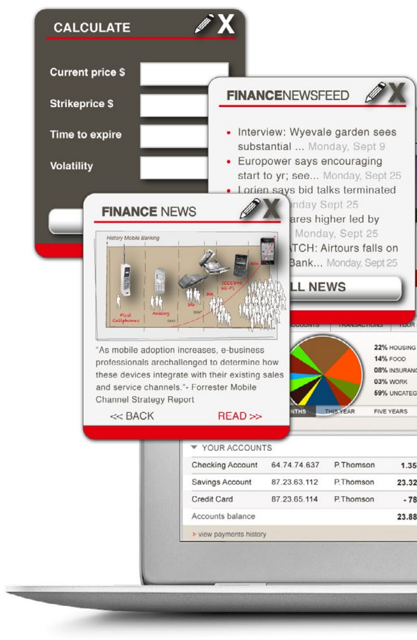
BACKBASE

Comparison

Widgets vs. Portlets



December, 2012

Comparison Widgets vs. Portlets

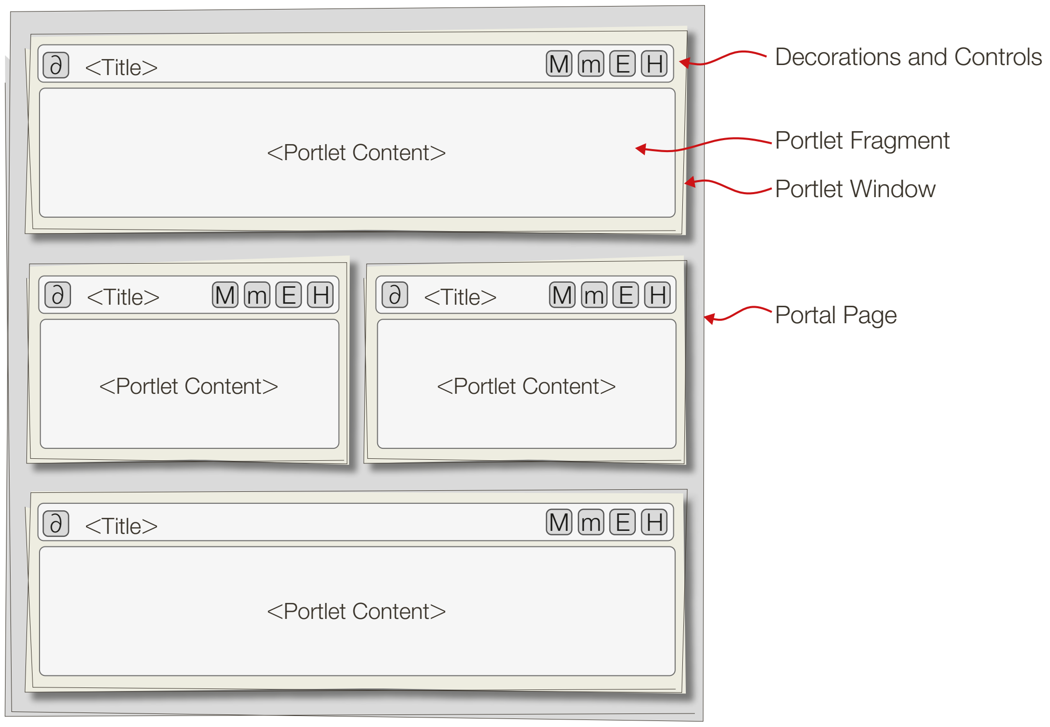
Introduction

This document compares portlets, web widgets, and Backbase widgets. It consists of two parts: the first part gives some background about portlets and web widgets, while the second part focuses on how Backbase widgets improve on both portlets and ordinary web widgets. Readers familiar with portlets and web widgets can skip directly to the second section.

Portlets

What Are Portlets?

"Portlets are web components--like servlets--specifically designed to be aggregated in the context of a composite page. Usually, many portlets are invoked to in the single request of a portal page. Each portlet produces a fragment of markup that is combined with the markup of other portlets, all within the portal page markup." (from the Portlet Specification, JSR-168).



