# Portals

Magnolia CMS Tech Brief

Magnolia International Ltd.

January 5, 2012

## Table of contents

## Introduction

Can a content management system be a portal? In this paper we pit the OpenSocial/mashup approach used by Magnolia CMS against traditional Java portal standards. We come to the conclusion that content management is the most intensive continuous activity on any website. A portal should be built with software that best supports that key activity.

## Can Magnolia CMS be used as a portal?

Yes, Magnolia CMS can be used as a portal. In fact, any good CMS is a possible option because portals need to mix apps and editorial content.

The primary purpose of portal software is to integrate apps. Typically this is a once-off task. Once the background business applications are converted into a format the portal can understand and assembled into a page, they tend to stay that way.

Content management, however, is a continuous task. New content needs to be written and published on a daily basis. Writing and publishing content is the most important job in the life cycle of a site. It creates substance and drives users to the site. Over time, content management becomes the most intensive activity on any website. A portal is no exception to this rule.

The challenge is to find a system that accomplishes both. A content management system is good at handling editorial content. Over time, the need to handle editorial content will tip the scales towards choosing a CMS.

At Magnolia we understand the need to integrate applications. Not everything is editorial content. Back-end systems provide dashboard components, statuses, charts and graphs that may be critical to users. Magnolia's portal approach relies on common OpenSocial application programming interfaces and modern,Web 2.0-style mashups. These technologies allow you to do everything you would expect from a portal:

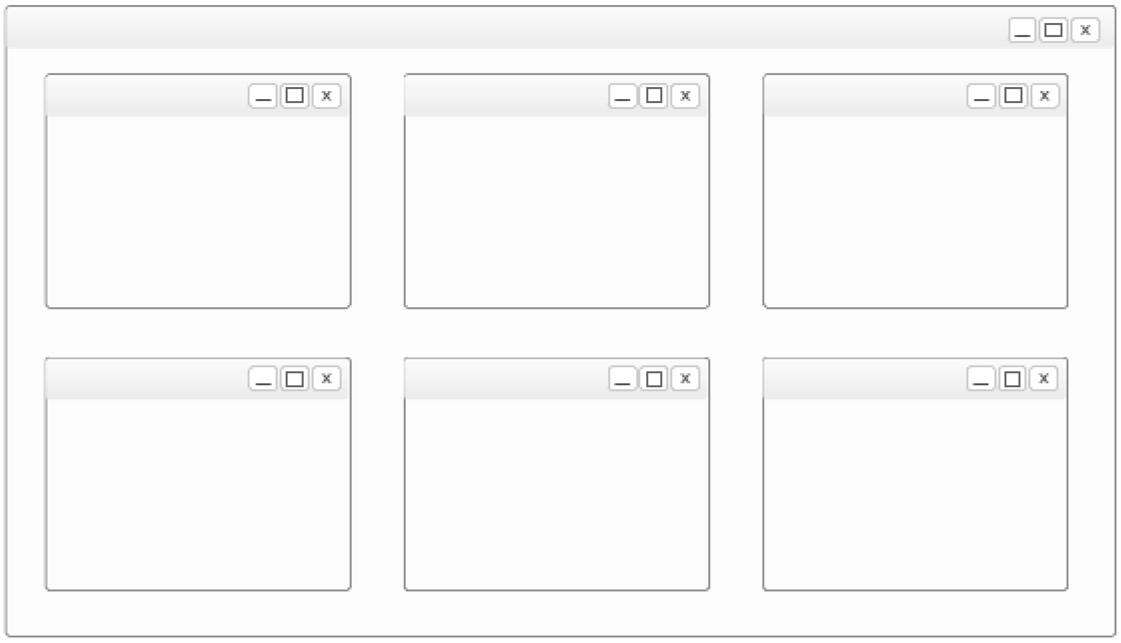
- Harvest content from various sources and mash it all up onto a single page.
- Embed applications and gadgets.
- Support interaction between applications.

