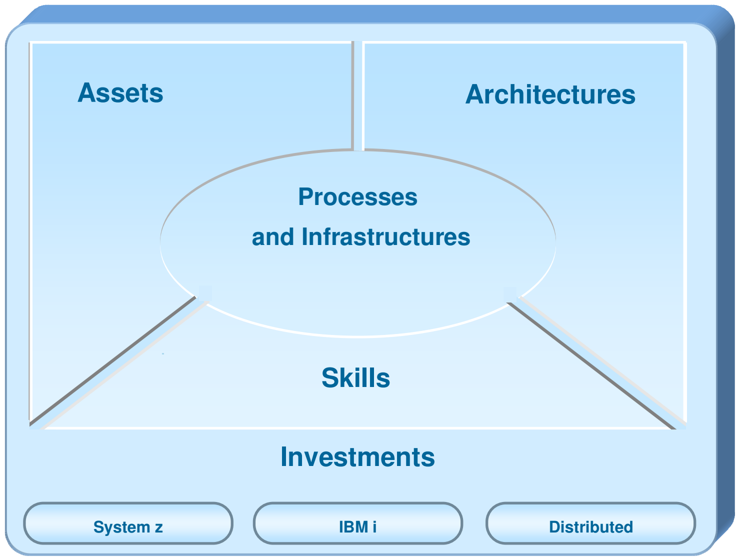
Modern Application Development Featuring Web 2.0 for System z

Rational Business Developer and EGL

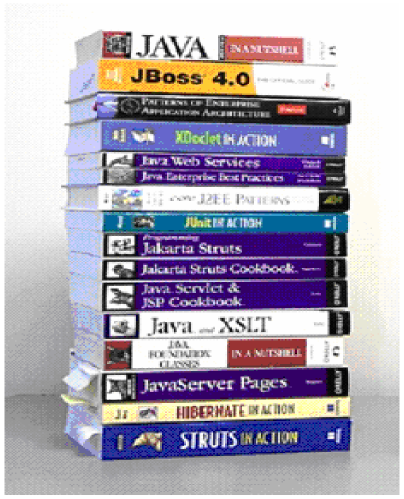
Rational. software

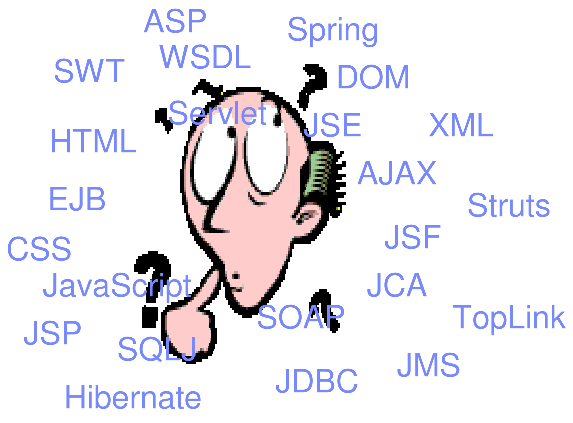
EGL for Enterprise Modernization

- Enables COBOL, RPG, 4GL developers to create Web applications and SOA solutions with minimum learning curve
- Provides a modern programming paradigm for “legacy” platforms, attract new staff to your trusted box
- Enables to easily connect to, wrapper and extend trusted valuable assets
- Affords maximum flexibility of deployment options and architectures
- Delivers a modern language that adapts more easily to changing technologies
- Can be used as a target for legacy programs conversion
- Enables migration from Natural or RPG



Why EGL? Because building applications today is not easy

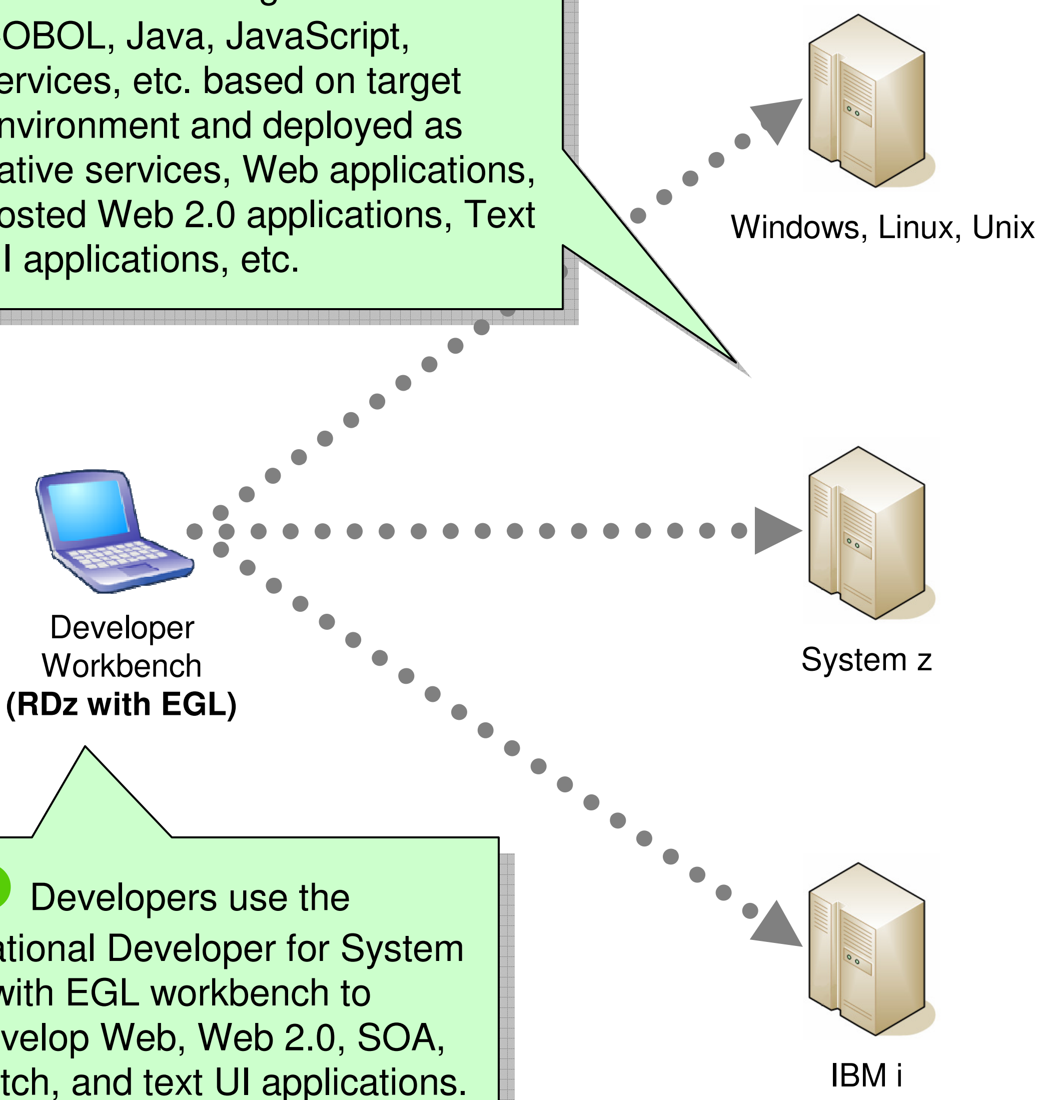




- Learn and master myriads of
  - programming languages and semantics
  - middleware interfaces
  - programming paradigms and styles
  - frameworks and libraries
- Constantly emerging new technologies

- Code at a more abstract and simpler level
- Easy to learn, modern and comprehensive language
- Keeps up with emerging technologies
- Inter-operates seamlessly with legacy

❷ EGL code is generated as COBOL, Java, JavaScript, services, etc. based on target environment and deployed as native services, Web applications, hosted Web 2.0 applications, Text UI applications, etc.



