



IBM Software Lab Services

# Developing Web 2.0 user interfaces with iWidgets and the Dojo Toolkit

**Martin Gale**

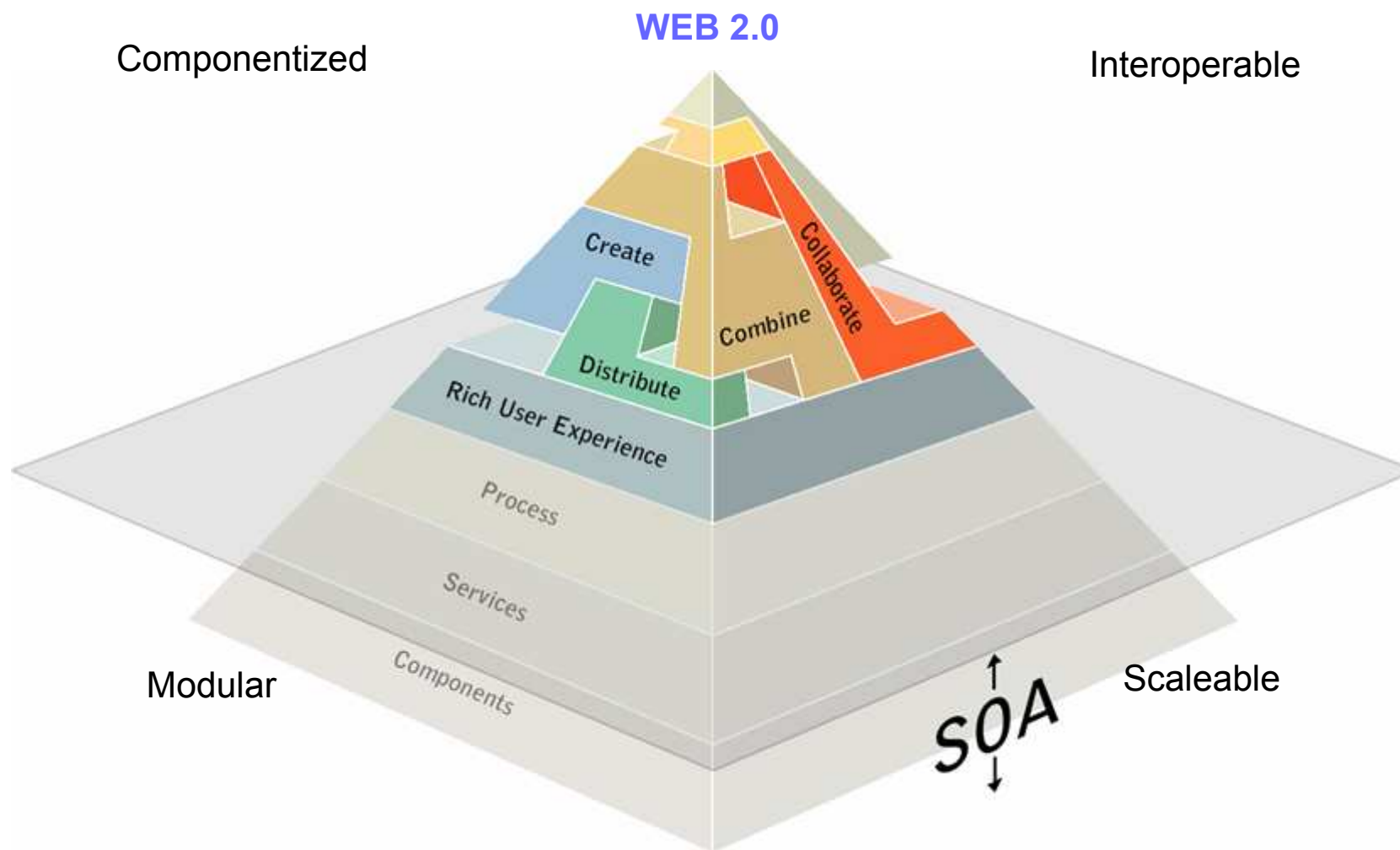
Master Inventor, IBM Certified IT Specialist

IBM Software Lab Services, Hursley

The session will introduce the the IBM recommended approach for developing Web 2.0 user interfaces

- The agenda for this session is:
  - An introduction to Web 2.0 and the key IBM products in the Web 2.0 space.
  - The Dojo Toolkit and its core concepts.
  - iWidget concepts and developing iWidgets for IBM Mashup Center.
  - Questions.

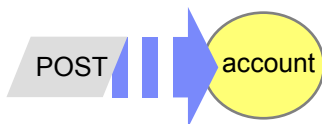
# Web 2.0 builds on the solid underpinning of enterprise SOA



## Web 2.0 technologies provide simple mechanisms for rapid (re)assembly of applications and new services



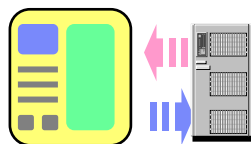
- **Feeds** – streams of information formatted to agreed standards providing a regular stream of updates



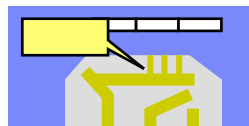
- **REST** – simple prescriptions for service design that increase the accessibility and re-usability of internet services



- **Widgets** – interoperable and reusable user interface components for flexible construction of the user interface



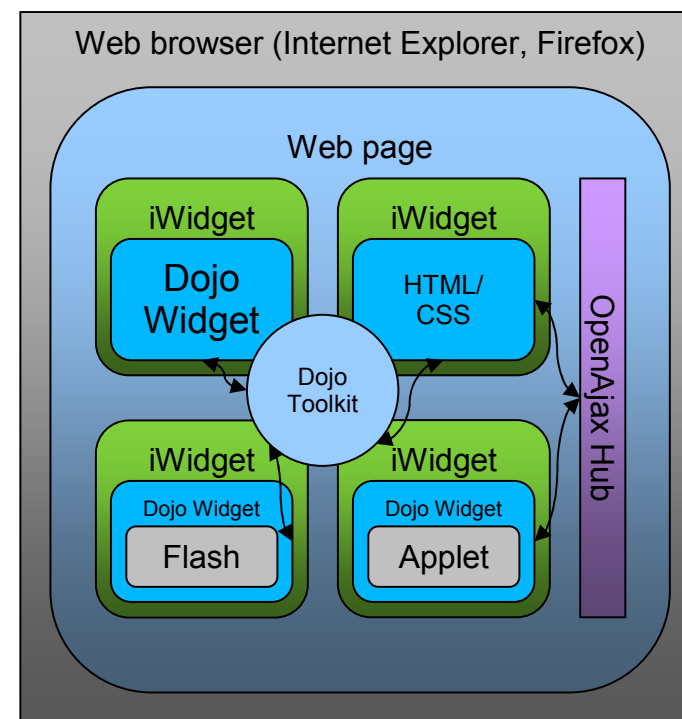
- **AJAX** – simple technologies used cleverly to create rich user interfaces that are rewarding to use in a browser



