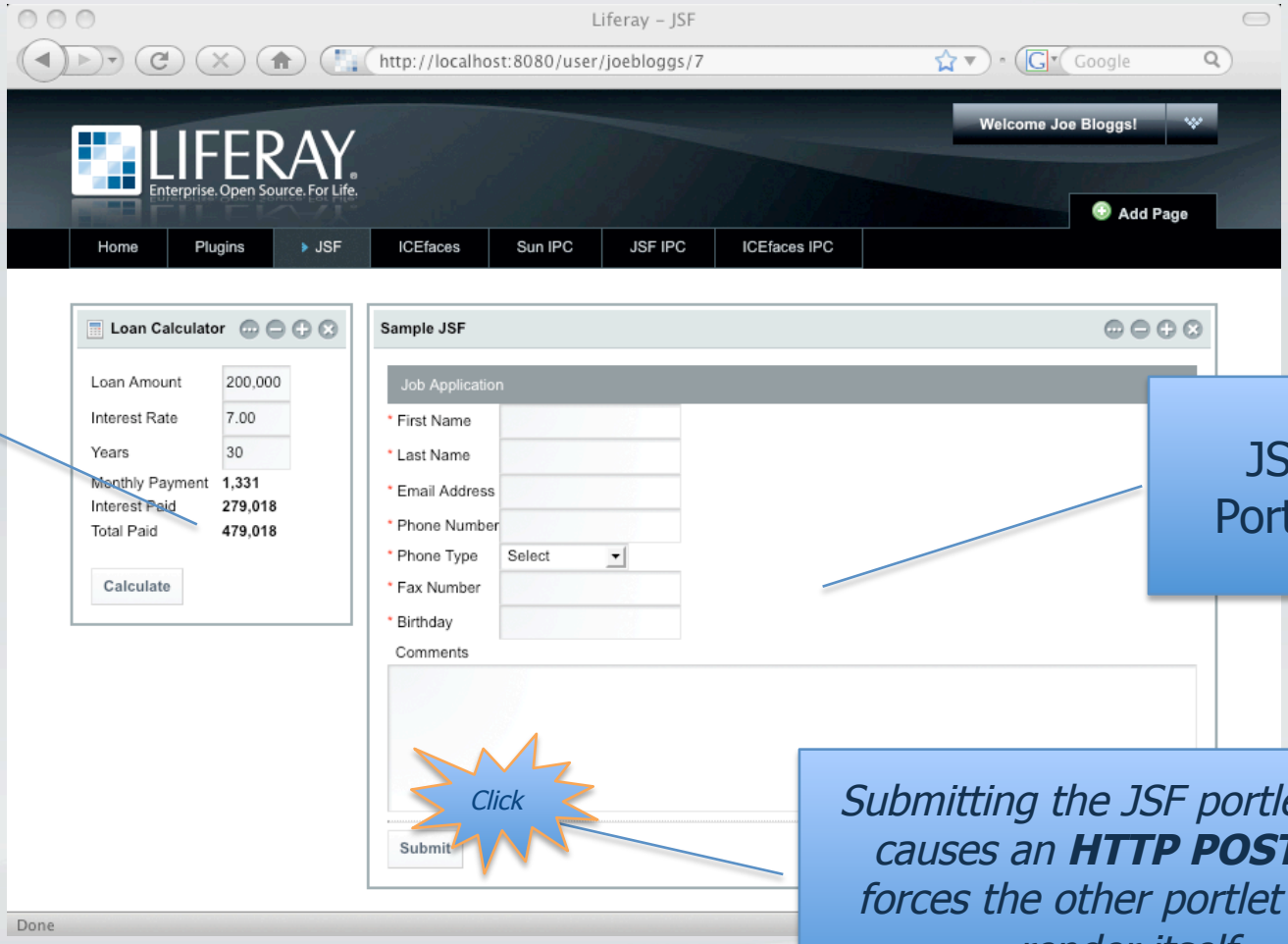
```
<!-- Sample fragment of markup that shows how to specify the -->
<!-- Sun OpenPortal JSF Portlet Bridge in portlet.xml -->
<portlet>
   <portlet-name>sample_jsf</portlet-name>
   <portlet-class>com.sun.faces.portlet.FacesPortlet</portlet-class>
   <init-param>
     <name>com.sun.faces.portlet.INIT_VIEW</name>
     <value>/xhtml/applicantForm.xhtml</value>
   </init-param>
   <init-param>
     <name>com.sun.faces.portlet.INIT_EDIT</name>
     <value>/xhtml/edit.xhtml</value>
   </init-param>
   <init-param>
     <name>com.sun.faces.portlet.INIT_HELP</name>
     <value>/xhtml/help.xhtml</value>
   </init-param>
   ...
</portlet>
```

LIFERAY
Enterprise. Open Source. For Life.

Portlet Form Submission

- Although portal pages can contain multiple portlets, only one portlet at a time can participate in form submission
  - Form submission in Portlet A causes Portlet B, Portlet C, … to re-render themselves
- Portlet form submission can cause a **disruptive end-user experience**

Demo #1 – JSF Portlet



JSP JavaScript Portlet

JSF Portlet

*Submitting the JSF portlet form causes an **HTTP POST** and forces the other portlet to re-render itself*

LIFERAY
Enterprise. Open Source. For Life.

ICEfaces to the Rescue!

ICEfaces

- ICEfaces is an **open source Ajax extension** to JSF
  - Ajax application framework
  - Robust suite of Ajax-enabled JSF UI components
- ICEfaces enables Java EE developers to easily create and deploy thin-client rich Internet applications (RIA)

LIFERAY
Enterprise. Open Source. For Life.

ICEfaces Portlets

- Liferay and ICEsoft have a partnership in place in order to support **ICEfaces portlets**

ICEfaces Portlets (Cont.)