Java and JavaScript generation
- WebSphere Application Server
- Apache Tomcat
- Java Runtime Environment

Java generation
- WebSphere Application Server
- USS
- z/Linux

JavaScript generation
- WebSphere Application Server

COBOL generation
- Batch
- CICS
- IMS

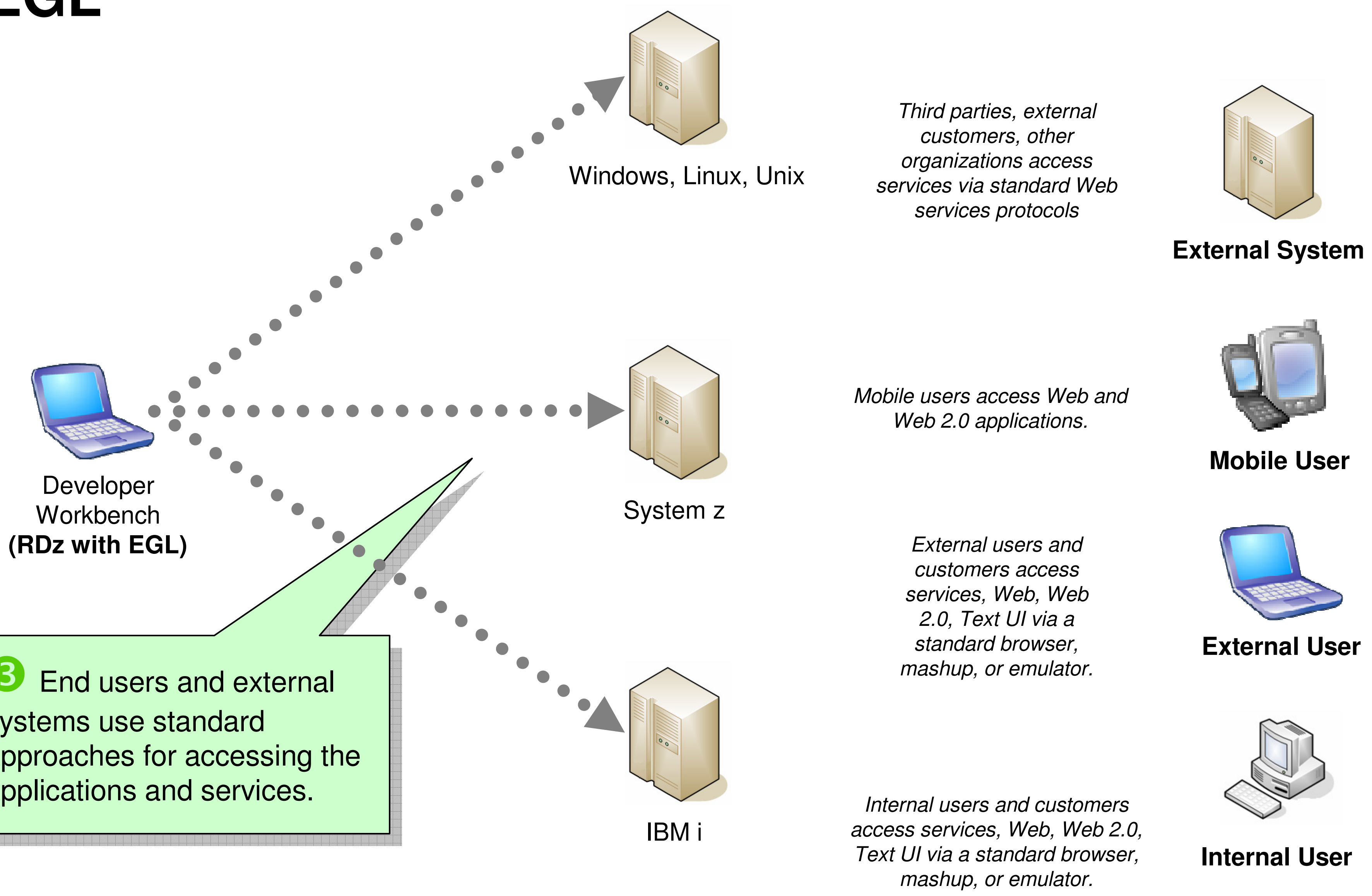
Java generation
- WebSphere Application Server
- Native IBM I

JavaScript generation
- WebSphere Application Server

COBOL generation
- IBM i

❶ Developers use the Rational Developer for System z with EGL workbench to develop Web, Web 2.0, SOA, batch, and text UI applications.

EGL



Third parties, external customers, other organizations access services via standard Web services protocols

Mobile users access Web and Web 2.0 applications.

External users and customers access services, Web, Web 2.0, Text UI via a standard browser, mashup, or emulator.

Internal users and customers access services, Web, Web 2.0, Text UI via a standard browser, mashup, or emulator.

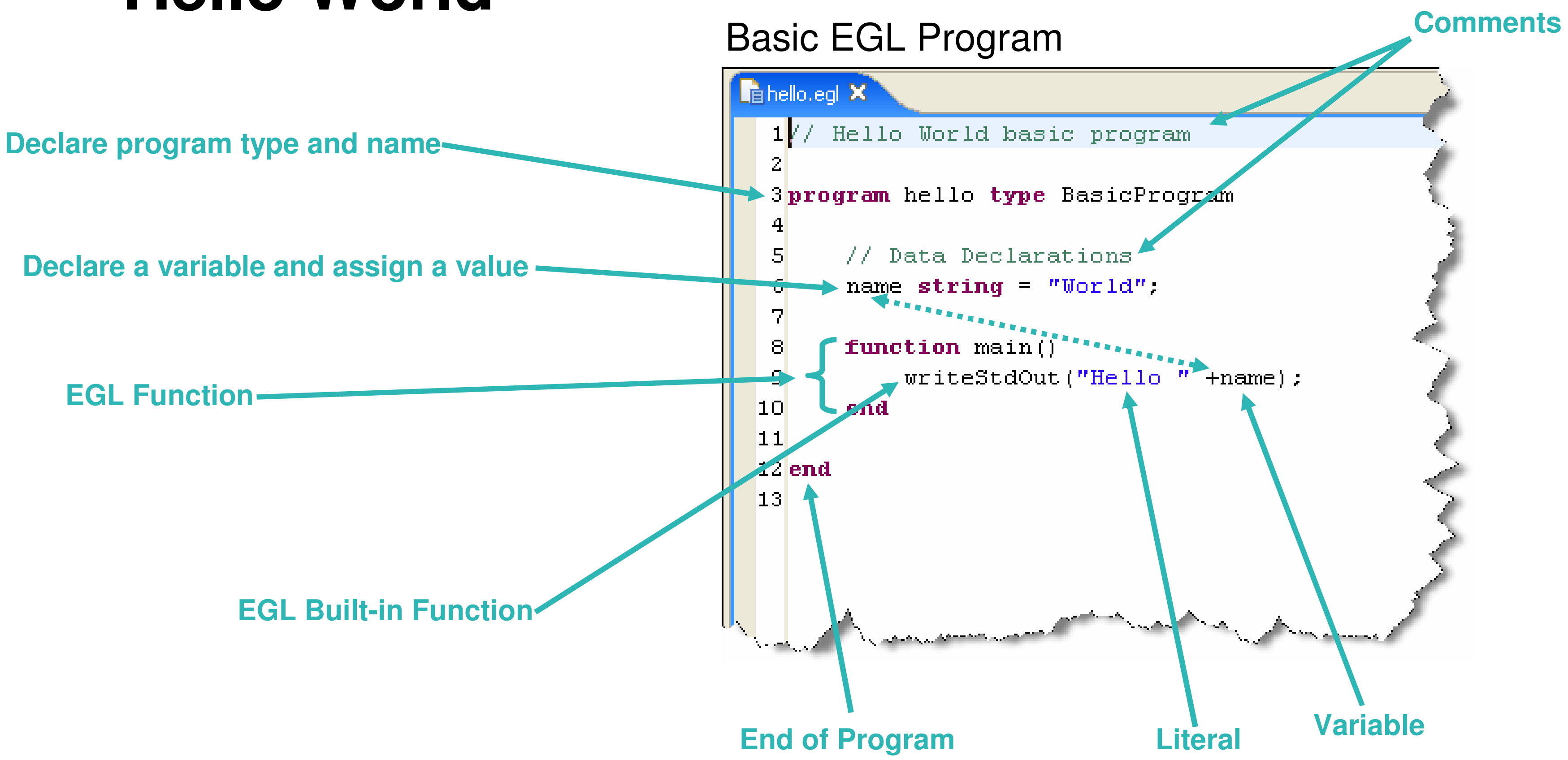
❸ End users and external systems use standard approaches for accessing the applications and services.

EGL - The power of the Language

Simple and familiar

Hello World

Basic EGL Program



```
// Hello World basic program

program hello type BasicProgram

    // Data Declarations
    name string = "World";

    function main()
        writeStdOut("Hello " +name);
    end

end

```

EGL the Language

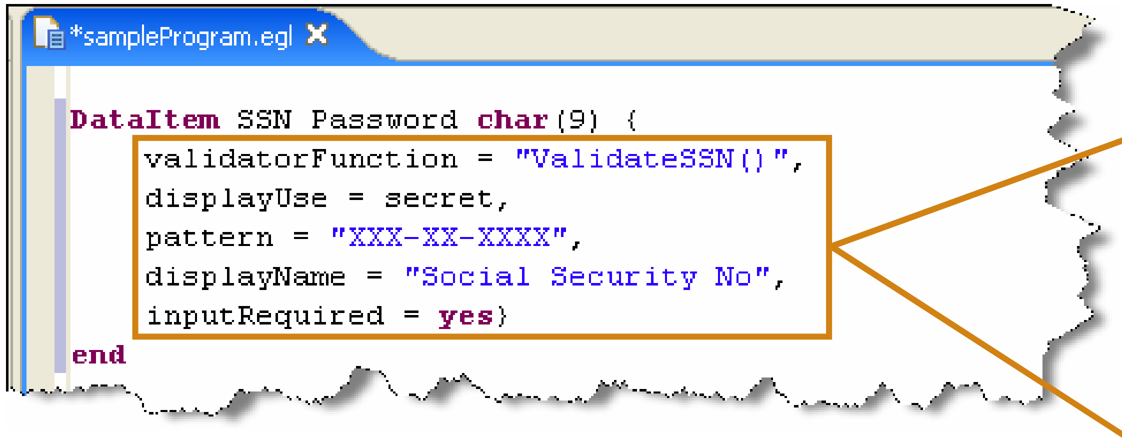
Powerful and complete

- **Rich data types**
  - Simple (int, string, boolean, etc.) or Complex (any, static arrays, dynamic arrays, dictionaries, array dictionaries, etc.)
- **Keywords**
  - Case, if-then-else, while, for loop, for loop cycling through a database result set, etc.
- **High power language capabilities**
  - Automated Casting (e.g. using AS operator)
  - Mixing data types in assignments and expressions
  - Exception handling
- **Rich libraries of built-in functions**
  - Math, string, date/time, system, etc.
- **Robust integrate with existing functions or low level APIs**
  - Call RPG, COBOL, C, etc.
  - Full Java interoperability
    - Invoke Java from EGL (map Java classes with EGL External Types)
    - Invoke EGL from Java

