Service-oriented Web application development with IBM WebSphere Portlet Factory software.
Overview

IBM WebSphere® Portlet Factory software provides leading-edge technology for speeding the creation of service-oriented applications—as portlets in IBM WebSphere Portal software—that integrate data from multiple sources, including databases, spreadsheets, data warehouses, enterprise applications (SAP, PeopleSoft, Siebel, IBM Lotus® Domino® server), and Web services. Recent additions to WebSphere Portlet Factory software broadly expand upon this core capability and focus on improving the practical use and management of Web services in a service-oriented architecture (SOA) environment. Creating applications for an SOA environment presents many challenges to development teams, including maintaining access to remote services (and enterprise applications), dealing with complex and differing design, test and deployment environments, and managing the inevitable changes to services in production use. WebSphere Portlet Factory software’s tooling is uniquely suited to helping developers exploit development in an SOA environment and maximize its value, including easing the testing process, creating multiple variations of a service and dynamically controlling the variants of a service to use.

WebSphere Portlet Factory software’s newly expanded set of features for building applications for service-oriented architecture provides several benefits. A dual-level architecture (service side and user interface [UI] presentation side) enables developers to cleanly separate the back-end services portion of an application from the presentation layer. Development teams can work efficiently and independently on both portions of the application without requiring constant interactions during either the development or testing phases. UI developers can also work disconnected from back-end systems or services, greatly simplifying their environment, helping ease the testing burden and speed their efforts.
New functionality

WebSphere Portlet Factory software has added a new set of builders that extends the functionality of its original services capabilities and makes SOA-based development even easier. The first services builder set enabled users to create Simple Object Access Protocol (SOAP)/Web Services Description Language (WSDL) compliant services, and to call out to SOAP/WSDL services, either within or external to the WebSphere Portlet Factory software. The new builder set greatly extends this functionality, enabling developers to even more quickly expose and use WebSphere Portlet Factory models as flexible and managed service providers. It also provides very high degrees of automation and back-end isolation on the UI (or service consumer) side of the services architecture equation. Automatic generation of test pages allows the service provider developer to verify that back-end access operations are working correctly. Additionally, the new builders set includes a powerful mechanism for dynamically assigning the service provider to be used by service consumer models, enabling implementation time and run-time service swapping without code changes.

There are three major benefits of cleanly separating the service provider and service consumer layers of an application into loosely connected models. First, developers working on either portion of an application can work independently, eliminating the need to closely coordinate work across front-end and back-end teams. Second, once an application has been built and deployed, you can automatically swap alternative service provider models at run time, either to vary the functionality of the application or to invoke a backup service, without requiring any modification of the consumer layers. Third, and perhaps most important, once back-end applications are encapsulated into a set of well-defined services, the services can be reused across a project and even across other types of applications within your organization. Thus, by adopting a service-oriented architecture and using WebSphere Portlet Factory software to create your services, you can help speed the creation of new applications by assembling them from these services, rather than having to code the applications from the ground up.
There are three primary builders that create this dual-layer, service-oriented architecture:

- **Service definition builder.** Establishes that a model is a service provider.
- **Service operation builder.** Creates an operation defined within the service. Multiple operations can be defined for a service (just as in a WSDL service).
- **Service consumer builder.** The builders for the UI side; used to invoke a service defined in a service provider model.

Figure 1 illustrates a simple implementation of a Web application that spans two models, one a service consumer, the other a service provider. The service provider model contains one or more service definitions and service operation builders that publicly expose services in the WebSphere Portlet Factory Web application (WebApp). These operations can invoke different back-end systems or Web services, enabling a single service to expose data from multiple sources. In the consumer model a service consumer builder creates the programmatic linkage to the service provider model and introspects the operations, making them accessible to other artifacts of the WebApp in the consumer model.

![Figure 1](image-url)
In addition to the primary builders listed on page 4, the following key features round out WebSphere Portlet Factory software's robust SOA functionality:

- **Stub service models.** A simple button in the service definition builder will automatically generate a stub service model, a self-contained snapshot of the service's operations, schemas and sample (or actual) data. This stub model can then be used for development and testing without requiring access to the original data source. As a result, developers can build and test fully functional models while working disconnected.

- **Service test builder.** The service test builder automatically generates the code and pages required for testing all the operations of a service. This enables the service developer to do his or her work without requiring a presentation layer or a test harness.

- **Service documentation builder.** The service documentation builder generates technical documentation about the services on either side of this architecture, either those created by a service provider model or those used by a service consumer model.

- **Dynamic service mapping.** This feature allows the service provider model to be dynamically swapped without changing the consumer side, enabling the use of the actual service provider model, a stub service model or an alternate service provider implementation. This provider model selection capability is available at run time as well as during the development process, giving maximum operational flexibility in an SOA environment. For example, there may be two sources of the same (or essentially the same) data, one from a data warehouse, perhaps, and another from a running application. Using profiling to control the model mapping mechanism, an application or portlet can be switched between the two service implementations for different roles, one requiring historical data, the other a real-time view. Figure 2 illustrates this capability.
• **Service interfaces.** To facilitate swapping of service provider models, this improved SOA builder set supports the notion of a service interface, similar in concept to a Java™ interface. This service interface capability provides assistance and verification when creating an alternate implementation of a service, to ensure that the services can be effortlessly swapped, without affecting the service consumer model or user interface.

Figure 2 illustrates the dynamic service mapping mechanism.
Summary

Much has been written about SOA as organizations begin to embrace the encapsulation of back-end systems into XML-enabled Web services. A lot of attention is given to the use of Web services as building blocks that can be assembled together into a loosely knit orchestration as part of a larger composite application. This is a strong value proposition for the use of Web services and one that is fully supported by WebSphere Portlet Factory software and its traditional use in the rapid creation and assembly of services. With a now even more flexible services builder set, WebSphere Portlet Factory software provides additional benefits for using a service-oriented approach in Web application development, namely the ability to easily create cleanly separated service providers and service consumers, and then to swap the providers to meet various use cases (development, testing, alternative services) without needing to modify the consumer side at all.

ibm.com/websphere/portletfactory/