Proceedings of the UK e-Science All Hands Meeting 2006, © NeSC 2006, ISBN 0-9553988-0-0

Application Reuse through Portal Frameworks

Mark Baker and **Rahim Lakhoo**

ACET
University of Reading

Distributed Systems Group
University of Portsmouth

{Mark.Baker@computer.org, Rahim.Lakhoo@port.ac.uk}

Abstract

Educational institutions, enterprises and industry are increasingly adopting portal frameworks, as gateways and the means to interact with their customers. The dynamic components that make up a portal, known as portlets, either use proprietary interfaces to interact with container that controls them, or more commonly today, the standardised JSR-168 interface. Currently, however, portal frameworks impose a number of constraints and limitations on the integration of legacy applications. This typically means that significant effort is needed to embed an application into a portal, and results in the need to fork the source tree of the application for the purposes of integration. This paper reports on an investigation that is studying the means to utilise existing Web applications through portlets via bridging technologies. We discuss the capabilities of these technologies and detail our work with the PHP-JavaBridge and PortletBridge-portlet, while demonstrating how they can be used with a number of standard PHP applications.

1. Introduction

Portal and portlet technologies are becoming increasingly popular with Web developers and orgnisations alike. Portals aim to provide a suite of applications in a common and customizable environment. Portal containers provide a number of portlet applications, which users can subscribe to or select. A portal would typically allow a user to organise a collection of portlets to their liking. Portlets themselves can be thought as mini Web applications, with GUI capabilities, such as being able to resize its window.

Portlets were once developed using vendor specific APIs and container, such as IBM's WebSphere. This resulted in portlets, which were confined to a single container. The JSR-168 [1] specification has had an impact on portal developers and their communities. Developers following the JSR-168 are now able to produce a single portlet application that is deployable across different portlet containers. However, developers are currently forced to either redesign or re-implement their existing applications with Java, which may be a deterrent for organisations to employ portlet technologies, especially if a viable Web enabled solution is already available.

Typically, scripting languages are used for Web development, which is a popular and well-established method of generating an application. Although portlets can use JavaServer Pages (JSP) to provide a GUI; this is not as widely used as other languages, such as PHP [2], Perl, Python and more recently Ruby. The number of tools and applications available to scripting languages is large compared to solutions offered based on JSP. Script based languages offer rapid development with low learning curves. Such an example is Ruby on Rails [3], which is a Web framework optimised for developer productivity.

More recently, developments have started to move towards bridging the gap between script based languages and Java. JSR-223 [4] aims to allow scripting language pages to be used in Java Server-side applications. JSR-223 is using PHP as their example scripting language. JSR-223 is not limited Web development or PHP. Groovy, JavaScript and command line interaction is currently also available. Currently JSR-223 is being distributed as part of Sun's JDK 1.6 "Mustang" beta. Other developments include the PHP-JavaBridge [5], which incorporates JSR-223, but is described as an XML-based network protocol, which can be used to connect to native script engines. PHP-JavaBridge features the ability for PHP applications to be executed within a J2EE container, such as Apache Tomcat. The PHP-JavaBridge will be discussed further in Section 2.2.

While JSR-223 and the PHP-JavaBridge project provide a means for Java to access script based languages, portal developers are also producing bridges. Apache Portals is developing Portals

Bridges [6], which will offer JSR-168 complaint portlet development using Web frameworks, such as Struts, JSF, PHP, Perl and Velocity. Bridges are being developed for each of the supported Web frameworks. In Portals Struts Bridge [8] support has been added to a number of portals, including JetSpeed, JBoss, GridSphere [9], Vignette Application Portal and Apache Cocoon. Portlet developers are also producing portlets, which are intended to allow regular Web sites to be hosted as JSR-168 compliant portlets. The PortletBridge-portlet [10] aims to proxy and rewrite content from a downstream Web site into a portlet.

The question is though, with these bridging technologies becoming more usable, can developers use them efficiently and avoid re-implementing existing applications within compliant portlets?

This paper will discuss the various implementations and outline our own experiences with some of these bridging technologies. Section 2 will give an insight to some of the bridging technologies introduced. Section 3 discusses our configuration and set-up of the bridging technologies. Section 4 presents some results of our implementation. Section 5 discusses some PHP Wiki applications tested with the PHP-JavaBridge. While Section 6, provides an insight to our future work involving a Single Sign-On (SSO), which interfaces to existing web applications from a portlet. Finally, Section 7 concludes the paper.