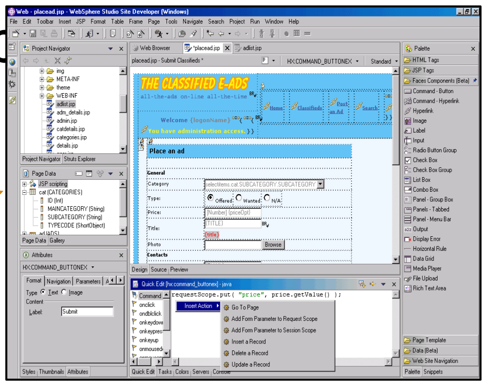
EGL - The Power of Declarative Programming

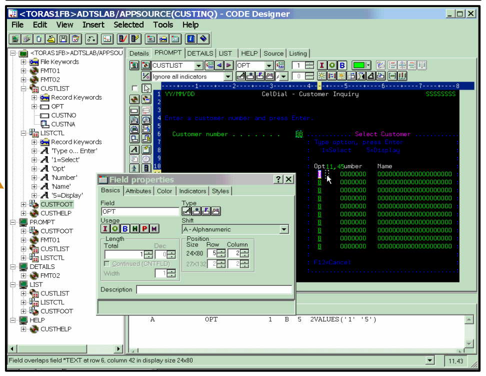
Annotations apply semantics in multiple contexts

- **Validation and Editing Rules**
  - Set properties in “Data Items”
  - Define formatting & validation rules in a common place
  - Reuse data items for Records, screens, web p



```
DataItem SSN Password char(9) {
    validatorFunction = "ValidateSSN()",
    displayUse = secret,
    pattern = "XXX-XX-XXXX",
    displayName = "Social Security No",
    inputRequired = yes}
end
```



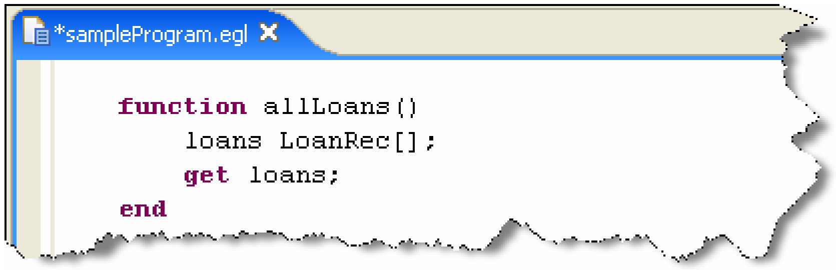


Validation is consistently applied whether the data is bound to a field in a JSF-based web page, field on a 3270 screen, or Rich UI page.

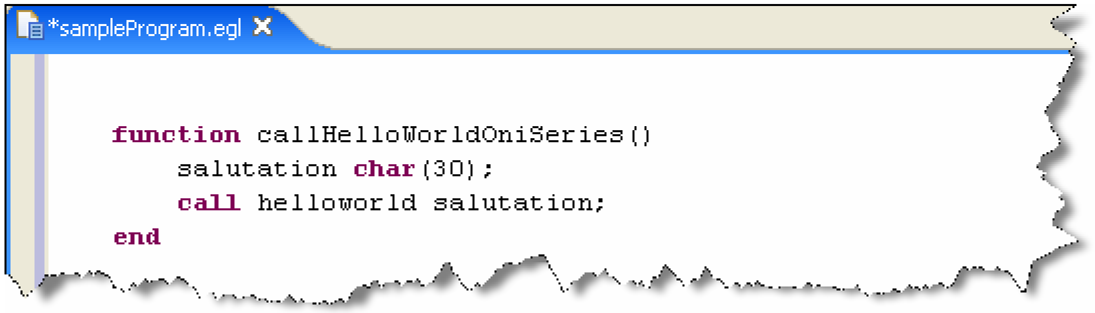
EGL - The Power of Abstractions

Abstractions

- **Data access**
  - Access SQL, Indexed, Relative, Serial, DL/I, and Service data through "Records"
  - Use common Verbs for data access (**Get, Add, Replace, Delete**)
  - Use common Error Handling
- **Remote Invocation**
  - Call COBOL, RPG, C, Java
  - Keep linkage info separated from code
  - Resolve data mapping and protocol invocation at runtime… NO code necessary!



```
function allLoans()
    loans LoanRec[];
    get loans;
end
```

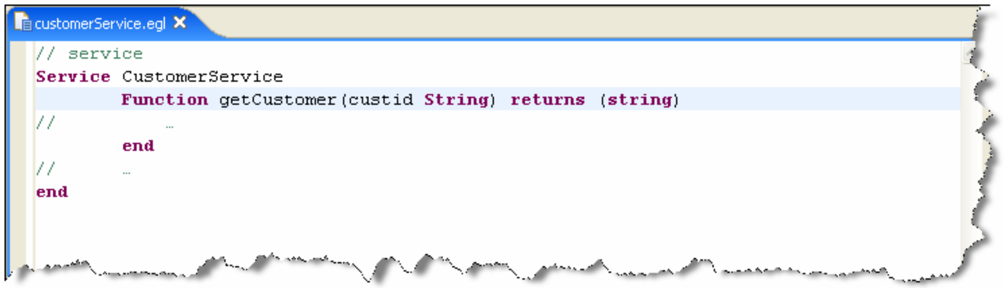


```
function callHelloWorldOniSeries()
    salutation char(30);
    call helloworld salutation;
end
```

The power of Services - Built into the language

Abstractions

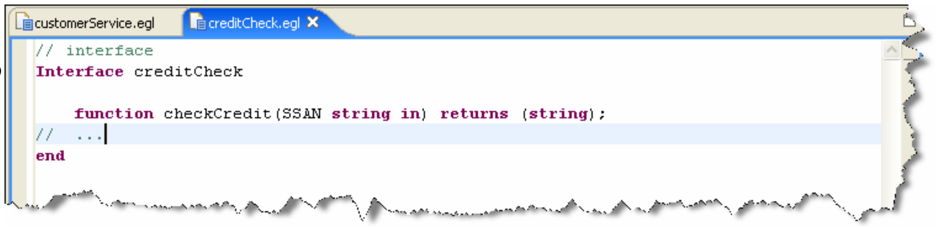
- **Service part:**
  - a generatable part containing code that will be accessed:
    - from EGL code by way of a local or TCP/IP connection (*EGL Service*).
    - from any code by way of an HTTP connection (*EGL Web service*).



customerService.egl

```
// service
Service CustomerService
        Function getCustomer(custid String) returns (string)
//          …
        end
//      …
end
```

- **Interface part:**
  - Used to access external services provide separation of concern.



customerService.egl creditCheck.egl

```
// interface
Interface creditCheck

   function checkCredit(SSAN string in) returns (string);
//  ...
end
```

EGL - The Power of Services

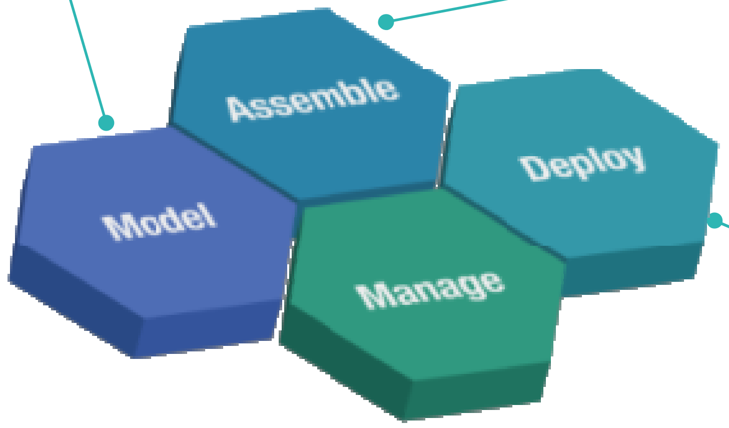
Cross platform language for business oriented services development.