## What is a portal?

There is no single definition of a portal. Most people understand the term to mean a platform that collects Web content and applications from multiple sources. It is a kind of container that you put many things into so that they are all conveniently available to users in a single place, in an integrated fashion.

It is important to understand that a portal is not a single piece of software. Rather, it is an entry point to multiple business processes and a collection of business information from many sources.

A typical portal looks like a dashboard. Once you've seen a few, they are quite easy to identify. A portal page typically consist of boxes that display information from various sources: statuses, graphs, charts, worksheets, calendars and so on. This content is often dynamic and interactive. It can include applications that provide functionality, such as forms, selectors and file upload controls.



## OpenSocial - Moving the social Web forward

At Magnolia we use OpenSocial to expose data and services in a portal. The OpenSocial specification is simple and elegant, and incorporates most of the capabilities found in the WSRP and JSR-168/286 specifications. Unlike the portlet specifications, particularly WSRP, OpenSocial enjoys wide adoption because it supports hundreds of thousands of gadgets that millions of users are already familiar with.

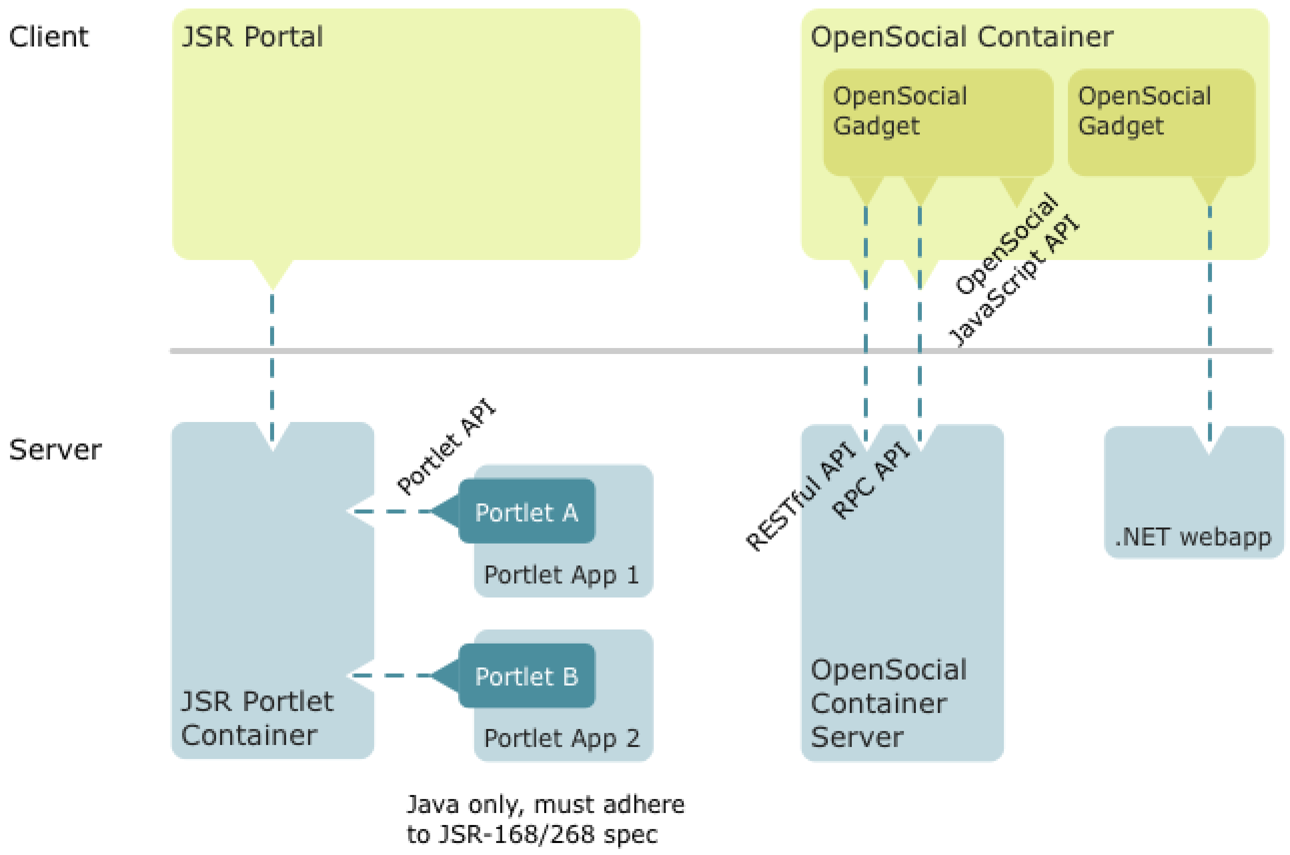
Google was the first to realize the potential of embedding content from other websites using simple XML and JavaScript. Together with MySpace and a number of other social networks, Google developed a common API for web applications. The OpenSocial specification for content syndication and social enablement evolved from this API.[1]