- Portlets built with ICEfaces **never perform an HTTP post** – instead, form submission is done via Ajax
- Because of this feature, portlets built with ICEfaces **don't disturb other portlets** on the same portal page
- The end result is a rich UI that does not disrupt the end-user experience

ICEfaces Portlet Bridge

```
<!-- Sample fragment of markup that shows how to specify the -->
<!-- ICEfaces Portlet Bridge in the portlet.xml file -->
<portlet>
    <portlet-name>sample_icefaces</portlet-name>
    <portlet-class>com.icesoft.faces.webapp.http.portlet.MainPortlet</portlet-class>
    <init-param>
       <name>com.icesoft.faces.VIEW</name>
       <value>/xhtml/applicantForm.iface</value>
    </init-param>
    <init-param>
       <name>com.icesoft.faces.EDIT</name>
       <value>/xhtml/edit.iface</value>
    </init-param>
    <init-param>
       <name>com.icesoft.faces.HELP</name>
       <value>/xhtml/help.iface</value>
    </init-param>
    ...
</portlet>
```

ICEfaces Extended Request Scope

- In a normal JSF webapp/portlet, request scope is very short-lived
- ICEfaces **Extended Request Scope** is longer in duration:
  - Starts when a JSF view is first requested
  - Terminates when any of the following occur:
    - Navigation to a different JSF view
    - ICEfaces Ajax connection timeout
    - Browser is dismissed by the user

Ext. Request Scope (Cont.)

- ICEfaces Extended Request Scope is very similar to JSF 2.0 "View Scope"
- The scope is also a **great match for portlets**, particularly those that do not participate in navigation from one JSF view to another

LIFERAY
Enterprise. Open Source. For Life.

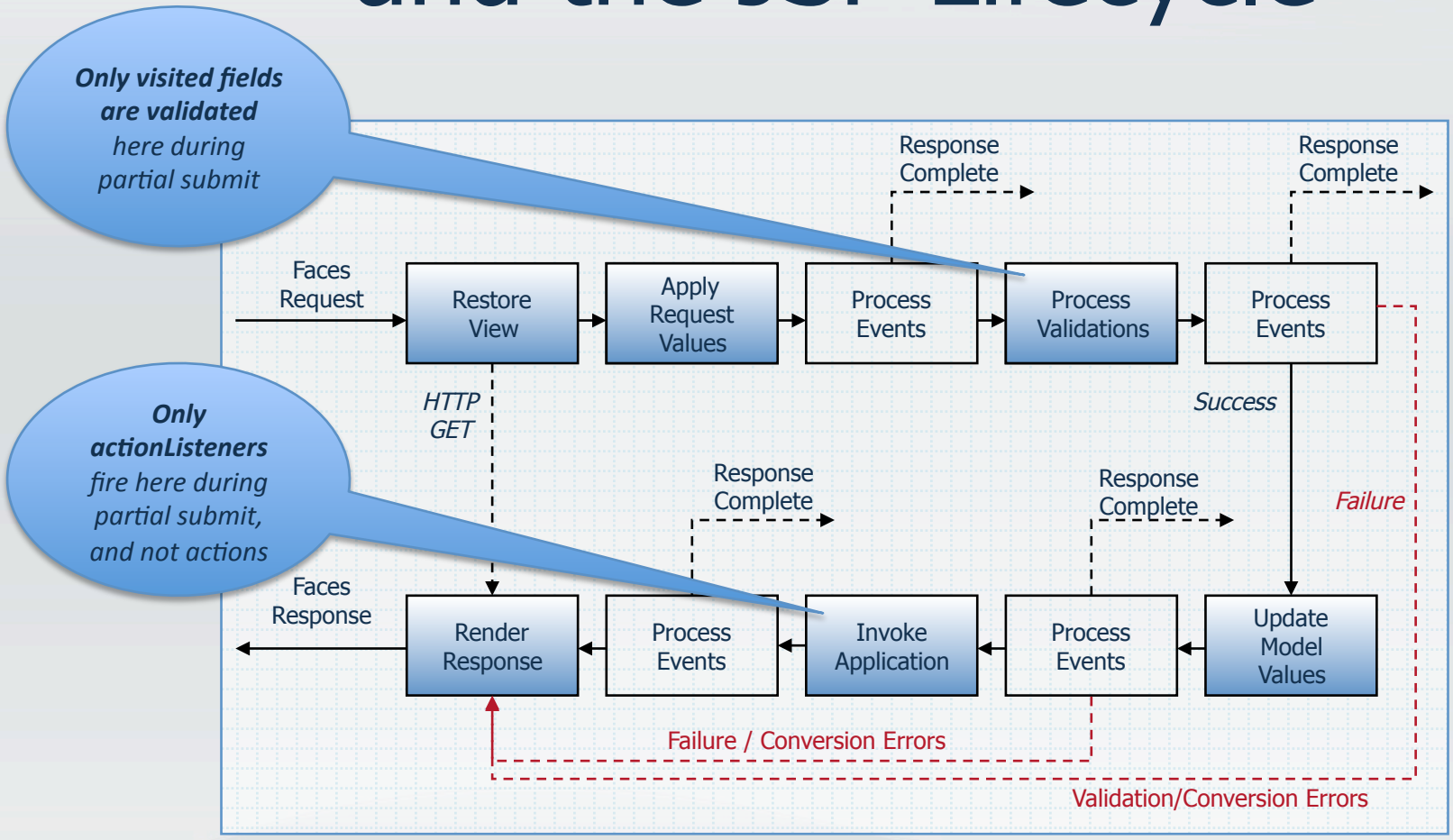
ICEfaces Partial Submit

- When the user presses the tab key in order to move from one field to another, the **`onblur`** JavaScript event is triggered
- When this occurs on an ICEfaces component with **`partialSubmit="true"`** the form is submitted via Ajax

Partial Submit (Cont.)

- During partial submit:
  - ICEfaces will invoke the JSF lifecycle
  - Form submission is "full" in the sense that all editable fields in the form are serialized and submitted
  - Form submission is "partial" in the sense that the form is only **partially validated**, meaning that only fields that have been visited by the user will undergo validation

Partial Submit and the JSF Lifecycle



LIFERAY
Enterprise. Open Source. For Life.