At development time…

- Focus on the business logic
- Implement SOA design elements: services and interfaces
- Leverage existing business developers for new SOA development
- Ignore deployment targets/technology while coding/testing

Leverage external web services…

- EGL Interfaces
  - represent external web services
  - Are created via import from WSDL
  - Allow the EGL developer to stay within the context of the EGL programming model



Deploy EGL services…

To any platform

- Java to WAS/Tomcat/etc.
- COBOL to CICS, iSeries

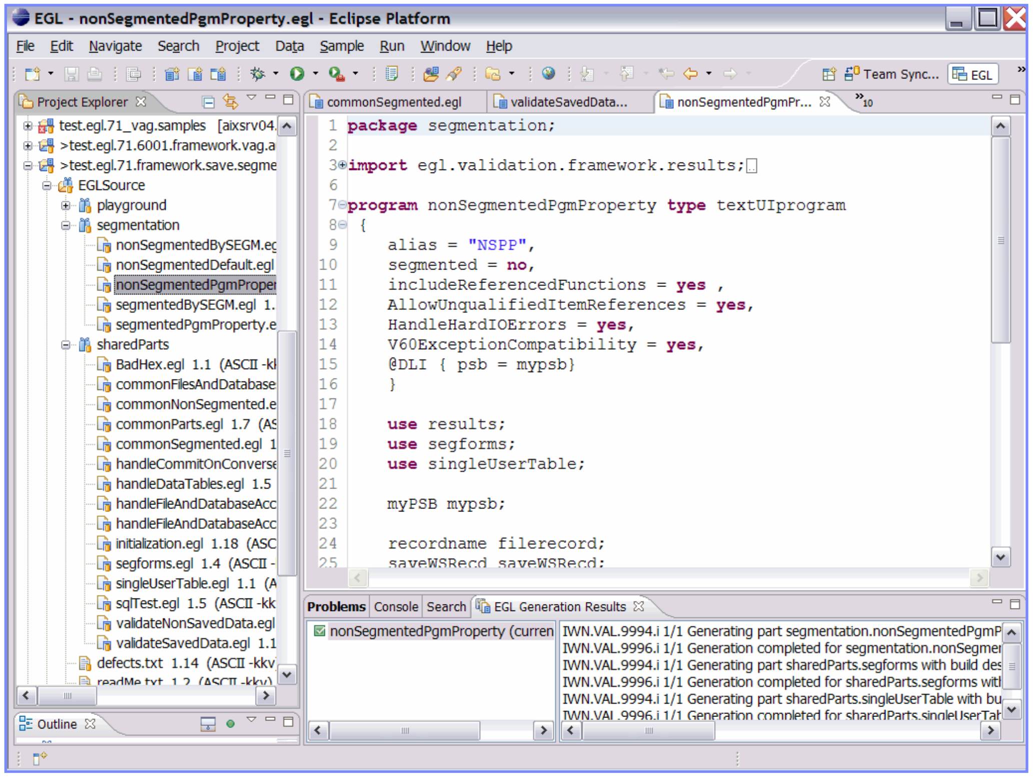
As…

- A Web service (uses SOAP)
- A private service (uses CICS ECI or TCP)
- Other SOA runtimes when they reach critical mass

EGL - The Power of Tools

First class Eclipse workbench

- Folders and views
- Smart EGL editor
- Code templates and snippets
- Code completion
- Import data items from tables
- SQL visualization and editing
- SQL validation
- References and declarations
- Open on selection
- Refactoring
- Cheat sheets and Dynamic help



```
test.egl.71_vag.samples  [aixsrv04.
>test.egl.71.6001.framework.vag.a
>test.egl.71.framework.save.segme
  EGLSource
    playground
    segmentation
      nonSegmentedBySEGM.eg
      nonSegmentedDefault.egl
      nonSegmentedPgmProper
      segmentedBySEGM.egl  1.
      segmentedPgmProperty.e
    sharedParts
      BadHex.egl  1.1  (ASCII -kl
      commonFilesAndDatabase
      commonNonSegmented.e
      commonParts.egl  1.7  (AS
      commonSegmented.egl  1
      handleCommitOnConverse
      handleDataTables.egl  1.5
      handleFileAndDatabaseAcc
      handleFileAndDatabaseAcc
      initialization.egl  1.18  (ASC
      segforms.egl  1.4  (ASCII -
      singleUserTable.egl  1.1  (A
      sqlTest.egl  1.5  (ASCII -kk
      validateNonSavedData.egl
      validateSavedData.egl  1.1
    defects.txt  1.14  (ASCII -kkv
    readMe.txt  1.2  (ASCII -kkv)
```

```
package segmentation;

import egl.validation.framework.results;

program nonSegmentedPgmProperty type textUIprogram
 {
    alias = "NSPP",
    segmented = no,
    includeReferencedFunctions = yes ,
    AllowUnqualifiedItemReferences = yes,
    HandleHardIOErrors = yes,
    V60ExceptionCompatibility = yes,
    @DLI { psb = mypsb}
    }

    use results;
    use segforms;
    use singleUserTable;

    myPSB mypsb;

    recordname filerecord;
    saveWSRecd saveWSRecd;
```

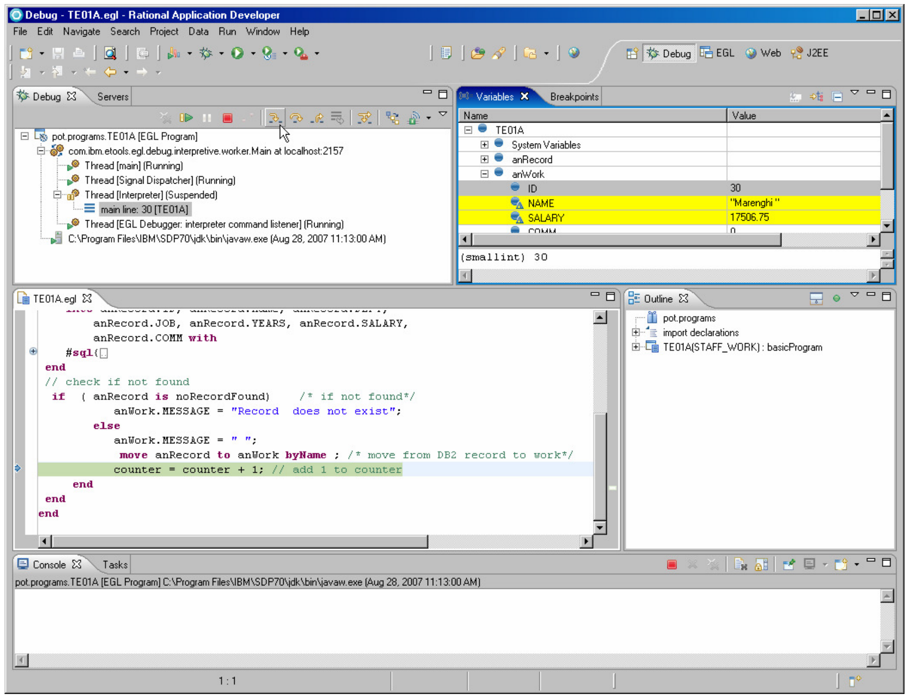
```
nonSegmentedPgmProperty (curren
```

```
IWN.VAL.9994.i 1/1 Generating part segmentation.nonSegmentedPgmP
IWN.VAL.9996.i 1/1 Generation completed for segmentation.nonSegmer
IWN.VAL.9994.i 1/1 Generating part sharedParts.segforms with build des
IWN.VAL.9996.i 1/1 Generation completed for sharedParts.segforms with
IWN.VAL.9994.i 1/1 Generating part sharedParts.singleUserTable with bu
IWN.VAL.9996.i 1/1 Generation completed for sharedParts.singleUserTab
```

EGL - The Power of Tools

Integrated debug environment

- **Debug entire application regardless of ultimate deployment targets**
  - Debug EGL, JSP, Java, etc.
- **Use features of the EGL source debugger**
  - Set breakpoints
  - Watch variables
  - Change variable values
  - Dynamic re-positioning
- **Additional EGL features**
  - Remote VSAM access


Debug - TE01A.egl - Rational Application Developer
File Edit Navigate Search Project Data Run Window Help
Debug EGL Web J2EE
Debug Servers
Variables Breakpoints
TE01A.egl
Outline
Console Tasks

```
pot.programs.TE01A [EGL Program]
  com.ibm.etools.egl.debug.interpretive.worker.Main at localhost:2157
    Thread [main] (Running)
    Thread [Signal Dispatcher] (Running)
    Thread [Interpreter] (Suspended)
      main line: 30 [TE01A]
    Thread [EGL Debugger: interpreter command listener] (Running)
  C:\Program Files\IBM\SDP70\jdk\bin\javaw.exe (Aug 28, 2007 11:13:00 AM)
```

