

The Portal as People-Centric SOA

Unifying the Enterprise across Java and .NET

May 2007

Contents

Executive Summary	1
The Portal as People-Centric SOA	1
Composite Applications and Workflows: Visible SOA	2
Portal Requirements	2
Introducing Mainsoft, Portal Edition	2
Mainsoft, Portal Edition: Enabling Front-End SOA	3
Federation Versus Integration: WSRP and Cross-Compilation	3
Multi-Language Java EE Platform	3
Technical Integrations: Don't Do Them with SOAP	4
People-Centric SOA: Unifying Services, People, and Platforms	4
References	4

Executive Summary

Increasingly, companies are turning to Java™ Enterprise Edition (Java EE) portals as the foundational starting point for their Service Oriented Architecture (SOA). With composite applications and workflows giving end users a single personalized view into the enterprise, portals offer an ideal platform for the most tangible and practical integrations.

However, not all integrations are created equal. An asymmetrical architecture using Web Services for Remote Portlets (WSRP) produces an asymmetrical environment in which composite application development is restricted to the Java components running locally on the Java EE portal, and functionalities such as single sign-on and universal branding are difficult to achieve. In addition, enterprises which choose a Java EE portal have limited opportunities to reuse their .NET skills and code.

When implemented as part of a Java EE portal environment, Mainsoft®, Portal Edition protects enterprise investments in .NET and Java skills and code. Together with portal standards such as WSRP and JSR 168/286, the Java Portlet Specifications, Mainsoft's enterprise-class .NET-Java EE interoperability software provides a direct path to a visible SOA and to a stable long-term SOA strategy.

Mainsoft's cross-compilation software enables .NET developers to produce standards-compliant Java portlets that run locally on Java EE portals, enabling .NET and Java developers to contribute equally to a visible SOA infrastructure. Composite applications

can be created using .NET and Java cooperative portlets. Regardless of whether services are written in C#, Visual Basic®, or Java, a Java EE portal built with Mainsoft, Portal Edition delivers a rich and well-integrated end-user environment, with equal access to the Java EE infrastructural services provided by the portal, such as role-based personalized interfaces, single sign-on, unified navigation, inter-portlet communications, and other portal services.

The Portal as People-Centric SOA

Managers want access to all their enterprise functionality as a seamless whole, and Service Oriented Architecture has emerged as the way to achieve this. SOA means business-oriented integration; in other words, SOA is built on coarse-grained, relatively large messages, which represent commonly-understood business concepts, as for example Purchase Order or Employee. Services do not need to be aware of each others' implementations or internal interfaces, and so avoid the limitations of proprietary, fine-grained, technically-oriented legacy APIs.

True SOA is difficult to achieve, and one of the hardest steps is defining services along meaningful business lines. Each service should consist of discrete functionality that provides true value, such as revenue reporting, shipping control, or human resources management. Yet there is intense pressure for technical constraints to define the services, creating service boundaries around technologies such as Java and .NET. Likewise, political struggles often mean that application borders are drawn around organizational units, as different departments work to control their own applications. This makes it difficult to bundle useful services out of pieces of the various applications.

But the users care only about their business needs, and they demand full access to enterprise applications, regardless of the implementation technology: Java, .NET, or any other.

An enterprise portal, such as IBM WebSphere® Portal, can answer the need for practical, user-centric SOA. It presents a Web application composed of portlets. Each portlet displays a well-defined unit of business functionality and shares information with other portlets as needed, showing relationships in the data and channeling users through a business process.

Focusing on tangible, useful applications makes it easy to define the services according to the needs of the business. Starting with legacy application interfaces, developers "portalize" a functionality using the portal's graphical toolkit, shaping the user interface for each service. This provides a valuable learning tool for users and executive sponsors: first, on the specific services provided by the enterprise, and second, on the



broader concept of correct business service definitions. On the technical side, it sets the stage for later integration of services at the business logic tier.