[1] **http://blogs.alfresco.com/wp/luissala/2008/12/09/goodbye-jsr-168-and-wsrp-portlets-hello-opensocial-gadgets/**

At Magnolia, we believe that this approach has all the benefits[2] that earlier standards promised, but none of the heavy-handedness:

- **Componentization**. Gadgets allow you to turn applications into portable components. OpenSocial provides a standard container to reassemble gadgets into views. As OpenSocial becomes more prevalent in the enterprise, you can mix more things together.
- **Easy** to develop, fast to learn. OpenSocial uses common Web technologies: HTTP, XML, AJAX and REST. OpenSocial Gadgets are well documented, and there are plenty of resources on the web to help developers get started.
- **Broad developer ecosystem**
- **Data interoperability**. Most enterprise apps may not seem “social” but nearly every application creates a network of relationships between people and things that may be useful.
- **Integration at client side** rather than server side.

The image below shows the benefits of client-side integration.



In the traditional JSR portal model, a portlet container hosts any number of portlets. Each portlet generates fragments of markup, which the portal ultimately pieces together to create a complete page that is presented to the user.

An OpenSocial application runs inside an OpenSocial container. The container provides an OpenSocial Javascript API as well as pass-through support (via standard JavaScript libraries) to the RESTful protocol API and RPC protocol API. These APIs are used by the container to communicate with the OpenSocial container server.

[2] http://techcrunch.com/2010/03/24/five-reasons-opensocial-will-change-the-enterprise/

The two approaches differ in where rendering happens. In the JSR portal model, markup fragments are rendered on the server side and passed to the client (and ultimately to the browser), whereas in the OpenSocial model rendering is mainly done on the client side.

The approaches also differ from a flexibility point of view. A JSR portlet must be coded in Java and must adhere to either the JSR-168 or JSR-286 specification. An OpenSocial gadget can be hosted on any server and architecture. It could be a .NET webapp running in a remote server, for example.

## Enterprise OpenSocial

Hold on. We keep talking about social. Isn't that for social networks? You have a serious business to run!

Initially,OpenSocial emerged from the demands of consumer-facing social networking sites, like MySpace, LinkedIn and Ning. It was envisioned as the glue that would bind social applications together. A number of developers started working on the specification thanks to its popular application. The specification made accessing and sharing user profile, relationship and activity data much easier.

Today, OpenSocial can also be used as a general Web application integration technology that provides open standards for browser-based components known as gadgets. For non-browser/UI data transfer, OpenSocial also includes a REST based server-to-server protocol.[3]

