Unleashing the "Big Iron" Power with Java

by Aviva Inc.

In its infancy, Java was a toy for creating gadgets in web pages. Many people did not realize the power of Java. Five years later, Java became a platform of choice for creating almost any type of application. Applications that can run on cellular phones, desktops, application servers, and of course, within your Web browsers. As Sun put it: "Write once, run everywhere".

But what about mainframes? It is a well-known fact that most of the worlds mission critical data and applications do not reside on Windows, Linux, Solaris or other Unix systems, but inhabit the "big iron", IBM mainframes. Some "legacy" applications perhaps written in COBOL 30 years ago, are still running on the mainframe. At that time sound software engineering practices and methodologies were not introduced.

One of the most important principles that drives application development in today's world is the clean separation of business logic and presentation. This principle was rarely applied when the "big iron" legacy applications were written. However, those applications work amazingly well (30 years of hammering bugs!) and still remain critical. The major challenges in using those "legacy" applications in our brave new world are: integrating with other data sources, and bringing the user experience to today's expectations, while keeping the data security and reliability intact.

Opening the "glass house"

Many of those born after 1980s maybe unaware of the term "glass house". In the technology world, it is the well-ventilated room full of large computers, servers, printers and tape units, with limited access to the general public.

Opening the "glass house" to the public means providing people remote access to the mainframe data and applications. (Extending the reach.) That certainly did not start with Java. In the beginning people were accessing mainframes using dedicated terminals. Then two important technologies introduced change.

The first, of course, was the Personal Computer. People soon found their desk real estate shrinking: the PC was sitting alongside with the bulky mainframe terminal. They were looking suspiciously similar–a screen, a keyboard and a box. However they were absolutely unable to exchange any data. Cumbersome work flow processes prevailed which involved looking at one screen and typing on a separate keyboard attached to another screen.

The introduction of terminal emulation software provided a solution to this dilemma. It runs on the PC, connects to the mainframe, and seems to behave just like a dedicated terminal, however it can move data between the mainframe and the applications running on the PC.

As the PC evolved, and as graphical interfaces became the norm, emulation products became more and more sophisticated, going far beyond simple "emulating". Aviva Inc. was one of the pioneers of this movement.

The second technology that caused yet a rethinking of the mainframe access was the Internet. Mainframe connectivity used to be governed by a complex and proprietary network architecture known as SNA, which was incompatible with the Internet. Obviously it had to evolve to the new worldwide network protocol: TCP/IP.

SNA was gradually replaced by TN3270, a protocol that encapsulates the mainframe presentation data in a Telnet-like envelope.

As the browser and HTML became the preferred way of viewing data, the inadequacy of the legacy user interface became unbearable. Therefore the "rejuvenation" era began. The original "green screens" were processed and radically transformed to a user friendly interface on a server and sent to the browser, or later to other devices, such as PDAs and mobile phones. During the transformation, server side Java technologies (servlets, JSP) started to play a role, although any other CGI-like technology would do it.

Java applets were also very much en-vogue. Emulation packages are known to be complex to configure, deploy and upgrade. A minor change of the configuration or a simple software upgrade could involve hours of working on thousands of computers. How nice it would be to handle all of these activities on a central Web server!

Java to Rescue

"Aviva for Java™", the secure host access solution for Java enabled Web browsers, dramatically increases the efficiency and productivity of administration. It enables the user to point the browser to a page and get the newly configured and upgraded package as a Java applet.

As electronic commerce started to flourish and as the "big iron" was involved in almost every transaction, the need to integrate mainframe based data with data from other sources emerged. As a result of such integration, the multitude and intricacies of the various data sources are completely hidden behind attractive Web applications from the user.

Java and in particular J2EE, is a prime candidate for the task for all the well-known reasons such as its high productivity, portability and security. Servlets, JSP, EJB, JDBC are all proven

technologies for building robust and platform independent applications. What about mainframe connectivity? There were no standard APIs and tools facilitating this difficult but important task.