Composite Applications and Workflows: Visible SOA

As portals mature, the integration of their services deepens, moving from simple juxtaposition of applications to composite applications to full business-process workflow.

The most basic portals simply aggregate portlets into a consistent interface. This gives the users the broad enterprise view that they demand, customized for their business role, as well as the added value of consistent branding, standard controls, user management, auditing, and persistence services. For example, salespeople would see their sales reports alongside their quarterly quotas, alongside a customer relationship management application; they would also see the same benefit forms and corporate news that is displayed to all employees.

In the next step on the maturity ladder, portals present new, fully integrated, composite applications, in which portlets are wired together “on the glass” through inter-portal communications. Tools such as WebSphere Portal Application Template allow business analysts to lay out and wire together these overarching Web applications without coding, making it easy to build simple organization-wide applications. For example, sales representatives using a composite application could see their estimated sales commission together with the compensation agreement and sales reports from which it was calculated. Likewise, a business line manager could see integrated reports on supply chain management, manufacturing, and shipping. These reports are generated in the portal without recourse to expensive and hard-to-use application integration tools.

Finally, the most mature portals present an entire human-driven workflow across the enterprise. These applications guide users through a business process, passing responsibilities around as needed, and obviating the need for manual effort. For example, when on-boarding a new employee, different roles such as human resources, IT, and the employee’s supervisor would pass tasks smoothly between them in the workflow. They would no longer need to copy data between applications, to remember the correct sequence of applications to use, or to hand off responsibility to each other manually.

The tangibility and visibility of the portal enables a phased progression towards deeper integration and workflow, since portal developers and content managers can create workflows on the glass. Because portals will soon also support Business Process Execution Language (BPEL), visual integrations blaze a path for later adoption of sophisticated “behind-the-glass” implementation of workflows at the business logic tier.

Portal Requirements

As the business user’s primary window into the enterprise, a complete front-end SOA environment within a portal must support stringent requirements.

1. It must deliver a seamless user experience. Visually, a portal must combine consistency with flexibility. On the one hand, every portlet must have cleanly integrated branding, layout, and rich user experience. On the other hand, the portal must be flexible enough to show each user an interface relevant to his or her role.

2. It must support both of the popular enterprise platforms: Java and .NET. Many organizations use both .NET and Java development technologies, because of corporate mergers or independent internal projects, and so a portal must give end users complete, transparent access to all services and user interfaces, regardless of the technology they were coded in.

3. Portal-based composite applications and workflows must operate cleanly across both Java and .NET. One approach to supporting both platforms is to run the ASP.NET applications on Microsoft’s® IIS and the Java applications on a Java EE portal server in tandem, integrating them using WSRP, so that ASP.NET applications can be visualized in the Java EE portal. However, a remotely served portlet will always have poorer functionality than one running directly on the server which presents it. This is especially true with WSRP version 1, which does not support inter-portal communications. As a result, composite application development in Java EE portals such as WebSphere Portal is limited to Java business components.

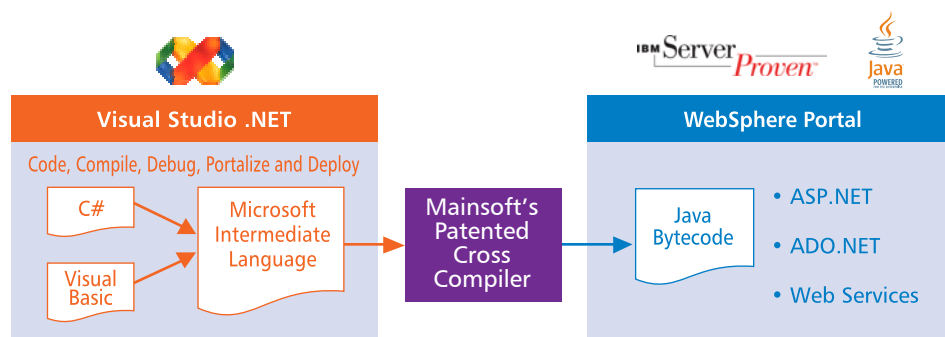
4. To be practical, the development process must preserve existing investments in .NET and Java skills and code. The adoption of a portal cannot require extensive re-training of developers or re-coding of user interfaces and underlying services, regardless of their implementation technology. Existing skills and code must continue to function. Rapid re-use of existing applications helps portals avoid the common “Teflon portal” syndrome, in which an enterprise launches its portal with a scattering of isolated enterprise applications and a few irrelevant starter portlets such as stock tickers or weather. Teflon portals often fail to “stick,” and so never gain the budget needed to expose new functionality.

