Eclipse EGL Development Tools Project
An Evolution in Business Application Development

Tuesday, June 29, 2010

Will Smythe, IBM
smythew@us.ibm.com
Have a question during the session?

www.webdialogs.com/join

Conference ID is 3402243

OR

Send an e-mail to tdramsey@us.ibm.com
This session is being recorded

If you have a problem with this, hang up now
Agenda

- EGL overview
- Eclipse EGL Development Tools (EDT) Project overview
- Initial IBM contribution to Eclipse
- Learn more and/or get involved!
Simplify Innovation with EGL

- EGL is a higher-level, application programming language
  - Designed for developing business-oriented applications (i.e. applications that access and display data from a database, mashup data from various services, present information to users via a Web interface, Web service, or report)
  - Provides a common programming layer across all tiers of the application
  - Shields developers from the complexities of runtimes
    - Allows developers to focus on business needs
  - Ideal for all types of developers (procedural, OO, JavaScript, etc)

- EGL is modern
  - Web 2.0/JavaScript and SOA built-in
  - Eclipse-based tools
Why EGL?

**Flexibility**
Delivers maximum platform and runtime independence

**Modern**
Eclipse-based tools simplify development of Web 2.0 style applications and Web services

**Productivity**
Allows developers to focus on business problems, not technology problems

**Skills**
Enables developers of all backgrounds to build modern Web applications and services

**Adaptability**
Delivers a modern language that evolves as technologies and runtimes change
EGL – Common Language Across Tiers

- **Compile**
  - COBOL
  - Java
  - JavaScript/HTML

- **Deploy**
  - Enterprise Systems: CICS, IMS, Power
  - Application Servers: Apache Tomcat, WebSphere, JEE

- **Run**
  - Server: CICS, IMS, Power
  - Client: Desktop and Mobile
EGL Code Example

- Objective: create a simple Web application that displays data from a database using a Dojo grid.
The following EGL code will eventually get generated as a Web service. It is responsible for returning all employee records from a database.

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 Code Example (Browser Side)

- The following EGL code will eventually get generated as JavaScript/HTML. It is responsible for calling the service and displaying the results to the user.

The user interface definition is written completely in EGL (not HTML).

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).

Notice how functions are bound to event types on UI widgets.

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: 30 years of innovation

What we have heard ...

- We prefer open languages because they are less risky to our business
- We believe innovation is accelerated by giving others the ability to participate
- We want to see an active community promoting awareness and growing skills

EGL to date ...

- Represents 30 years of innovation and continued investment by IBM
- Introduced in 2004 as the follow on to CSP and VisualAge Generator
  - Designed for extensibility
- Thousands of customers using EGL and predecessor technologies
- Migration paths have preserved customers’ investments
Open source project on Eclipse.org for the EGL language, compiler, source editing tools, debugger, and generators for Java and JavaScript

- Initial contribution based on technology in IBM Rational Business Developer
- Core EGL development by IBM and other participants in the open on Eclipse
- Project launched June 7, 2010
Proposed Initial Contributors

- Tim Wilson, IBM (project lead)
- Tony Chen, IBM
- Paul Harmon, IBM
- Matt Heitz, IBM
- Brian Svihovec, IBM
- Joe Vincens, IBM
- Paul Pilotto, ASIST
- Bernd van Oostrum, ASIST
- Wim Delen, ASIST
- David Van Herzele, ASIST
- Aaron Allsbrook, ClearBlade
- John Trevisan, ClearBlade
- Rhys Priddle, FBDA
- Carlos Prieto, NEXTEL Engineering Systems
- Bruno Sanz, NEXTEL Engineering Systems
- Roland Zurawka, PKS
- Oleg Arsky, Synchrony Systems
- Corey Grimes, Synchrony Systems
- Richard Doust, Synchrony Systems

Interested in contributing to the project?! Let us know!
EDT Project

- **Scope of the Eclipse EDT project is ...**
  - First-class, Eclipse-based development tools for the EGL language, comparable in scope to the tools provided for the Java language by the [Java Development Tools project](#)
  - EGL language parser, compiler and generators, which can be invoked via the tools (e.g. by project builders) or command line (for headless compilation and generation)
Proposed Initial Contribution

- **With the initial contribution:**
  - Application developers will be able to develop, debug, and test EGL-generated Java and JavaScript applications and deploy to a Dynamic Web Project
    - These capabilities exist today in the [EGL Community Edition](http://www.eclipse.org/egl) tool
  - **For example, developers can create:**
    - SOAP or RESTful Web services that perform some transaction and/or access a database
    - Rich, Web 2.0 applications that use open source widgets (like Dojo) and call services (EGL-generated or other)
    - Batch style Java programs

- **Tools**
  - EGL source editor and related views
  - Integration into Eclipse Project Explorer view
  - Debugger (of EGL source)
  - Eclipse Data Tools integration
  - Deployment capabilities

- **Compiler and generators**
  - EGL source compiler
  - Java generator (targeting JEE environments)
  - JavaScript generator (targeting modern Web browsers)
Eclipse View Integration

EGL parts are organized by package in the Project Explorer view.

Compile and other problems appear in the Problems view alongside other workspace problems.