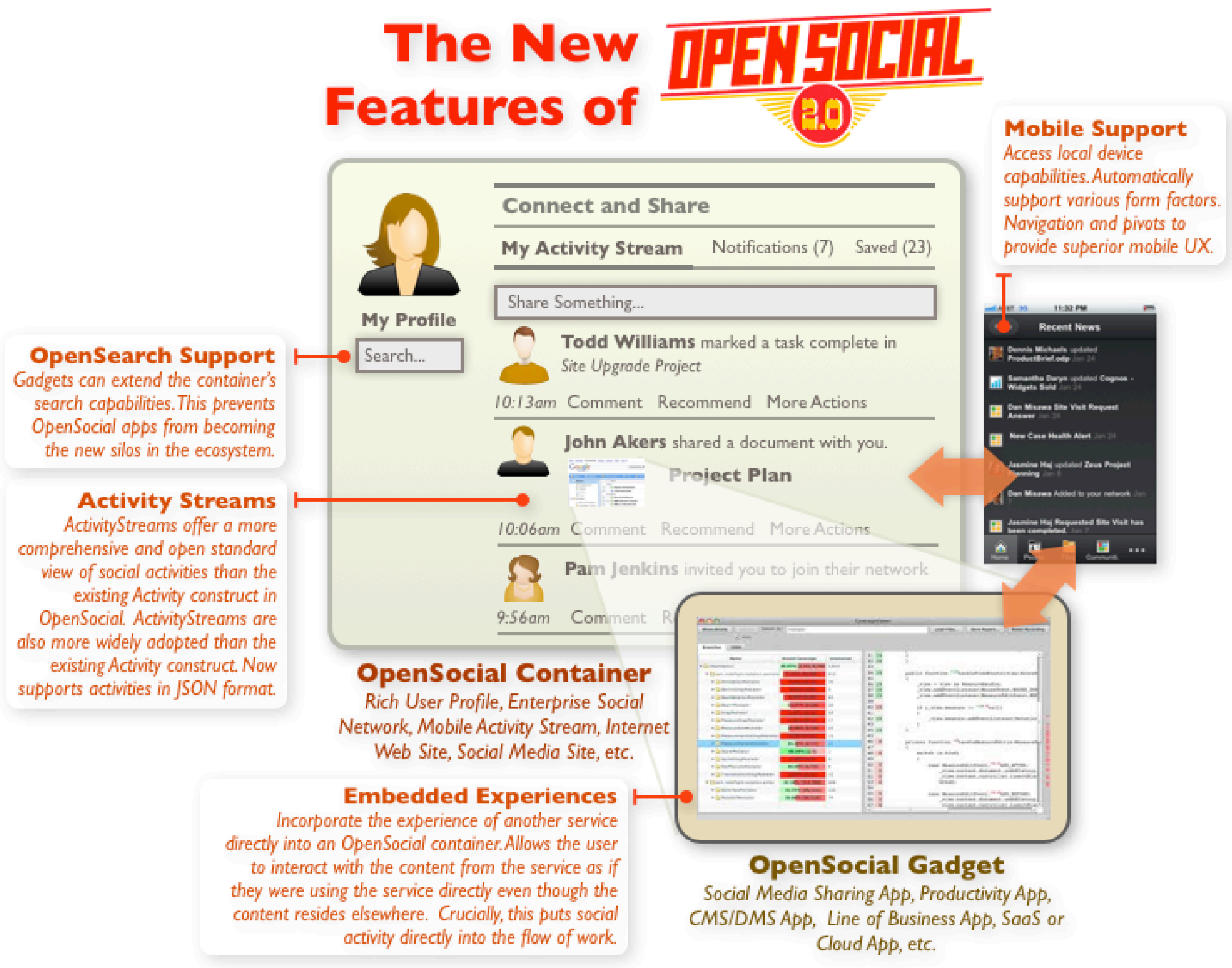
OpenSocial 2.0 is very attractive for enterprise use. It has been embraced by Alfresco, Atlassian, Cisco, eXo, IBM, Jive, Lockheed Martin, SAP, SocialText and others who have built containers for it.[4] Version 2.0 of the specification focused particularly on business use and introduced a number of new features:

- Activity Streams support: a mechanism for defining rich and detailed social activities.
- Simplified gadget format.
- Embedded Experiences: running a service in a gadget.
- OAuth 2 support: OAuth is a common security mechanism that allows you to grant a site access to your private resources. OAuth is about giving access to your stuff without sharing your identity at all (or its secret parts). Even Facebook uses it.[5]
- Common Container: a new specification for the container that enables better interaction with gadgets.