Portlets are pluggable web application components like weather reports, discussion forums, or stock quotes. They generate HTML markup that is then assembled into a single page. A server adds CSS stylesheets, cookies, and resources, and delivers the page to a browser. Portlets allow both static and dynamic binding: portal applications can make portlets available in a registry at runtime for the user to select and position. Portal servers typically let users customize pages by rearranging, showing, or hiding portlets, and they provide single sign-on and role-based personalization.

Portlets

Portlets implement a Java interface to the standard portlet API and they can plug into any standard-compliant portal server: for example, the Java Portlet specification v2.0 (JSR-286) defines a runtime environment for portlets and the corresponding Java API. While the term portlet originally designated Java Portlets, other technologies like ASP.NET Web Parts use a similar model.

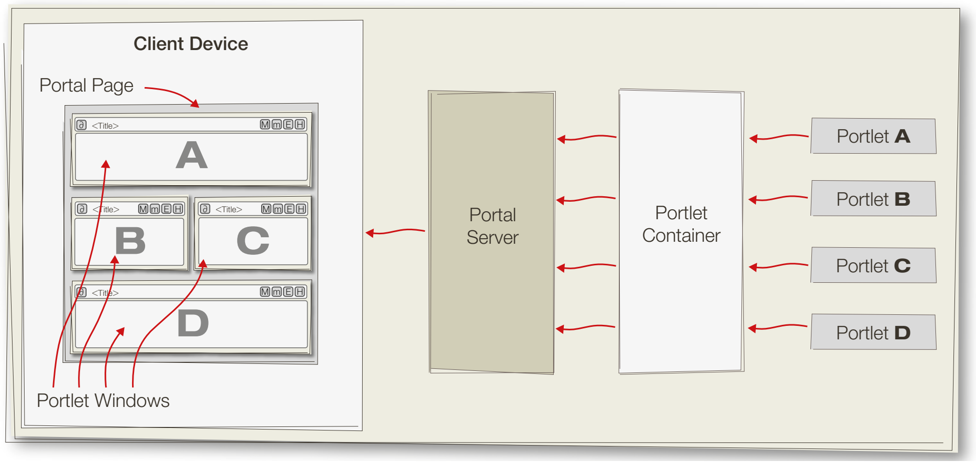
Example: Java source for Hello World portlet (JSR 168):

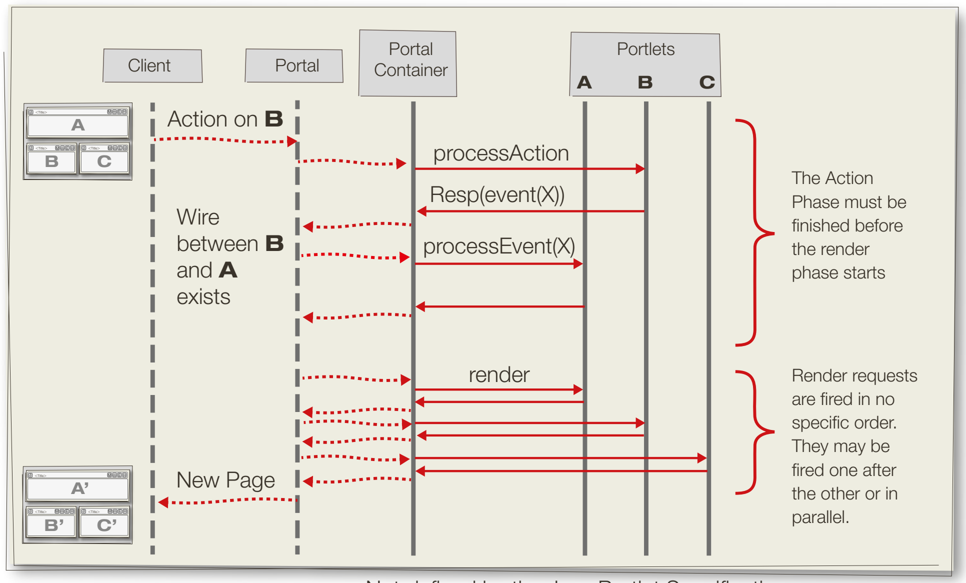
Example Portlet

```
import javax.portlet.*;
import java.io.*;

public class HelloWorld extends GenericPortlet {
    public void init (PortletConfig portletConfig) throws UnavailableException,
PortletException {
        super.init(portletConfig);
    }

    public void doView(RenderRequest request, RenderResponse response) throws
PortletException, IOException {
        response.setContentType("text/html");
        PrintWriter writer = response.getWriter();
        writer.println("<p class='wpsPortletText'>Hello Portal World!</p>");
     }
 }
```

What Is a Portal Container?

