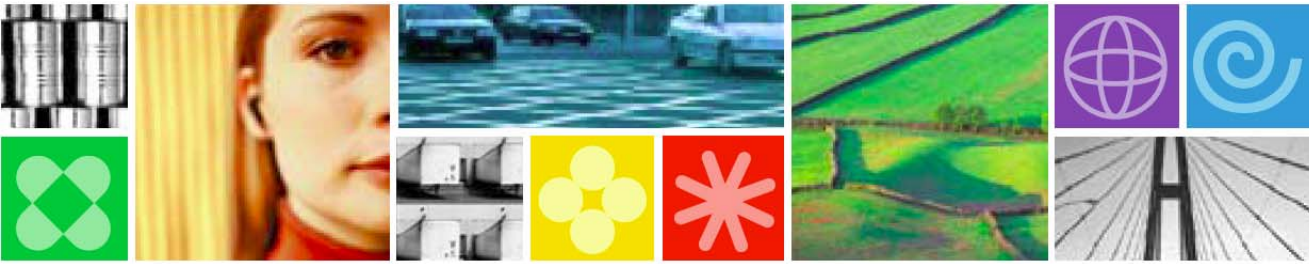
Developing Web 2.0 applications using Mashup Tools



Updated Feb, 10 2010

Agenda

08:40 - 09:40 - Build a smarter foundation for future investments
09:40 - 09:50 - Break (10 min)
09:50 - 10:50 - Smart Reuse- Transform green screens to Web, SOA, mobile, and portal
10:50 - 11:00 - Break (10 Min)
11:00 - 12:00 - Speed the development of multiplatform applications

12:00 - 01:00 - Lunch (1 hour)

1:00 - 2:00 - Developing Web 2.0 applications using Mashup Tools
2:00 - 2:10 - Break (10 Min)
2:10 - 3:10 - Smart Work on System z: Enhance teamwork with multiplatform SCM tools
3:10 - 3:20 - Break (10 Min)
3:20 - 4:20 - Let's tie it all together and play in the sandbox
3:20 - 4:30 - Close

Agenda

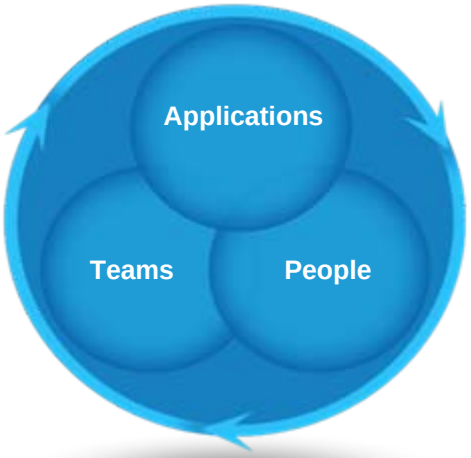
- **EGL Overview**
- **Support for Web 2.0 and Rich User Interfaces**
- **EGL CE**

EGL – Simplify Innovation

- **EGL is a modern, higher-level programming language designed for quickly developing new business applications and workload**
  - The goal is to shield developers from complexities typically associated with modern application development
  - Spend more time innovating and less time fighting with technology
- **A true cross-platform, cross-tier language targeted at all types of developers**
- **Flexible deployment options**
  - Compiles to Java, COBOL, or JavaScript
  - Deploy to JEE environments, CICS, IMS, IBM i, and more
- **Web 2.0 and SOA built-in**
- **Eclipse-based tools**
- **EGL is also an excellent target language for migrated traditional applications**

Benefits of EGL

- Flexibility**: Affords maximum platform independence and architecture support**
- Rich user interfaces: **Enables business developers (COBOL, RPG, VB, 4GL programmers) to create extremely rich, Web 2.0 user interfaces, along with Service-Oriented, multi-platform applications with a very short learning curve**
- Integration**: Enables developers to easily connect to, wrapper and extend trusted, valuable assets**
- Productivity**: Encourages developers focus on business problems, not technology problems**
- Adaptability**: Delivers a modern language that adapts more easily to changing technologies**
- Migration**: EGL is ideally suited for migration of COBOL, RPG, apps and developers**



EGL Language Example

Hello World

Basic EGL Program

hello.egl

```
1  // Hello World basic program
2
3  program hello type BasicProgram
4
5      // Data Declarations
6      name string = "World";
7
8      function main()
9          writeStdOut("Hello " +name);
10     end
11
12 end
13
```

Comments

Declare program type and name

Declare a variable and assign a value

EGL Function

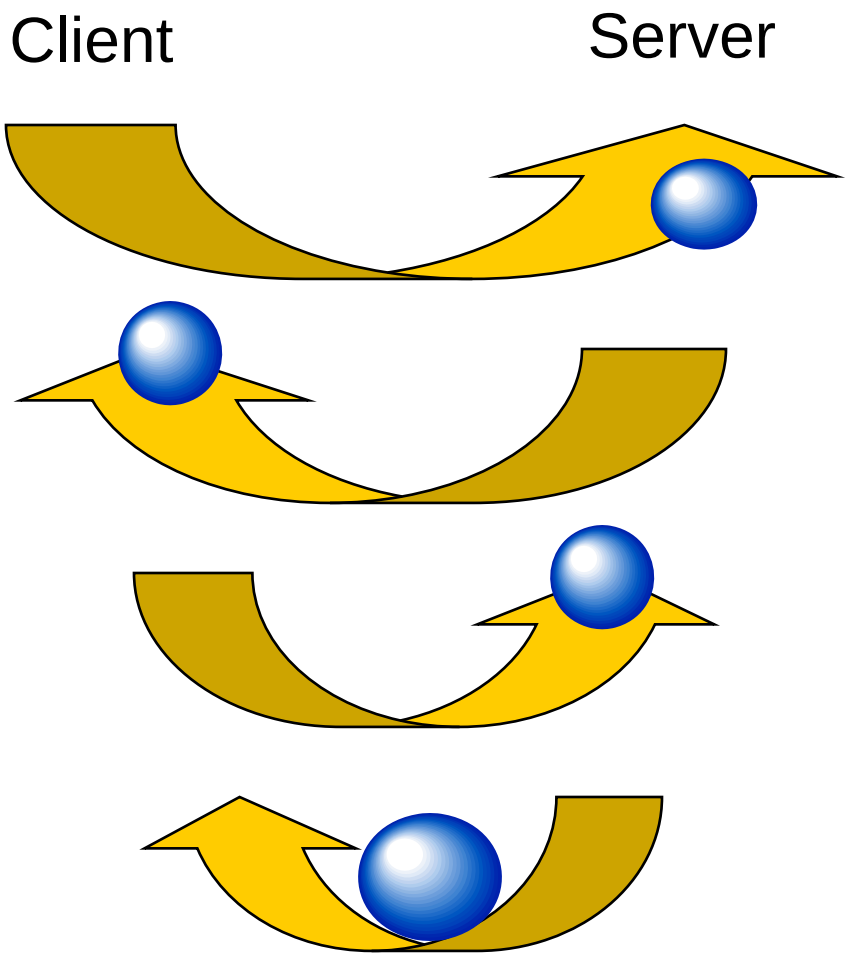
EGL Built-in Function

End of Program

Literal

Variable

Evolution of Computing



Mainframe computing

“Dumb” green screen clients
Omnipotent big mainframe servers

Client-server computing

“Smart” Personal Computer clients
Simple file and database servers

Web (1.0) computing

Light Web Browser clients
Rich application and database servers

Web 2.0 computing

Rich Internet Application clients
Lighter application and database servers

Web Applications Today

- **Web applications are no longer static, server-generated collections of pages**
- **Web 2.0 and Rich Internet Applications (RIAs) represent the next generation of Web applications**
  - Provides capabilities of a desktop application, but with the manageability of a Web application
  - Enabled by technology like JavaScript and Ajax
  - Lightweight and built on open standards
- **Provides a richer user experience, compared to traditional Web 1.0 applications**
  - Simplified, but powerful user interfaces
  - Mashed up, related data from multiple sources
  - More processing happening on the client (e.g. validation)
  - Collaborative
- Provides a new vehicle for delivering richer, more powerful business applications
  - Web 2.0 and RIAs are not just for college kids