Direct2DOM Rendering

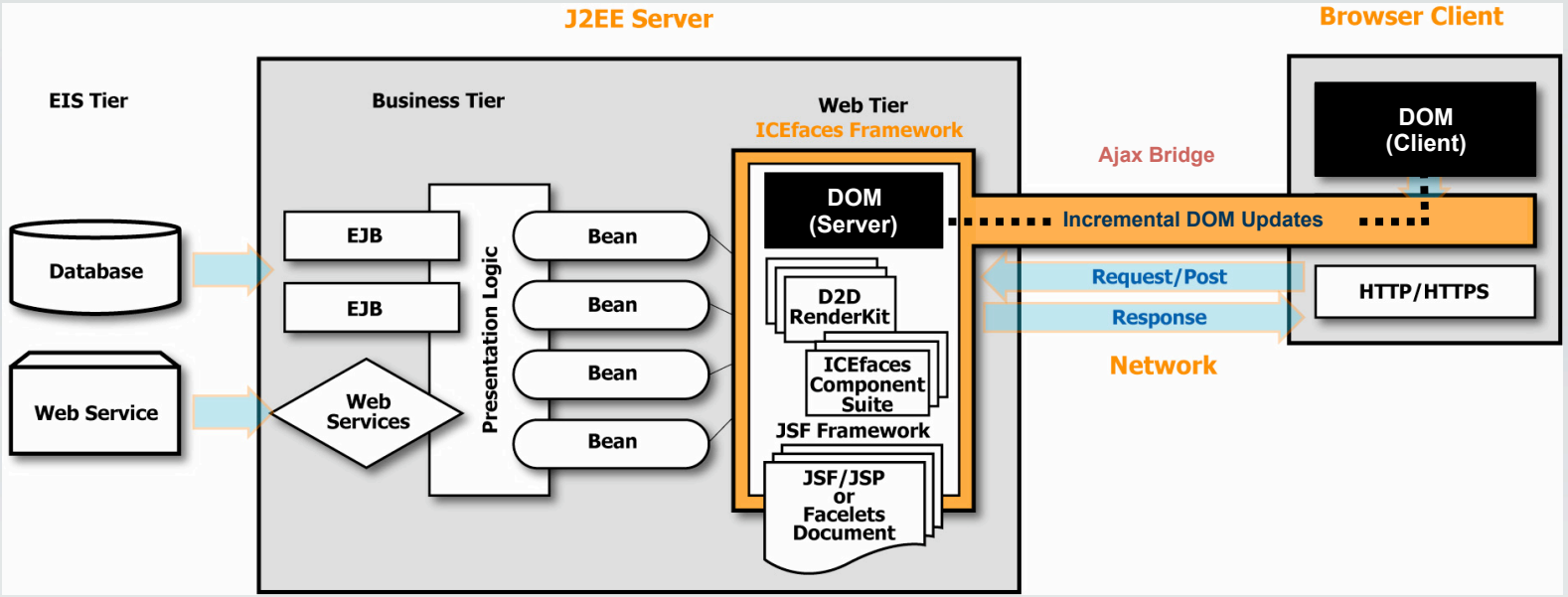
- Standard JSF components render markup directly to the response
- ICEfaces provides a JSF **render-kit** that causes components to render themselves into a server-side DOM

LIFERAY
Enterprise. Open Source. For Life.

Direct2DOM (Cont.)

- After the "Render Response" phase of the JSF lifecycle, ICEfaces will determine the differences between the server side DOM and the DOM in the browser
- ICEfaces will then use its Ajax Bridge to supply the browser with incremental DOM updates
- This technique insulates developers from the task of writing JavaScript

ICEfaces Direct2DOM Rendering



Direct-to-DOM insulates Java developers from the task of writing JavaScript…

LIFERAY
Enterprise. Open Source. For Life.

ice:portlet

- Portlet containers like Liferay Portal control the output of the following elements:

```
<html> ... </html>
<head> ... </head>
<body> ... </body>
```

- In order to ensure that ICEfaces portlets do not interfere with these tags during navigation from one JSF view to another, the ice:portlet tag must be used

ice:portlet (cont.)

```
<!-- Sample fragment of markup that shows how to -->
<!-- surround the ice:form with ice:portlet -->
<f:view
   xmlns:f=http://java.sun.com/jsf/core
   xmlns:ice="http://www.icesoft.com/icefaces/component">
   <ice:portlet>
     <ice:form>
          ...
     </ice:form>
   </ice:portlet>
</f:view>
```

Demo #2 – ICEfaces Portlet

JavaScript onblur event invokes partial submit on visited fields

File upload progress indicator driven by ICEfaces Ajax Push

*Other portlets on the page remain **undisturbed***

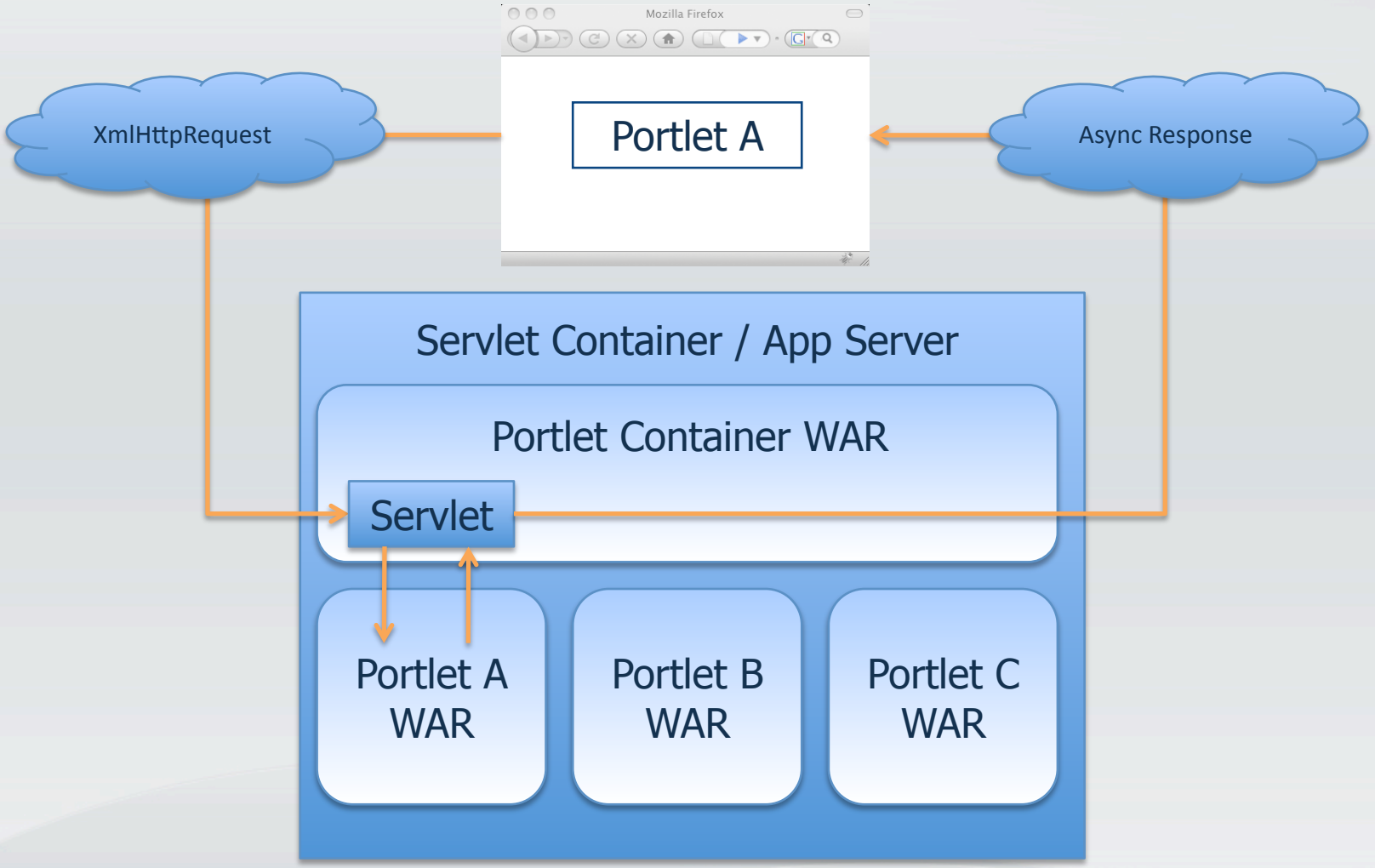
Direct2DOM incremental page updates provide client-side validation for free