Name	Value
TE01A	
System Variables	
anRecord	
anWork	
ID	30
NAME	"Marenghi "
SALARY	17506.75
COMM	0

```
(smallint) 30
```

```
        anRecord.JOB, anRecord.YEARS, anRecord.SALARY,
        anRecord.COMM with
    #sql{
 end
 // check if not found
  if  ( anRecord is noRecordFound)     /* if not found*/
         anWork.MESSAGE = "Record  does not exist";
      else
         anWork.MESSAGE = " ";
          move anRecord to anWork byName ; /* move from DB2 record to work*/
         counter = counter + 1; // add 1 to counter
     end
 end
end
```

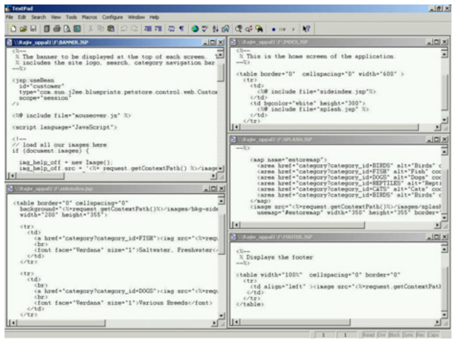
```
pot.programs
import declarations
TE01A(STAFF_WORK) : basicProgram
```

```
pot.programs.TE01A [EGL Program] C:\Program Files\IBM\SDP70\jdk\bin\javaw.exe (Aug 28, 2007 11:13:00 AM)
```

1 : 1

Accelerating Application Delivery

Hand-Coded


TextPad
File Edit Search View Tools Macros Configure Window Help
\\Rajiv_uppali\F\BANNER.JSP
\\Rajiv_uppali\F\INDEX.JSP
\\Rajiv_uppali\F\SPLASH.JSP
\\Rajiv_uppali\F\sideindex.jsp
\\Rajiv_uppali\F\FOOTER.JSP

```
<%--
 % The banner to be displayed at the top of each screen.
 % includes the site logo, search, category navigation bar
--%>

<jsp:useBean
  id="customer"
  type="com.sun.j2ee.blueprints.petstore.control.web.Custom
  scope="session"
/>

<%@ include file="mouseover.js" %>

<script language="JavaScript">

<!--
// load all our images here
if (document.images) {

  img_help_off = new Image();
  img_help_off.src = '<%= request.getContextPath() %>/image
```

```
<%--
 % This is the home screen of the application.
--%>

<table border="0"  cellspacing="0" width="600" >
  <tr>
    <td>
      <%@ include file="sideindex.jsp"%>
    </td>
    <td bgcolor="white" height="300">
      <%@ include file="splash.jsp" %>
    </td>
  </tr>
```

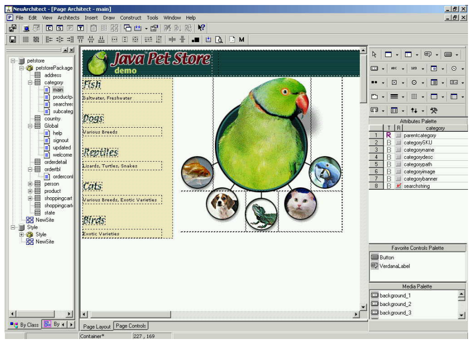
```
--%>

    <map name="estoremap">
      <area href="category?category_id=BIRDS" alt="Birds" c
      <area href="category?category_id=FISH" alt="Fish" coo
      <area href="category?category_id=DOGS" alt="Dogs" coo
      <area href="category?category_id=REPTILES" alt="Repti
      <area href="category?category_id=CATS" alt="Cats" coo
      <area href="category?category_id=BIRDS" alt="Birds" c
    </map>
    <image src="<%=request.getContextPath()%>/images/splash
      usemap="#estoremap" width="350" height="355" border="
```

```
<table border="0" cellspacing="0"
  background="<%=request.getContextPath()%>/images/bkg-side
  width="200" height="355">

  <tr>
    <td>
      <a href="category?category_id=FISH"><img src="<%=requ
      <br>
      <font face="Verdana" size="1">Saltwater, Freshwater</
    </td>
  </tr>

  <tr>
    <td>
      <br>
      <a href="category?category_id=DOGS"><img src="<%=requ
      <br>
      <font face="Verdana" size="1">Various Breeds</font>
    </td>
  </tr>
```

```
<%--
 % Displays the footer
--%>

<table width="100%"  cellspacing="0" border="0"
  <tr>
    <td align="left" ><image src="<%=request.getContextPat
    </td>
  </tr>
</table>
```

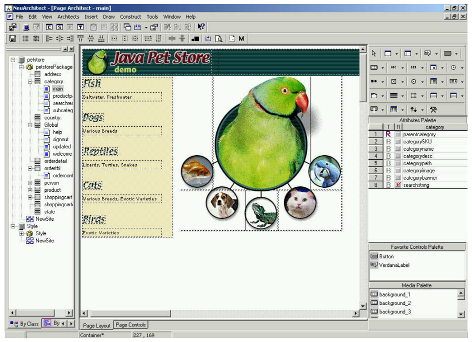
507 Hours

Java/J2EE IDE



330 Hours

RBD



60 Hours

- RBD is **dramatically faster** than traditional development*
- RBD is more productive than MS VS 2005**

* *Internal benchmark using Sun PetStore application*

** *Branham Study April 2008*

EGL Rich UI

- **Why is it difficult to build Web 2.0 style applications today?**
  - Currently domain of “tech heads”
  - Need to know many low level intricate technologies
  - Compound the skill/tool silos and fragmentation
- **Why EGL Rich UI?**
  - Hide complexity of JavaScript, Ajax, JSON, etc.
  - Fully open and extensible
  - Easily integrate/consume any service (REST or SOAP)
  - Single language end-to-end (front-end to back-end)
  - Includes visual composition, libraries of RUI widgets
  - Instant deploy/visualize while you code
  - Extends existing System z data and processes to Web 2.0

RUI Programming - EGL

1. Single Language

- EGL in 3 tiers:
  1. Data and Logic
  2. Soap/Rest Services
  3. Declarative UI
- Just 1 language
- Skill transfer

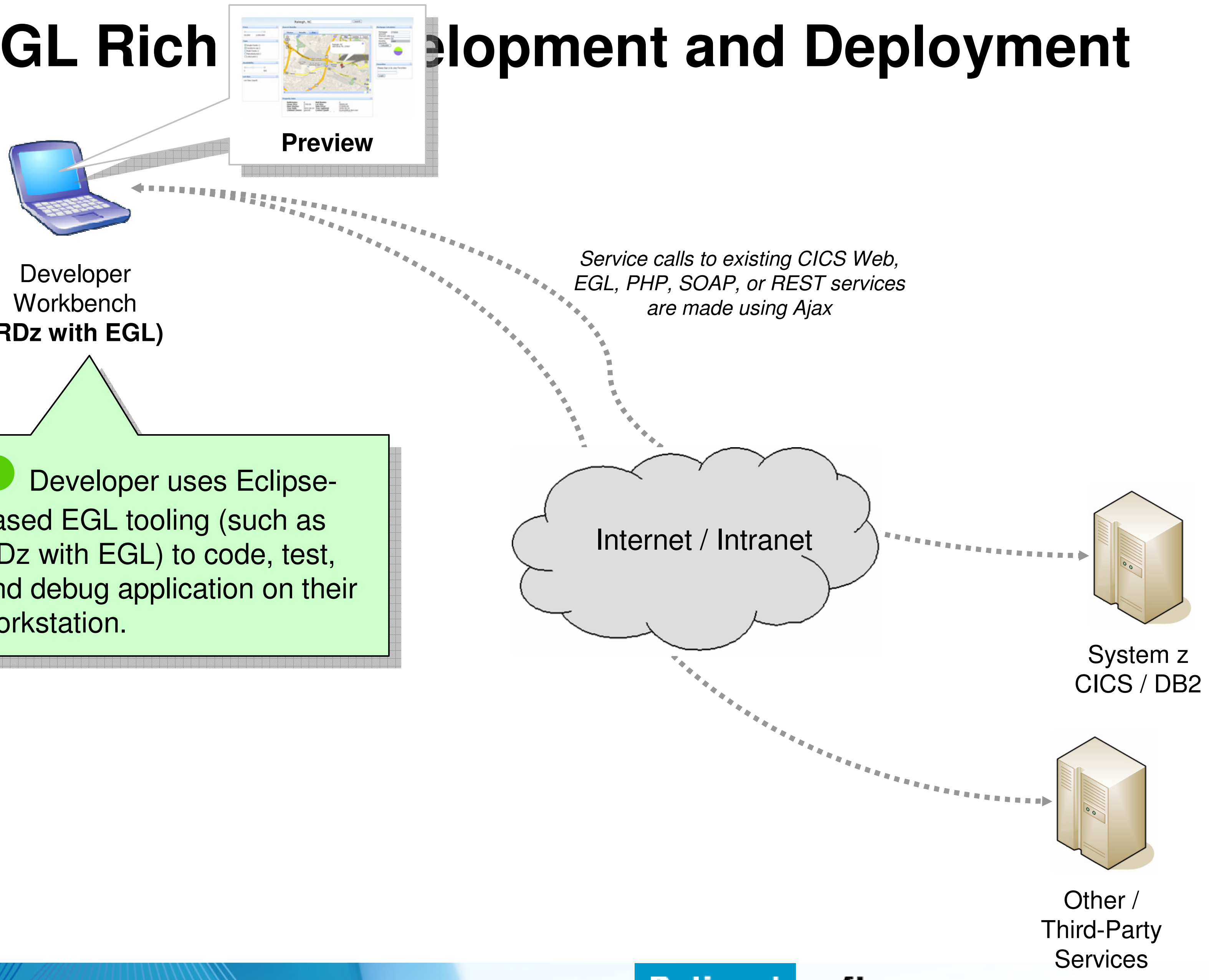
2. Rich User Interfaces

- Declarative UI
- Ajax support
- Rich set of widgets
- Easily extensible
- Mashup using SOA