2. Bridging Technologies

In this section we discuss and outline the reference implementation of JSR-223, then we move on to detail both PHP-JavaBridge and the PortletBridge-portlet in terms of architecture, functionality and features.

2.1 JSR-223

The JSR-223 specification is designed to allow scripting language pages to be used in Java server-side applications and vice versa. It basically allows scripting languages to access Java classes. JSR-223 is currently a public draft specification and may therefore change. The material presented in this section is based on the draft specification. In order to assess JSR-223's reference implementation we examine it in relationship with PHP as the scripting language and Apache Tomcat as the servlet container.

JSR-223 introduces a new scripting API `javax.script` that consists of interfaces and classes, which define the scripting engines and a framework for their use in Java. JSR-223 also provides a number of scripting engines for different script based languages. Currently, under the reference implementation version 1.0 [11], there are servlet engines for Groovy, Rhino and PHP. The `GroovyScriptEngine`, `RhinoScriptEngine` and `PHPScritpEngine` classes in the API represent the servlet engines, respectively.

PHP uses a Server API (SAPI) module, which defines interactions between a Web server and PHP. PHP 4 includes Java capabilities provided either by a servlet SAPI extension module or by integrating Java into PHP. The Java SAPI module enables PHP to be run as a servlet while using Java Native Interface (JNI) to communicate with Java. However, unlike PHP 4, PHP 5 does not include the Java integration support in its current release, although it is available under the development branch. The version of PHP 5 included with the reference implementation of JSR-223 does not use the more traditional CGI/FastCGI as its server API. Instead, it makes use of the new PHP Java scripting engine.

JSR-223's reference implementation is based on PHP 5.0.1. The current version of PHP is 5.1.2. Even though there is a mismatch, versions of PHP can be compiled for inclusion into JSR-223. Very few PHP extensions are provided by JSR-223. Any additional PHP extensions desired can be compiled against the version of PHP packaged with JSR-223.

With respect to the installation of the JSR-223, Apache Tomcat is configured by the installation script to process `.php` files. When executing PHP applications with JSR-223, the PHP servlet engine calls the PHP interpreter and passes on the script. Although JSR-223 contains a PHP executable binary, it is not used. Instead, the JSR-223 servlet uses JNI for native library files, namely `php.so` or `php.dll` under Windows. The advantage to this method is that the performance of PHP under JSR-223 is similar to that of a native Apache Web server with `mod_php`.

JSR-223 has examples of session sharing between JSP and PHP, amongst others. Thus JSP and PHP applications may exchange or share information via sessions. Both types of session may be attached to a single PHP

application, through the use of the `use_trans_sid` option, resulting in the Web server's response containing both session IDs.

The PHP servlet engine provides additional objects as predefined variables to aid interoperability. JSR-223 requires that objects, such as an HTTP request, should be available from scripting languages. From a PHP script, these predefined variables reflect the same variables available from a JSP page, namely:

- `$request`
- `$response`
- `$servlet`
- `$context`

Although there can be communication between PHP and Tomcat sessions, there are limitations on what a session can hold. A Tomcat session may only hold scalar values, i.e. strings and numbers. It may not contain database connections, file handlers or PHP objects, but may contain Java objects.

Other limitations of the reference implementation are most noticeable when migrating or using existing PHP applications. PHP applications usually make extensive use of predefined variables, however `$_SERVER['REQUEST_URI']` or `$_SERVER['QUERY_STRING']` are not defined, instead the request variables can be found in `$_SERVER['argv']`. Another limitation is relative paths within PHP pages, which can be worked around in two ways. One of which is to set the `include_path` in the `php.ini` initialisation file. The other way is to modify PHP applications to use absolute paths.

Although JSR-223 is still a reference implementation under review, it does make substantial progress towards the interoperability of Java and script-based languages. This brings forth a new dimension of Java interoperability, which can be useful for developers and end users alike. Despite JSR-223's limitations, it does provide foundations for other implementations.