LIFERAY
Enterprise. Open Source. For Life.

Portlet 2.0 and Ajax

- Portlet 2.0 provides the ability to issue XmlHttpRequest calls that go through the portlet container
- Benefit:
  - Provides complete access to portlet state
- Drawbacks:
  - Developer must manually update the DOM
  - No support for Ajax Push
  - Does not support Ajax-based Inter-Portlet Communication

Portlet 2.0 and Ajax



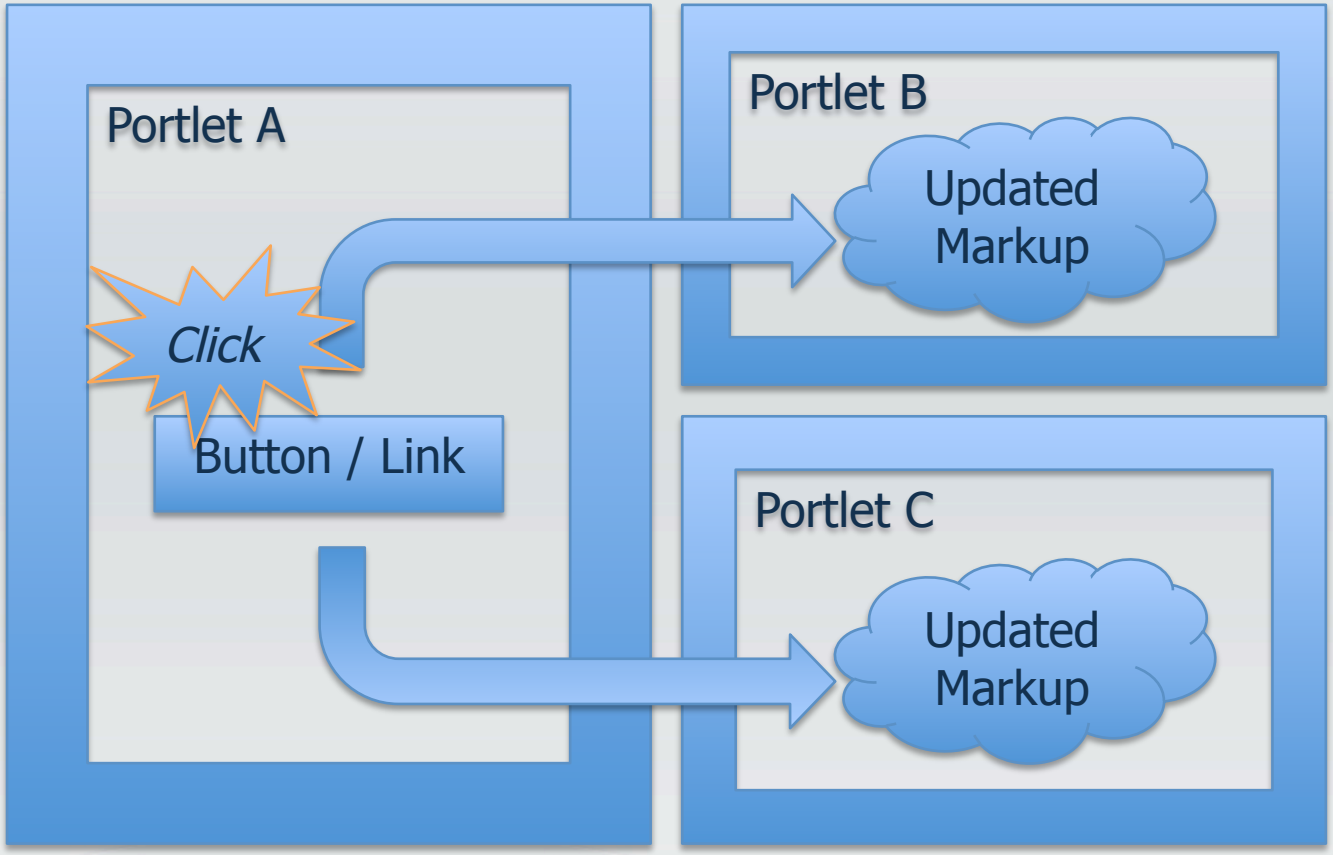
LIFERAY
Enterprise. Open Source. For Life.

Inter-Portlet Communication

- Inter-Portlet Communication (IPC) is a technique for **sharing data** between portlets and building **composite applications**
  - Enables building of composite applications by aggregating different portlets that share data
  - User interactions in Portlet A can affect the rendered markup in Portlet B, Portlet C, …
- IPC can be achieved by **client-side** and **server-side** techniques

LIFERAY
Enterprise. Open Source. For Life.