The Outline view shows the structure of the EGL part.
EGL Source Editor

- Content assist
- Folding
- Code formatting
- Import organizer
- Syntax highlighting
- Customizable templates
- Code refactoring (limited)
- Problems reported as you type
- Undo/redo
Rich UI Visual Editor

- **EGL Rich UI handlers are EGL parts that compile into JavaScript**
  - Contain both Web UI layout and UI event logic
- **WYSIWYG visual editor for composing Rich UI handlers**
  - DOM-based
  - Drag and drop from palette
  - View and modify UI hierarchy via the Outline view
  - Set properties and create UI event handler functions from the Properties view
- **Palette**
  - IBM, Dojo, and developer-contributed widgets
Rich UI Visual Editor Preview

- Preview tab shows running, fully-interactive application
- EGL services (within the same or a referring project) can be called without deploying anything to a server
Dojo Integration

- Allows developers to use Dojo Toolkit widgets in applications, but code within the EGL programming model
  - Think of EGL as providing a layer that:
    - Insulates developers from the underlying complexities of JavaScript and Dojo
    - Provides a common, normalized API interface across all widget libraries (EGL and third-party)
  - Allows data to be provided from EGL native records and primitive variables
- Demonstrates the EGL concept of External Types, which allows EGL code to interface with non-EGL artifacts (JavaScript widgets, Java code, etc)
Debug

- Integrated with Eclipse Debug UI
- Developers can set breakpoints and step through both “client” side and “server” side code
- Developers step through and debug EGL code, not generated code!

Eclipse debugger stopping when breakpoint is reached in client-side code

Setting breakpoint via EGL source editor

Variables view
Debug (Services)

Setting breakpoint in service code

Eclipse debugger stopping when breakpoint is reached
Data Tools Integration

- EGL tools work with Eclipse Data Tools to help developers:
  - Create records that represent a database view or table
  - Validate SQL statements
  - Build prepared statements

EGL records can be created and populated from database view/table metadata

Resulting EGL record

```
record Contact type SQLRecord
(keyName="contact_id")

  contact_id int
  contact_group int
  first_name string
  last_name string
  title string
  company_name string
  email_address string
  office_phone string
  mobile_phone string
  other_phone string
  address1 string
  address2 string
  city string
  state string
  zip string
  country string
  photo_url string
  notes clolh

end
```
Deployment

- Once applications are ready for deployment, the EGL Deployment Descriptor editor allows the developer to specify a “target” project (standard Eclipse Dynamic Web project) that will receive generated HTML/JavaScript, Java Web services, etc.

- Projects can have more than 1 deployment descriptor
  - Allows deployment of application to different environments (e.g. test, QA, production)
Web Service Support

- EGL service parts can be generated as Java-based SOAP and/or REST Web services.

Wizard that controls the type of Web service to create for an EGL service during deployment.

Eclipse Web Services Explorer showing testing of EGL-generated SOAP service.
Proposed Schedule

- **June 2010** – initial proposal on Eclipse.org

- **Summer 2010** – creation of project

- **December 2010** – final delivery of initial contribution

- Goal: part of 3.7 (Indigo) simultaneous release
EGL Strategic Direction

EGL Source Code → EGL Compiler → Model

Extensible Code Generation Framework

<table>
<thead>
<tr>
<th>Java</th>
<th>Java</th>
<th>JavaScript</th>
<th>C/C++</th>
<th>.Net</th>
<th>.COBOL</th>
<th>…</th>
</tr>
</thead>
</table>

Pluggable Generators

Pluggable Extension Examples

Pluggable Generators

Eclipse
Possible
IBM

JEE
OSGi based runtimes
Gears

CICS/IMS
Learn more about EGL, the initial contribution, and future direction!

www.eclipse.org/proposals/egl
EGL Community Edition

- Simplify development of Web 2.0 solutions – **for free!**
- Eclipse-based development environment
  - Small download, simple install
  - WYSIWYG visual editor
  - Instant application previewing without deploying to a server
  - Full debug support for client and server-side code
- Rich Web user interfaces using Dojo
  - Fully extensible architecture supports other popular JavaScript widget libraries
- Build Java-based Web services without coding in Java
  - Take advantage of EGL’s powerful keywords for accessing data in most popular databases, including MySQL
- **Spend more time innovating and less time fighting with technology!**

[Check out the EGL CE video on YouTube!](http://www.ibm.com/software/rational/cafe/community/egl/ce)

EGL Distance Learning 2010

- Learn EGL by taking IBM’s no charge (free) education course!
  - Explore the emerging cross-platform rapid development technology 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

- **Topics**
  - Foundations of EGL
  - Web 2.0 development with EGL Rich UI

**2010 Dates**
- February 15
- March 29
- May 3
- June 21
- July 26
- September 13
- November 15

Up to 3 hours of instruction per day
3 to 5 hours of independent hands-on per day

Register today!
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!
We want your feedback!

Please take a quick survey:

Thank you!
EGL tools for shops of all sizes

<table>
<thead>
<tr>
<th>Eclipse EGL Tools¹</th>
<th>RBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Premier, IBM-supported development environment for EGL. Included in Rational Developer for z and Rational Developer for i for SOA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>EDT</th>
<th>RBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse-based development tools</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Compilation to Java</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Compilation to JavaScript</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Compilation to COBOL</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Deployment to WebSphere Application Server</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Deployment to Apache Tomcat</td>
<td>✔</td>
<td>✔²</td>
</tr>
<tr>
<td>Deployment to IBM CICS and IMS</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>DB2, MySQL, and Derby database support</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>SQL Server, Oracle database support</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>EGL Rich UI (RIA) visual editor</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Text UI tooling</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>EGL JSF support</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Simplified install and updates via IBM Installation Manager</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Support from IBM</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

¹proposed list of capabilities in initial contribution. Subject to change.
²MySQL support planned for future release. Subject to change.
EGL in Action (Side-by-Side Comparison)

**EGL Rich UI**

```egl
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
```

**HTML and JavaScript**

```html
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">
<head>
<meta charset="UTF-8"/>
<title>GoogleMap Example</title>
<script src="http://xgmaps.google.com/v2.x"></script>
</head>

<body onload="initialize()" onunload="GUnload()">
<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>
<div id="map_canvas" style="width: 500px; height: 300px"></div>
</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.