Simplify Web 2.0 with IBM Rational EGL Community Edition

Will Smythe (smythew@us.ibm.com)
EGL Product Line Manager
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
Challenges with Web 2.0 Development

- Why is it difficult to build Web 2.0 Rich Internet Applications 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.
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
- Eclipse-based development, testing, and debugging
- Consume all types of Web services
EGL – IBM’s Newest Business Language

- EGL is a higher-level programming language designed for developing powerful, modern applications
  - The goal is to shield developers from complexities typically associated with Web 2.0 application development
- A true cross-platform, cross-tier language targeted at all types of developers
  - Traditional, procedural, object-oriented
- Provides flexible deployment options
  - Compiles to Java, COBOL, or JavaScript
- Web 2.0 and SOA built-in
- EGL is also an excellent target language for migration of existing, traditional applications.
Developers use Eclipse-based **EGL Community Edition** to code, preview, and debug application on their workstations. EGL services are compiled as **Java** and RUIs are compiled as **JavaScript** on-the-fly during preview.
EGL Rich UI Architecture

For deployment, RUI code is compiled into HTML and JavaScript, services are compiled into Java as SOAP or REST Web services. Code is then deployed as a JEE application into a Web application server.
EGL Rich UI Architecture

1. Users access the URL for the application using a standard Web browser and the HTML and JavaScript is downloaded to the client. All client/server interaction occurs via REST and SOAP service calls (using Ajax). Services can be also external/third-party.
EGL Rich UI

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

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

goButton Button { text = "Go!", onClick := goButton_clicked }
gobutton clicked { children = [ addressField, goButton ] }

map GoogleMap { width = "500px", height = "400px" };

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

HTML and JavaScript

```html
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
        <title>Google Map Example</title>
        <script src="http://maps.googleapis.com/maps/api/js?v=2.x"
            type="text/javascript"></script>
        <script>
            var map = null;
            var geocoder = null;

            function initialize() {
                if (BrowserIsCompatible()) {
                    map = new GoogleMap();
                    map.setCenter(38.907151, -77.036490);
                    map.setMap(map);
                    map.addMarker([38.907151, -77.036490], map);
                    map.setZoom(13);
                }

                function showAddresses(addresses, addresses)
                    if (geocoder) {
                        geocoder.geocode({ address: address }, function(results, status) {
                            if (status == google.maps.GeocoderStatus.OK) {
                                geocoder.geocode({ address: address }, function(results, status) {
                                    if (status == google.maps.GeocoderStatus.OK) {
                                        var marker = map.addMarker(results[0].geometry.location, map);
                                    }
                                });
                            }
                        });
                    }
                }
            }
        </script>
    </head>

    <body onload="initialize()" onunload="onUnload()">
        <form action="#" onsubmit="showAddress(this.address.value); return false"/>
            <input type="text" size="60" name="address" value="1600 Pennsylvania Ave, Washington DC" />
            <input type="submit" value="Go!" />
        </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

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

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.

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.

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.

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.

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.
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!**
EGL Community Edition Overview

- EGL Community Edition (CE) is a **free**, Eclipse-based tool that simplifies the development of Web 2.0 applications
  - Simplifies development of open standards-based Web user interfaces
    - Utilize the popular Dojo widget toolkit without learning HTML or JavaScript
  - Supports simplified development of Java-based Web services
    - Build services without learning Java, XML, JSON, etc.
    - Easily interact with existing programs or databases using EGL’s simplified APIs
      - Supports DB2, MySQL, and Derby
  - Package and deploy as a standard JEE application

- **Easy to consume, free to use**
  - Small download, simple install – get up and running quickly!
  - Available Tuesday, September 8, 2009 on the EGL Cafe (http://ibm.com/rational/eglcafe)

- **Why EGL?**
  - EGL takes a more open approach to Web 2.0 development
    - EGL does not try to replace HTML or JavaScript (unlike Flash/Flex and Silverlight)
  - EGL is completely extensible, allowing for unlimited extension and flexibility
  - Build rich, modern Web applications without writing a single line of HTML or JavaScript
EGL Community Edition FAQ

- Why is IBM doing this?
  - IBM understands that for a programming language to be successful, it needs to be accessible by the broadest possible group of people.

- Who is it for?
  - EGL CE is intended for anyone who wants to develop RIAs quickly and efficiently. Since EGL syntax is easy for people to learn and can integrate across different business tiers, people from all development backgrounds (Java, PHP, COBOL, etc) find they are very productive using EGL.

- Why would I be interested?
  - Let's face it - RIA / Web 2.0 application development is not easy. With EGL, you can spend more time building cool apps, and less time digging through JavaScript reference manuals.

- How is EGL CE different from our purchasable EGL tools (Rational Business Developer, Developer for System z, Developer for i for SOA Construction)?
  - No support from IBM
  - CE lacks “enterprise level” capabilities
    - No WebSphere Application Server support, no support for generating COBOL for System z or IBM I, no JavaServer Faces (JSF) support.
Why EGL Community Edition?

- **One language**
  - Develop your complete application in a single language (EGL)

- **Instant gratification**
  - Test and debug server and client-side code, without running on or deploying to a server

- **Easy access to data**
  - Access information in databases using straight-forward, but powerful EGL keywords like get, add, and update

- **Built on Web services**
  - Easily integrate with existing SOAP or REST services, or develop your own

- **JavaScript without the JavaScript**
  - Build powerful JavaScript-based applications without writing a line of JavaScript
## RBD vs. EGL CE Feature Matrix

<table>
<thead>
<tr>
<th>Feature / capability</th>
<th>RBD</th>
<th>EGL CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBOL generation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Java generation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>JavaScript generation (Rich UI)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>JavaServer Faces (JSF)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Tools for editing Text/Console UIs</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Creation and consumption of SOAP and REST Web services</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>External program call support</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BIRT / EGL text reporting</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Install/updates via IBM Installation Manager</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>IDE translation (Group1 + others)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Support from IBM</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
EGL Community Edition and EGL Open

EGL Open is part of the long-term EGL strategy, but this plan is subject to change.
Next Steps
EGL Distance Learning 2009

- Learn EGL in 2 weeks with IBM’s no charge (free) remote course.
- Explore the emerging cross-platform rapid development environment from IBM that enables you to build Web, Web 2.0, and SOA solutions using EGL.
- Discover how quickly you can learn EGL and build state-of-the-art applications, all without getting bogged down in the technical complexities of middleware and runtime platforms.

**Week 1:**
- Foundations of EGL technical workshop

**Week 2:**
- Web 2.0 development with EGL Rich UI technical workshop

**Dates**
- May 4
- June 15
- July 27 (part 2 only)
- September 14
- November 30

2 hours of instruction per day
3 - 5 hours of independent hands-on per day

**Register now!**
http://www.ibm.com/software/rational/cafe/docs/DOC-3004
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!
Thank You