IPC Illustration



LIFERAY
Enterprise. Open Source. For Life.

Client-Side IPC

- Client-side IPC can be achieved with JavaScript
  - Liferay provides an event system based on the jQuery JavaScript API
  - Can be fortified with Ajax calls in order to acquire data that is not-yet in the browser's DOM

LIFERAY
Enterprise. Open Source. For Life.

Client-Side IPC (Cont.)

- Benefits:
  - Simple publisher/subscript event mechanism
  - Rich user experience as Portlet A triggers DOM update in Portlet B, Portlet C, …
  - Network activity only takes place if Ajax is used to acquire data
  - No full page submit

LIFERAY
Enterprise. Open Source. For Life.

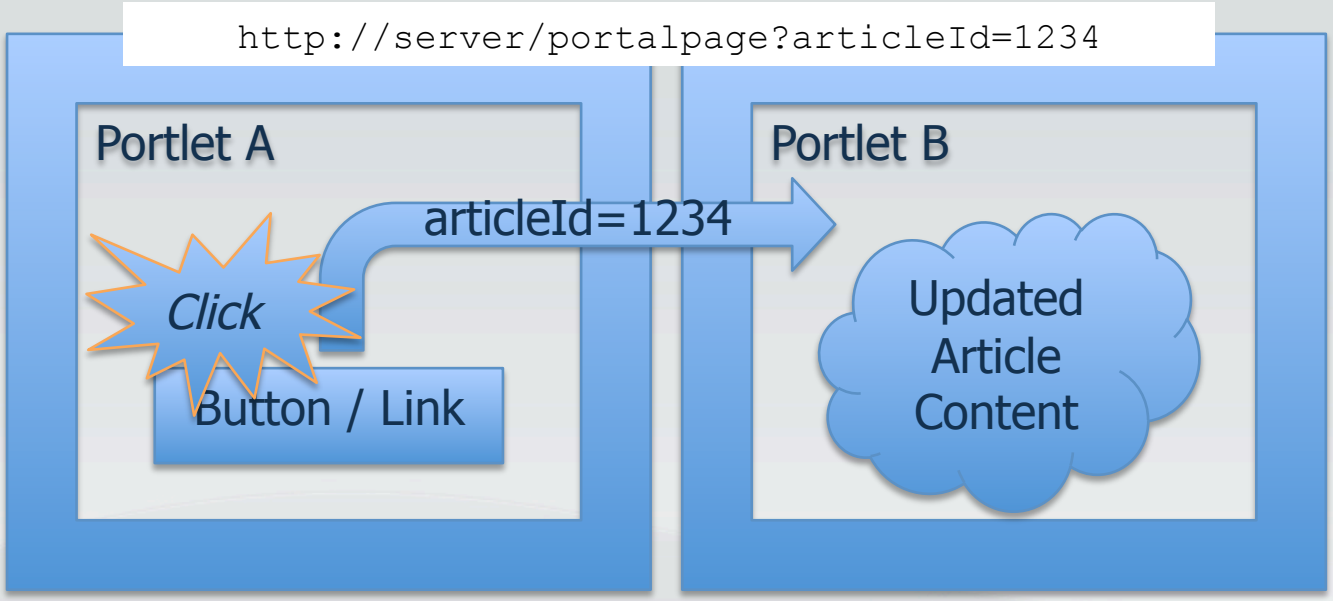
Client-Side IPC (Cont.)

- Drawbacks:
  - Only one user (and one web browser) participates in IPC
  - Have to write JavaScript for Ajax interactions
  - Have to write JavaScript to update DOM in affected portlets
  - Potential risk of business logic being exposed on the client
  - Portlet development is partly in JavaScript, partly in Java, which can be difficult to maintain sometimes

Server-Side IPC
Public Render Parameters

LIFERAY
Enterprise. Open Source. For Life.

- Portlet 2.0 defines the ability for portlets to set public/shared parameter names in the URL controlled by the portal



Public Render Parameters (Cont.)

- Benefits:
  - Easy to implement
- Drawbacks:
  - Requires full page submit
  - Only practical for passing small amounts of data, such as the "id" of a record in the database
  - Passing request parameters from page to page is not very JSF-*ish*

Server-Side IPC Events

LIFERAY
Enterprise. Open Source. For Life.

- Portlet 2.0 provides the publish/subscribe method for portlets to communicate via events
- Benefits:
  - Portal acts as broker and distributes events and payload (data) to portlets
- Drawbacks:
  - Can be challenging to implement
  - Not yet supported by JSF portlet bridges
  - Requires full page submit
  - Payload must be serialized by the portal when events are passed to listeners in other classloaders

Server-Side IPC
JSF Session Scope

LIFERAY
Enterprise. Open Source. For Life.

- Trying **JSF session scope** for IPC might be the most natural thing for a JSF developer to try, but it **doesn't work**!
- **Why not**? Because the Portlet API defines session scope in two ways:
  - PortletSession.PORTLET_SCOPE: Data cannot be accessed by other portlets
  - PortletSession.APPLICATION_SOPE: Data can be accessed by other portlets

And…

Server-Side IPC
JSF Session Scope (Cont.)

- JSF portlet bridges (including the ICEfaces bridge) default JSF session scope to be PortletSession.PORTLET_SCOPE
- Consequently, JSF session scope doesn't work for IPC ☹

Server-Side IPC
Shared Portlet Session Scope

- JSF portlets can take matters into their own hands and store data for IPC in PortletSession.APPLICATION_SCOPE
- Benefit:
  - Sharing data in a stateful user session managed by portal
- Drawback:
  - Can't share data acrosss WARs

LIFERAY
Enterprise. Open Source. For Life.