3. Mobility

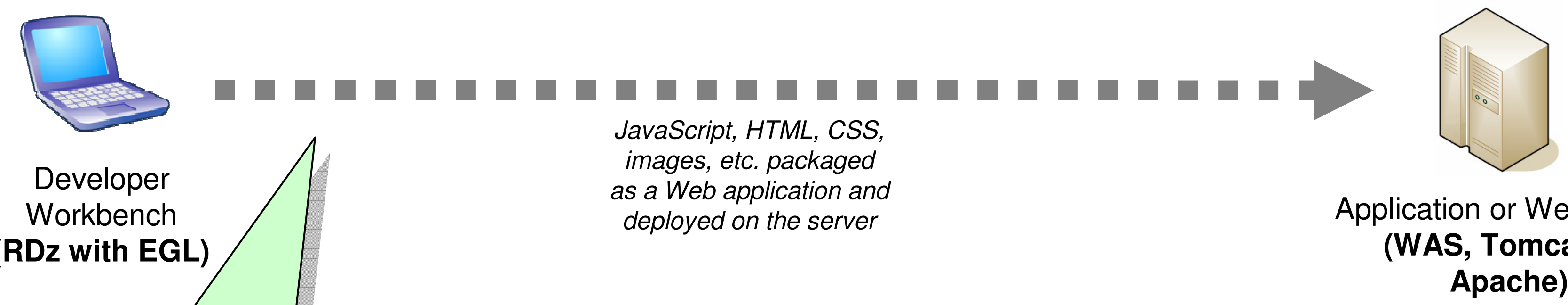
- Disconnect/Sync
- Easy installation
- No version pain
- Flexibility

EGL Rich ...elopment and Deployment



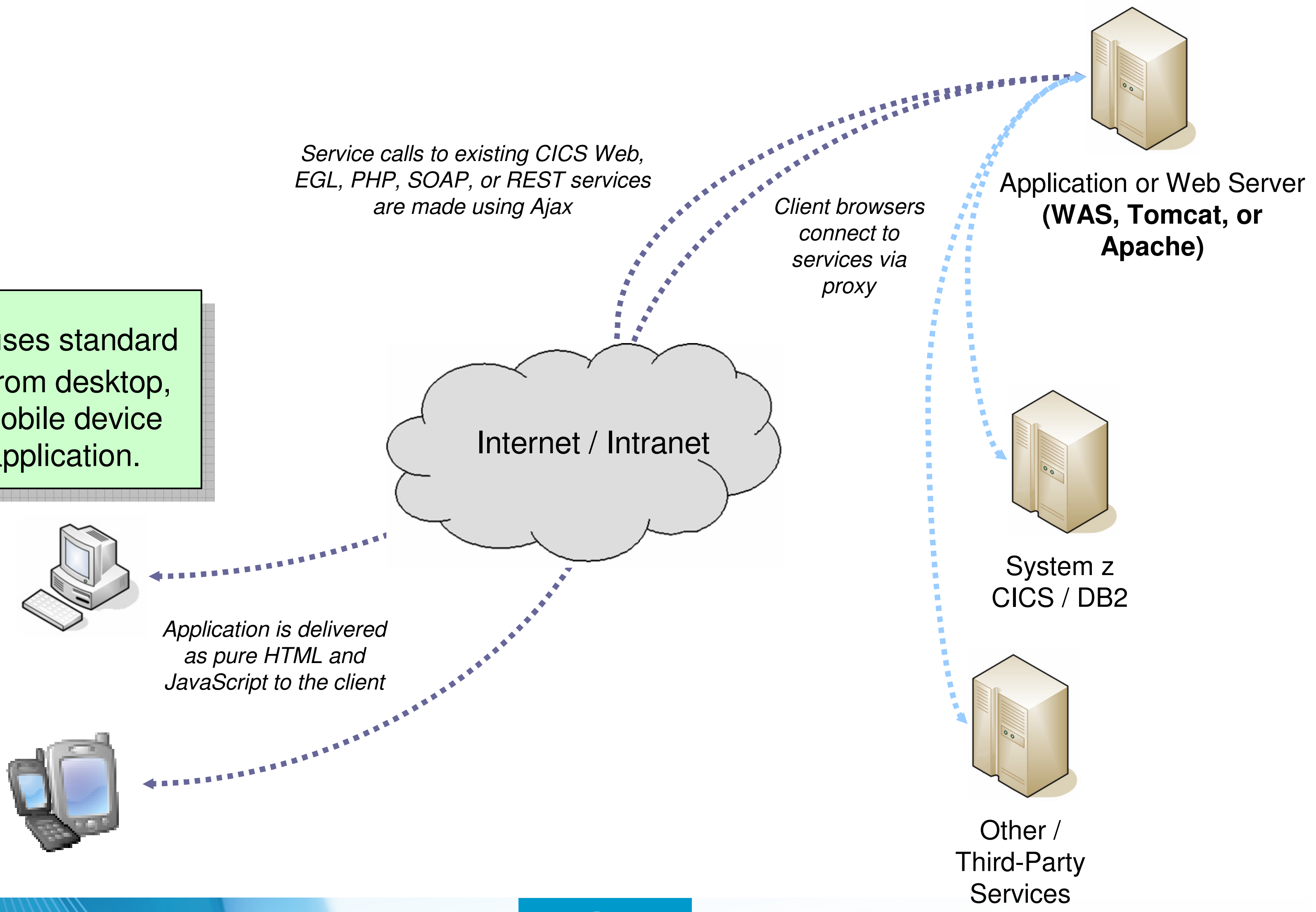
➊ Developer uses Eclipse-based EGL tooling (such as RDz with EGL) to code, test, and debug application on their workstation.

EGL Rich UI Development and Deployment


Developer
Workbench
(RDz with EGL)
JavaScript, HTML, CSS, images, etc. packaged as a Web application and deployed on the server
Application or Web Server
(WAS, Tomcat, or Apache)

❷ Developer runs Deployment wizard to create a Web application containing HTML and JavaScript (generated from EGL code). Application is deployed to an application server.

EGL Rich UI Development and Deployment

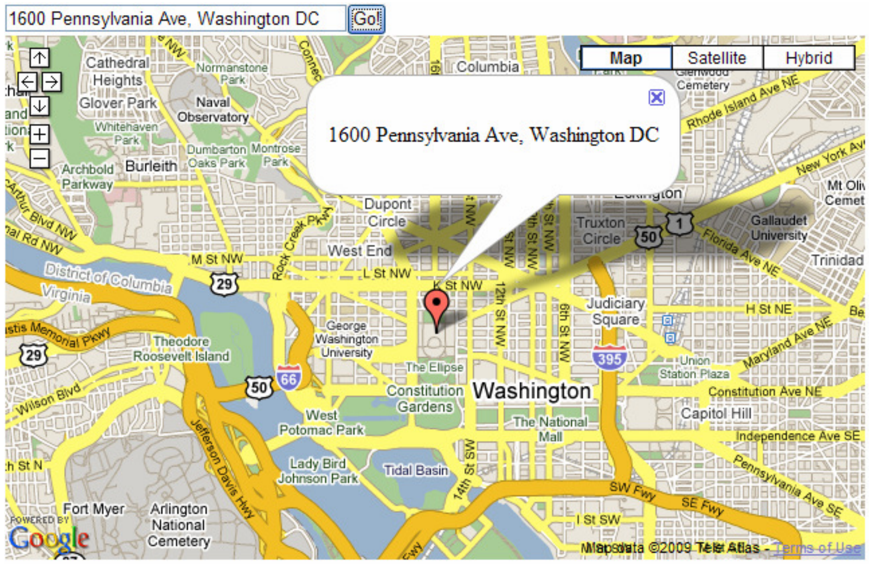

Service calls to existing CICS Web, EGL, PHP, SOAP, or REST services are made using Ajax
Client browsers connect to services via proxy
Application or Web Server **(WAS, Tomcat, or Apache)**
Internet / Intranet
System z CICS / DB2
Application is delivered as pure HTML and JavaScript to the client
Other / Third-Party Services

❸ End user uses standard Web browser from desktop, notebook, or mobile device to access the application.