- **Mashups** – applications consisting of services and information that are rapidly assembled to meet dynamic needs

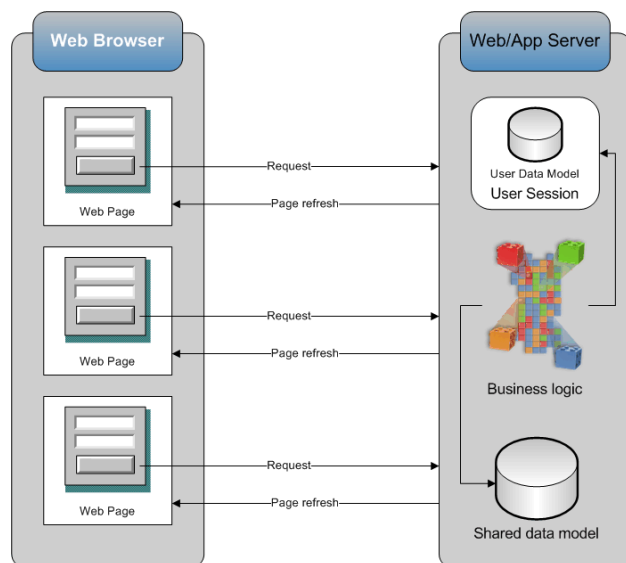
## IBM client strategy for Web 2.0

- Key themes for the IBM strategy for Web 2.0 user interfaces are
  - Open, standards-based
    - W3C, DOM, CSS and JavaScript
  - Leverage core browser technology as the platform and augment with AJAX frameworks
    - Dojo Toolkit
- Accommodates proprietary third party technology without mandating wholesale adoption of a proprietary runtime.
  - Flash, Flex, Silverlight...

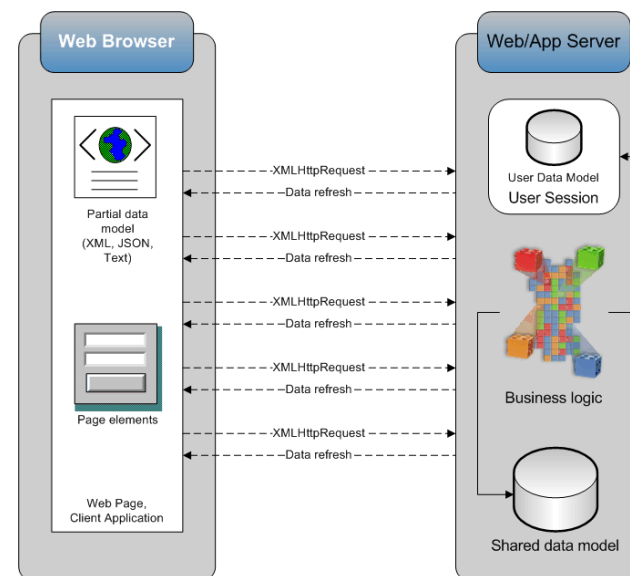


# The Web 2.0 user interface evolution

## Traditional web interaction



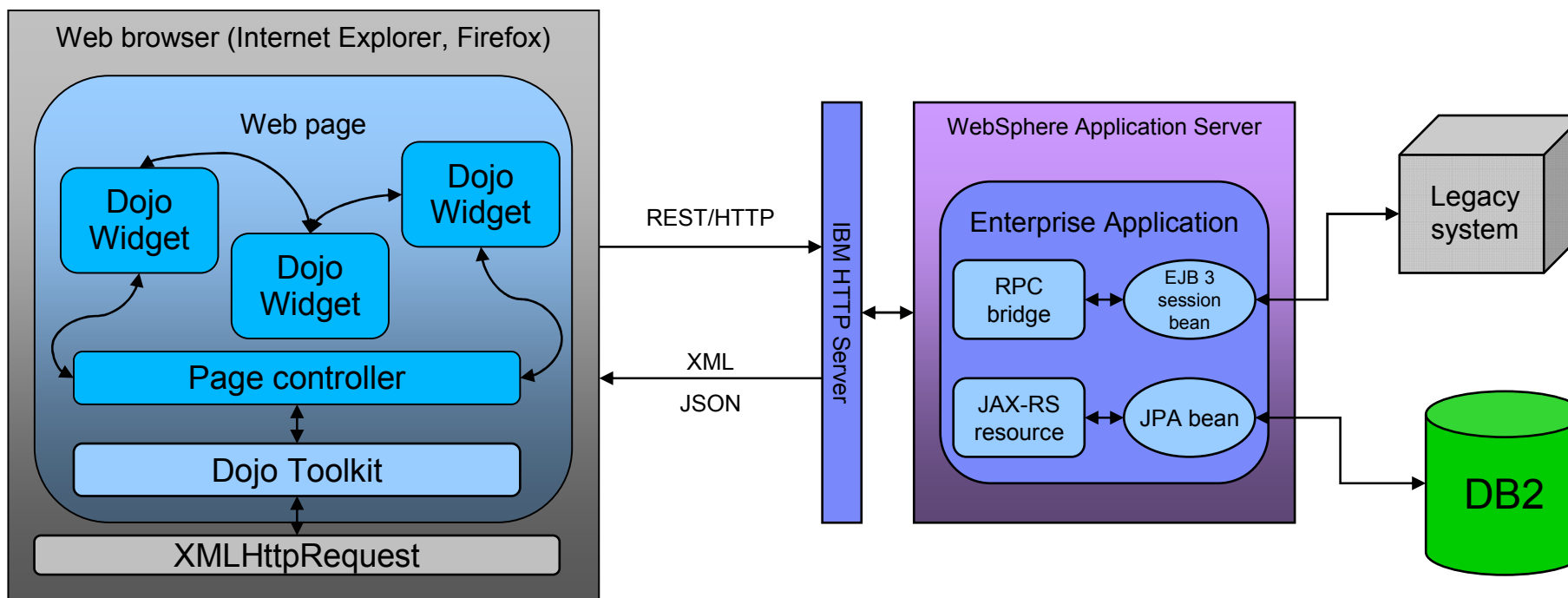
## Web 2.0 AJAX interaction



- “Round tripping” provision of the user interface – client/server interaction obvious to the user.
- User interface flow logic managed on the server.
- Application server primarily serves HTML pages.
- Reuse only within the application server infrastructure.