Example Web 2.0 http://www.google.com/finance

Challenges

- **Why is it difficult to build Web 2.0 Rich Internet Applications (RIA) today?**
  - Currently domain of “tech heads”
  - Developer must learn multiple complex technologies
    - JavaScript, Ajax, JSON, SOAP
  - Compound the skill/tool silos and fragmentation
  - Most solutions are either front-end or back-end focused, but not both
    - Results in code duplication and manual efforts to keep code in sync
  - Most solutions are built on Web 1.0 style architectures
    - Not an ideal programming model for building RIAs

- RIA creation typically required lots of time, tools, and languages … until now.

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=
    <title>Google Maps API Example: Simple Geocoding</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2.x
    <script type="text/javascript">

     var map = null;
     var geocoder = null;

     function initialize() {
       if (GBrowserIsCompatible()) {
         map = new GMap2(document.getElementById("map_canvas"));
         map.setCenter(new GLatLng(37.4419, -122.1419), 13);
         geocoder = new GClientGeocoder();
       }
     }

     function showAddress(address) {
       if (geocoder) {
         geocoder.getLatLng(
           address,
           function(point) {
             map.setCenter(point, 13);
             var marker = new GMarker(point);
             map.addOverlay(marker);
             marker.openInfoWindowHtml(address);
           }
         );
       }
     }
    </script>
  </head>