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
```


1600 Pennsylvania Ave, Washington DC
Go!
Map
Satellite
Hybrid
1600 Pennsylvania Ave, Washington DC
Washington
POWERED BY Google
Map data ©2009 Tele Atlas - Terms of Use

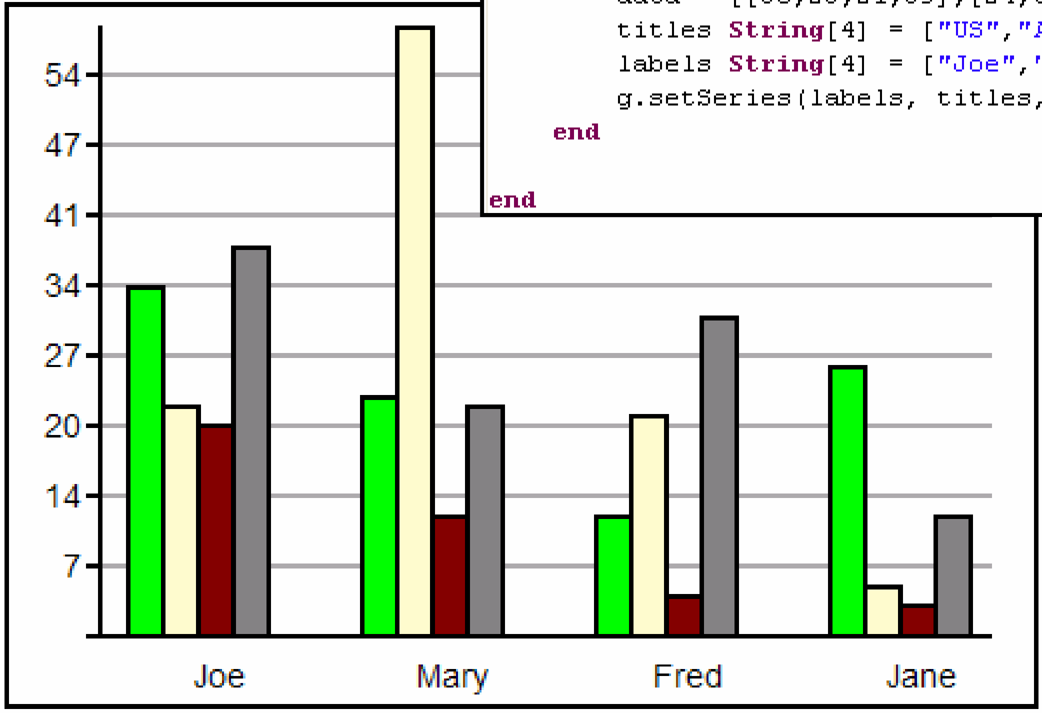
HTML and JavaScript

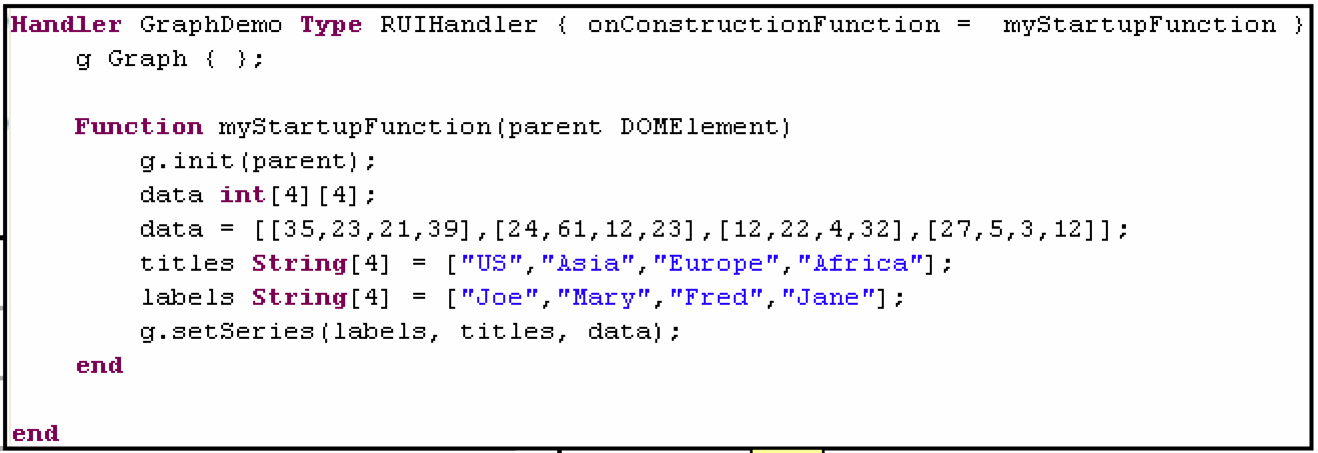
```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-
com:vml">
  <head>
    <meta http-                              tml; charset=UTF-8"/>
    <title>Goog                              g</title>
    <script src                              api&amp;v=2.x
    <script typ

    var geocode

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


      if (ge
        geocoder
          address
          functio
            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.

EGL Simple Example (Drawing Graphics)



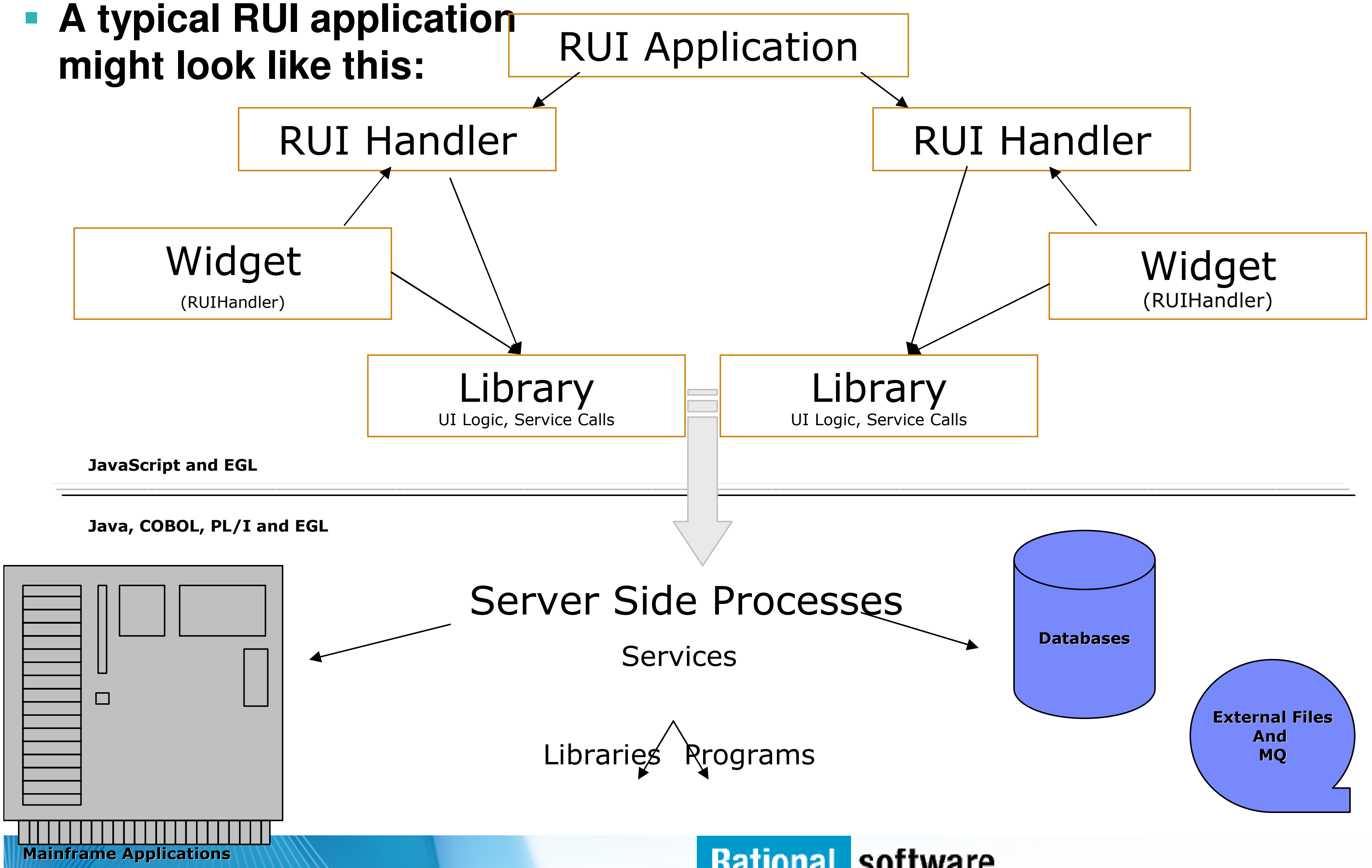


```
Handler GraphDemo Type RUIHandler { onConstructionFunction =  myStartupFunction }
    g Graph { };

    Function myStartupFunction(parent DOMElement)
        g.init(parent);
        data int[4][4];
        data = [[35,23,21,39],[24,61,12,23],[12,22,4,32],[27,5,3,12]];
        titles String[4] = ["US","Asia","Europe","Africa"];
        labels String[4] = ["Joe","Mary","Fred","Jane"];
        g.setSeries(labels, titles, data);
    end

end
```