- Smooth user experience – client/server interactions achieved without a full page refresh.
- User interface flow logic managed on the client.
- Application server primarily serves data.
- Reuse through modular AJAX toolkits on the client and RESTful services.

## An example of a modern JEE architecture for Web 2.0



## What are AJAX frameworks and why do we need them?

- There are a number of challenges posed by the browser environment including
  - differences between browser brands
  - lack of a consistent component/packaging model
  - limitations of the standard HTML widgets
  - extensions to UI behaviour always require scripting
  - lack of a natural separation of concerns between presentation and data
- AJAX frameworks raise the level of abstraction above the base HTML/JavaScript runtime.
- There are in the order of hundreds currently available today, by and large through Open Source.

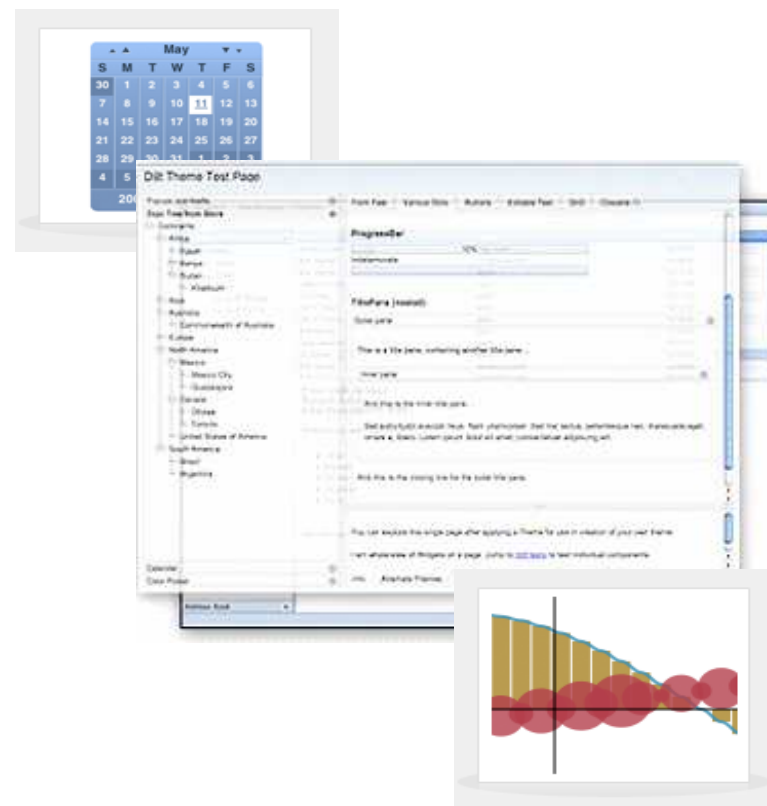




## Introducing the Dojo Toolkit

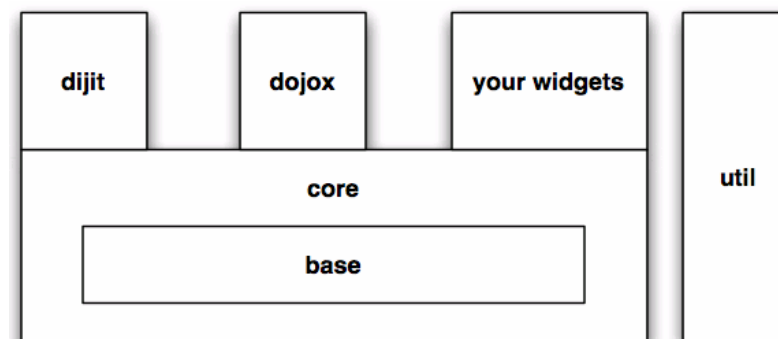
- The Dojo Toolkit is IBM's choice as best of breed AJAX framework.
  - Open source, freely available from <http://www.dojotoolkit.org>
  - Built on open web standards such as HTML, JavaScript and CSS.
  - Seeks to insulate the developer from browser differences and quirks and promote modular, open web UIs.
- Key features include:
  - Extensible and flexible component model
  - Allows declarative UI extensions
  - Rich user interface components and themes.
  - Support for accessibility and internationalisation.

# dōjō



# Dojo Architecture

- **Base**
  - Component model and packaging.
  - Inheritance, dependencies and class model.
- **Core**
  - Commonly used foundation services.
  - XMLHttpRequest Wrappers.
  - Wipes/slides.
  - Pub/sub event model.
- **Dijit**
  - “Dojo” + “Widget” = “Dijit”
  - Rich user interface widgets.
- **DojoX**
  - New extensions to the framework.
  - Charting.
  - Rich data grids.
- **Util**
  - A collection of Dojo utilities including compression, math and offline storage.