2.2 The PHP-JavaBridge

The PHP-JavaBridge is as the name implies a PHP to Java bridge. The PHP-JavaBridge uses JSR-223 concepts while adding important aspects, such as enabling communications with already existing PHP server installations. The PHP-JavaBridge is described as an XML-based network protocol designed to communicate with native scripting engines, which have a Java or ECMA 335 [13] virtual machine.

The PHP-JavaBridge XML protocol is used to connect to PHP server instances. The PHP server instance can be a J2EE container, such as Apache Tomcat or a PHP enabled Web server, such as Apache or IIS. PHP version 5.1.2 is included with the PHP-JavaBridge as a CGI/FastCGI component, with native library files, such as `php-cgi-i386-liunx.so` for Linux. The bridge is not limited to the included CGI environment. The operating system's native installation of a PHP CGI/FastCGI can also be used, or an alternative set of binaries can replace those included. Although PHP is the focus of the PHP-JavaBridge project, it does include other features, including the ability to contain PHP scripts with JSF (since version 3.0 of the PHP-JavaBridge) and RMI/IIOP. Examples of these capabilities are included in the standard distribution.

Unlike the reference implementation of JSR-223, the PHP-JavaBridge does not use JNI. Instead, HTTP instances are allocated by the Web server or from the Java/J2EE backend. The PHP instances communicate using a "continuation passing style", which means that if PHP instances were to crash it would not takedown the Java server or servlet at the backend. Because the PHP-JavaBridge communicates via a protocol, it does have a certain amount of additional functionality. There can be more than one HTTP server routed to a single PHP-JavaBridge or each HTTP server can have its own PHP-JavaBridge. When the PHP-JavaBridge is running in a J2EE environment, it is also possible to share sessions between JSP and PHP. Although the PHP-JavaBridge can use Novell MONO or Microsoft .NET as a backend, this is beyond the focus of this section.

As mentioned, the PHP-JavaBridge is flexible with regards to the type of PHP server or communication method that it uses. The PHP-JavaBridge also has four operating modes, some of which implement practical solutions for production servers. The different operating modes of the bridge have been categorised as:

- Request.
- Dynamic.
- System.
- J2EE.

When the PHP-JavaBridge is operated in Request mode, the bridge is created and

destroyed on every request or response. While the bridge is running in Dynamic mode the bridge starts and stops synchronously with a pre-defined HTTP server. System mode is when the bridge is run as a system service. This mode is installed as an RPM under Red Hat Enterprise Linux. Finally, the J2EE mode is when the bridge is installed in a J2EE server, such as Apache Tomcat. Other modes and configurations are possible, though not discussed, are here [5], including executing a standalone bridge and configurations for Security Enhanced Linux distributions.

When using a J2EE environment, PHP applications can be packaged with a PHP-JavaBridge into a `.war` archive. The archive will contain the configuration of the bridge, the PHP scripts and if needed a CGI environment. This makes Web applications for Java, based on PHP, easy to deploy.

The PHP-JavaBridge does not encounter the same issues with PHP extensions, as the JSR-223. This is due to the configuration flexibility of the PHP environment. When utilising an existing PHP installation, any extension module can be installed in the normal manner, with only little or no configuration required of the PHP-JavaBridge. However, the ease of integrating an existing PHP application into a J2EE environment can be hampered by the application itself. A PHP application, in some cases, is not fully aware of the J2EE environment, such as host names containing the additional port in the URL. Some predefined PHP variables are also not available in certain circumstances, or are not correct under the PHP-JavaBridge environment.

The performance of the PHP-JavaBridge [5] shows that the overall execution time is similar to that of native scripting engines executed from the command line with `jrunscript`. The results also show that there is a performance overhead when using the PHP-JavaBridge communication protocol. However, the overhead introduced is not substantial and considering that communications between different hosts is only 50% slower than communications on a single host, it is fair to ignore the overheads in favour of flexibility.

Developers using either JSR-223 or the PHP-JavaBridge can produce hybrid Java and PHP applications. Session sharing between JSP and PHP is also available with both technologies. However, the PHP-JavaBridge offers greater flexibility and integration when communicating with existing PHP applications or servers.

2.3 The PortletBridge-Portlet

While JSR-223 and the PHP-JavaBridge concentrate on the interoperability of scripting languages and Java. The PortletBridge-portlet focuses on creating a generic method of allowing Web sites to be viewed via a JSR-168 compliant portlet. Typically, portals, such as uPortal provide the means to view Web sites via Iframes. Yet, the JSR-168 specification does not recommend their use, as there is no control over the Iframe for the portal. Typically, links within an Iframe are external from the portal environment, which can result in inconsistencies with a user's experience.