A mainframe connectivity toolkit for Java–Aviva Host Integration SDK™

Aviva, with its vast experience in mainframe connectivity, realized that tools and APIs for programmatic mainframe access were needed to complete the facilities offered by Java for building e-commerce applications. Such tools and APIs are missing not only from Java, but also from all the competing platforms and languages. It was for these fundamental reasons that Aviva developed Host Integration SDK™ (SDK™).

Some may question, why a Java and not a .NET toolkit? Aviva considered Java a more promising platform not only for the well known reasons but also for one important element that has entered into the equation: IBM's firm commitment to Java. IBM pushed new life into the "big iron" by fully supporting Java on the mainframe. Today, mainframes are no longer dinosaurs, they run Unix and Linux sub-systems, they come with state-of-the-art Java VMs, feature powerful J2EE compliant application servers and so on. While non-Java solutions would allow applications running on other machines to tap mainframe data (using TCP/IP), Java solutions can run directly on the mainframe itself.

It is really a user's choice to select where the application or its discrete components will run. Some may prefer a distributed solution to avoid intruding or overloading the mainframe however others will choose to run on the mainframe for its many advantages: the power and legendary reliability of the "big iron", simplified topology, reduced network maintenance and so on.

The SDK™ is a comprehensive solution for providing mainframe connectivity to Java applications. While the emphasis is on server-side applications, the tools and API's are "container neutral". It means that the code will function properly if used from any type of Java application, in the most general sense, be it a servlet, a stand-alone application or even an applet.

The same host connectivity components offered to developers by the SDK™, have been used by Aviva to build an end-user type of product: Aviva for Java™. The fact that the SDK™ components have been used in such a commercial product for a long time makes us very confident with respect to their stability and completeness. What is in the SDK™?

- A rich set of versatile APIs that encapsulate the mainframe connectivity stack, including SSL. The various APIs are tailored for specific needs. The Aviva Class Library is the core of all other APIs and is geared towards server side applications. Aviva XML API is geared towards alternate presentations (for instance HTML) and B2B applications. Aviva Open Host Interface Objects are our contribution to IETF standardization efforts. Aviva Beans add a GUI on top of ACL, facilitating Rapid Application Development and IDE integration. Aviva Beans are more client side oriented, but invaluable during development and troubleshooting of server side applications (Everything is visualized!)
- Powerful development tools, including the Aviva Developer Assistant. Aviva realized that writing mainframe data access code is a challenging task, even when provided with powerful and elegant APIs. Aviva Developer Assistant is a code generation tool that records your mainframe screen navigations and creates an elegant abstraction: a hostlet. The hostlet is a Java class (actually a Java bean) that hides the complexities of a specific mainframe application screen scraping process. For instance, manipulating a CustomerInfoHostlet will consist essentially in instantiating it, providing the connectivity parameters, calling a setCustomerId() method, followed by executeNavigation(), followed by getCreditLimit(), getAccountBalance() a.s.o. Besides generating the hostlet, Aviva Developer Assistant generates your choice of fully functional calling code (prototypes): stand-alone application, servlet, JSP, RMI server, EJB, Oracle portlet or web service. The process is driven by templates, which are easy to create and modify. Therefore users can create their own templates.

Conclusion

Aviva Developer Assistant brings to the application developer the missing link: the link with the mainframe. This link is mostly ignored in mainstream Java literature and neglected because of its complexity. Almost every developer facing the need to integrate the mainframe data had to struggle and reinvent the wheel. Aviva believes that ignoring the mainframe link will not make it disappear; therefore we provide the Java developer community the means to access and integrate this essential data source. The Aviva Host Integration SDK™ provides developers with libraries that are pure Java, truly platform independent, highly scalable and non-intrusive. The developer decides the architecture of his own application (which probably does much more than accessing mainframe data) and the destination platform, while the SDK™ components will seamlessly just fit in.

As a Senior Software Architect for Aviva Inc, Eugene Aresteanu is playing a major role of defining the technical architecture for Aviva software products and solutions. His experience ranges from hands-on design, programming, product road mapping, and architecture. For the past 15 years at Aviva, Mr. Aresteanu pioneered and led the architecture, design, and development of the Aviva software products for various platforms. Mr. Aresteanu is a strong believer in Java's promise "Write once, run everywhere" and he thinks Java has a great potential in mainframe access field.