  <body onload="initialize()" onunload="GUnload()">
    <form action="#" onsubmit="showAddress(this.address.value); return false"
      <p>
        <input type="text" size="60" name="address" value="1600 Pennsylvania
Washington DC" />
        <input type="submit" value="Go!" />
      </p>
      <div id="map_canvas" style="width: 500px; height: 300px"></div>
    </form>
  </body>
</html>
```

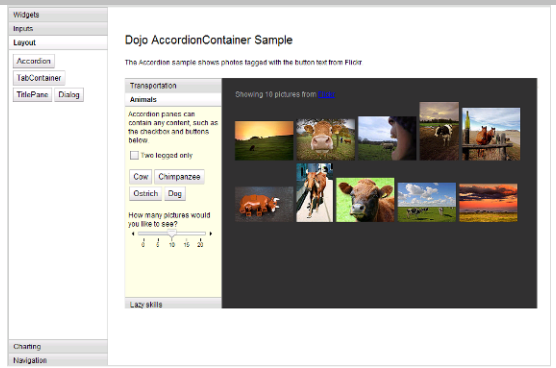
Rich User Interfaces with EGL

- **Simplify creation of Rich Internet Applications**
  - Deliver end-to-end Web 2.0 quickly in a single language
  - Build rich user interfaces to modernize existing applications
- **Generates standard JavaScript and Ajax**
  - EGL does NOT replace HTML or JavaScript
- **Easy-to-learn language**
- **Fully open and extensible**
- **Use a rich, extensible widget library**
  - **Including support for Dojo**
- **Eclipse-based development, testing, and debugging**
- **Consume all types of Web services**

EGL + Rich UI - Open and Extensible

- **Fully open and extensible**
  - Utilize existing Java or JavaScript libraries if needed
- **Rich UI based on Web Standards**
  - REST, SOAP, JSON, OpenAjax, Dojo, etc
- **UI Libraries at the EGL Café**
  - Download third-party libraries
  - Write your own and upload them
  - Import into the visual editor palette
- **Plans for open implementation**
  - Allow third parties to extend EGL, develop their own version

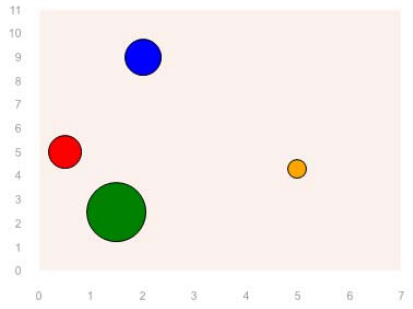
Dojo Support



- **Dojo is a popular and powerful open source JavaScript library used throughout the Web**
- **IBM has created a sample EGL Dojo widget library that enables developers to easily use Dojo widgets within their EGL applications**
  - No knowledge of Dojo or JavaScript required
  - Fits within the EGL programming model
  - Demonstrates extensibility of EGL architecture
  - Enables faster development
  - Available as a sample on the EGL Café and is included in EGL Community Edition

Code sample showing creation of a rich bubble chart:

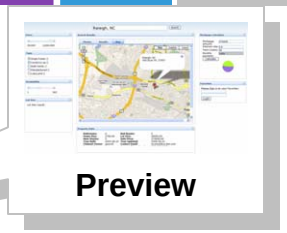
```
new DojoBubbleChart{
    themeColor = dojo.widgets.DojoLib.CHART_COLOR_THEME_ORANGE,
    minX = "0", maxX = "7",
    minY = "0", maxY = "11",
    width = "400",
    height = "300",
    data = data
}
data BubbleChartData[] = [
    new BubbleChartData { x=0.5, y=5,   size=1.4, color="red",    tooltip="Gas" },
    new BubbleChartData { x=1.5, y=2.5, size=2.5, color="green",  tooltip="Mortgage" },
    new BubbleChartData { x=2,   y=9,   size=1.5, color="blue",   tooltip="Electric" },
    new BubbleChartData { x=5,   y=4.3, size=0.8, color="orange", tooltip="Cable" }
];
```



Provided Widgets

- **Accordion Container**
- **Bar Graph**
- **Bubble Chart**
- **Button**
- **Check Box**
- **Color Palette**
- **Combo Box**
- **Content Pane**
- **Context Menu**
- **Currency Text Box**
- **Date Text Box**
- **Dialog**
- **Grid**
- **Horizontal Slider**
- **Line Graph**
- **Menu**
- **Pie Chart**
- **Progress Bar**
- **Radio Group**
- **Tab Container**
- **Time Text Box**
- **Title Pane**
- **Tree**

EGL Rich UI Development and Deployment



Developer
Workbench
(RDz with EGL)

Service calls to existing CICS Web, EGL, PHP, SOAP, or REST services are made using Ajax

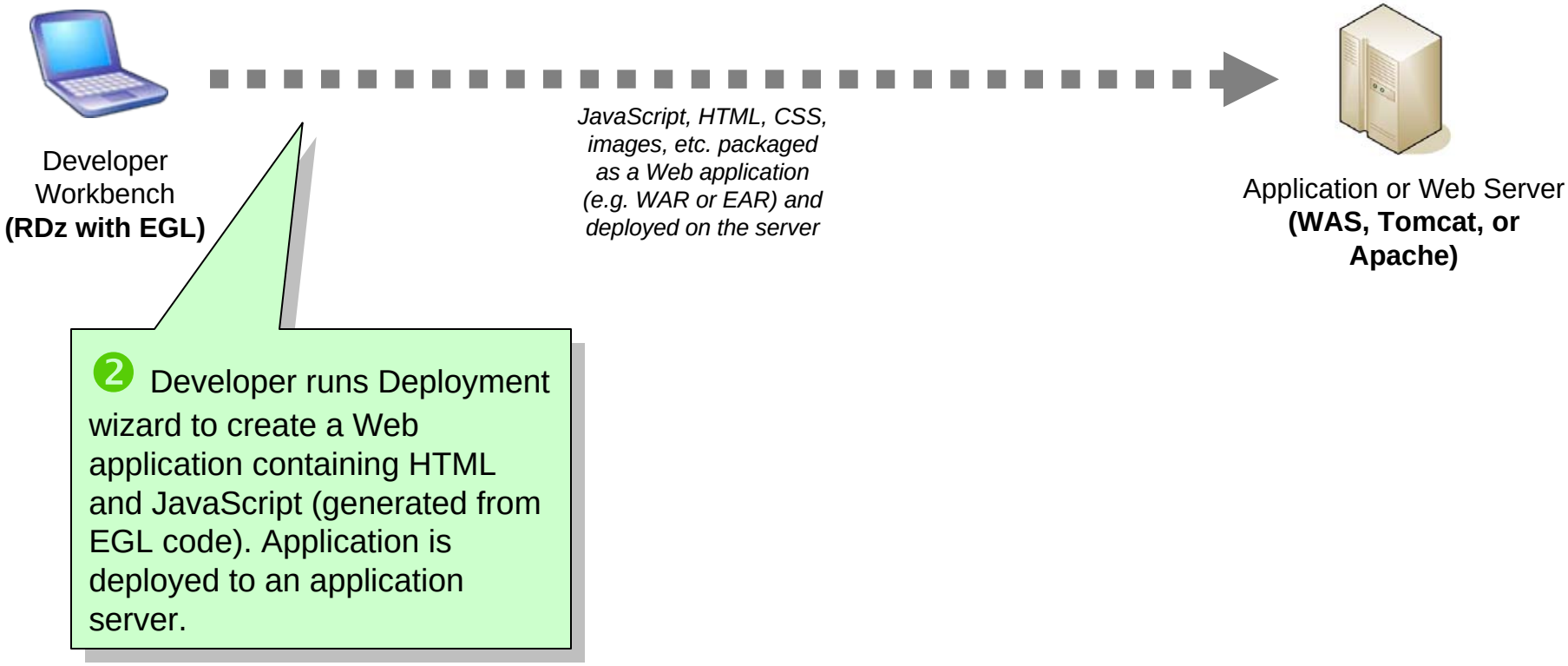
❶ Developer uses Eclipse-based EGL tooling (such as RDz with EGL IDE) to code, test, and debug application on their workstation.

Internet / Intranet

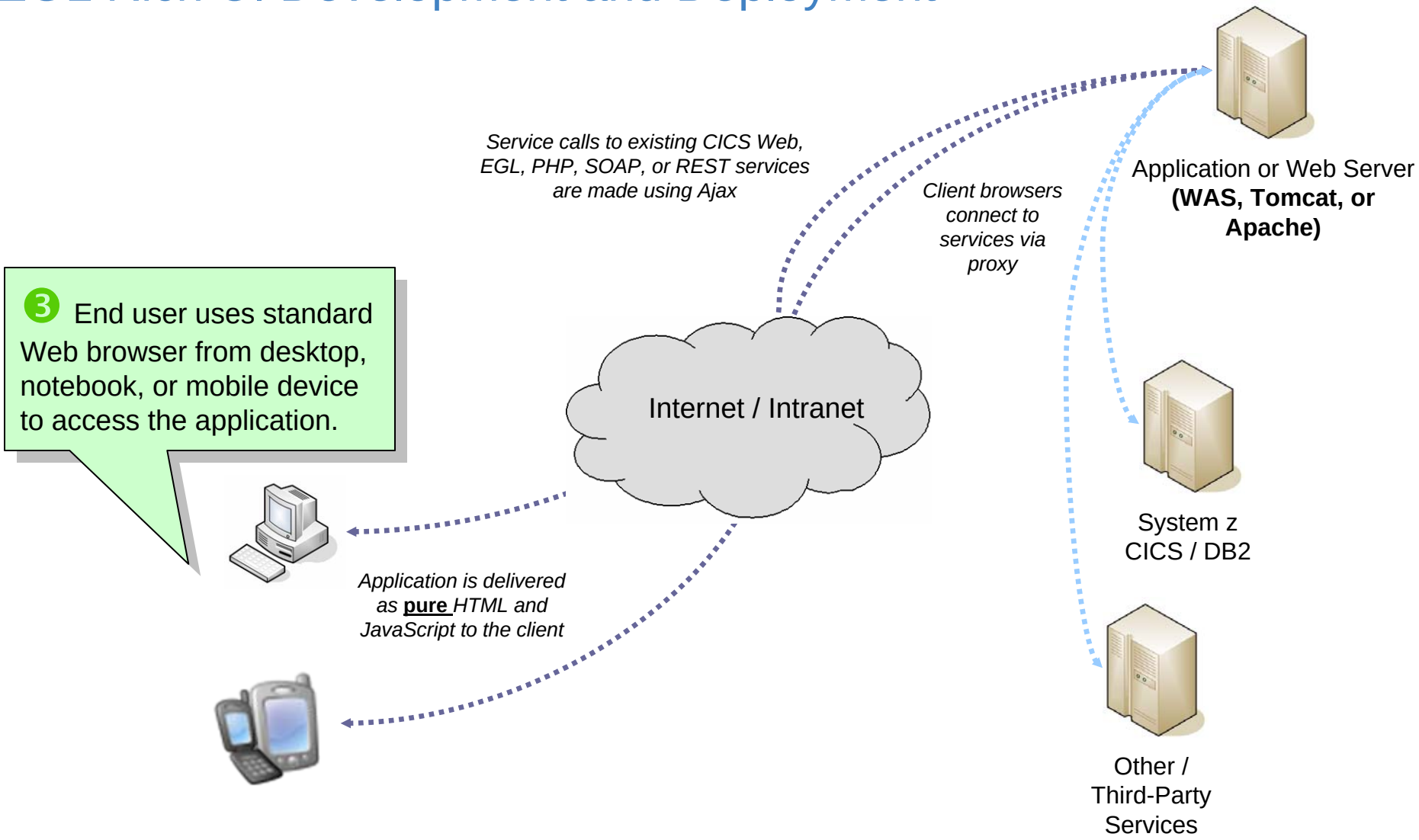
System z
CICS / DB2

Other /
Third-Party
Services

EGL Rich UI Development and Deployment



EGL Rich UI Development and Deployment



EGL in Action (Side-by-Side Comparison)

EGL Rich UI

```
handler MyRuiHandler type RUIhandler { initialUI = [ addressForm,
map ] }

 addressField TextField { text = "1600 Pennsylvania Ave, Washington
DC", width = 250 };

 goButton Button { text = "Go!", onClick ::= goButton_clicked };
 addressForm Box { children = [ addressField, goButton ] };

 map GoogleMap { width = "500px", height = "300px" };

  function goButton_clicked (e Event in)
    addresses String[] = [ addressField.text ];
    map.showAddresses(addresses, addresses);
  end
end
```

HTML and JavaScript

```
<html xmlns="htt                    ="urn:schemas-microsoft-
com:vml">
  <head>
    <meta http-                     html; charset=UTF-8"/>
    <title>Goog                     g</title>
    <script sr                      api&amp;v=2.x

    var map = nu
    var geocoder = nu

    function initialize() {
      if (GBrowserIsCompatible()) {
        map = new GM                        vas"));
        map.setCe
        geocoder

    function s
      if (geocode
        geocoder
          address
          functi
            map.s
            var m
            map.a
            marke
          }
        );
      }
    }
    </script>
  </head>