The PortletBridge-portlet is also known as a 'Web Clipping Portlet'. The PortletBridge-portlet proxies Web site content and renders the downstream content into a portlet. An interesting concept from the PortletBridge-portlet is using XSLT to control or transform the rendering of the downstream content. This is ideal for portlet environments, as portlets do not typically cover an entire page. Developers can select the parts they wish to render in a portlet, which is unlike an Iframe. These differences are also highlighted with the ability to resize the PortletBridge-portlet, which is also not possible in an Iframe.

The PortletBridge-portlet also allows developers to control, which links within the downstream content are proxied and which are not. Images or Flash animations from remote resources can also be proxied by the PortletBridge-portlet. A regular expression is used to define the links to proxy within the portlet and which links are external to the portlet. The rewriting of the downstream content is not limited to links. JavaScript and CSS can also be handled by the XSLT or with regular expressions. By default, only the body of a Web page is rendered by the PortletBridge into a portlet. This follows the JSR-168 specification in which no header or footer tags are allowed into portlets. However, it is still possible to manipulate the content of a page's header/footer with XSLT and have it rendered into the portlet.

The PortletBridge-portlet uses the CyberNecko HTML parser [14], which allows HTML pages to be accessed via XML interfaces. It is built around the Xerces Native Interface (XNI) that has the ability to communicate "streaming" documents. The PortletBridge-portlet also

defines a "memento" that holds user states, such as cookies. The memento is shared between the PortletBridge portlet and the PortletBridge servlet. For this interaction to occur the J2EE container, in this case Apache Tomcat needs to have `crossContext` enabled.

Currently the PortletBridge-portlet lacks the features to support complex XHTML and is a relatively young open source project. Nevertheless, it does present a viable option for making Web sites or applications available through a portlet. In addition, due to the adherence of the PortletBridge-portlet to the JSR-168 specification, it is also possible to use this technology in various different portals or even consume the portlet with Web Services for Remote Portlets (WSRP) [15]. Because the PortletBridge-portlet is a generic solution that is confined to portals, it is also a practical option to combine the PortletBridge with other bridging technologies such as the PHP-JavaBridge to provide a fully featured application environment.

3. BibAdmin Portlet Configuration

Non-Java applications, in our case, PHP, need to be redesigned or re-implemented with Java for use as a portlet. Using the techniques described in this section, a developer can reduce the amount of work needed to have a PHP application available as a portlet under a J2EE environment.

Our configuration consisted of the following components:

- Debian Linux with PHP5-CGI binaries,
- A PHP application,
- MySQL server 4.1.x,
- Apache Tomcat 5.5.16,
- PHP-JavaBridge 3.0.8r3,
- PortletBridge-portlet CVS,
- GridSphere 2.1.x.

Our installation of PHP included the MySQL extension, as it is common for PHP applications to be part of a LAMP/WAMP stack. An arbitrary multi-user application was chosen, namely BibAdmin 0.5 [16]. BibAdmin is a Bibtex bibliography server. The need for a multi-user application was to match the security framework of the portal and is required for future work outlined in Section 5. GridSphere is a JSR-168 compliant portal with good user support, which we have used extensively in other projects.

The PHP-JavaBridge was installed in Apache Tomcat as a `.war` file. The `php-servlet.jar` and `JavaBridge.jar` were also placed in `$CATALINA_HOME/shared/lib` as per the installation instructions [5]. The PHP-CGI binary and the MySQL PHP extension were used from the Debian Linux installation, instead of those originally included with the PHP-JavaBridge. This then provided a feature rich PHP environment to applications running with the PHP-JavaBridge.

When using PHP applications under the PHP-JavaBridge the PHP's configuration file (`php.ini`) is not in a default location. Instead it is located under `$CATALINA_HOME/webapps/JavaBridge/WEB-INF/cgi`, referenced as `php-cgi-i386-linux.ini`. Any PHP configurations are placed in the PHP configuration file, such as the configuration of the MySQL extension.