[3] **http://blog.opensocial.org/2009/12/enterprise-opensocial-white-paper-now.html**

[4] **http://www.infoq.com/news/2011/08/OpenSocial-2**

[5] **http://developers.facebook.com/docs/authentication/**


The New Features of OPENSOCIAL 2.0
Mobile Support
Access local device capabilities. Automatically support various form factors. Navigation and pivots to provide superior mobile UX.
OpenSearch Support
Gadgets can extend the container's search capabilities. This prevents OpenSocial apps from becoming the new silos in the ecosystem.
Activity Streams
ActivityStreams offer a more comprehensive and open standard view of social activities than the existing Activity construct in OpenSocial. ActivityStreams are also more widely adopted than the existing Activity construct. Now supports activities in JSON format.
Connect and Share
My Profile
Search...
My Activity Stream
Notifications (7)
Saved (23)
Share Something...
Todd Williams marked a task complete in Site Upgrade Project
10:13am Comment Recommend More Actions
John Akers shared a document with you.
Project Plan
10:06am Comment Recommend More Actions
Pam Jenkins invited you to join their network
9:56am Comment
Recent News
OpenSocial Container
Rich User Profile, Enterprise Social Network, Mobile Activity Stream, Internet Web Site, Social Media Site, etc.
Embedded Experiences
Incorporate the experience of another service directly into an OpenSocial container. Allows the user to interact with the content from the service as if they were using the service directly even though the content resides elsewhere. Crucially, this puts social activity directly into the flow of work.
OpenSocial Gadget
Social Media Sharing App, Productivity App, CMS/DMS App, Line of Business App, SaaS or Cloud App, etc.

From http://blogs.zdnet.com/Hinchcliffe

## Mashups aggregate and add value

Magnolia CMS uses the mashup approach to aggregate portal content. Again, rather than adhering to an aging JSR-168/286 portal standard, we take the more modern path. Mashup is a Web page or application that uses and combines data, presentation and functionality from two or more sources to create new services. Mashups are wildly popular in modern Web apps.

Mashups make it possible to integrate business and data services quickly. You can combine company-internal services with external or personalized information, and make those services tangible to users through user-friendly browser interfaces.[6]

[6] http://www.ibm.com/developerworks/lotus/library/mashups-patterns-pt1/

To understand why mashups are better suited to mixing apps and editorial content, let's compare them to JSR portals.[7]

| | **Mashup** | **JSR Portal** |
|---|---|---|
| **Technology** | Uses newer, loosely defined Web 2.0 techniques | Uses older technology: an extension of the traditional Web server model. |
| **Content aggregation** | Uses APIs provided by different content sites to aggregate and reuse content in various ways. | Approaches aggregation by splitting the Web server role into two phases: markup generation and aggregation of markup fragments |
| **Content dependencies** | Can operate on pure XML content and also on presentation-oriented content (e.g. HTML) | Aggregates presentation-oriented markup fragments (HTML, WML, VoiceXML, etc.) |
| **Location** | Content aggregation can take place either on the server or on the client | Traditionally, content aggregation takes place on the server |
| **Aggregation style** | "Melting Pot" style: Individual content may be combined in any manner, resulting in arbitrarily structured hybrid content | "Salad bar" style: Aggregated content is presented 'side-by-side' without overlap. |
| **Event model** | CRUD operations are based on REST architectural principles, but no formal API is necessary. | Read and update event models are defined through a specific portlet API. |
| **Standards** | Base standards are XML, interchangeable with REST or Web Services. RSS and Atom are commonly used. More specific mashup standards such as EMML are emerging. | Portlet behavior is governed by standards (JSR 168, JSR 286 and WSRP), but portal page layout and portal functionality are undefined and vendor-specific |

## Business mashups

Business mashups differ from consumer mashups in the level of integration with business computing environments, security and access control features and governance. Using business mashups in commercial software-as-a-service (SaaS) offerings is a growing trend.