  <body onload="initialize()" onunload="GUnload()">
    <form action="#" onsubmit="showAddress(this.address.value); return
false">
      <p>
        <input type="text" size="60" name="address" value="1600 Pennsylvania
Ave, Washington DC" />
        <input type="submit" value="Go!" />
      </p>
      <div id="map_canvas" style="width: 500px; height: 300px"></div>
    </form>
  </body>
</html>
```

All code, including UI and controller logic, is written completely in EGL.

The complexity of the Google Map APIs are hidden from the developer, so the developer can focus on the actual business requirement and not technical complexities.

Developing RIAs by hand requires developers to become experts in multiple technologies – HTML and JavaScript. Neither was designed for the kinds of applications being developed today!

EGL Rich UI Example (Server Side)

EGL has a **service** keyword that enables developers to define services, which are then compiled into Java and deployed as a REST or SOAP service.

EmployeeService.egl

```
package services;

service EmployeeService

    // Returns all employee records from the database
    function getRecords() returns (EmployeeRecord[])
        employees EmployeeRecord[0];
        get employees;
        return (employees);
    end

    // Adds a new employee to the database
    function addRecord(newEmployee EmployeeRecord in)
        add newEmployee;
    end

end

record EmployeeRecord type SQLRecord {tableNames = [["EMPLOYEE"]], keyItems=

    EMPNO string            {column="EMPNO", maxLen=6};
    FIRSTNME string         {column="FIRSTNME", sqlVariableLen=yes, maxLen=12};
    MIDINIT string          {column="MIDINIT", isSqlNullable=yes, maxLen=1};
    LASTNAME string         {column="LASTNAME", sqlVariableLen=yes, maxLen=15};
    HIREDATE date           {column="HIREDATE", isSqlNullable=yes};
    SALARY decimal(9,2)     {column="SALARY", isSqlNullable=yes};

end
```

Functions declared in services are available to be called externally. In this example, the **getRecords** function returns an array of all employee records.

EGL makes it simple to interact with databases. In this example, the "**get**" keyword is used to populate an array of employee records from a database (connection settings are stored outside the code). Other keywords (like "add" and "update") are used to easily add new records to the database or update an existing record.

Records are EGL parts that represent data. In this example, the **EmployeeRecord** part is an SQL record, which means it is tied to a table (or tables) in a database. As you can see, the table name is specified and fields within the record are bound to columns in the table.

EGL Rich UI Example (Browser Side)

The user interface is written completely in EGL (not HTML). This **code is compiled into JavaScript and HTML** on-the-fly during development.

EgleeView.egl

```
package uis;

import egl.ui.rui.*;
import dojo.widgets.*;
import services.*;

handler EmployeeView type RUIhandler {initialUI = [ displayButton, Grid ] }

    // Button (when clicked, service will be called and data will be displayed)
    displayButton DojoButton{ text = "Display", onClick ::= displayButton_onClick };

    // Dojo grid
    Grid DojoGrid{
        columns = [
            new DojoGridColumn { displayName = "First Name", name = "FIRSTNME"},
            new DojoGridColumn { displayName = "Last Name", name = "LASTNAME"},
            new DojoGridColumn { displayName = "Salary", name = "SALARY"}
        ]
    };

    // Function called when Display button is clicked
    function displayButton_onClick(event Event in)
        myService EmployeeService {}; // service instance
        call myService.getRecords() returning to displayGrid; // asynchronous service call
    end

    function displayGrid(retResult EmployeeRecord[] in)
        Grid.data = retresult as any[]; // display results in the grid
    end

