EGL Entwicklungsumgebung

Rational. software

Die IBM Rational Migration Extension for System i: Eine zukunftsweisende Lösung für Ihre RPG-Anwendung

Mit EGL und RMEi hat IBM eine Plattform geschaffen, die den Weg der Migration vernünftig und ökonomisch beschreitet.

„Wie zukunftssicher ist die Programmiersprache RPG? Lohnt sich ihre Weiterentwicklung überhaupt noch?“

Über diese Fragen wurde in den letzten Jahren viel diskutiert und spekuliert. Auch wenn RPG immer noch viele Weiterentwicklungen erfährt, so schienen Java und andere objektorientierte Programmiersprachen auf dem Vormarsch zu sein. Doch RPG hat nach wie vor viele Freunde, weil es – im Gegensatz zu Java oder .NET – speziell auf die Programmierung kaufmännischer Anwendungen zugeschnitten ist.

Trotzdem bewegte sich der Markt lange in die andere Richtung und es schien so, als sollten die zahlreichen stabilen und bewährten RPG-Anwendungen möglichst rasch durch Neuentwicklungen ersetzt werden. Aber der Schein trog: Das Resultat solcher Umstellungsprojekte war oft ernüchternd und viele ehrgeizige Vorhaben fanden vorzeitig ihr Ende.

Das hat einen simplen Grund: Zwischen prozeduraler und objektorientierter Welt herrscht ein Paradigmenbruch, der Migrationen zu einem heiklen Unterfangen macht – für Programme und Programmierer gleichermaßen. Viele sagen, es sei unmöglich, eine Brücke zwischen diesen Welten zu bauen. Allem Enthusiasmus zum Trotz, den die Open-Source-Gemeinde versprüht: Es hat sich gezeigt, dass natives Java nicht die Business-Sprache der Zukunft ist. Unbestritten ist aber auch, dass RPG alles andere als elegant ist, wenn es um die Neuentwicklung zum Beispiel von Web 2.0-Anwendungen geht.

IBM baut eine Brücke zwischen traditioneller und objektorientierter Welt

In diese Lücke stößt nun IBM Rational mit der Business-Sprache EGL (Enterprise Generation Language). EGL ist eine Programmiersprache der neuesten Generation, die sowohl den Client- als auch den Server-seitigen Bedarf bei der Entwicklung von Geschäftsanwendungen berücksichtigt: Moderne Frontends, basierend auf JSF und Ajax-Technologie, lassen sich ebenso herstellen wie klassische Batch-Programme mit Text-User-Interface à la 5250. Das Schöne daran: Aus EGL heraus kann man direkt Java-Programme erzeugen, aber weiterhin auch native i-Programme.

Doch ganz besonders wichtig: EGL ist keine objektorientierte, sondern eine prozedurale Programmiersprache, in der die modernen SOA-Aspekte implizit eingebunden sind. Die bisher fehlende tragfähige Brücke zwischen traditioneller und objektorientierter Welt wäre somit gebaut.

Die EGL Entwicklungsumgebung ist in folgenden Paketierungen erhältlich:

- *IBM Rational Business Developer (RBD)*
- *IBM Rational Developer for System i for SOA Construction (RDi for SOA)*

Soweit, so gut. Doch wie lassen sich bewährte RPG-Anwendungen sicher und effizient nach EGL migrieren, um nachhaltig auch deren Return of Invest zu sichern? Hierfür bietet IBM die Lösung mit dem Produkt:

- *IBM Rational Migration Extension for System i (RMEi)*

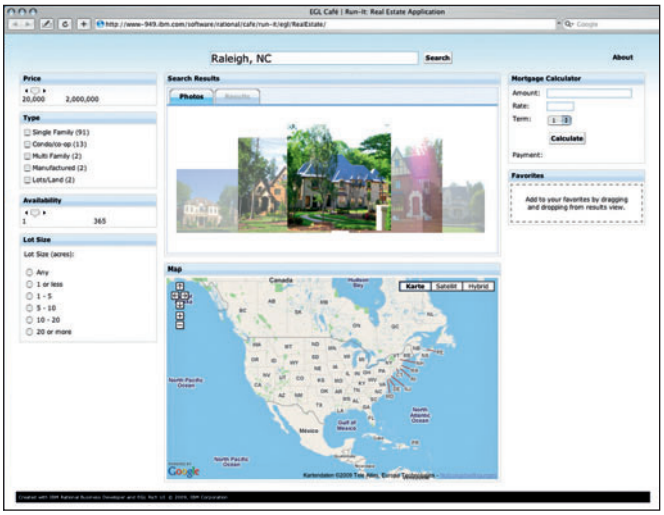
„Ganz wichtig: EGL ist keine objektorientierte, sondern eine prozedurale Programmiersprache.“

Mit dem RMEi können die vorhandenen RPG-Programme nach EGL migriert werden. Der große Vorteil beim Umstieg nach EGL ist: Kein Paradigmenbruch mehr zwischen prozeduraler und objektorientierter Welt. Dieser überforderte die meisten Anwendungsentwickler und führte in der Praxis zu schwer wartbaren Ergebnissen. Eine Modernisierung der Benutzeroberfläche, wie von zahlreichen Kunden in den letzten Jahren durchgeführt, ist zwar ein guter Anfang – doch reines Screenscraping allein erhöht nur die Komplexität, ohne die Funktionalität nachhaltig zu verbessern!