PHP 5 has some differences compared to PHP 4, which can lead to application incompatibilities. BibAdmin is a PHP 4 application, which was migrated to PHP 5 [17]. Certain PHP functions used in BibAdmin required modifications to reflect API changes in PHP 5, such as `mysql_fetch_object()`. Other changes included some PHP predefined variables. The PHP variable `$_SERVER['SERVER_NAME']` was changed to `$_SERVER['HTTP_HOST']`. This change was used to reflect the additional port number in the URLs needed for BibAdmin administrative tasks when creating a new user. Another useful PHP variable was `$_SERVER['PHP_SELF']`, which not only provides the PHP page, but also the Web application directory, i.e. `/BibAdmin-0.5/index.php`. BibAdmin under a normal PHP 5 setup was compared to BibAdmin under the PHP-JavaBridge. No differences were found in terms of functionality or usage.

The PortletBridge-portlet is typically installed from a Web archive. However, with GridSphere, the PortletBridge-portlet was first expanded into a directory. This was due to a missing root directory in the PortletBridge-portlet package. The created directory is copied into Apache Tomcat's `webapps` directory. Some additional changes to the `web.xml` file were needed for the PortletBridge-portlet to operate correctly in GridSphere. In addition, the GridSphere UI tag library is required.

A simple XSL style sheet was used that defines what transformations are carried out on the

downstream content from a Web site. In the Edit mode of the portlet, the configuration of the portlet can be altered. The preferences include proxy server authentication, the initial URL to load and XSL style sheets. The Edit mode of the PortletBridge-portlet is only used for development purposes. The end user would not generally use these preferences. The portlets proxy configurations are useful for larger or more complex networks. Different proxy authentication methods can be used, such as Windows NT LAN Manager (NTLM).

BibAdmin was configured with a MySQL database connection setting and packaged as a `.war` file. Included in the Web archive's `WEB-INF` was the CGI components, `web.xml` and any PHP-JavaBridge dependencies.

Figure 1, shows the final layout of the configuration for reusing a Web application in a portlet under a J2EE environment.

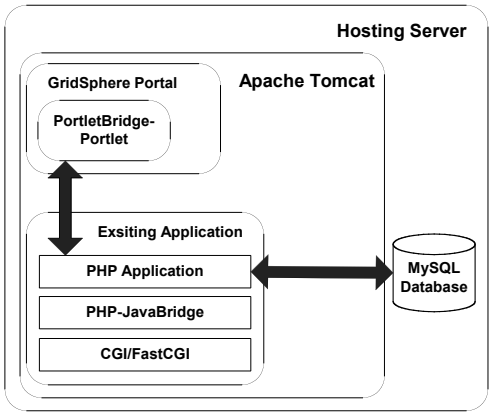

Figure 1: BibAdmin Portlet Configuration

The flexibility offered by the PHP-JavaBridge ensures that it is also possible to use the more common Apache Web server and PHP combination. The Apache Web server could also be located on the same or a different host. Communications with the Apache Web server would be with the XML protocol via sockets. This configuration can be de-coupled even further by having the portal and PortletBridge-portlet located on a different host. It was decided that a J2EE environment would provide the most challenges and thus best demonstrate the set-up.

4. The BibAdmin Portlet Results

It was observed that the PHP application, BibAdmin, was equally functional under the PHP-JavaBridge as it was under a normal Apache Web server. Although, this is only true after the minor changes needed to BibAdmin to make it aware of its different environment. Figure 2, shows the BibAdmin portlet login page. Figure 3, is another screenshot of the BibAdmin performing as a portlet.

Some parts of the PortletBridge-portlet are not complete, such as XHTML support. This hampers the number of sites that can be proxied by the PortletBridge. Currently PHP logins are also an issue. The PortletBridge developers have been contacted about this problem and it is under investigation. However, the login issue is not completely relevant to our needs, as it does not fit in with the portals security framework. A user should be automatically be logged into BibAdmin with the appropriate credentials, provided by the portal security framework. This area of research is discussed further in our future work under Section 6.