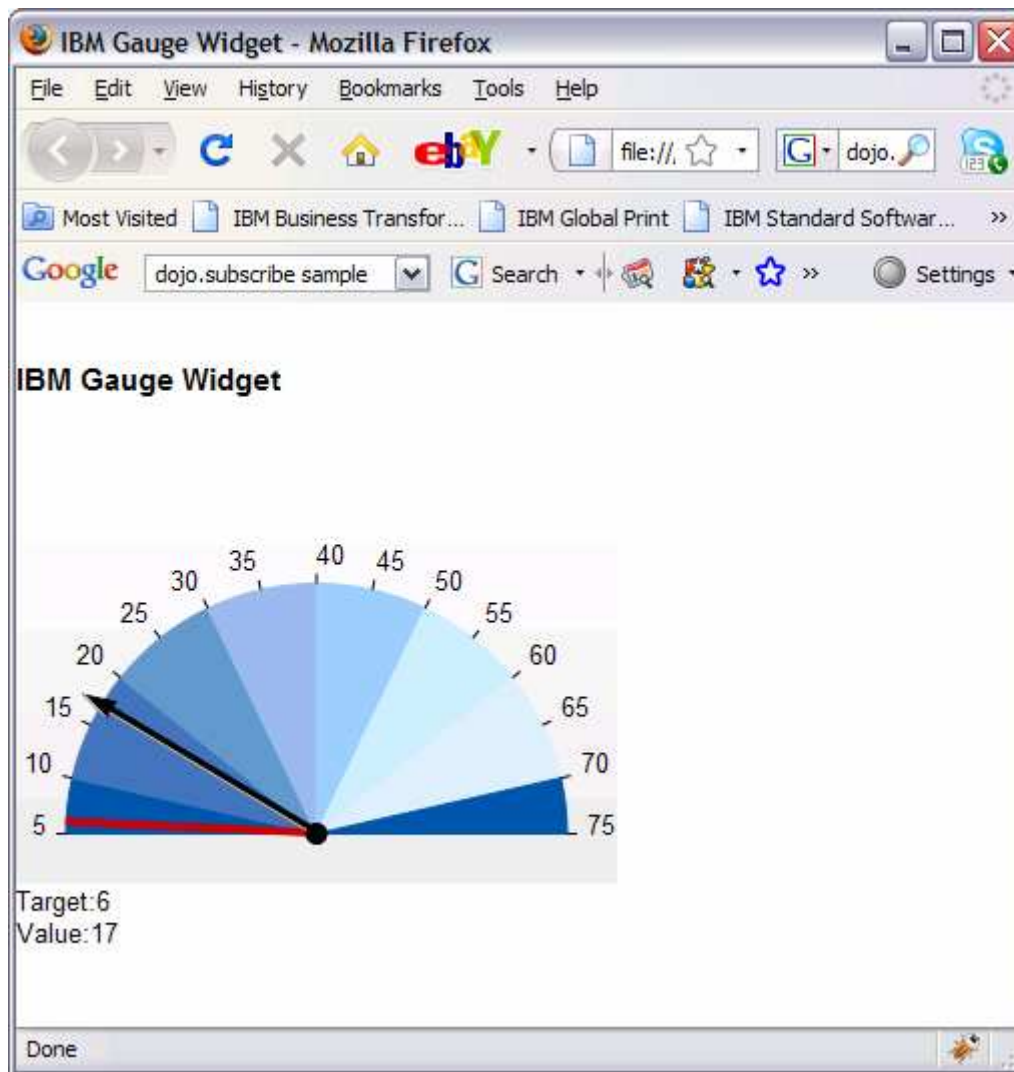


```
<html>
<head>
  ...
  <title>IBM Gauge Widget</title>
  <style>
    @import "../dojo/dijit/themes/tundra/tundra.css";
  </style>
  <script type="text/javascript"
    src="../dojo/dojo/dojo.js"></script>
  <script language="JavaScript">
    dojo.require("ibm_gauge.widget.AnalogGauge");
  </script>
  ...
</head>
<body>
  ...
  <div dojoType="ibm_gauge.widget.AnalogGauge"
    id="testGauge"
    gaugeWidth="300"
    gaugeHeight="200"
    cx="150"
    cy="175"
    radius="125"/>
  ...
</body>
```

HTML page (.html file)

```
dojo.provide("ibm_gauge.widget.AnalogGauge");
dojo.require("dojox.gfx");
dojo.require("ibm_gauge.widget._Gauge");
dojo.declare("ibm_gauge.widget.AnalogGauge",
  ibm_gauge.widget._Gauge, {
  ...
  constructor: function(args) {
    this.id = args.id;
    this.radius = args.radius;
  },
  ...
  postCreate: function() {
    dojo.subscribe(
      "ibm/widgets/gauge/"+this.id+"/set",
      this,
      function(value) {
        this.setGaugeValue(value);
        dojo.publish("events/changed",
          [value]);
      }
    )
  }
  ...
});
```

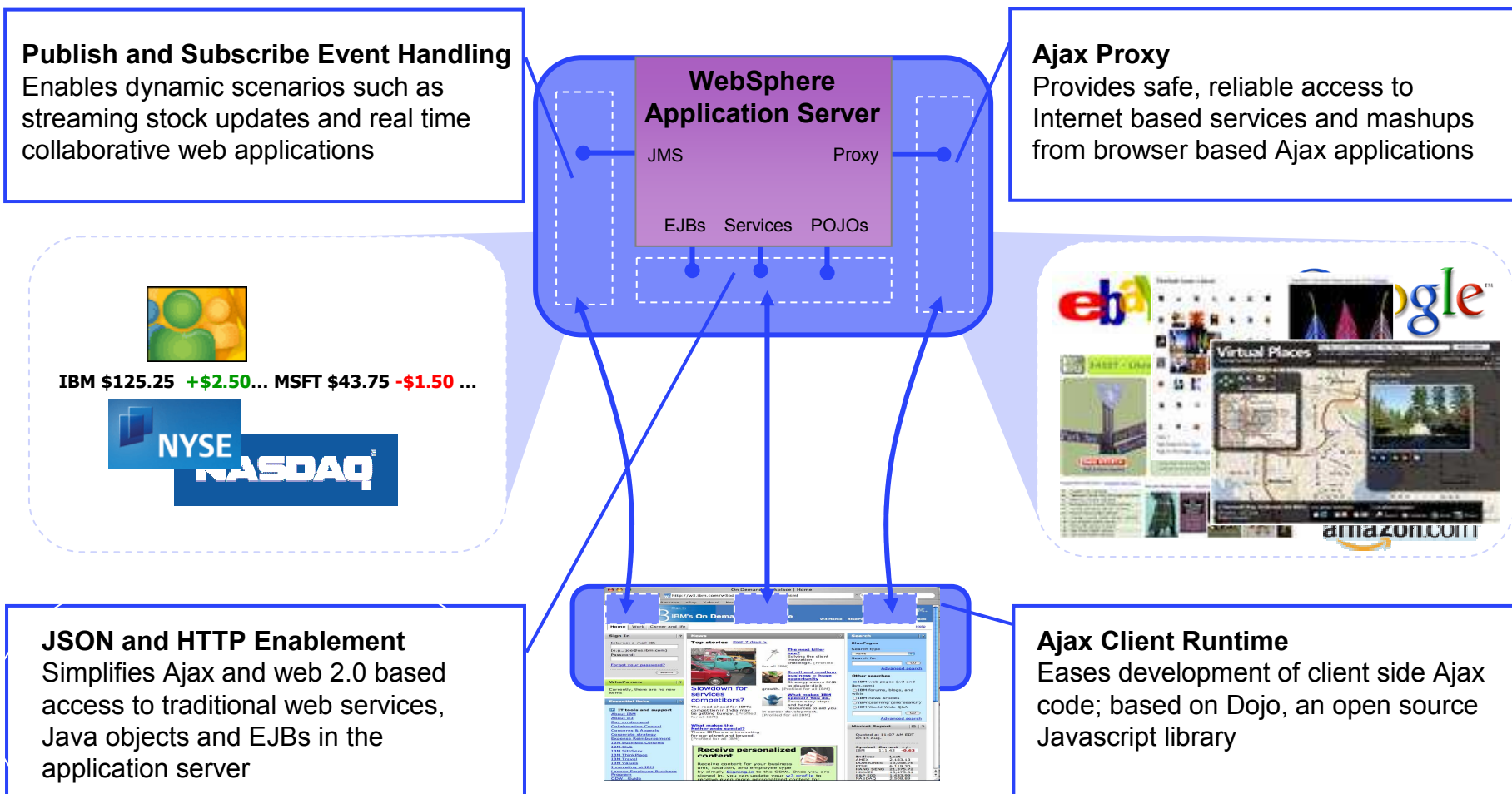
Dojo Widget class (.js file)