JSF Application Scope

- Storing shared data in JSF application scope is another choice for IPC
- Benefit:
  - Not restricted to a single user – perfect for a Chat portlet
- Drawback:
  - Can't use the stateful features of the session to rely on memory getting freed up when the session is invalidated
  - Can't share memory between different portlet WARs

Server-Side IPC
Sharing Data Between WARs

- Liferay Portal normally lives in the ROOT context of the servlet container
- Liferay provides the PortalClassInvoker utility that can provide access to static data that lives in the ROOT context
- For more information see blog entry on sharing data between portlets in different .WARs

http://www.liferay.com/web/ngriffin/blog/-/blogs/sharing-data-between-portlets?_33_redirect=/web/ngriffin/blog

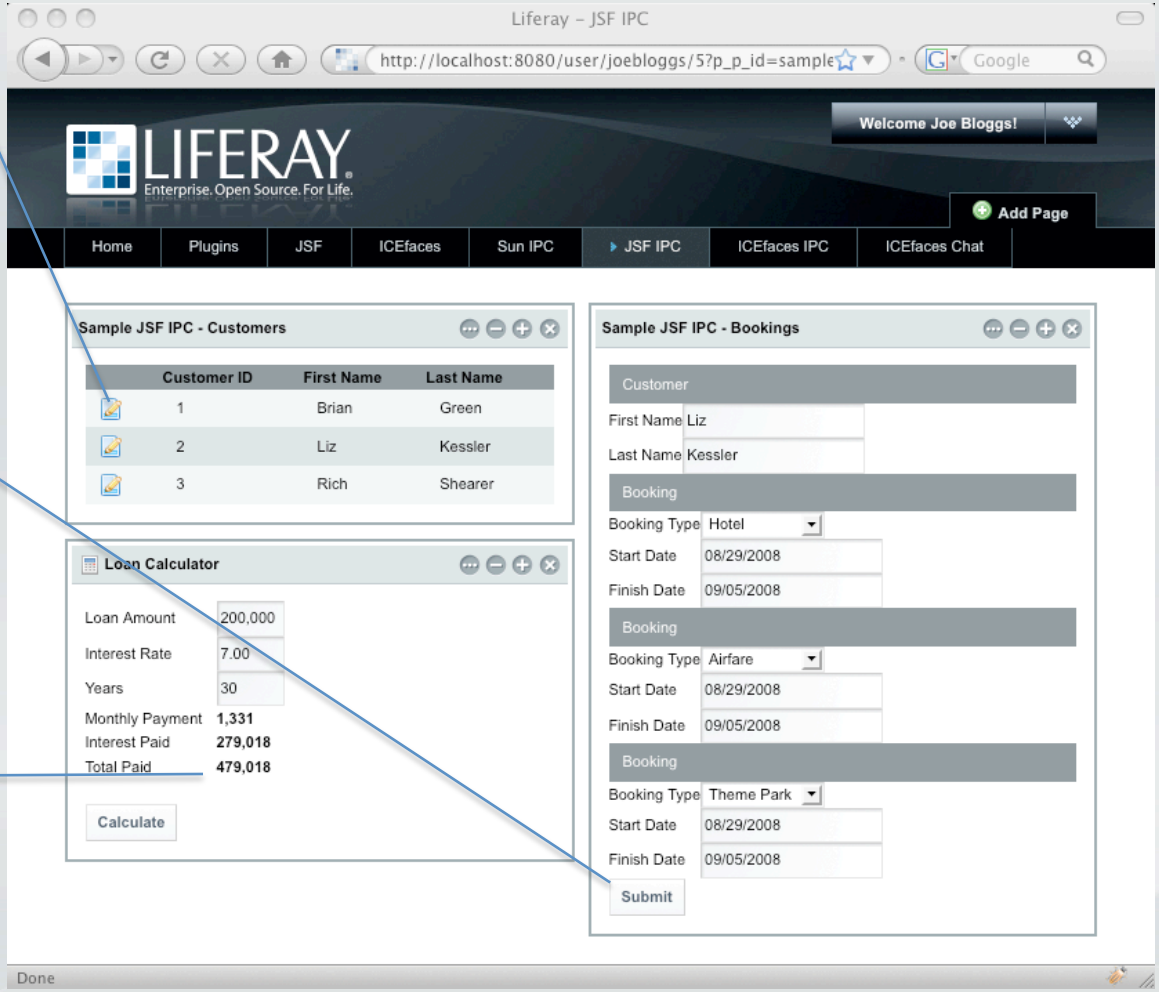
LIFERAY
Enterprise. Open Source. For Life.

Demo #3 – Standard JSF IPC

User must submit the form (HTTP POST) in order to inform the Bookings portlet of a new selection

User must submit the form (HTTP POST) in order to inform the Customers portlet of a name change

Other portlets on the page ***are disturbed***



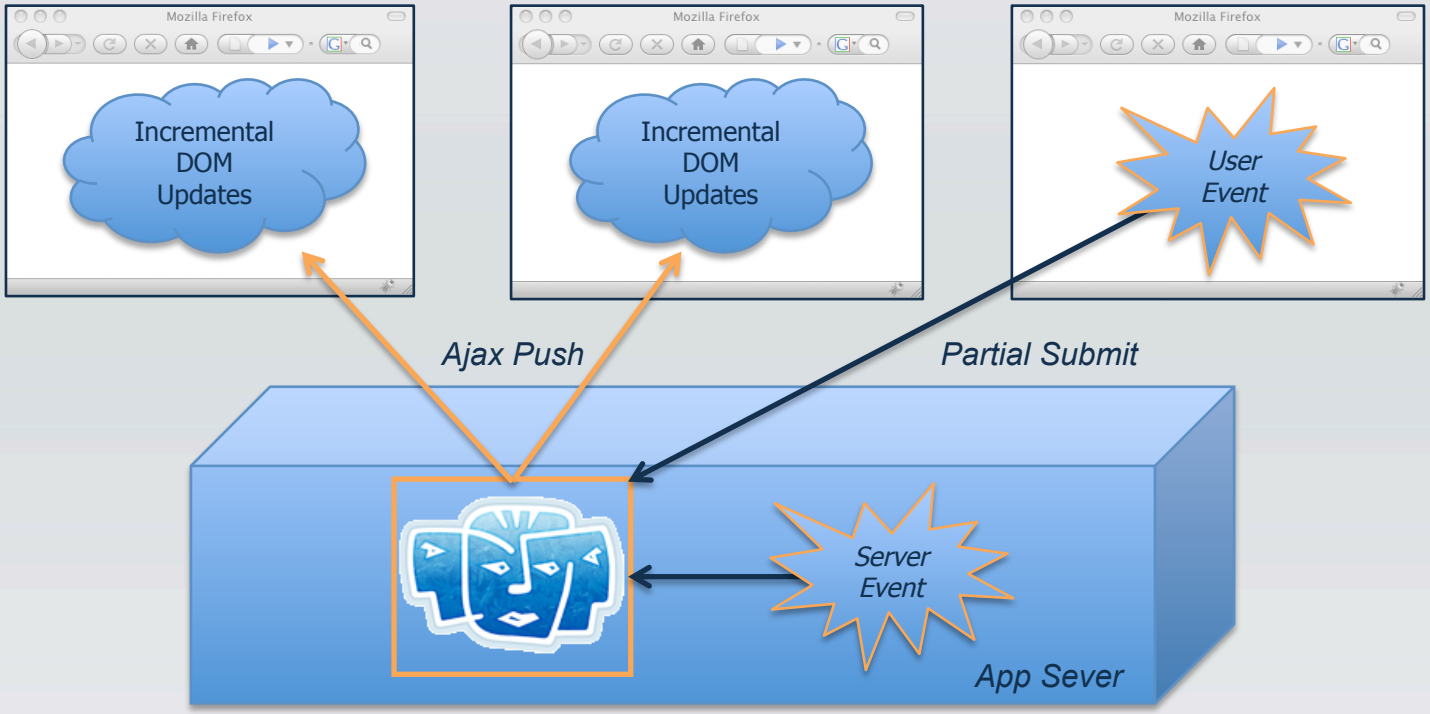
LIFERAY
Enterprise. Open Source. For Life.