Here's a great video by IBM about what mashups can do for enterprise users. http://www.youtube.com/watch?feature=player_embedded&v=3kDnbhKb2ow.

VPRO, a Dutch public broadcaster who works with mashups is a Magnolia specific example. Pinkpop is the largest pop festival in the Netherlands. VPRO's work on Pinkpop festival website is a perfect example of how mashups aggregate content and add value. VPRO created the website using Magnolia CMS, CouchDB and Camel. The site harvests pictures and video uploaded by festival goers from the Internet and serves the harvested media back to viewers in a single, browsable repository. The mashup site lets festival attendees

[7] http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)#Mashups_versus_portals

re-live the moments afterwards and those who couldn't make it can experience the event at home. Check out the presentation VPRO gave at the 2010 Magnolia Conference. http://www.magnolia-cms.com/community/magnolia-conference/archive/2010/program/presentation-day/pinkpop.html

## Portal features

Here is a list of typical features implemented by portals and how Magnolia CMS achieves them.

### Single sign-on (SSO)

Single sign-on (SSO) makes the user's life easier. The user signs in once and gains access to all systems without being prompted to log in again at each of them. Sign into the portal and you are authenticated to access all the services it provides. This is a big boon for anyone trying to remember dozens of logins. Facebook Connect is an example of SSO. It is a set of APIs from Facebook that enable Facebook members to log onto third-party websites, applications, mobile devices and gaming systems with their Facebook identity. It is a growing trend to share at least some data about site members so people do not have to enter the same identifying information again and again on different sites.

Magnolia CMS provides CAS (Central Authentication Service) single sign-on protocol as the first option. Its purpose is to permit a user to access multiple applications while providing credentials (such as user ID and password) only once. It eliminates future authentication requests when the user switches applications during a particular session. Unauthenticated users are diverted to the authentication service and returned only after successful authentication. You can also implement SSO with a custom authentication solution.

### Integration

A portal collects many pieces of information into one convenient user interface. Data from multiple distinct systems is exposed as components (widgets, gadgets, portlets, media). Since a portal is built using Internet technology, navigation between components is easy and intuitive.

Magnolia CMS aggregates content using the mashup approach. OpenSocial gadgets and native CMS content are placed onto the portal page in “melting pot" style. Individual content may be combined in any manner, resulting in arbitrarily structured hybrid content. Magnolia's OpenSocial Container allows editors to place an OpenSocial gadget anywhere on the page just like any other page component. The editor only needs to know the gadget's URL. Gadgets don't need to be in a fixed location either; editors can move them around like any other page component.

Besides applications and services that are OpenSocial gadgets, Magnolia CMS can integrate:

- Spring applications using the Blossom module
- Separate, independent content repositories via REST-based Web services
- External RSS feeds using the RSS Aggregator module and render them on any page
- Resources on the local file system using the WebDAV module

### Federation

In a federated environment, all content sources and repositories provide content using the same standard. This is a collective agreement that makes mashing it all up a whole lot easier. The sources where the data is stored and created may have completely different internal structures, but the data is at least delivered on common ground.

Magnolia CMS supports Content Management Interoperability Services (CMIS). It is a specification that helps content management systems access each other's content. We implement it using Apache Chemistry which makes any JCR implementation CMIS compliant automatically .

## Customization and personalization

Customization puts the look and feel, and much of the portal content into the hands of the user. Users can choose which components they want to see in the portal and move them around. They can prioritize content and services to suit their preferences. Customization is based on *explicit data* that users provide knowingly, for example by rating content favorably or setting their preferences to display certain components. iGoogle is an example of a highly customizable Web portal based on OpenSocial. You can add and remove gadgets as you please, and customize much of their content. Want a clock that displays the local time where your friends live around the globe? Here you go.