# A more advanced sample

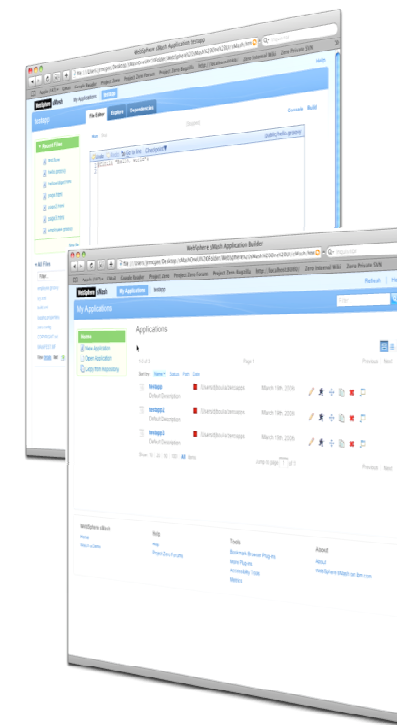
The screenshot displays the 'Field Instrumentation Diagram Editor' software running in a Mozilla Firefox browser. The interface includes a menu bar (File, Edit, View, Window, Help), a toolbar, and a main workspace. On the left, a tree view shows a project structure with folders for 'Diagrams and Enterprises', 'My Diagrams', and 'Other Diagrams'. The main workspace contains a P&ID diagram with several components: a pressure transmitter labeled '13-ZSH 045E', a control valve labeled '13-ESV 045', and a needle gauge with a scale from 5 to 30. A red line connects the valve to the gauge. On the right, a 'Palette' lists various instrument symbols such as 'Title Banner', 'Output Text', 'Indicator', 'Thermometer', 'NeedleGauge', 'Line Chart', 'PID Runtime Diagram Co', 'Controller', 'SafetyDevice', 'Valve', 'Well Probe', 'Temperature Transmitter', 'Pump', 'Distillation Tower', 'Flow Meter', 'C Valve', 'Shared Display', 'Unknown', 'Primary Element', and 'Replay'. At the bottom, a 'Properties View' shows the selected 'NeedleGauge' element's attributes: width (204), height (109), and three color bands (0054AA, 4477BB, 6699CC).

# WebSphere Application Server Feature Pack for Web 2.0

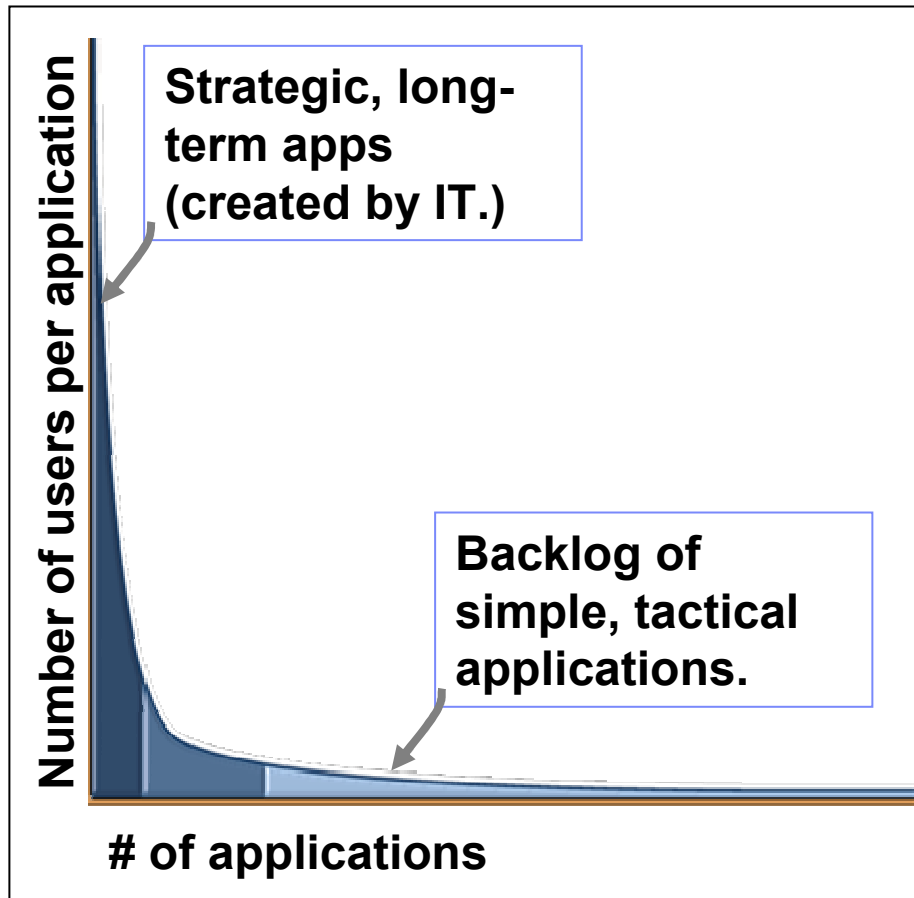


# WebSphere sMash

- WebSphere sMash is an Agile Web Application Platform
  - Architected around Dynamic Scripting, REST, Rich Web Interfaces, AJAX, and Feeds
  - Optimized for
    - Speed
    - Simplicity
    - Agility
- Key Scenarios
  - Composing, and “cobbling together” pre-existing assets (PHP assets, services, feeds, code snippets) using dynamic scripting languages and simple consumption principles based on REST.
  - Rapid development of dynamic web applications that are assembled from enterprise assets and publicly available APIs.



## Business/IT challenges in the Web 2.0 world



### ***Business Challenges***

- Lack of agility – IT can't respond fast enough to business requirements
- Spreadsheet “apps” hard to manage, share, keep current
- Silo-ed, outdated information

### ***IT Challenges***

- IT is backlogged and suffering from too many business requests
- Underground or “shadow” IT
  - Security violations
  - Loss of information



## What is a Mashup?

A “mashup” is a lightweight web application created by combining information or capabilities from more than one existing source to deliver new functions & insights.