ICEfaces Ajax Push

- ICEsoft pioneered **Ajax Push**, and made it part of the design of ICEfaces from the very beginning
  - Sometimes referred to as "Comet" or "Reverse Ajax"
- ICEfaces webapps/portlets can use Ajax Push to trigger **server-initiated rendering**

LIFERAY
Enterprise. Open Source. For Life.

Ajax Push Illustrated



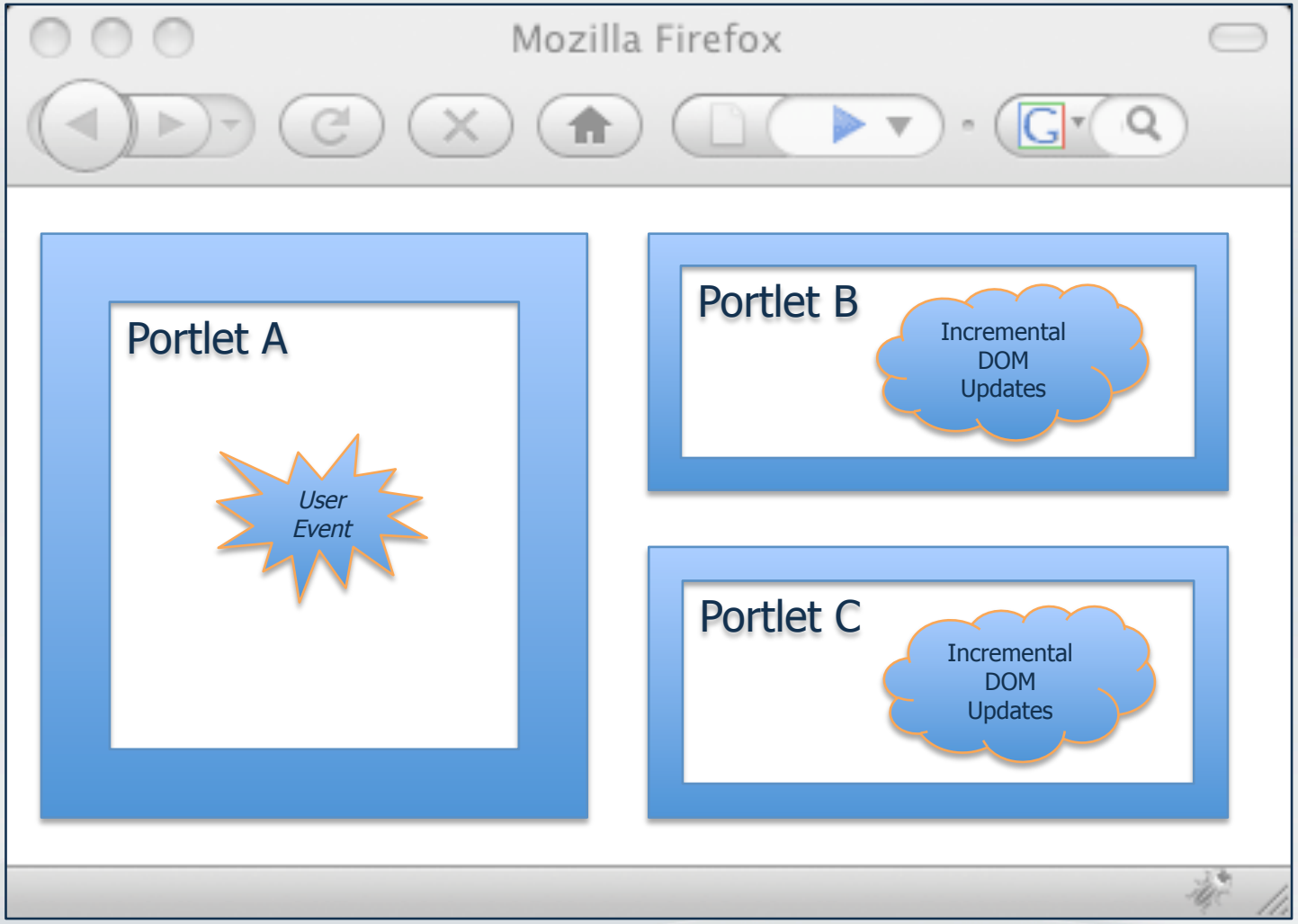
LIFERAY
Enterprise. Open Source. For Life.

Ajax Push IPC

- ICEfaces Ajax Push is a compelling technique for IPC
- Benefits:
  - Easy to implement
  - Rich UI experience for the end user
  - Behaves like client-side IPC, but has none of the drawbacks!
  - Other portlets on the page are undisturbed
  - Not just inter-portlet, but inter-portlet, inter-user communication!
- Drawbacks:
  - None!

LIFERAY
Enterprise. Open Source. For Life.