Portlet containers are the runtime environments where portlets are instantiated, used, and destroyed. They compose portlet fragments into HTML pages and handle portlet events.




Client
Portal
Portal Container
Portlets
A
B
C
Action on B
processAction
Resp(event(X))
Wire between B and A exists
processEvent(X)
render
New Page
A'
B'
C'
The Action Phase must be finished before the render phase starts
Render requests are fired in no specific order. They may be fired one after the other or in parallel.

Not defined by the Java Portlet Specification

What are JSR-168 and JSR-286?

JSR-168 and JSR-286 are the Java Portlet Specifications. They define a contract between the portlet container and the portlets, and provide a convenient programming model for Java portlet developers.

The Java Portlet Specification V1.0 was developed under the Java Community Process (JCP) as Java Specification Request JSR-168. It introduces the basic portlet programming model with:

- two phases of action processing and rendering that conform to the Model-View-Controller pattern;
- portlet modes that enable the portal to advise the portlet on what task it should perform and what content it should generate;
- window states that indicate the amount of portal page space to be assigned to the portlet-generated content;
- a portlet data model that allows the portlet to store information about the view, session, and user customizations;
- a packaging format bundling portlets and other Java EE artifacts to make them deployable on portal servers;
- integration of different web-based applications supporting the delivery of information and services.

JSR-286 is the Java Portlet specification v2.0 as developed under the JCP. It was developed to improve on the shortcomings of JSR-168 and it aligns with WSRP version 2.0.[1]. Some major features include:[2]

- inter-portlet communication through events and public render parameters;
- dynamically-generated resources served directly through portlets;
- AJAX or JSON data served directly through portlets (in JSR-168, these data had to be served through servlets outside the portlet context);
- portlet filters and listeners.

What Is WSRP?

Web Services for Remote Portlets (WSRP) “defines a set of interfaces and related semantics which standardize interactions with components providing user-facing markup, including the processing of user interactions with that markup. This allows applications to consume such components as providing a portion of the overall user application without having to write unique code for interacting with each component”.

The WSRP protocol provides a standard for portlets running remotely across different hosts. It defines common interfaces to display remote portlets inside the pages of a portal. The portlet container and the portal interact through SOAP messages.

For example, a web site may use portlets in remote portlet containers to allow registered users to turn on or off portions of a page, add features, or delete them. Thanks to WSRP, end-users may not notice that different portlets reside in a remote portlet containers.

Are Portlets Interoperable with Other Technologies?

Java Portlets and ASP.NET Web Parts have similar goals, but are not interoperable. WSRP addresses this issue at the protocol level by exposing remote portlets as web services. Communications between the portal server (WSRP consumer) and the portlets (WSRP producer) occur via SOAP, so developers can use different languages and runtime frameworks. In practice, however, portlets from different producers are rarely combined because of different standard implementations.

Can I Use AJAX with Portlets?

Portlets can use AJAX for improved responsiveness and can produce documents that resemble widgets running in a browser container. However, the portlet specifications mainly cover the server-side APIs and services, not the HTML and Javascript sent to the browser. As a result, AJAX solutions are often built in an ad-hoc fashion and are not reusable.

What Are the Main Problems of Using Portlets?

According to Gartner (Report No. G00166378, “Get Ready for the ‘Portlet-Less’ Portal”), projects using portlets may fail because of:

- Budget overruns: portals using portlets are typically expensive and, depending on the type of portal, can take several months to deploy;
- Lack of structure, sub-par content, no connections to relevant applications, or bad user experience;
- Lack of a user experience tailored to the intended audience. Companies might question why they have to spend significant amounts of money on what seems little different from the legacy web environment.
- Overlapping investments in multiple portals that fail to meet expectations;
- Complaints about the user experience, especially when compared to other portals found on the web. Customers may ask why their portals do not “look like iGoogle”.