end
```

Design Source Previe

Notice how functions are bound to event types on UI widgets. In this example, when this button is clicked, the **displayButton_onClick** function is called.

Notice how columns are declared on the Dojo grid. The “name” field tells EGL which field in the record (in this case, “EmployeeRecord”) to display in the column.

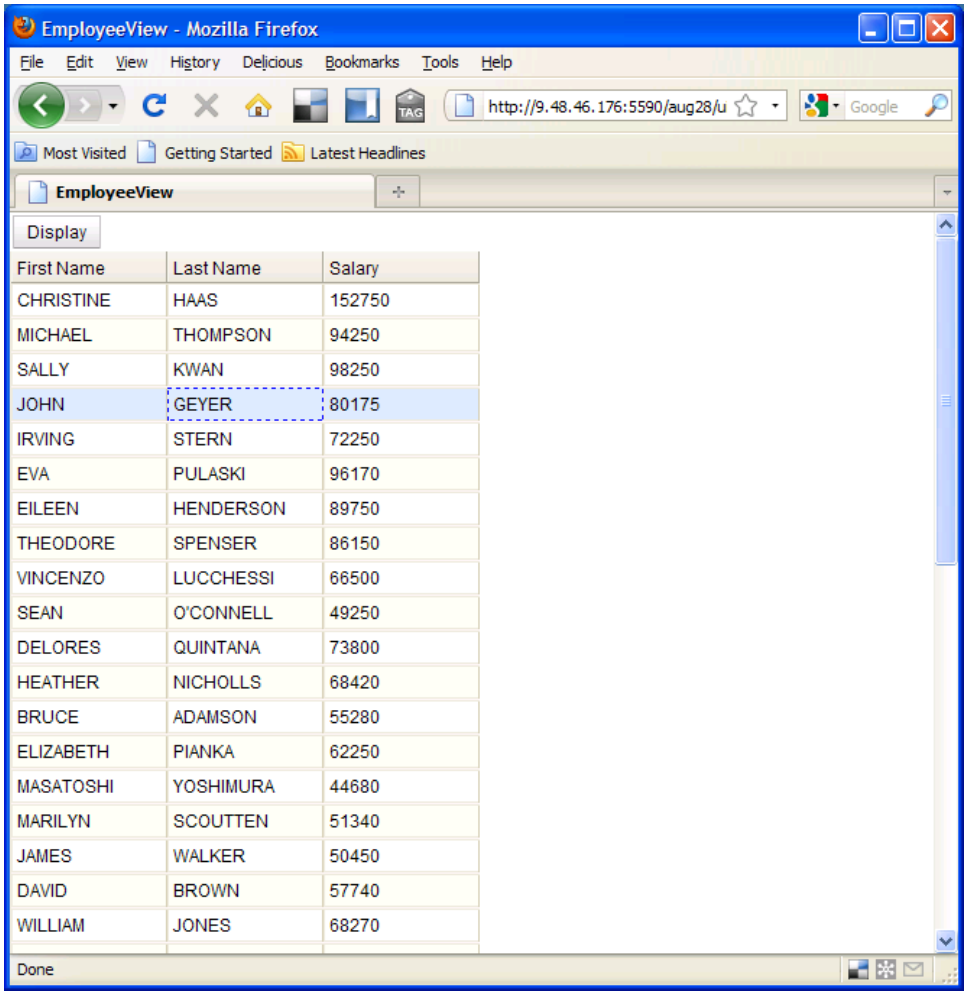
EGL uses a **declarative programming** style to make creating new objects (in this case, UI widgets) easy. In this example, a simple button and Dojo grid are defined.

Notice how an instance of the previously-created service is declared directly in our UI code. Also, notice the **call** statement that asynchronously invokes the **getRecords** method. This statement will be compiled into a **JavaScript Ajax** statement.

The array of employee record is passed directly to the Dojo grid widget (although this record is defined in our service code, it will be compiled into JavaScript since it is referenced by the UI code). The grid widget will automatically populate the grid based on the columns defined earlier and the data in the records.

EGL Rich UI Example (End Result)

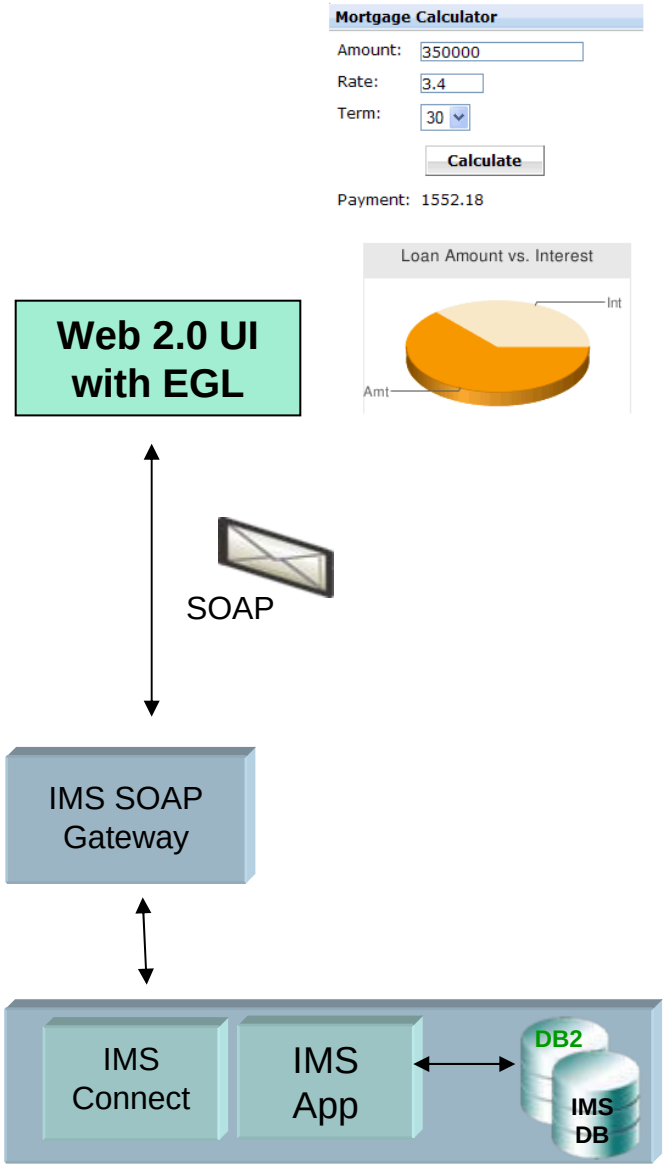
- **Clicking the Display button will cause a Web service to be invoked on the server. This service will pull records out of a database table and return them to the client. Once returned, the records will be displayed in the Dojo grid.**
- **Key points:**
  - Data can be represented the same way in both server and client code.
  - Web services can be easily created and invoked from the client side.
  - EGL makes it simple to interact with a database.
  - EGL does not replace HTML or JavaScript!
- **EGL allows you to** spend more time innovating and less time fighting with technology!



First Name	Last Name	Salary
CHRISTINE	HAAS	152750
MICHAEL	THOMPSON	94250
SALLY	KWAN	98250
JOHN	GEYER	80175
IRVING	STERN	72250
EVA	PULASKI	96170
EILEEN	HENDERSON	89750
THEODORE	SPENSER	86150
VINCENZO	LUCCHESSI	66500
SEAN	O'CONNELL	49250
DELORES	QUINTANA	73800
HEATHER	NICHOLLS	68420
BRUCE	ADAMSON	55280
ELIZABETH	PIANKA	62250
MASATOSHI	YOSHIMURA	44680
MARILYN	SCOUTTEN	51340
JAMES	WALKER	50450
DAVID	BROWN	57740
WILLIAM	JONES	68270

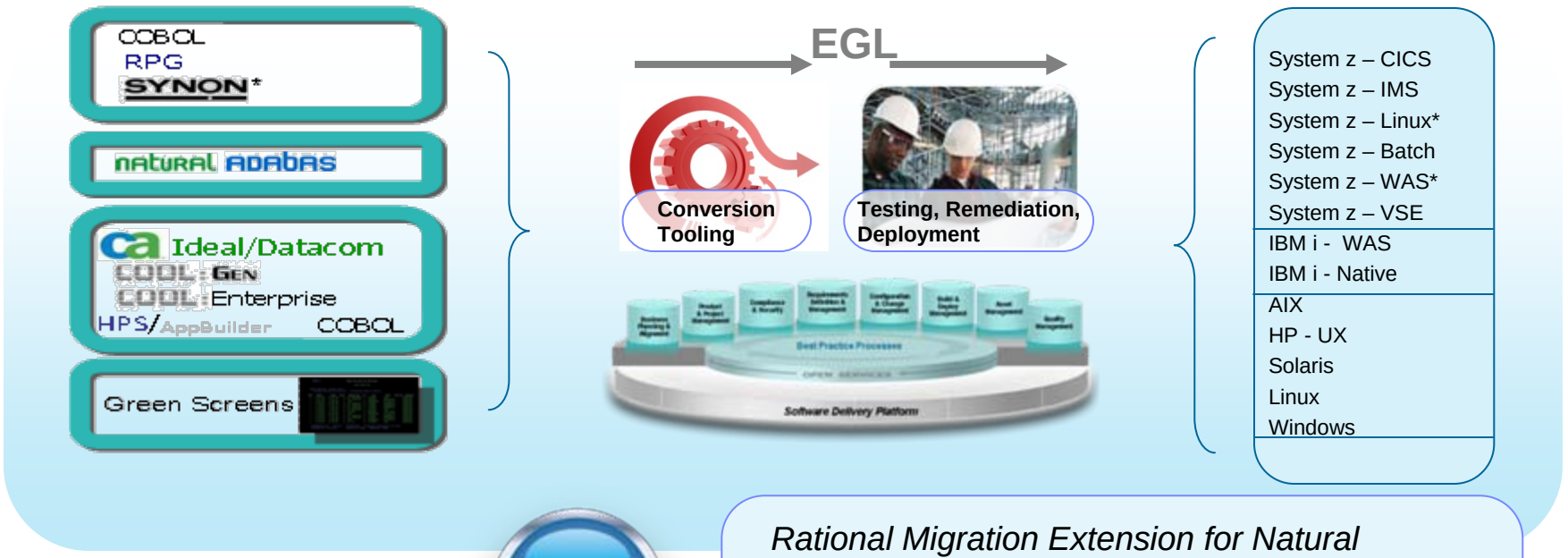
EGL and IMS

- **Integrate Web services from IMS SOAP Gateway to build rich Web 2.0 user interfaces that incorporate popular widget libraries like Dojo.**
- **Generated COBOL code can run in:**
  - Message Processing regions
  - Batch Message Processing (BMP) regions
  - IMS Fast Path (IFP) regions