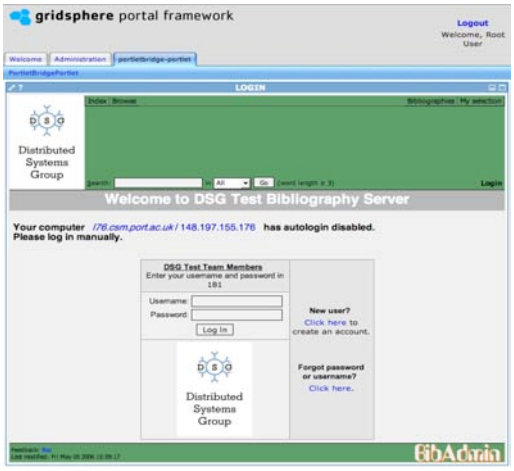

Figure 2: The BibAdmin Portlet

5. PHP Wiki Systems

This technique of reusing Web applications as portlets has been applied to other PHP applications. A Wiki offers an online resource, which allows multiple users to add and edit content collectively. It is also a suitable application to include in a portal and to test our bridging system. PHP Wiki applications include, PHPWiki [18] and DokuWiki [19].



Figure 3: The BibAdmin Portlet - BibTex Use

Both of these Wikis work under the PHP-JavaBridge, however they do not render correctly within the PortletBridge-portlet. This rendering error is due to the lack of support in the PortletBridge-portlet for XHTML. Figure 4 and Figure 5 show screenshots of PHPWiki and DokuWiki running under the PHP-JavaBridge, respectively.



Figure 4: PHPWiki with the PHP-JavaBridge



Figure 5: DokuWiki with the PHP-JavaBridge

As can be seen from figures that PHP applications work under the J2EE environment. The errors seen in the PortletBridge-portlet are primarily focused on the rendering. Not that it does not render, just that it is not usable. Once the PortletBridge-portlet has included XHTML support, these issues should be resolved.

6. Future Work

Our future work will concentrate on integrating PHP application further within portal frameworks. One area that we feel is important to develop urgently is a generic Single-Sign-On (SSO) system. Users can only access Web application portlets after being authenticated, by the portals security framework. It should then not be necessary to re-authenticate the user again with the Web application. Instead, the SSO system will use the security framework in the portal to provide an automated and trusted login for the Web application. Our SSO system will use a combination of the PHP-JavaBridge and the PortletBridge-portlet to pass information between a Portal and the PHP application. With this system a user's experience of Web applications via portlets will be transparent.

7. Conclusions

This paper has presented Java and portlet bridging technologies. Both types of bridging technologies have had their features and capabilities reviewed. We have also demonstrated how to reuse existing Web applications as portlets. Using a combination of these technologies, it is also possible to merge and manipulate the best features of Java, and scripting based languages. Developers can also concentrate on an applications usability rather than integration of technologies. This technique provides rapid portlet development resulting in highly usable portlets.

Portal developers can use the techniques presented, to not only provide new and existing tools along side within a common environment, but also allow applications to be used as a portlet and Web application simultaneously.

References

[1] JSR -168 specification, http://www.jcp.org/en/jsr/detail?id=168
[2] PHP, http://www.php.net
[3] Ruby on Rails, http://www.rubyonrails.org
[4] JSR-223: Scripting for the Java Platform, http://www.jcp.org/en/jsr/detail?id=223
[5] PHP-JavaBridge, http://php-java-bridge.sourceforge.net/
[6] Apache Portals, Portals Bridges, http://portals.apache.org/bridges/
[7] Apache Struts, http://apache.struts.org
[8] Struts Bridge, http://portals.apache.org/bridges/multiproject/portals-bridges-struts/index.html
[9] GridSphere Portal, http://www.gridsphere.org/
[10] PortletBridge-Portlet, http://www.portletbridge.org/
[11] JSR-223 Reference Implementation, http://jcp.org/aboutJava/communityprocess/pr/jsr223/index.html
[12] Rhino, http://www.mozilla.org/rhino/

[13] ECMA-335 standard, http://www.ecma-international.org/publications/standards/Ecma-335.htm
[14] CyberNeko HTML parser, http://people.apache.org/~andyc/neko/doc/html/index.html
[15] WSRP, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp
[16] BibAdmin, http://www-sop.inria.fr/axis/personnel/Sergiu.Chelcea/software.php?soft=bibtex
[17] PHP Manual, Migrating to PHP 5, http://php.ftp.cvut.cz/manual/en/migration5.php
[18] PHPWiki, http://phpwiki.sourceforge.net/phpwiki/
[19] DokuWiki, http://wiki.splitbrain.org/wiki:dokuwiki