While the terms customization and personalization are often used interchangeably, strictly speaking personalization is more about matching content with the user without their direct involvement. "Hands free", you could say. Personalization matches services and content to the user based on the user's profile and past behavior. It involves mining and analyzing *implicit* data about you. In intranet and enterprise portals, personalization is often based on user attributes such as department, functional area or role.

This video of personalizing Google News is a great example.
http://news.google.com/support/bin/answer.py?answer=1146405&hl=en

## Access control

Access control applies the appropriate permissions. In an extranet portal, only employees are permitted to access company's proprietary information. External users such as clients have access only to information specifically provided for them. Access rights may be provided by a portal administrator or by a provisioning process. Access control lists manage the mapping between portal content and services over the portal user base.

In Magnolia CMS, security is controlled with a built-in access management system. The system is based on the Java Authentication and Authorization Service (JAAS). You can set permissions for all types of users, whether real people or processes, and control access to resources such as Web pages, documents, data, forums and templates. Permissions are controlled through a combination of users, groups, roles and ACLs.

## Search

Portal search is typically closer to enterprise search than the open Web search everyone is familiar with. Content from multiple enterprise-type sources such as content management systems, shared folders, wikis and email is made searchable. Much of this content is not openly available on the Web. The search engine uses connectors and adapters to tap into the proprietary sources. Portal search is federated in the sense that a search query is broadcast to a group of disparate databases and content sources and the results are presented in a unified format.

Magnolia CMS uses the default search that ships with the Jackrabbit repository to cover the basics. Content from workspaces such as website and DMS, including content found inside documents, is indexed by the Lucene indexer and made available to search. You can also index external repositories and provide interesting features such as related content. Another alternative is to replace the default search with Apache

Solr[8] to get faceted search, dynamic clustering, database integration and geospatial search.

### Summary

Magnolia CMS is a good choice for building a portal because you get:

- Pixel-perfect control over layout.
- A light-weight framework.
- An easy to implement, modern, flexible and cost-effective system.

All features commonly associated with portals are available in Magnolia CMS.

## JSR-168/286 portal standards and their constraints

Portals developed in Java have traditionally been defined by two main standards: JSR-168 followed by JSR-286. The first is a Java specification covering the basic portlet programming model while the second improves on some initial shortcomings. The third standard, Web Services for Remote Portlets (WSRP), is a network protocol standard designed for communications with remote portlets. It is not Java-specific. The JSR standards are commonly used to define a portlet and WSRP to define a portlet’s operations and relations with remote containers.

A JSR standards-based portal is a rather outdated form of content aggregation technology. It was designed as an extension of traditional, dynamic Web applications, in which the process of converting content data into marked-up Web pages is split into two phases: generation of markup fragments and aggregation of the fragments into pages. Each markup fragment is generated by a portlet, and the portal combines them into a single Web page. Portlets may be hosted locally on the portal server or remotely on a separate server.

Portal technology defines a complete event model covering reads and updates. A request for an aggregate page on a portal is translated into individual read operations on all the portlets that form the page. If a submit button is clicked on any portlet on a portal page, it is translated into an update operation on that portlet alone (processAction on a local portlet or performBlockingInteraction on a remote, WSRP portlet). The update is then immediately followed by a read on all portlets on the page.

If all that sounds rather heavy, it is. Portal technology is about server-side, presentation-tier aggregation. It cannot be used to drive more robust forms of application integration such as a two-phase commit.[9]

## CMS integration into existing portal context

But what if you already have an existing portal? It was expensive to build but it serve users reasonably well by aggregating information from dozens of back-end systems. The thought of starting from scratch with Magnolia CMS may seem unthinkable, involving considerable time and money, and throwing away all your previous work and investment.

We don’t discredit portals as such but we do have our own take on how they should be built: with OpenSocial and mashups. Portals exist for good reason. They provide a single access point to web content and applications. They provide a personalized experience to each user, helping users deal with information

[8] **http://wiki.magnolia-cms.com/display/WIKI/Magnolia+Apache+Solr+integration**

[9] **http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)**

overload.