Introducing Mainsoft, Portal Edition

Mainsoft, Portal Edition provides a set of .NET extensions for Java EE portals and offers a pragmatic solution for a symmetric .NET/Java portal architecture. Based on patent-pending technology developed over the last five years, Mainsoft, Portal Edition includes a seamless plug-in to the popular Visual Studio® development environment, which enables .NET developers to write .NET code in ordinary C# or Visual Basic using the usual ASP.NET controls and .NET libraries, as well as Java EE portal services exposed through .NET interfaces.

Mainsoft, Portal Edition works by cross-compiling .NET Intermediate Language into native Java bytecode, producing JSR 168 compliant Java portlets that run the same as any other portlet running locally in the Java EE server. It includes an ASP.NET runtime that enables .NET developers to code against their familiar APIs:

- The Java EE portal’s look-and-feel is exposed as standard ASP.NET themes, Java data sources as ADO.NET, and portal services such as People and Location Awareness as drag-and-drop .NET controls.
- ASP.NET role-base security is transparently mapped to WebSphere Portal Membership and Authentication providers.
- Java-standard APIs, such as the Portal User Management Architecture (PUMA), are transparently accessed through interfaces of ASP.NET providers.
- Access to widely used .NET enterprise services, such as SQL Reporting Services, is available through C# portlets provided in source code form.



Architecture of Mainsoft, Portal Edition

More broadly, .NET developers can call on all Java library functionalities, whether encoded as JSR168/286 APIs, portal infrastructure services APIs, or generic Java class libraries. Such full functionality could also be achieved by re-coding the .NET applications in Java. But the cost of this approach can be prohibitive, both in manpower and in time-to-value. Mainsoft, Portal Edition, on the other hand, allows the migration to happen rapidly and automatically at compile-time, without introducing the risks inherent in a rewrite. Typically, only 0.5% of code needs to be altered, usually as a way of adding extended Java functionalities that are not available in .NET.

With Mainsoft, Portal Edition, .NET developers can continue working side-by-side with Java developers indefinitely, both deploying code to the same portal server. Even non-specialists can contribute to the people-centric SOA of the portal: Business analysts can build composite applications and workflows with inter-portlet communications between .NET and Java components, using interactive tools such as WebSphere Portal Application Template.

Mainsoft, Portal Edition: Enabling Front-End SOA

A sustainable, long-term portal strategy recognizes the need for coherence in visible functionality. Each enterprise information system may maintain its own user interface, but their unavoidable single point of contact is the user, who must see clear, consistent composite applications and workflows. An architecture based on a Java EE portal server, such as IBM WebSphere Portal, and Mainsoft, Portal Edition recognizes that even when heterogeneity exists on the back-end, the user needs consistency and ease-of-use above all.