ETA	Libelle POD	Libelle fournisseur
2007-10-01	SINGAPORE	LANDBOND MARKETING (F.E.) PT
2007-10-01	SINGAPORE	LANDBOND MARKETING (F.E.) PT

ETD	ETA	Transit Time	POL	Libelle POL	POD	Libelle POD	Navire	NO Voyage	Compagnie	Feeder Arriv
2007-10-05	2007-10-05 3		VNSGN	HO CHI MINH CITY THBKK	BANGKOK	HETHI BHUM 5110			SUPPLIER CARRIER	

- Rapid creation (days not months)
- Reuses existing capabilities, but delivers new functions + insights
- Requires limited to no technical skills
- Often mixes internal and external sources

## IBM Mashup Center

*A comprehensive mashup platform, supporting line of business assembly of simple, flexible, and dynamic web applications – with the management, security, and governance capabilities IT requires*

- **Create** new applications by reusing existing data and services
- **Unlock** Enterprise, Web, Personal and Departmental Information
- **Develop** widgets from enterprise systems
- **Discover** and share mashups, widgets, feeds, and services
- **Transform** information into new feeds

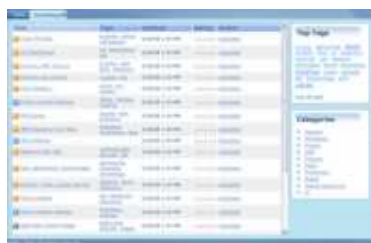


# IBM Mashup Center Components



**Assembly  
Centric**

**Lotus Mashups:** Quickly and easily create and assemble mashups on-the-glass. Create dynamic widgets.



**Common Catalog:** Sharing & discovery of mashable assets.



**Information  
Centric**

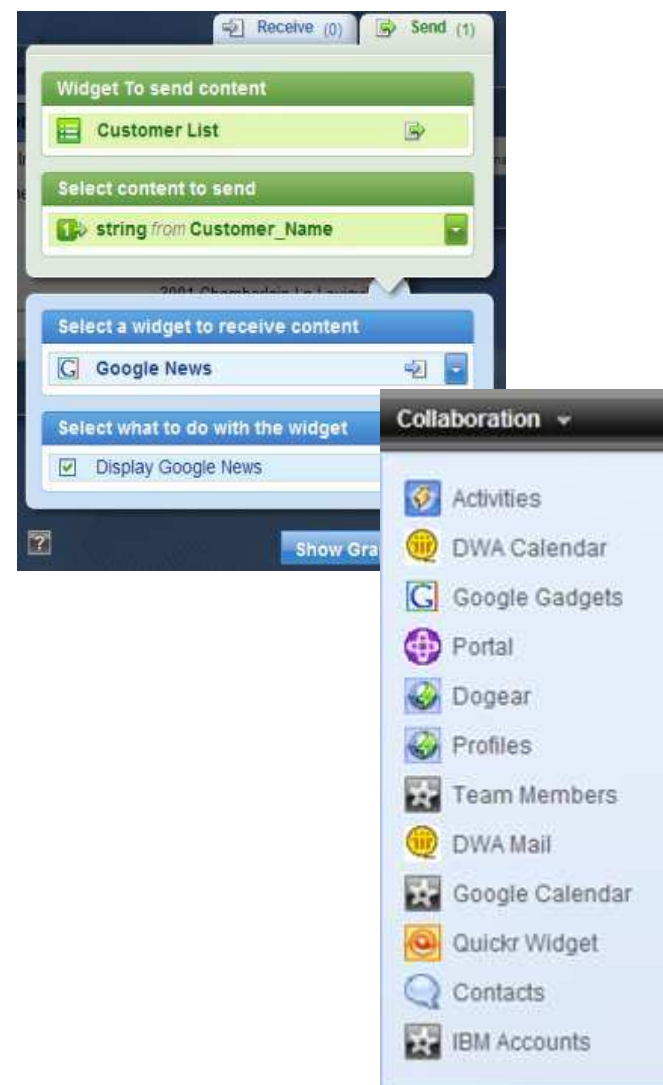
**InfoSphere MashupHub:** unlock and share web, departmental, personal and enterprise information for use in REST-style Web2.0 applications. MashupHub includes visual tools for transforming and re-mixing REST-style feeds.

## Mashups build on top of AJAX frameworks to provide coarse grained components for visualising data

- Flexible applications such as mashups bring together data feeds with AJAX widgets that render the information in a consumable form in a browser environment.
- The visualisation widgets must be sufficiently flexible and “wireable” to adapt to different feeds and scenarios.
  - A variety of different data feed sources and styles.
  - A variety of different application scenarios.
- For the broadest choice, the mashup user should be able to choose from feeds and widgets developed both in house and by third parties.
- In order for these components to be consumed with the minimum integration effort, a standard interface is required.

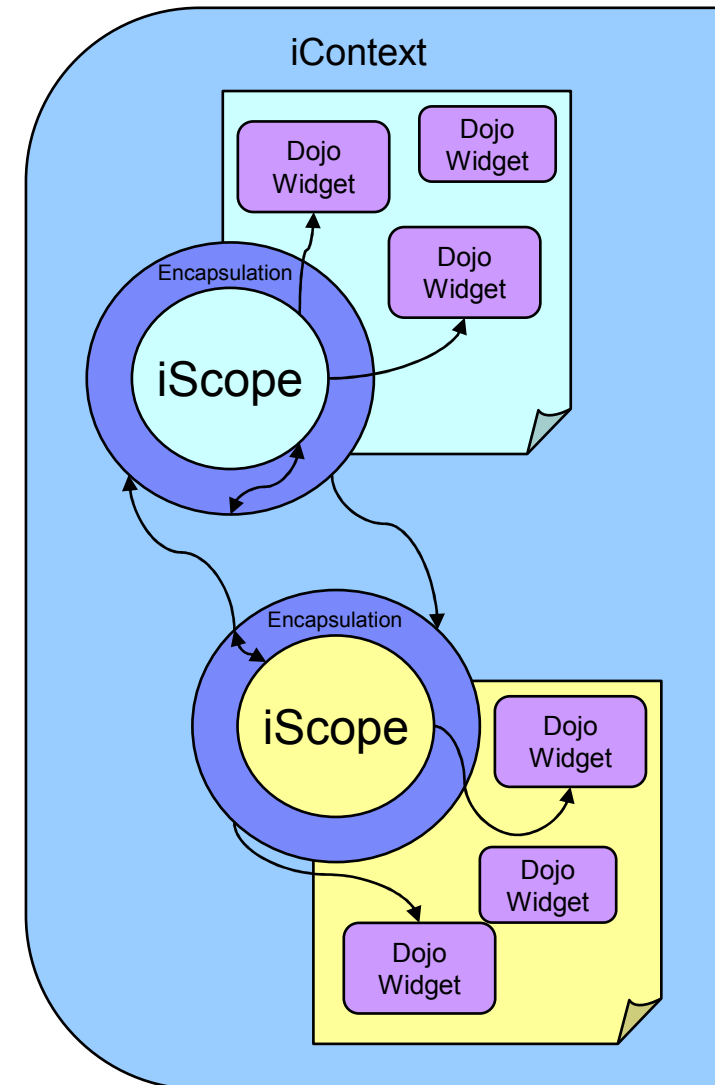
## What are iWidgets?