EGL nutzt die Stärken beider Welten –
prozedural und objektorientiert, effizient und integrativ

Mit einem Umstieg von RPG nach EGL fahren RPG-Entwickler viel besser als mit der direkten Migration von RPG nach Java. Denn EGL ist für sie sehr leicht erlernbar und hochproduktiv. Weiterer Vorteil: Sie brauchen kein weiteres Know-how im Bereich der komplexen Java- und OO-Technologien.

Dennoch bietet EGL – basierend auf Eclipse – eine der modernsten Entwicklungsoberflächen überhaupt. Es erhöht zum einen die Entwicklerproduktivität, da es mindestens 25 Prozent effizienter ist als eine native Java-Entwicklung. Zum anderen schafft es Akzeptanz sowohl bei den Java- als auch bei den RPG-Programmierern.



Die EGL-Entwicklungsumgebung

```
1  program caseSample type BasicProgram
     // Data Declarations
     testArray string [5] {"A", "B", "C", "D", "F"}     // string array of input values
2    FLDD string = "D";                                 // argument for when "FLDD"
     tstFld string;                                     // input field passed from function main() to functi
11
     function main()                                    4
       i int;                                           // index variable to control for statement
3      for (i from 1 to 5 by 1)                         // loop through FOR statement 5 times
6        tstFld = testArray [i] ;                       // assign element "i" of testArray[] to tstFld
         caseStatement (tstFld) ;                       // call caseStatement Function using tstFld as input
       end  // end of for loop                         5
     end  // end of main() function

     function caseStatement (tstF string)               // accept a sring field as input to the function

       case (tstF)                                      // define a case statement with tstF as the comparison operator
7        // condition 1
         when ("A", "B")                                // compare tstF to the literals "A" and "B"
8          // note when two arguments are present they are treated like an OR:
           // execute the next statement when tstF == "A" or tstF == "B"
           writeStdOut ("Condition 1: When tstF contains: "+ tstF +" - tstF == \"A\" or \"B\" ");
         // condition 2
         when  ("C")                                    // compare tstF to the literal "C"
           // execute next statement when tstF == FLDD
           writeStdOut ("Condition 2: When tstF contains: "+ tstF +" - tstF == \"C\" ");
         // condition 3
         when (FLDD)                                    // compare tstF to a variable
           // execute next statement when tstF == FLDD
           writeStdOut ("Condition 3: When tstF contains: "+ tstF +" - tstF == \"D\" ");
         otherwise                                      // execute when no when conditions match
           writeStdOut ("Otherwise:   When tstF contains: " + tstF + " - does NOT match A, B, C, or D");
       end // end of case statement
     end // end of caseStatement() function
9  end // end of program
```

1. Program declaration and type
2. Program variables
3. Main() Function - entry point
4. Local function variable
5. Function call
6. Simple FOR loop
7. User function
8. Case Statement
9. End of Program
10. Set Block {...}
11. Array declaration [] or [n]

Typische EGL-Sprachkonstrukte

Aus EGL lässt sich entweder nativer Code oder Java generieren. Nativer Code ist optimal für High Performance-Jobs auf dem System i wie Batch-Anwendungen oder interaktive Applikationen mit sehr großer Benutzerzahl. Java dagegen ist völlig plattformunabhängig. Die Entscheidung für Java oder Cobol kann jederzeit ad hoc getroffen werden. Eine einmalige vorherige Festlegung braucht es nicht.

EGL hat viele bewährte Konzepte aus RPG übernommen und kombiniert diese mit modernsten Standards für User Interfaces, Services, Architektur und Plattformunabhängigkeit. Alles ist in Eclipse integriert und auch für RPG-Programmierer nach zwei bis vier Wochen gut zu beherrschen. Dann können RPG-Programmierer endlich all das tun, was bisher ihren Java- und .Net-Kollegen vorbehalten war – wohlgemerkt auf der AS/400 und ohne sich mit komplexer Technologie herumschlagen zu müssen. EGL schlägt so die Brücke zwischen der alten und neuen IT-Welt.