By deploying .NET services locally on the Java EE server, Mainsoft, Portal Edition enables rich, consistent user interfaces, since all portlets can use the common portal infrastructure for shared branding and look-and-feel. It also supports the modern AJAX Web architecture, which allows Web interfaces to respond quickly and interactively to user input. Web interfaces, known as thin clients, no longer need be the poor cousins of fat-client desktop applications. AJAX depends on frequent, narrowly-defined communication between the browser and the server, and so cannot work when a portlet is served remotely with WSRP; but with Mainsoft and the open-source ASP.NET AJAX framework (Anthem.NET), a Java EE portal server can provide the full richness of AJAX to portlets coded in .NET and in Java. Support for Microsoft's AJAX framework (Atlas) is scheduled for the first part of 2008.

Federation Versus Integration: WSRP and Cross-Compilation

Mainsoft's cross-compilation capabilities complement the WSRP standard, which is ideal for federating external services and portal-to-portal interoperability. However, WSRP does not support inter-portlet communications, leaving each portal as a functional silo. When composite application development as well as a rich, end-user experience are requirements, Mainsoft's cross-compilation approach provides the requisite rich integration, composing an organization's applications through inter-portlet communications.

Version 2 of WSRP, primarily centered on inter-portlet communications, is scheduled to come out in late 2007, and it is expected to reach wide deployment in 2009. Mainsoft will support WSRP version 2 to enable the federation of remote .NET Web and portal applications. Even then, Mainsoft's cross-compilation will serve as an essential complement to WSRP version 2 by providing symmetric access to infrastructural services, and by enabling technical integrations with the requisite fine-grained communications.

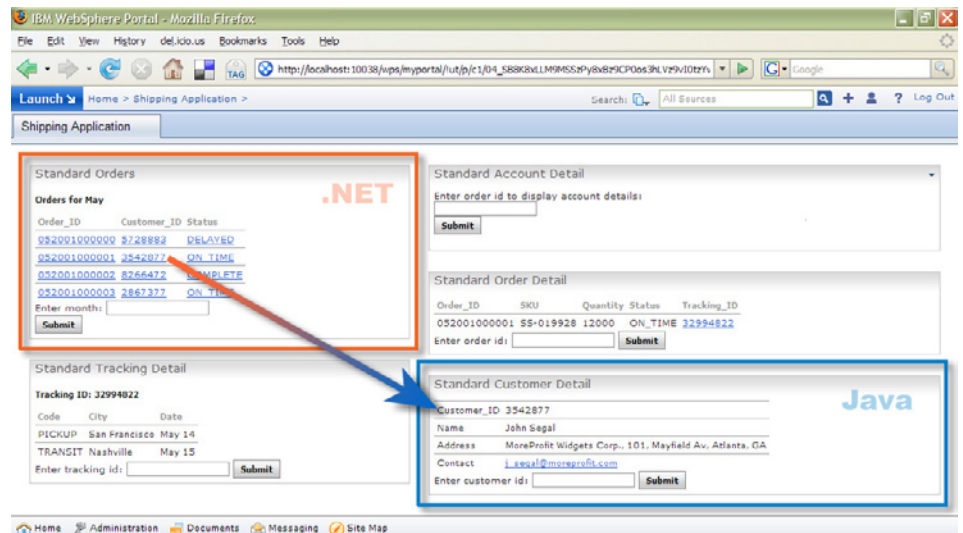
In the meantime, enterprises looking to federate SQL Reporting Services within WebSphere Portal can use Mainsoft's .NET Extensions for WebSphere Portal.

The Java Portlet Specification, JSR 168, defines a standard and open API into Java EE portal servers, encouraging healthy competition between vendors. Mainsoft, Portal Edition implements this standard for .NET-coded portlets. It also provides inter-portlet communications, which are not supported by JSR 168. Mainsoft will also enable cross-compilation and execution of .NET code on JSR 286 containers. JSR 286 will add support for standard inter-portlet communications, caching, and direct access to the underlying request and response.

With support for Web services standards and cross-compilation capabilities, Mainsoft supports the full range of .NET-Java EE interoperability technologies for Web applications: the federation of .NET remote assets into Java EE portals and tight integration of .NET-Java applications on JSR-compliant containers.