Where Can I Learn More about Portlets?

There are a number of resources online, including:

- *Wikipedia,*
  http://en.wikipedia.org/wiki/Portlet
- *Sunil Patil, What Is a Portlet,*
  http://oreilly.com/java/archive/what-is-a-portlet.html
  O'Reilly, September 14, 2005
- *Stefan Hepper, Oliver Köth, What's new in the Java Portlet Specification V2.0 (JSR 286)?,*
  http://www.ibm.com/developerworks/websphere/library/techarticles/0803_hepper/0803_hepper.html
  18 Mar 2008
- *Stefan Hepper and Stephan Hesmer, Introducing the Portlet Specification, Part 1,*
  http://www.javaworld.com/javaworld/jw-08-2003/jw-0801-portlet.html
  JavaWorld.com, 08/01/03
- *Stefan Hepper and Stephan Hesmer, Introducing the Portlet Specification, Part 2,*
  http://www.javaworld.com/javaworld/jw-09-2003/jw-0905-portlet2.html
  JavaWorld.com, 09/05/03
- *Deepak Gothe, Understanding the Java Portlet Specification 2.0 (JSR 286): Part 1,*
  Overview and Coordination Between Portlets,
  http://developers.sun.com/portalserver/reference/techart/jsr286/jsr286.html
  Oracle Sun Developer Network (SDN), January 2010
- *Deepak Gothe, Understanding the Java Portlet Specification 2.0 (JSR 286): Part 2, Serving*
  Resources and Other New Features,
  http://developers.sun.com/portalserver/reference/techart/jsr286/jsr286_2.html
  Oracle Sun Developer Network (SDN), January 2010
- *Deepak Gothe, Understanding the Java Portlet Specification 2.0 (JSR 286): Part 3,*
  Extensions,
  http://developers.sun.com/portalserver/reference/techart/jsr286/jsr286_3.html
  Oracle Sun Developer Network (SDN), January 2010
- *Portlets in Action*
  http://portlets-in-action.blogspot.nl/
- *OASIS Web Services for Remote Portlets (WSRP) TC*
  https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

BACKBASE

Web Widgets

What Are Web Widgets?

A web widget is a mini-application that can be embedded into third-party sites by any user with authorship rights on a page (e.g. a webpage, blog, or profile on a social media site). Widgets can be installed by simply adding some code: for example, Google offers a number of widgets (called gadgets) that you can add to a page by copying and pasting a <script> tag into the HTML source code.

The idea of web widgets is not new: embeddable chunks of code have existed since the early developments of the World Wide Web. Early web widgets provided functions such as link counters and advertising banners.

What Are W3C Widgets?

W3C defines a widget as "an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user's machine or mobile device. A widget may run as a stand-alone application (meaning it can run outside of a Web browser)".

W3C widgets are ZIP packages bundling HTML, cascading stylesheets (CSS), Javascript files, and other resources. They can be deployed in HTML files with a Multipurpose Internet Mail Extensions (MIME) type of "application" or "widget". Once downloaded, they can be re-used just like installed applications, thereby reducing bandwidth usage.

The W3C "Widget Packaging and Configuration" specification (http://www.w3.org/TR/widgets/) describes configuring, packaging, and deploying widgets. You can use widgets for small applications like calendars, weather reports, chats, and more.

The W3C "Widget API Specification" (http://www.w3.org/TR/widgets-apis/) defines an application programming interface (API) to, for example, access widget metadata or persist data.

Where Can I Learn More about Web Widgets?

- *ABC, A Brief Widget History,*
  http://www.niallkennedy.com/blog/2007/09/widget-timeline.html
- *W3C Widgets Current Status,*
  http://www.w3.org/standards/techs/widgets#w3c_all
- *Native Web Apps Community Group Home Page,*
  http://www.w3.org/community/native-web-apps/
- *Misconceptions about W3C Widgets,*
  http://marcosc.com/wp-content/uploads/2011/10/Misconceptions-about-W3C-Widgets.pdf
- *Paco Azevedo Mendes, Evaluation of widget-based approaches for developing rich internet applications,*
  http://wiredspace.wits.ac.za/handle/10539/9484

Web Widgets vs. Portlets

How Are Web Widgets and Portlets Similar?