The available options, from simple to advanced, for integrating apps and content are:

1. **iFrame portlet**: All portlets host content from a different website or local servlet inside a portlet window. An iFrame portlet achieves this by embedding an HTML iframe within a portlet. The frame consumes HTML views directly from the external website or servlet.
2. **Reverse proxy iFrame portlet**: Behaves in the same way as an iFrame portlet except that it uses reverse proxy service URLs instead of the original source URLs. By using a reverse proxy service component, more sophisticated content can be served by setting content rewriting configurations or enabling cross-domain scripting.
3. **XML + XSLT transformation**: Use a Web service to produce XML output from the source and transform it on the portal server side to renderable HTML, for example using XSLT. Useful for aggregating XML based content.
4. **JSR-168/286**: Convert the source to traditional JSR-168/286 portlet format which can interact with the CMS. Avoid this approach. It is heavy and rigid.
5. **OpenSocial + mashups**: Modern and lightweight approach favored by Magnolia. This is the most advanced, flexible and future-proof approach.

## Replacing the CMS

In existing portal installations the integrated apps are rarely the problem. They may not be pretty, but portal apps work fine and serve users the information they need. The problem lies in content management.

Users tasked with editing portal content are not happy if the content management tool falls short of what is needed to get the job done well. An add-on CMS tool can never be as versatile and workable as a dedicated CMS product. So the question is, can you replace just the content management tool?

Magnolia CMS can replace a portal’s built-in CMS tool because Magnolia has the potential to publish content as OpenSocial gadgets. For example, IBM WebSphere Portal supports the assembly of gadgets in the Mashup Center[10]:

> *IBM Mashup Center helps users create web applications from existing information sources by dragging and dropping widgets onto the page, and then wiring them together on-the-glass. Users can extend their mashup environment by incorporating custom developed widgets provided by IT, widgets available on the external IBM Mashup Catalog, or widgets from across the Web, including any of the thousands of Google Gadgets.*[11]

This means that IBM WebSphere Portal can display Magnolia CMS content. Both parties support the same OpenSocial specification.

According to Gartner, by 2015 gadgets will surpass portlets as the most prevalent portal component model and at least 25% of new enterprise portal projects in Global 2000 firms will use open-source horizontal portal frameworks.[12]

Portal vendors have realized this and started to support the consumer-Web widget standards to leverage widely available components. Portal vendors that have declared support for OpenSocial integration are:

[10] http://www.ibm.com/developerworks/lotus/library/mashups-opensocial/index.html
[11] http://en.wikipedia.org/wiki/IBM_Mashup_Center
[12] http://www.gartner.com/technology/reprints.do?id=1-17RTIFE&ct=111025&st=sg

- IBM: WebSphere Portal
- Liferay via Apache Shindig[13]
- JBoss: GateIn Portal[14]

If you don't want to replace your current portal, it may still be worth replacing your content management tool:

1. Render Magnolia CMS content in a portal inside an iFrame or request it with an AJAX call. Magnolia page components are autorenderable. This means you can copy the path of a component, paste it in the address bar of your browser, and see the rendered component. Try this URL from our demo site, it will render the abstract of the Articles section page.
2. Pull or push Magnolia content into the portal using OpenSocial's RESTful protocol.
3. Pull OpenSocial apps into the portal. Provide access to OpenSocial apps as they would appear in iGoogle or any other OpenSocial container, but from within the portal UI. App functionality should not be hindered by being rendered within portal environment. Leverage existing ways to publish gadgets and OpenSocial apps via generated JavaScript snippets.

   (Adapted from Integrating OpenSocial and Salesforce[15])

[13] http://www.liferay.com/web/jorge.ferrer/blog/-/blogs/introducing-opensocial-and-how-liferay-can-benefit-from-it

[14] http://www.jboss.org/gatein/

[15] http://www.slideshare.net/cschalk/integrating-opensocial-sales-force-presentation