Eine Umstellung nach EGL kann die Lebensdauer einer Anwendung um gut zehn weitere Jahre strecken, wodurch sich ein Return of Invest (ROI) sehr gut darstellen lässt.

„Eine Umstellung nach EGL kann die Lebensdauer einer Anwendung um gut zehn weitere Jahre strecken."

Wann ist eine Umstellung nach EGL sinnvoll?

- *Wenn die Grundsatzdiskussion aufkommt, ob man die Eigenentwicklung durch Standardsoftware ablösen soll. Häufig der Fall, wenn neue Manager in der IT oder in der Geschäftsleitung aktiv werden.*
- *Wenn neue Anforderungen nicht mehr in RPG realisiert werden, sondern unter Java oder .NET.*
- *Wenn es eine größere Zahl von Anwendern gibt, die ständig zwischen GUI und Green Screen Oberflächen verschiedener Anwendungen hin und her wechseln müssen.*
- *Wenn Programmanpassungen und Erweiterungen nicht mehr zeitnah ausgeführt werden können – und die Entwicklung zum erfolgskritischen Flaschenhals wird.*

Das EGL-Cafe – der Treffpunkt für die EGL-Community (http://www-949.ibm.com/software/rational/cafe/index.jspa)

EGL – Highlights im Überblick

- EGL kombiniert die Stärken beider Welten: die moderne, plattform-unabhängige Java-Entwicklung mit der Einfachheit und hohen Produktivität von RPG.
- EGL ist DIE strategische Businesssprache der IBM.
- Mit den IBM Migration Roadmaps (für VAGen, Natural, Cobol, RPG) werden Risiken eines Umstiegs minimal und beherrschbar.
- SOA ist zentraler Bestandteil der Sprache und der Entwicklungsumgebung.
- Erlaubt volle Konzentration auf die Business-Anforderungen und erspart die Auseinandersetzung mit der Technologie.
- Generierung nach COBOL für maximale Performance.
- Generierung nach JAVA für interaktive Programme und Frontend.
- Steile Lernkurve: 2 Wochen Schulung, nach 4 Wochen produktives Arbeiten möglich.
- Offene Sprache und Toolset – hohe Interoperabilität und Flexibität, um andere Sprachen einzubinden und plattformunabhängig zu deployen.
- Einfache, prozedurale Sprache, mit vielen Assistenten und „Build-In-Functions".
- Bei der OMG zur Standardisierung eingereicht.

„Gerade für große Firmen bietet EGL noch den Mehrwert, dass sich damit auch andere Programmier-sprachen wie Natural, VAGen, Informix 4GL und Cobol konsolidieren lassen.“

Sanfte Migration von RPG nach EGL – die IBM Solution Roadmap für System i

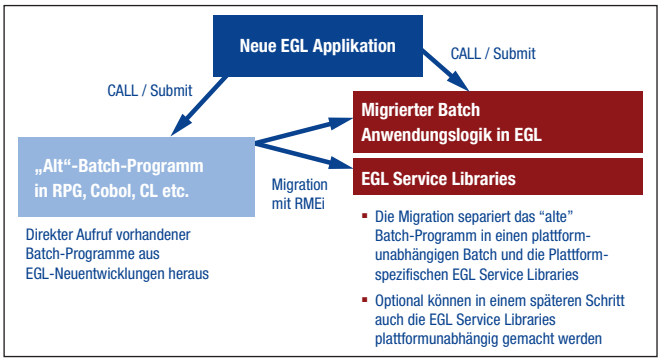
Anwender des System i brauchen für die Migration eine Lösung, die sowohl die Anwendung als auch das Know-how und Profil ihrer Entwickler berücksichtigt.

In den Köpfen der Entwickler steckt ein enormer Wert: das Wissen über das Unternehmen und die Abläufe. Neben der vorhandenen Anwendung muss dieses Wissen auch in der modernisierten Anwendung nutzbar bleiben.

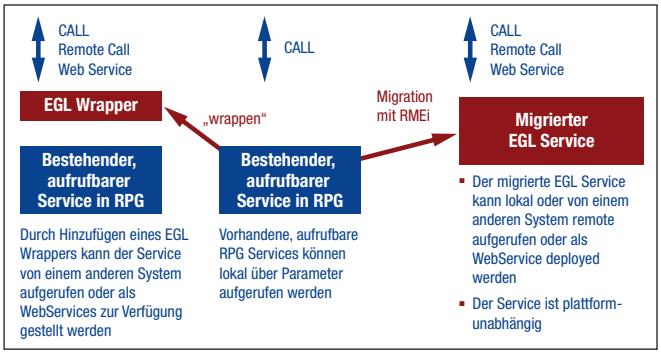
Genau diesen Bedarf adressiert die RMEi. Das Prinzip einer „Migration in unterschiedlich großen Schritten“ wird optimal durch verschiedene, kombinierbare Szenarien unterstützt:

- *RPG und EGL integrieren sich sehr elegant über Datenbank und Program-Calls. So lässt sich leicht eine Verbindung zwischen Alt und Neu herstellen.*
- *Dank Web-API-Technologie ist sogar die Integration auf UI Ebene möglich, so dass der Endanwender von Anfang an einen durchgängigen Workflow erlebt.*
- *Durch die direkte Code-Migration von RPG nach EGL kann man relevante Anwendungsbereiche auch direkt in die neue Welt überführen, ohne alles neu schreiben zu müssen.*

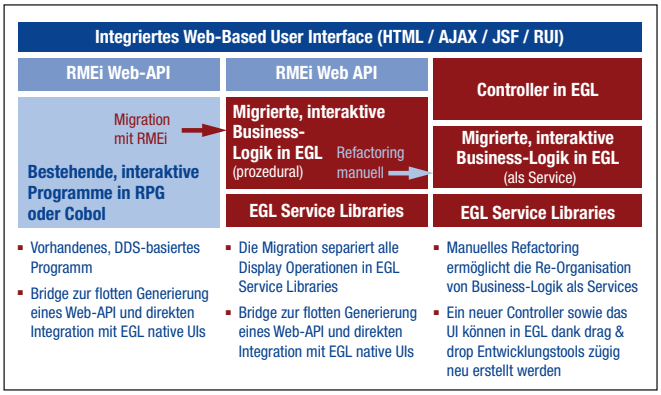
Mit EGL und RMEi hat IBM eine Plattform geschaffen, die den Weg der Migration vernünftig und ökonomisch beschreitet.



Aufruf und Migration von RPG-Batches



RPG-Services aufrufen, wrappen oder migrieren



Integration auf User-Interface-Ebene mittels Web-API-Technologie oder schrittweiser Migration

Da RMEi eine schrittweise Modernisierung unterstützt, kann man auch umfangreiche Anwendungen in leicht verträgliche Häppchen zerlegen. Gerade für große Firmen bietet EGL noch den Mehrwert, dass sich damit auch andere Programmier-sprachen wie Natural, VAGen, Informix 4GL und Cobol konsolidieren lassen. So reduzieren sich die Wissensanforderungen. Dabei werden alle IBM-Plattformen von Power Systems und System z unterstützt.

Zu Beginn der Migration findet man in der Analyse-Phase heraus, welche der verschiedenen Migrationsmöglichkeiten optimal zur Kundenanwendung passen. Dies bildet die Grundlage für die anschließende Migration und garantiert einen nachhaltigen Projekterfolg.

EGL Making a Difference: Community Health and Counseling Services accelerates delivery of HIPAA-compliant medical software with EGL

Kundenbeispiel im Überblick

- *Die Aufgabe*
  Als das Gesundheitsministerium des amerikanischen Bundesstaates Maine (Department of Health and Human Services – DHHS) ein Managed-Care-System im Bereich der Verhaltensmedizin einrichtete, musste CHCS (Community Health and Counseling Services) innerhalb kurzer Zeit Managed-Care-Module für sein System der elektronischen Patientenakten entwickeln.

- *Die Lösung*
  Mit Hilfe der IBM Rational Enterprise Generation Language (EGL) implementierte CHCS ein webbasiertes Tracking-System mit vorherigen Genehmigungen und schuf damit die Voraussetzung für eine schnelle Reaktion auf zukünftige Anforderungen durch neue gesetzliche Vorschriften.

- *Die Vorteile*
  Nach weniger als drei Monaten hatte ein kleines Entwicklerteam von CHCS mit Hilfe von EGL bereits ein erstes System fertiggestellt – mehr als vier Monate vor dem vom Bundesstaat Maine vorgegebenen Termin. Durch eine beschleunigte interne Softwareentwicklung ist das Unternehmen nun in der Lage, besser und schneller auf veränderte gesetzliche Bestimmungen zu reagieren.

„Durch eine beschleunigte Softwareentwicklung ist das Unternehmen CHCS nun in der Lage, besser und schneller auf veränderte gesetzliche Bestimmungen zu reagieren.“

Die komplette Referenz finden Sie auf folgender Website:

http://www-949.ibm.com/software/rational/cafe/docs/DOC-2720

IBM Deutschland GmbH
Pascalstraße 100
70569 Stuttgart
ibm.com/de

IBM Österreich
Obere Donaustraße 95
1020 Wien
ibm.com/at

IBM Schweiz
Vulkanstraße 106
8010 Zürich
ibm.com/ch

Die IBM Homepage finden Sie unter:
ibm.com/de

IBM Form RAB14018-DEDE-00 (04/2009)