Portlets and widgets have many similarities:

- they both provide a user interface to services running on back-end systems, though there are cases where they perform some application logic as well;
- they can both pass information and context to other portlets or widgets, and consume context from other components;
- end-users and administrators can personalize which portlets and widgets appear on a page, how they look, and how they behave.

How are Web Widgets and Portlets Different?

Portlets are server-side component models, while widgets are client-side component models that run in a browser container. Even though portlets can generate markup that runs client-side like a widget, the portlet specifications mainly cover the server-side APIs and services, not the HTML and Javascript sent to the browser. Similarly, the widget specification covers the client-side APIs and services, and not the HTML and Javascript generated server-side.

Where Can I Learn More about Differences between Widgets and Portlets?

There is an ongoing discussion on the web about these differences. You may check some of the following blog entries:

- *Rob Will, Making Sense of Portlets and Widgets,*
  https://www.ibm.com/developerworks/mydeveloperworks/blogs/WebSpherePortal/entry/making_sense_of_portlets_and_widgets1?lang=en
  Posted on August 11, 2009
- *David Megginson, Widgets vs. Portlets,*
  http://quoderat.megginson.com/2008/07/14/widgets-vs-portlets/
  Posted on July 14, 2008
- *Michael Porter, Widgets vs Portlets, Portals - Perspectives on Collaboration and the Connected Enterprise,*
  http://blogs.perficient.com/portals/2010/06/16/widgets-vs-portlets/
  June 16th, 2010
- *Tycho Luyben, Hot or Not: - Widgets in the Java portlet world,*
  http://www.theserverside.com/news/thread.tss?thread_id=46071
  July 27, 2007
- *Technologie Roadmap: Portlet JSR286 vs Widget/Gadget,*
  http://programmers.stackexchange.com/questions/52722/technologie-roadmap-portlet-jsr286-vs-widget-gadget
- *Gadgets and Widgets as an alternative to Portlets,*
  http://apoorv.info/2008/12/05/gadgets-and-widgets-as-an-alternative-to-portlets/
  December 2008
- *What is the difference between a portlet and a servlet?,*
  http://stackoverflow.com/questions/1480528/what-is-the-difference-between-a-portlet-and-a-servlet
  StackOverflow, 2009
- *Jason Wyatt, iWidgets: Kinda Like Portlets,*
  http://jasonwyatt.tumblr.com/post/200237967/iwidgets-kinda-like-portlets
  September 29, 2009

BACKBASE

The Backbase Approach

The previous section discussed strengths, weaknesses, and use cases of portlets and web widgets. This section focuses on Backbase Portal, its widgets, and how they compare to web widgets and portlets.

Widgets, in particular W3C widgets, are applications that can run independently or within web pages. This flexibility often goes unnoticed, but it comes at the cost of complex standards: the W3C widget standard has no less than 54 requirements. Enabling widgets to work together with a coherent look-and-feel may be challenging: modifying the web services from which the widgets receive data requires modifying the widgets as well.

Portlets can adapt complex content more easily because they run server-side. However, their comprehensive APIs add layers of complexity as well, as do the standards that regulate instance life-cycle management, inter-portlet communication, etc. In order to reduce complexity, portlet implementations are seldom fully standard-compliant, and portlets from different vendors are rarely compatible. Finally, portlets do not standardize rich client and rich user experience.

The Backbase Portal approach combines the best ideas from web widgets and portlets. It offers server-side integration and adaptation, reusing some ideas of portlets, but comes with both client and server APIs, doing away with much of the complexity of web widgets and portlets. The Backbase Portal widget model comes with the following benefits over portlets and web widgets:

- More options. Backbase Portal can aggregate applications in the browser (like ordinary web widgets), on the server (like portlets), or combine both approaches. Backbase Portal offers a number of options ranging from pure server-side rendering, where HTML code is generated server-side and sent to the browser, to pure client-side rendering, where the browser downloads individual widgets and constructs the page.
- Leanness. Backbase Portal is a compact product. Backbase applies lean principles to eliminate waste and only include the functionality that customers really need; it stands out in the industry as a lean portal solution based on widgets and Web Oriented Architecture (WOA).