- **Quickly build new Text UIs (TUIs)**
- **Use EGL data access keywords such as get, add, replace, delete to easily work with data.**
  - DL/I support with tooling to customize DL/I statements
- **Full transaction support**
- **Invokes IMS programs remotely**
  - EGL handles transaction invocation and data conversion
- **Work with serial and print files**
- **Use a single IDE (Rational Developer for System z with EGL) to develop EGL and IMS solutions.**



Application transformation tools and services

- Reduce maintenance and support costs, enable platform flexibility, and move to a modern language and tool set with Rational Migration Extension
  - **Enable modern development with EGL's built-in support for Web 2.0 and SOA**
  - **Gain the benefits of integration with other modern tools from Rational**



WEB 2.0

Rational Migration Extension for Natural
Rational Migration Extension for CA Technologies
Rational Migration Extension for Rich UI

EGL Community Edition

- **Simplify development of Web 2.0 solutions –** for free!
  - Code client and server side code is in one language (EGL)
- **Eclipse-based development environment**
  - WYSIWYG visual editor
  - Instant previewing without deploying to a server
  - Full debug for client and server-side code
  - Small download, simple install
- **Rich Web user interfaces using Dojo**
  - Fully extensible architecture
- **Build Java-based Web services without the Java**
  - Take advantage of EGL's powerful keywords for accessing data in most popular databases, including MySQL
- **Fully functional, but no-cost**
- Spend more time innovating and less time fighting with technology!

Statistics

- **1,200 downloads** in the first 5 days!
- Thousands of views of "Hello World with EGL" video on YouTube
- Press coverage: InfoWorld, SearchSOA, The Register, InternetNews.com, Dr. Dobb's, System i Network, D'Technology, and others.

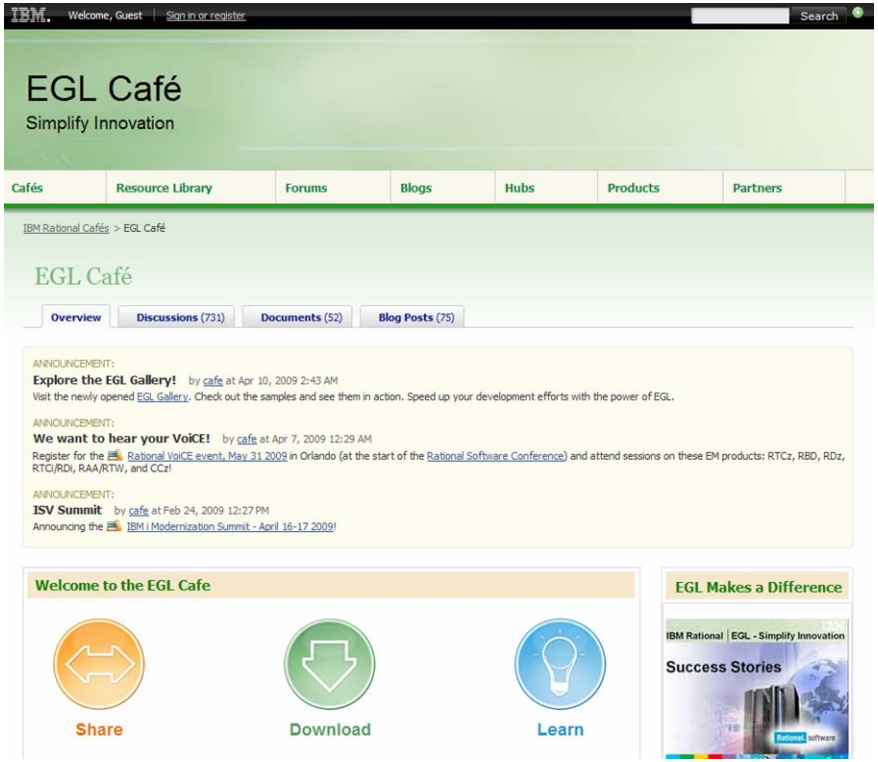
Check out the EGL CE video on YouTube!

Download EGL Community Edition today!
http://**www.ibm.com**/software/rational/cafe/community/egl/ce

EGL Café

- **Online community for EGL developers, partners, and clients**
- **Discussion forums**
- **Gallery of sample applications and widgets**
- **Presentations, videos, and articles**
- **Blogs by IBMers and partners**
- **Success stories**
- **Become part of the community today!**

http://ibm.com/rational/eglcafe



Resources: Download, Learn, Presentations, Video/viewlet, Sample Code

Community: Clients, Partners, Influencers, Press, News and Events