Multi-Language Java EE Platform

Mainsoft's cross-compilation software establishes C# and Visual Basic as fully supported languages for the Java Virtual Machine (JVM). In fact, running non-Java languages on the JVM is a major priority of Sun Microsystems and the Java community. Java 6 added support for plugging in multiple



scripting languages (with the JSR 233 standard), already including Python, Ruby, Visual Basic, JavaScript, and others. The upcoming Java 7 is scheduled to have even deeper support for non-Java languages (JSR 292).

Thus, the JVM becomes a true abstraction layer, in keeping with an ongoing industry trend towards virtualization and open systems. The virtual machine is no longer locked to one language. Instead, the IT organization is free to de-couple development decisions from production decisions and use Visual Studio for development and a Java EE portal server for deployment.

Because Mainsoft generates native Java bytecode, it creates portals that behave in every way like those composed of ordinary Java-language portlets. Benchmarks show that these .NET/Java portals have a zero performance hit compared to all-Java or all-.NET portals. Indeed, by taking advantage of Java EE high performance and scalability, they can even outdo the original .NET application performance. This is in contrast to multi-portal approaches, using WSRP for integration, in which performance is reduced by network activity and by rendering HTML to and from SOAP/XML. **See the full Performance Study** online (<http://www.mainsoft.com/solutions/pdfs/PerformanceStudy.pdf>).

The pure-Java-bytecode runtime also allows developers to debug uniformly across the whole application. They work with a single, consistent system, regardless of the language of the source code.

are well suited to these goals. Technical integrations are best implemented as direct method calls within the JVM. Mainsoft makes these direct invocations possible.

People-Centric SOA: Unifying Services, People, and Platforms

The portal gives enterprises a long-term basis for people-centric SOA, starting with aggregated service-oriented user interfaces, and ramping up to composite applications and full on-the-glass workflows. But a portal can only fulfill the SOA promise when all enterprise services are equally accessed through the enterprise portal, regardless of whether they are written in Java or in .NET, and when all developers can continue to apply their existing skills and code. Mainsoft, Portal Edition, creates a symmetric, standards-based architecture in which portlets in multiple software languages can be composed into coherent composite and workflow applications.

References

- IBM White Paper: **"WebSphere Portal: An on-ramp to a service oriented architecture"** (http://www-07.ibm.com/sg/soa/downloads/WebSphere_Portal.pdf).
- Gartner Research: **"A portal may be your first step to leverage SOA"** (http://www.gartner.com/DisplayDocument?doc_cd=130149).
- Forrester Research: **"Choosing the best option for .NET-Java/J2EE interoperability"** (http://www.mainsoft.com/solutions/pdfs/Forrester_Tech_Choices.pdf).
- Laurence Moroney: **"Jumpstart SOA: Pragmatic approaches to integrating .NET and Java components within WebSphere Portal,"** Java Developer's Journal, Nov. 2006 (http://www.mainsoft.com/news/articles/JDJ_JumpstartSOA_Nov2006.pdf).
- Resources for Mainsoft, Portal Edition, can be found on the <http://www.mainsoft.com> and <http://dev.mainsoft.com/> Web sites, including success stories, technical specifications, samples applications, and detailed tutorials.

Technical Integrations: Don't Do Them with SOAP

Newcomers to SOA often think of it as a new version of older remoting technologies such as RMI, DCOM, or CORBA: a way to call a function from one application or machine to another. But Web services, though ideal for integration of business services, are not well suited for integration at a *technical* level.

Loosely coupled business service integration, which is at the heart of SOA, passes only coarse-grained, business-oriented messages. Though this is perfect for portlets, technical services such as themes, styles, and user management, or the composition of business services from low-level API calls need to be tightly coupled and fine-grained, with relatively smaller units of data passed into object-oriented methods. These require speed and precisely-specified interfaces, but XML Web services are too slow and loosely defined; binary, low-level technologies

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.