- iWidget is IBM-developed standard to describe reusable and configurable browser UI components.
- An iWidget
  - shares the page with other components.
  - is usually designed to be wired to other components on the page.
  - is often constructed from finer-grained widgets from a toolkit such as Dojo
- The iWidget specification builds on XML, web standards and AJAX toolkits to define meta-data describing
  - modes of operation for the component (view, edit)
  - any application events emitted and consumed.
  - configurable attributes understood by the component.
- iWidgets are the application building blocks in IBM Mashup Center



## Components of an iWidget

- An iWidget is instantiated in a supporting iWidget runtime environment e.g.
  - IBM Mashup Center
  - WebSphere Portal with the iWidget Portlet
- An iWidget typically consists of
  - an XML meta-data descriptor and rendering markup
  - a JavaScript class called the `iScope` to provide the programmatic logic for the widget such as
    - responding to and emitting events from its surrounding framework known as the `iContext`.
    - managing the lifecycle of the iWidget.
    - application logic to underpin the user interface.
- At runtime, the `iContext` supplied to the `iScope` by the runtime environment provides the means for the iWidget instance to interact with its surroundings such as
  - a localised `getElementById()` method for accessing UI elements.
  - accessors for attributes defined for the specific iWidget instance.



## Components of the iWidget XML definition

- The XML definition describes the widget configuration with details such as
  - the viewing modes the iWidget supports.
    - `view` is the most commonly used mode and is the mode in which the iWidget displays its UI.
    - `edit` to modify settings about the iWidget.
    - `help` to provide help to the user of the iWidget.
  - References to any required resources such as JavaScript files or CSS style sheets.
    - iWidgets often have at least one referenced resource which is the .JS file containing the `iScope` class.
  - Markup content for each of the required modes. This is the basic display logic for the iWidget and can contain other widgets within it.
- The name of the JavaScript class implementing the programmatic logic for the widget as the `iScope`.

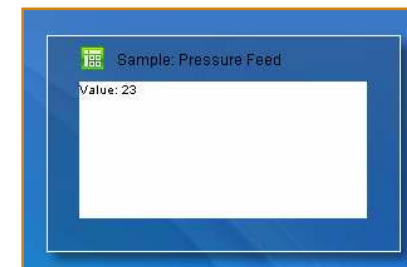
```
<iw:iWidget
  id="helloWorld"
  xmlns:iw="http://www.ibm.com/xmlns/prod/iWidget">
  <iw:content>
    Hello World
  </iw:content>
</iw:iWidget>
```

## An example iWidget definition

```

<iw:iwidget id="samplewidget" xmlns:iw=http://www.ibm.com/xmlns/prod/iWidget
  iScope="lm.samplewidget" allowInstanceContent="true" supportedModes="view edit"
  mode="view" lang="en">
  <iw:resource uri="samplewidget.js" />
  <iw:content mode="view">
    <![CDATA[
      <div id="container">Value: <span id="valueNode">0</span>
    </div>
    ]]>
  </iw:content>
  <iw:content mode="edit">
    <![CDATA[
      <div style="background-color: white; padding: 2px; border: solid #4078C2 1px;">
        <table border="0">
          <tr> <td colspan="2">
            <h2>Data feed</h2>
          </td>
        </tr>
        <tr><td>Poll interval (secs):</td>
          <td><input id="pollInterval" size="5"></td>
        </tr>
        <tr><td colspan="2" height="5"></td></tr>
        </table>
        <span style="margin-right:12px">
          <a href="javascript:iContext.iScope().cancel();"
            id="_IWID_CONF_CANCEL" class="lotusAction">Cancel</a>
        </span>
        <input id="_IWID_CONF_SAVE" title="Save settings"
          class="lotusFormButton" style="margin-right:10px;"
          type="button" value="Save" name="save"
          onclick="iContext.iScope().saveParams();return false;"/>
      </div>
    ]]>
  </iw:content>
</iw:iwidget>

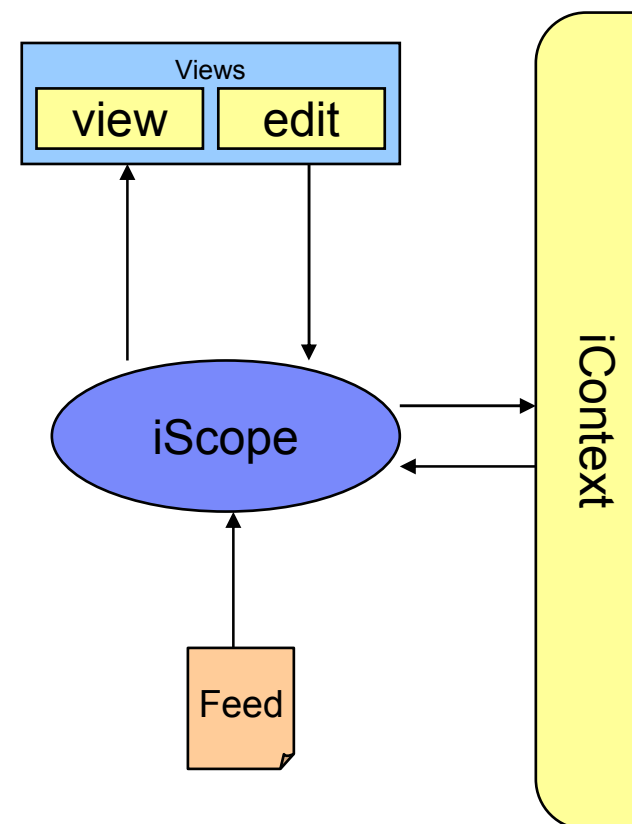
```





## The iScope class

- In model-view-controller terms you can think of the `iScope` JavaScript class as the controller for the `iWidget`.
- The `iScope` orchestrates how the `iWidget` responds to events, retrieves its data and manipulates the UI.
- An `iScope` defines a set of standard lifecycle events for an `iWidget` such as
  - `onLoad` fired when the `iWidget` instance has fully loaded.
  - `onView` fired when the view mode has been rendered.
  - `onEdit` fired when the edit mode has been rendered.
- The `iScope` also contains any event handlers for the `iWidget` to react to events occurring within the `iContext`.
  - The handlers may be wired using the XML descriptor.