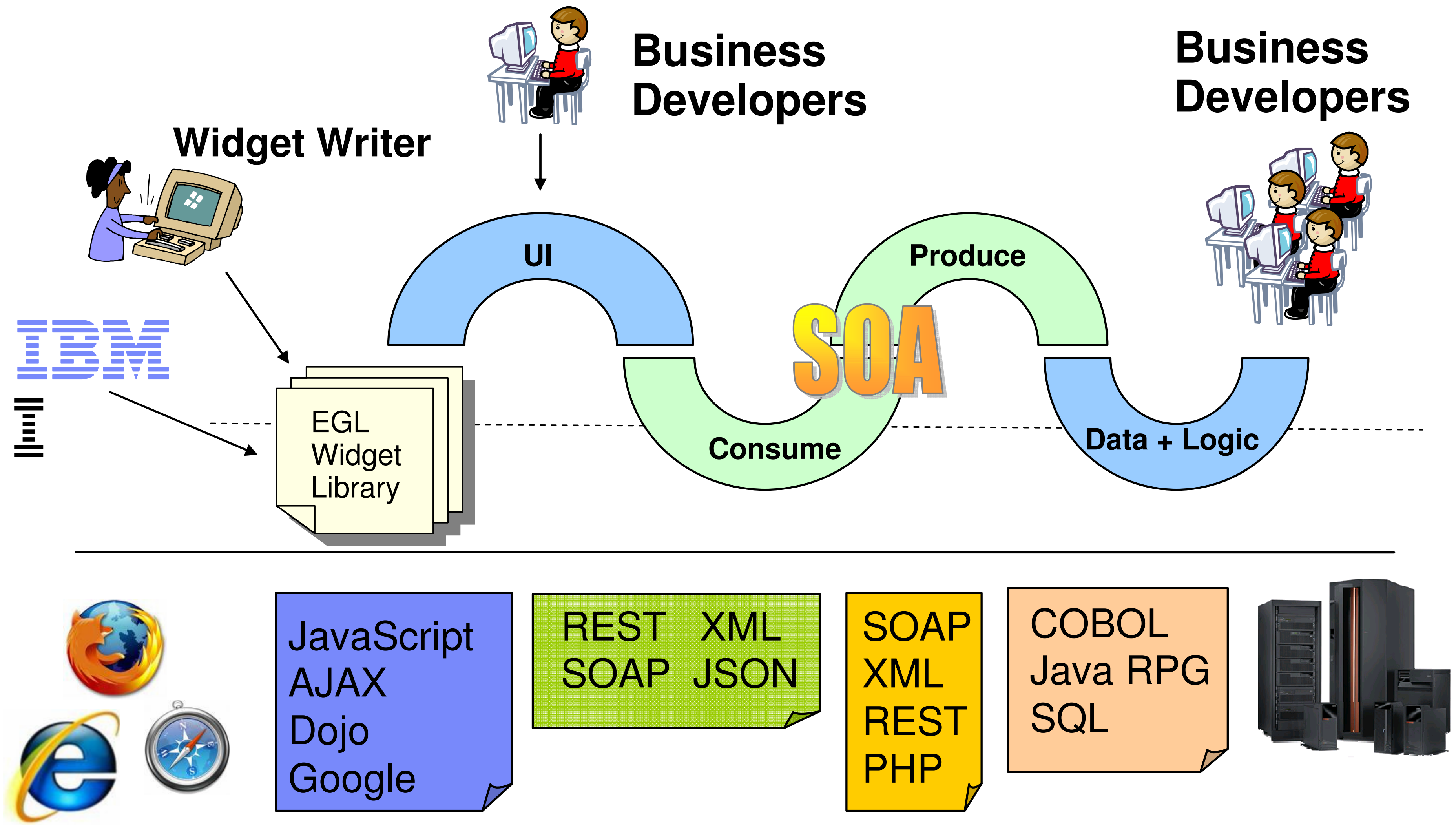
Graph widget was user-written in 120 lines of Javascript

RUI Programming – Overview

- A typical RUI application might look like this:



EGL: Shielding Complexity



Demo Tasks

Task 1 - Adding Mortgage Calculator

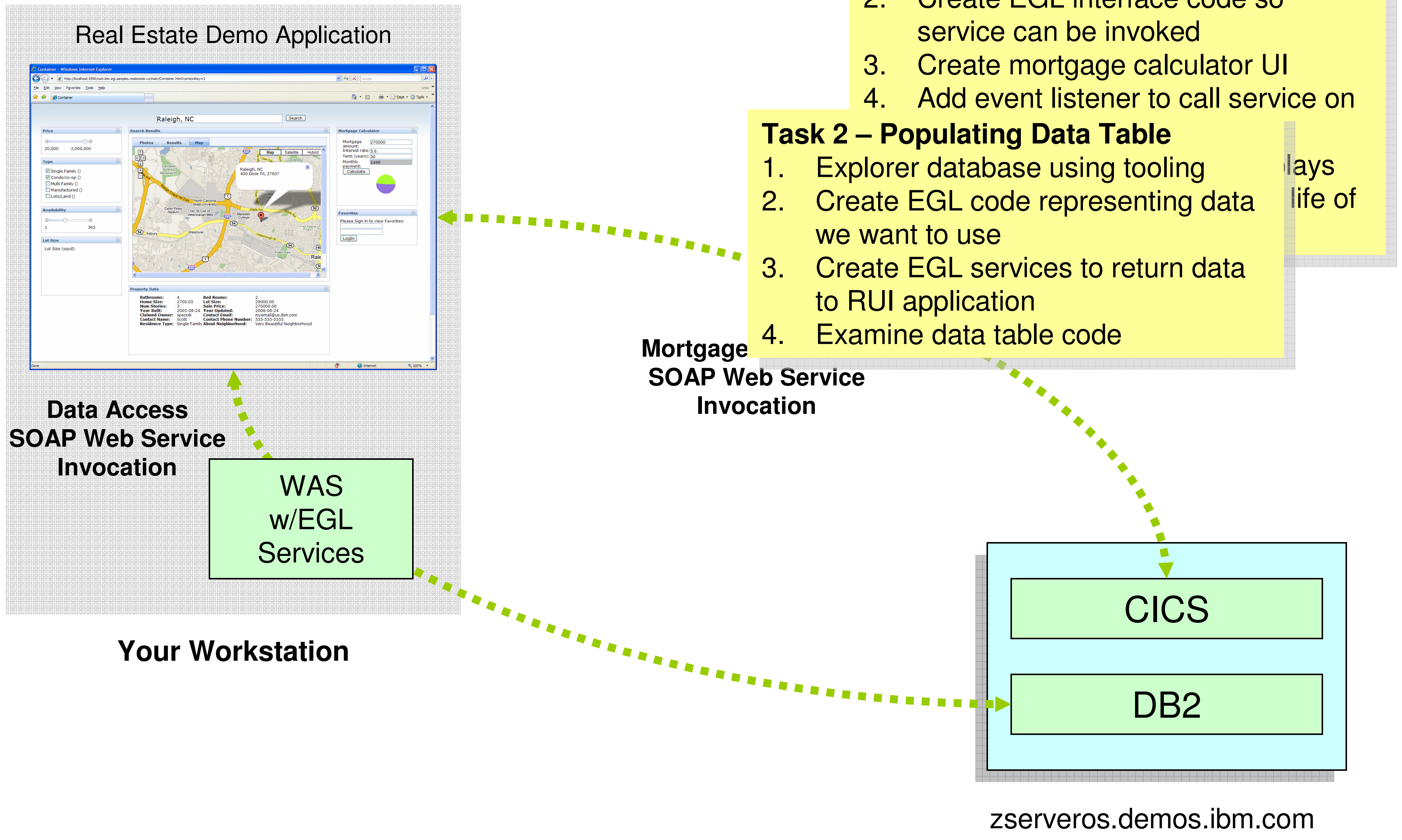
1. Consume CICS Web service in EGL Rich UI project
2. Create EGL interface code so service can be invoked
3. Create mortgage calculator UI
4. Add event listener to call service on

lays

life of

Task 2 – Populating Data Table

1. Explorer database using tooling
2. Create EGL code representing data we want to use
3. Create EGL services to return data to RUI application
4. Examine data table code



EGL Rich UI

Mortgage Calculator, Data Table, and Google Map