Collaboration: Blogs, Forums, Tips and Techniques Comments, Ratings

Testimonials: Case Studies, Celebrations!

Proof Of technology

Discovering the value of EGL to develop cross-platform applications and accelerate the adoption of Web 2.0

Presentations

- *** Day 1 Presentation topics ***
- - Introduction
- - EGL Overview
- - EGL Details, Using the Web Wizards
- - Generating Java or COBOL
- *** Day 2 Presentation topics ***
- - Integrating with existing Legacy systems (Java, COBOL/CICS, RPG, etc..)
- - EGL and SOA (Service Oriented Architecture)
- - EGL and Web 2.0
- - EGL and UML (Unified Modeling Language)
- - On Request: Migrating to EGL (from CSP or Vis
- - Conclusions

Workbooks

- *** Day 1 Labs ***
- - LAB 1 - Create, test and debug an EGL Server progr
- - LAB 2 - Using Java Server faces (JSF) with EGL to c
- (Optional) - Using Text User Interface to call EGL Ser
- - LAB 3A - Generating Java from JSF/EGL Client and Server programs).
- Or
- - LAB 3B - Generating COBOL from EGL Server program.
- *** Day 2 Labs ***
- - LAB 4 – Using a wizard to create a Web application that accesses a databases.
- - LAB 5 – Calling COBOL from EGL – An example using COBOL CICS.
- Or
- - LAB 6 - Calling Java from EGL.
- Or
- - LAB 6B - Calling RPG from EGL
- - LAB 7 - Creating and Consuming Web Services with EGL
- - LAB 7A - Building a Web 2.0 application with EGL
- - LAB 7B - Modernize existing CICS with SOA and Web 2.0 using Rational Developer for System z with EGL.
- Or
- - LAB 7C - Modernize existing RPG with SOA and Web 2.0 using EGL and Rational Developer for i for SOA.
- - LAB 8 - UML to EGL Transformation, using Rational Software Architect and Rational Business Developer.
- - LAB 9 - Using EGL to work with MQ and generating COBOL/CICS
- - LAB 10 - (on request) Migrating VisualAge Generator to EGL (Optional).

Duration : 2 days

Free of Charge

POT Link

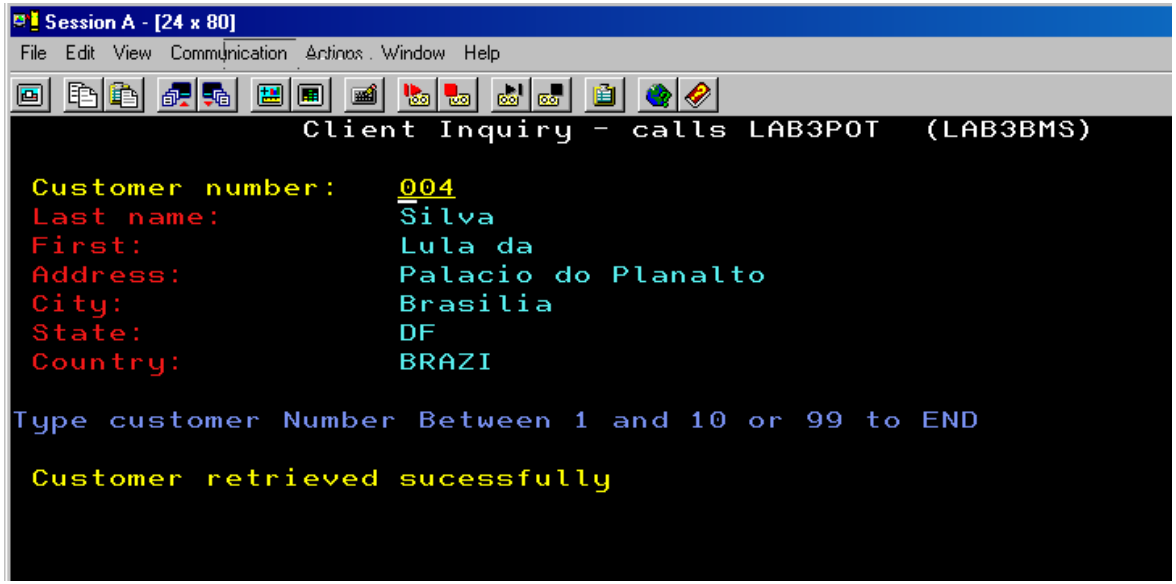
http://nasoftware.ibm.com/tec/assets.nsf/doc/BCUO-7P8M9D

Questions?

Moving legacy COBOL/CICS/VSAM to iPhone

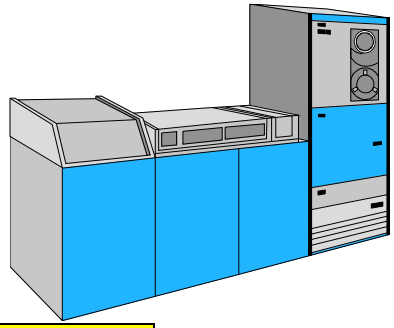
http://testiphone.com/?url=http://zserveros.demos.ibm.com:9080/iPhone/CICS-en_US.html

Existing **COBOL/CICS 3270 Application**



z/OS Texas

ZSERVEROS



EXEC CICS LINK LAB3POT

COBOL Client

EXEC CICS READ
FILE ('POTVSAM')
INTO (WORK)

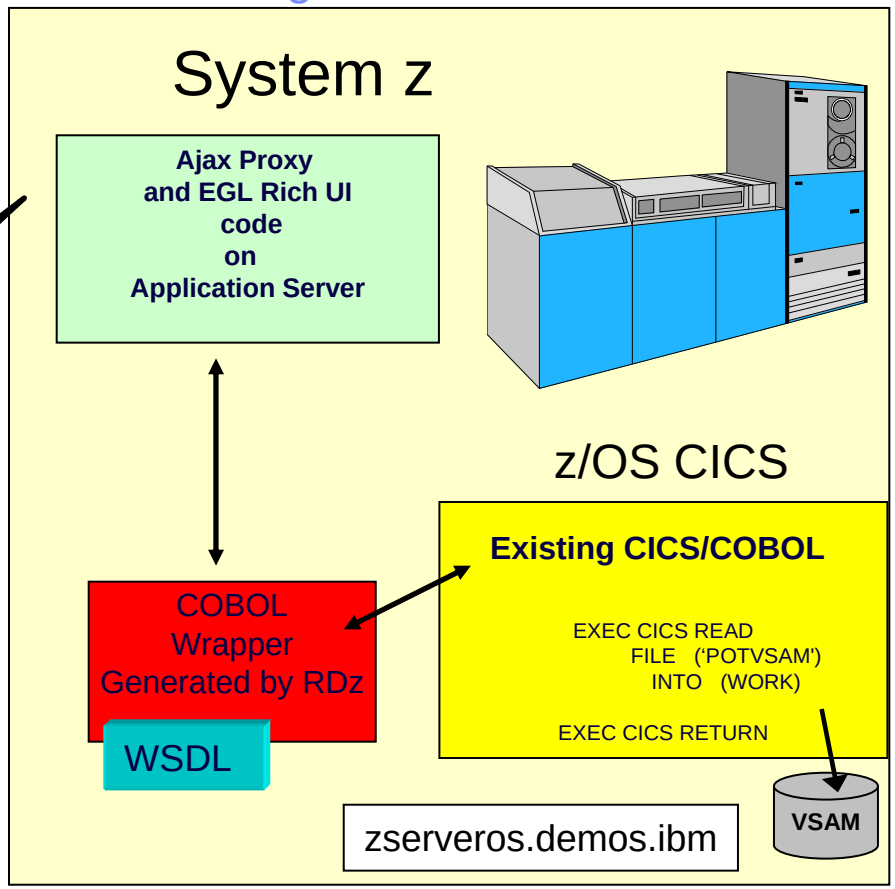
EXEC CICS RETURN

VSAM

COBOL Server

Link to PCOM

Invoking CICS Web services from iPhone



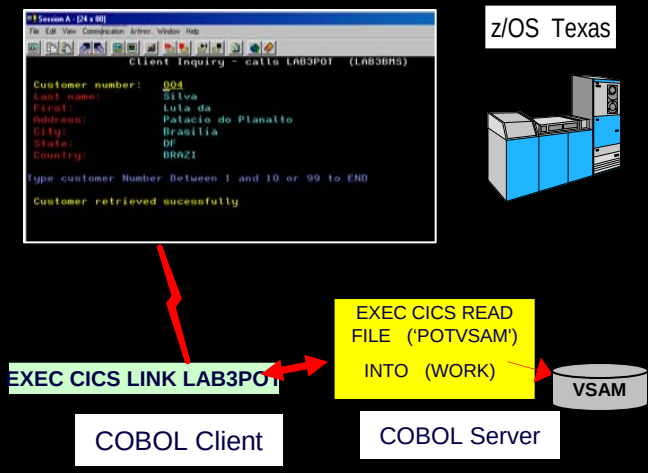
rbarosa@us.ibm.com

http://zserveros.demos.ibm.com:9080/iPhone/egl.html

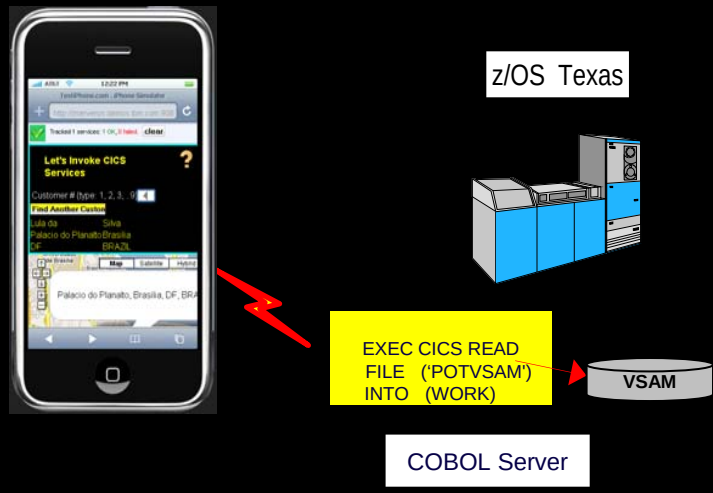
More at:
http://www.ibm.com/developerworks/websphere/techjournal/0909_barosa/0909_barosa.html

9 Minutes

PART #1. Create a CICS Web Service and WSDL using Rational Developer for System z (RDz)

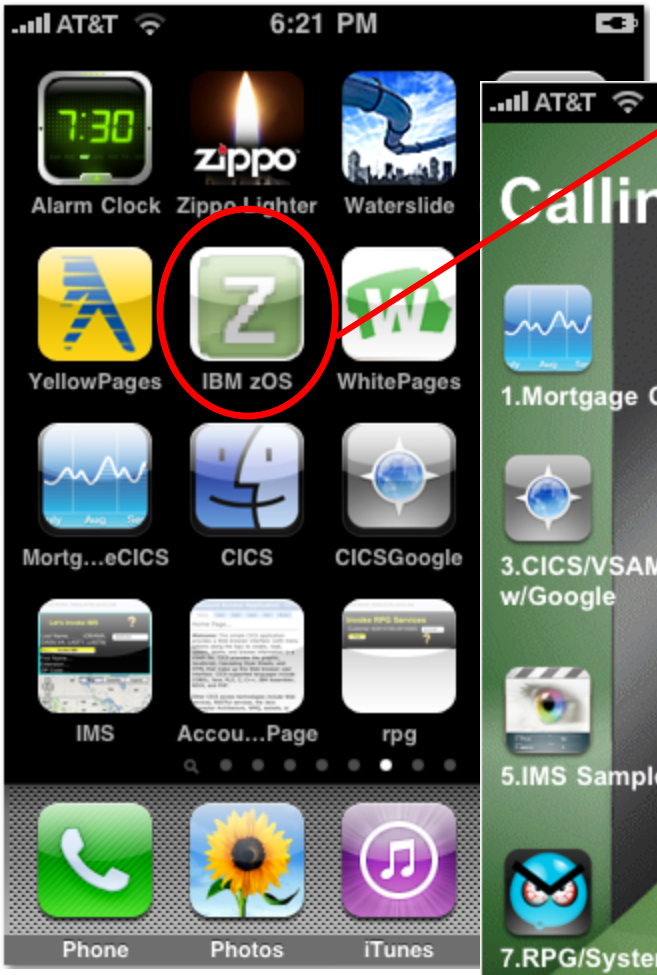


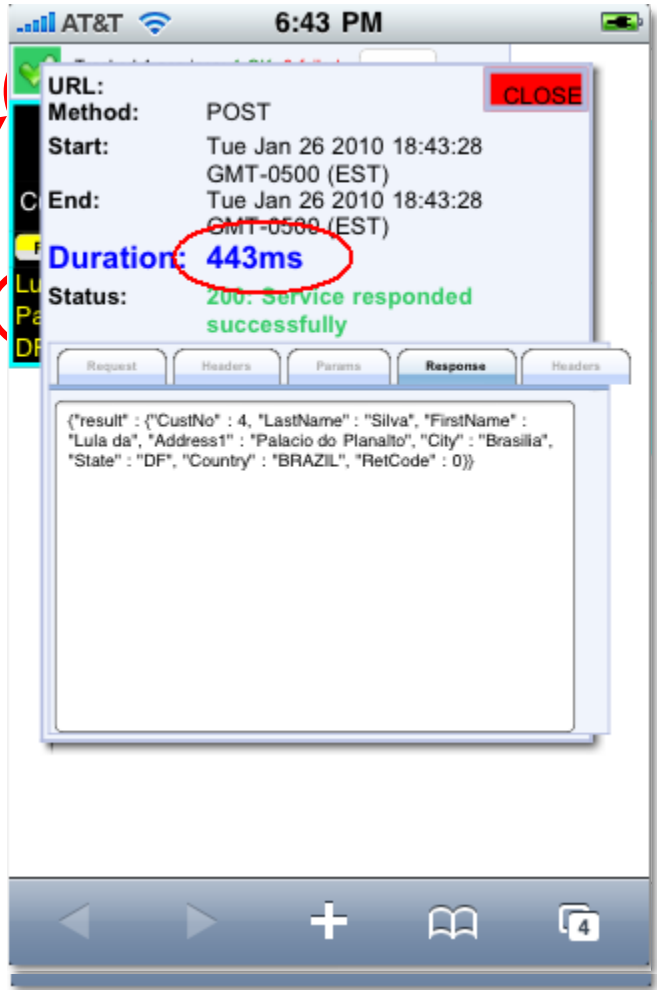
PART #2. Create a Web 2.0 Interface using Rational Developer for System z with EGL (RDz/EGL)



Invoking CICS Web services from iPhone

(Real iPhone screen captures)





http://zserveros.demos.ibm.com:9080/zos/egl.html

Thank You

Agenda

08:40 - 09:40 - Build a smarter foundation for future investments
09:40 - 09:50 - Break (10 min)
09:50 - 10:50 - Smart Reuse- Transform green screens to Web, SOA, mobile, and portal
10:50 - 11:00 - Break (10 Min)
11:00 - 12:00 - Speed the development of multiplatform applications

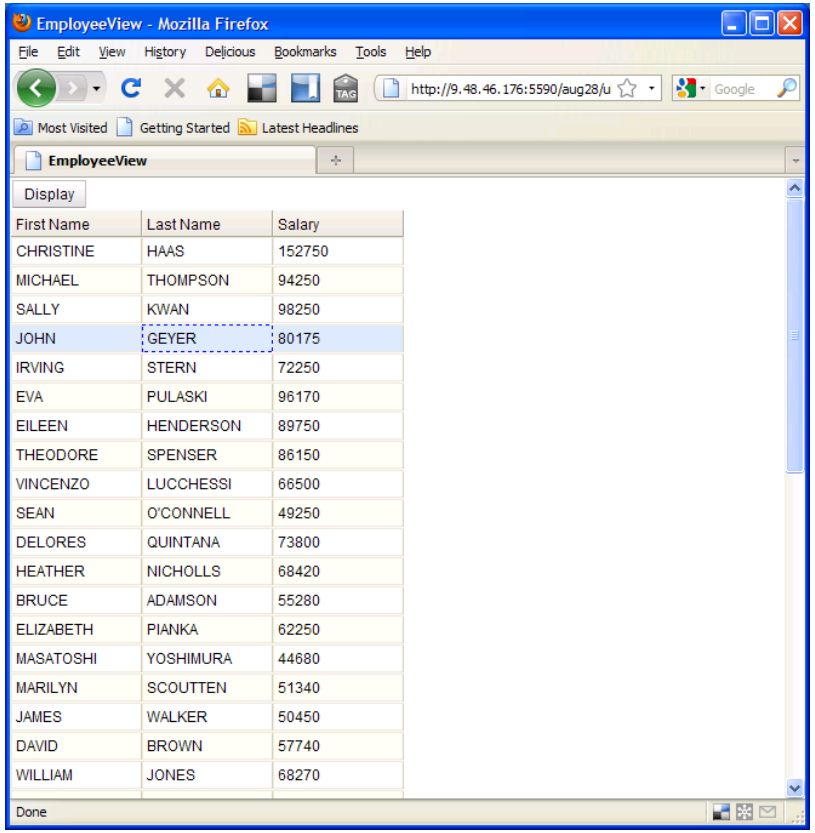
12:00 - 01:00 - Lunch (1 hour)

1:00 - 2:00 - Developing Web 2.0 applications using Mashup Tools
2:00 - 2:10 - Break (10 Min)
2:10 - 3:10 - Smart Work on System z: Enhance teamwork with multiplatform SCM tools
3:10 - 3:20 - Break (10 Min)
3:20 - 4:20 - Let's tie it all together and play in the sandbox
3:20 - 4:30 - Close

Orginal Slides - Backup

EGL Rich UI Example

- **Objective: create a simple Web 2.0 style application to display data from a database in a Dojo grid.**



First Name	Last Name	Salary
CHRISTINE	HAAS	152750
MICHAEL	THOMPSON	94250
SALLY	KWAN	98250
JOHN	GEYER	80175
IRVING	STERN	72250
EVA	PULASKI	96170
EILEEN	HENDERSON	89750
THEODORE	SPENSER	86150
VINCENZO	LUCCHESSI	66500
SEAN	O'CONNELL	49250
DELORES	QUINTANA	73800
HEATHER	NICHOLLS	68420
BRUCE	ADAMSON	55280
ELIZABETH	PIANKA	62250
MASATOSHI	YOSHIMURA	44680
MARILYN	SCOUTTEN	51340
JAMES	WALKER	50450
DAVID	BROWN	57740
WILLIAM	JONES	68270

EGL Rich UI and Services

- Services are the key to any modern, flexible IT architecture
- **EGL Rich UI is built on services**
  - All interaction from the client (browser) to the server is performed via Web service calls
- **EGL includes first-class support for creating and consuming Web services**
  - EGL provides a “service” keyword – a developer simply codes the functions/methods desired for the service
    - The type of service (REST, SOAP, CICS, etc) does not need to be decided up-front
  - Existing services can be easily consumed in EGL
    - All without requiring the developer to learn the details of SOAP and HTTP messaging
- **Rational provides tool to expose existing logic on enterprise systems as services**
  - Services from existing RPG programs can be created with Rational Developer for i for SOA Construction