## Some snippets from a sample iScope class

```
onLoad: function() {
  this.feedURL =
    this.iContext.getiWidgetAttributes().
    getItemValue("feedURL");
  console.info("feedURL = "+this.feedURL);
  this.domID = "_" +
    this.iContext.widgetId + "_";
  dojo.subscribe (
    "events/"+this.domID+"/poll",
    this, "pollForData");
}
```

```
onview: function() {
  if (this.feedURL) {
    console.info("Feed URL is
    "+this.feedURL);
    this.pollForData();
  }
}
```

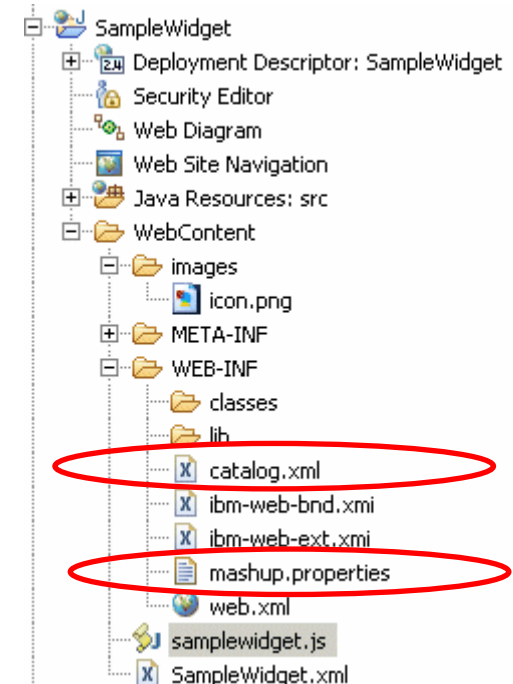
```
onedit: function() {
  this.iContext.
    getElementById("pollInterval").value =
    this.pollInterval;
}
```

## Core services provided by the iContext

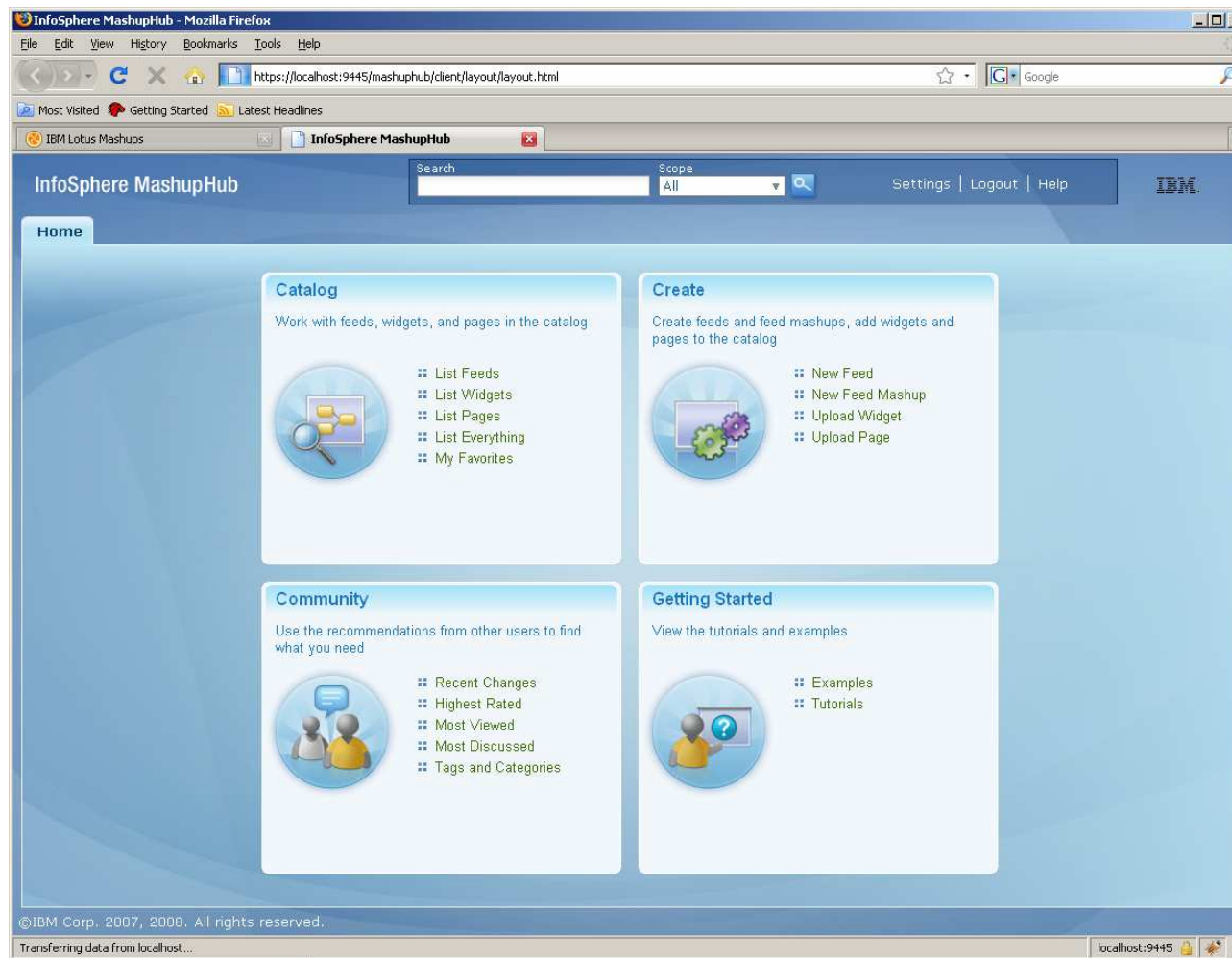
- The iContext is the view of the iWidget instance context supplied at runtime.
- The iContext provides two core mechanisms of co-ordination with other iWidgets on the page.
  - `iEvents` that can be propagated to and consumed by other iWidgets depending on their capabilities.
  - Shared state for the instance that can be interrogated or listened to by other iWidgets.
- The iContext also provides a number of services to the iWidget instance to facilitate participation in a shared page
  - `getElementById()`, `getElementByClass()` for DOM manipulation local to the iWidget.
  - `requires()` for loading resources used by an iWidget, including versioning and caching.

## Packaging the iWidget for deployment in IBM Mashup Center

- iWidgets can be deployed to IBM Mashup Center in two ways.
  - A standard JEE WAR file.
  - An OSGi bundle.
- The WAR structure is augmented with two additional configuration files for deployment in the Mashup Center.
  - A properties file indicating the context root for the iWidget.
  - A catalogue descriptor telling Mashup Center the labelling and categorisation for the iWidget in the catalog.



# Deploying an iWidget in IBM Mashup Center



Any questions?

Thank you 😊

- e-mail
  - [martin\\_gale@uk.ibm.com](mailto:martin_gale@uk.ibm.com)
- Blog
  - <http://galem.wordpress.com>