- Easy integration. Backbase Portal relies on REST-oriented approaches for simple and future-proof integration with existing systems. REST orientation uses open standards to increase interoperability with other products.
- Better alignment of business and IT. Backbase Portal puts both business users and developers in control of application development. Developers have full control over the development of services and widgets, and business people can use Backbase Portal Manager to add widgets, configure pages, and quickly add value and business insights to the portal site. Compare this to normal portlet and widget engines, where usually only developers and administrators can make changes.

The rest of this section compares the Backbase widget model to web widgets and portlets along the following dimensions:

- Rendering and application composition
- Interoperability
- Server requirements
- Back-end service adaptation and augmentation
- Multi-device support
- Performance
- Inter-widget/portlet communication and common authentication/authorization
- User experience

Rendering and Application Composition

Portlets typically assemble HTML fragments into single pages, add resources like CSS stylesheets and cookies, and deliver the page to the client browser. Some client-side processing is possible, but portlet standards are mainly concerned with server-side APIs.

Web widgets are, in essence, web pages embedded in larger web pages, generally within iFrames. They receive content via separate HTTP connections and have their own resources. The browser composes the final pages.

Backbase Portal uses a more elaborate GUI component model than web widgets and portlets: portals are organized into pages, pages include one or more - possibly nested - containers, and containers include widgets. Depending on the particular settings and implementation, developers can decide whether the final composition takes place on the portal server, in the browser, or both. Widgets can share resources across the whole page. Finally, Backbase widgets do not use iFrames.

Interoperability

Portlets loosely follow a range of protocols and standards like JSR-168, JSR-286, and WSRP. In practice, however, different implementations of portlet standards are rarely compatible.

Web widgets normally use REST-oriented standards, and exchange messages in XML, JSON, HTML, or other simple text formats. Integration with custom or more complex protocols like SOAP is usually difficult and requires some server-side adaptations.

Backbase Portal mainly uses open REST-oriented standards like HTTP, XML, or JSON. REST-oriented solutions have several advantages: they are simple, standard-based, easy to adopt, and can use HTTP cache and proxy servers to handle high loads. For more complex needs, the Backbase Mashup Services module uses custom data providers to make custom protocols and data structures available to Backbase widgets.

Server Requirements

Portlets require installation and maintenance of specialized and often expensive portlet containers.

Web widgets do not usually require special servers.

Backbase Portal requires Portal Server, but its basic footprint is small, and many of its components are optional. Backbase widgets are simpler and more powerful than pure widget solutions. The Backbase Mashup Services module offers proxying, content transformation, and caching services to overcome the same-origin policy; it provides widgets with content in appropriate formats (like JSON), and improves performance.

Back-End Service Adaptation and Augmentation

Portlets can add server logic to augment and adapt back-end services, but this requires adapting the data to the Portlet APIs. Web widgets usually offer data that requires little processing, like RSS feeds, and do not require complex logic or complex data operations.

Backbase Portal provides a number of options depending on how much additional server-side logic is required. In the simplest case, Backbase widgets can access remote servers directly or through a simple proxy. Alternatively, Backbase Portal Mashup Services enables a range of back-end service augmentations, adaptations, and transformations. This makes widgets lighter than in the case of pure client-side data processing.

Multi-Device Support

Rendering portlets on multiple devices requires server-side processing of user agent information. Web widgets, instead, generally need to access the browser API to support multi-device rendering. Backbase Portal offers a flexible, high-level, declarative way to specify multi-device rendering. Each Backbase Portal item (page, container, or widget) can have different templates for different devices. The portal server selects the right template at runtime depending on the user device, so that Backbase widgets do not have to include any special logic for multi-device rendering.

Performance

Portlets employ a server-side component model. While they can use AJAX, the rendering and processing normally happen server-side.

Web widgets employ a client-side component model. They normally consume more client resources, because they offload the processing from the server to the client. They also open more HTTP connections because they use AJAX to communicate with the server.