Ajax Push for IPC

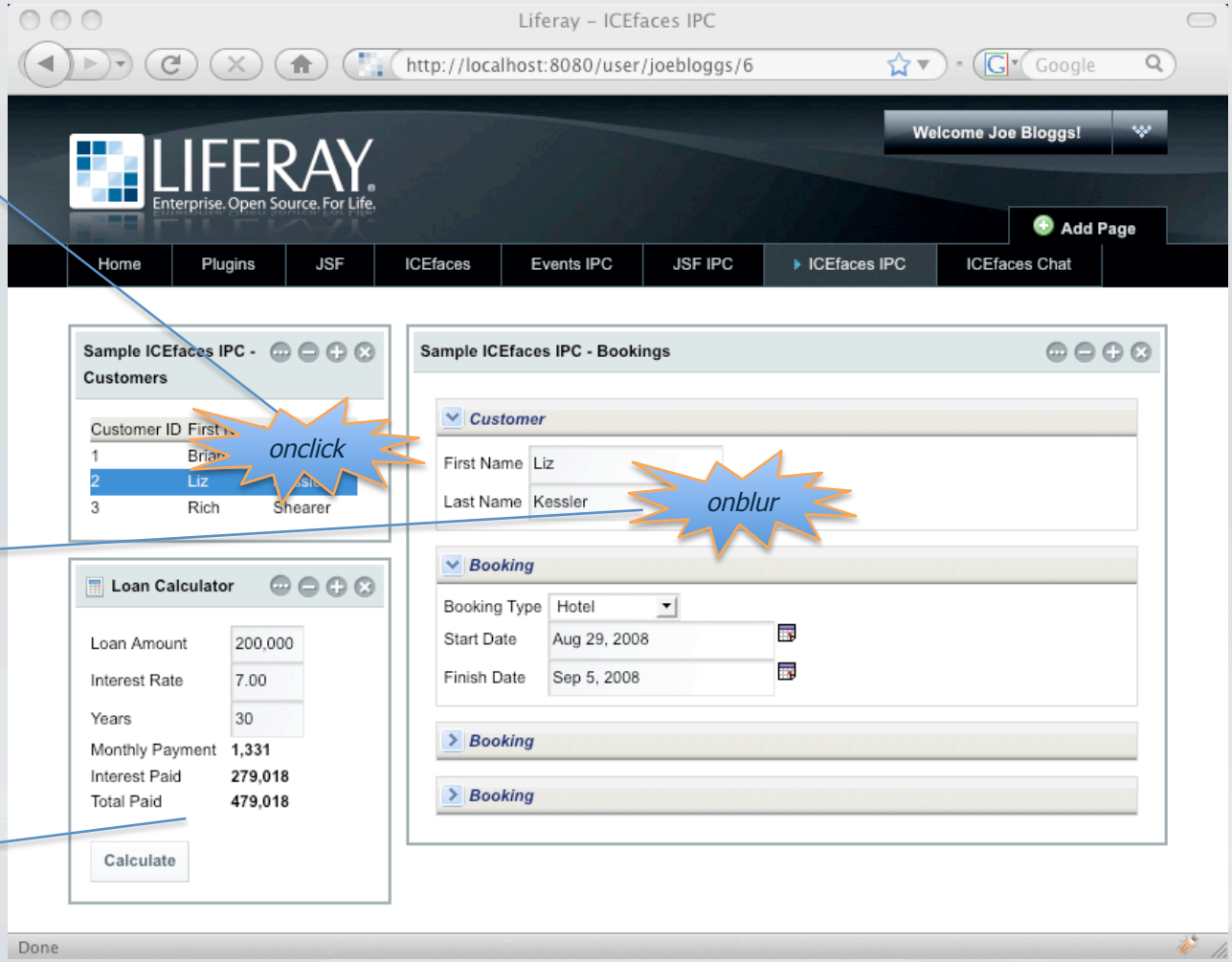


Demo #4 – Ajax Push IPC

Partial submit triggers Ajax Push, informing the Bookings portlet of a new customer selection

Partial submit triggers Ajax Push, informing the Customers portlet of a name change

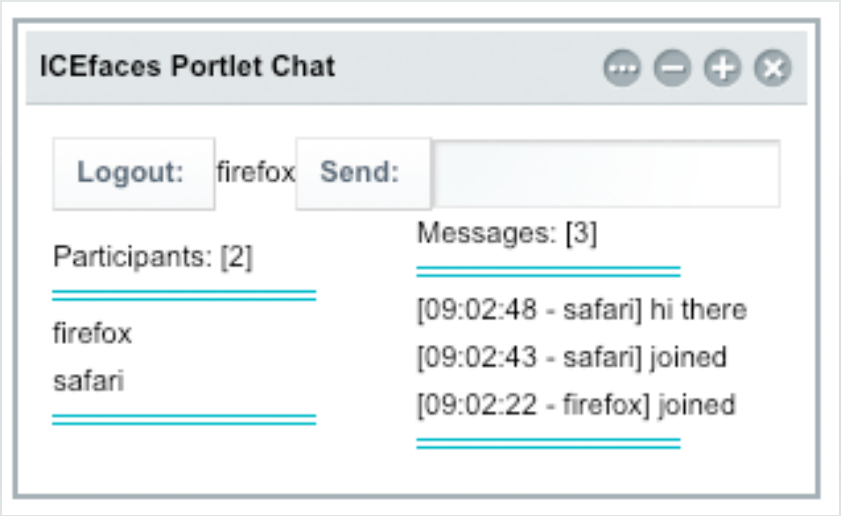
Other portlets on the page are ***undisturbed***



LIFERAY
Enterprise. Open Source. For Life.

Demo #5

- Shared data (chat log) stored in **JSF application scope**
- Different portal users can chat with each other



Liferay + ICEfaces Deployment Options

LIFERAY
Enterprise. Open Source. For Life.

Servlet Containers:

- Apache Tomcat
- Webtide Jetty

Database Servers:

- MySQL®
- Oracle®
- Microsoft® SQL Server™
- IBM DB2™
- Sybase®
- SAP®
- JavaDB (Apache Derby)

Application Servers:

- Sun GlassFish™ AS
- JBoss® AS
- BEA®/Oracle® WebLogic AS
- Oracle® AS
- IBM WebSphere® AS

Operating Systems:

- Windows®
- Linux®
- Sun Solaris®
- IBM AIX™

Summary

- ICEfaces portlets provide a rich UI that does not disturb other portlets on the same portal page
- ICEfaces Ajax Push is a compelling technique for IPC within Liferay Portal

Questions?

- Thank you for attending!