Backbase Portal is not biased towards any of these options, but enables the flexible configuration of client/server processing balance: when rendering client-side, it uses AJAX extensively; when rendering server-side, it makes no AJAX calls at all. It can also combine both solutions for improved flexibility.

Inter-Widget/Portlet Communication - Common Authentication/Authorization

Portlets communicate with each other server-side. They use common authentication and authorization so that users do not have to sign on to each portlet separately.

Web widgets can use the HTML5 postMessage APIs or libraries like the jQuery postMessage plugin to communicate within a page. Common authentication/authorization is usually more problematic.

Backbase Portal supports inter-widget communication through a flexible, loosely coupled, publish-subscribe mechanism shipped with its client library. Backbase widgets can still use other libraries and standard inter-widget communication mechanisms, and they can take advantage of common authentication/authorization.

Responsiveness, User Experience, and Accessibility

Portlets normally render a page server-side and deliver it to the client browser. This reduces responsiveness because each request requires reloading the page. Many portlet products can render portlets and pages using AJAX for improved responsiveness, but such options are limited and not standardized.

Web widgets feel more responsive than portlets because they are built through independent browser calls to server-side services.

Backbase supports modern user experience (UX) and interaction patterns. It enables creating highly responsive interfaces, but supports pure server-side rendering as well. It enables developers to select the best approach depending on the bandwidth, server load, and desired patterns of user interaction. Backbase Portal also supports progressive enhancement - a strategy for web design that emphasizes accessibility, semantic HTML markup, and external stylesheet and scripting technologies. Progressive enhancement uses web technologies in a layered fashion to give everyone access to the basic content and functionality of a web page, while also providing an enhanced version of the page to those with more advanced browsers or greater bandwidth.

Learning Curve

The learning curve for widgets is usually lower than for portlets, because widgets have fewer APIs and use common XML and Javascript technologies, while portlets have a richer, more comprehensive API, instance life-cycle management, inter-portlet communication, etc. In general, development time for widgets tends to be shorter as well.

The learning curve for Backbase Portal is between the learning curve for portlets and that for web widgets. Backbase Portal relies on open standards familiar to many developers, but requires learning a few Backbase-specific APIs and formats, such as the Backbase Portal REST API.

About Backbase

Backbase delivers portal software that provides a new user experience layer on top of underlying infrastructure and IT systems. It gives companies the opportunity to create interactions that link customers to relevant information and applications that fit their needs and preferences. With its modern, widget-based architecture Backbase Portal provides the flexibility and speed to create modern portals that truly empower the customer.

Unlike most traditional IT portal vendors, Backbase has created a contemporary, business-driven portal solution that makes portal management easy for e-business professionals. This means faster time to market and more flexibility to optimize online channels with less IT support.

The unique Backbase approach enables organizations to drive self-service, fuel online revenues, and turn their online banking channel into a true Customer Experience Platform. Global companies such as ABN Amro, AIG, Al Rajhi Bank, Costco, GE, Barclays, ING, KPN, Motorola, ViaWest and Visa have improved their online customer interactions and maximized online customer experience, retention, and conversion, by leveraging Backbase Portal.

Backbase was founded in 2003 and is privately funded with operations in New York, Amsterdam, Moscow, and Singapore.

North American HQ	European HQ	Moscow	Singapore
350 Broadway, Suite 1107	Jacob Bontiusplaats 9	Lesnaya Plaza Business Center	3 Church Street
New York, NY 10013, United States	1018 LL Amsterdam	125047 Moscow, Russia	Level 25 Samsung Hub
Toll-Free Number: +1 866 800 8996	The Netherlands	4th Lesnoy Pereulok, 4	Singapore 049483
Office Number: +1 646-478-7538	Phone: + 31 20 465 8888	Phone: +7 495 641 37 70	Phone: (65) 6692-9110
sales-us@backbase.com	sales-eu@backbase.com	moscow@backbase.com	singapore@backbase.com

Discover Backbase

Contact us via

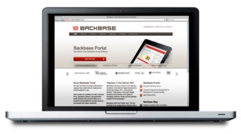
discover@backbase.com

*Join our **free** webinars*

backbase.com/webinars

Interesting white papers at:

backbase.com/bank20



Visit our website at:

www.backbase.com