Frequently Asked Questions about uPortal

Bill Brooks
California Polytechnic State University
Application & Information Management

1. FAQ Administration

1. What is this document?

This Frequently Asked Questions (FAQ) list will attempt to answer some of the most common questions relating to uPortal. Although this FAQ contains some technical information, you should use it only as a source for answers to as-yet-unanswered questions about uPortal and not a comprehensive reference document. See the Appendix section, below, for a comprehensive list of uPortal-related resources.

2. Who maintains this FAQ list?

Bill Brooks <wbrooks at calpoly.edu> with help from other active participants in the uPortal development effort. We encourage contributions of questions (and answers), comments, and corrections.

3. How can I contribute to this FAQ?

You can send your contributions, suggestions for improvement, etc to Bill via e-mail (see above) and he'll incorporate them.

4. Where can I find this FAQ on the World-Wide Web?

You can find the FAQ on the JA-SIG website, located at http://www.ja-sig.org/ You can always find the latest version in the `docs/` directory of the uPortal source code repository. See the uPortal web site for details about participant access to the source code repository.

2. General Questions about uPortal

1. What exactly is uPortal?

Simply put, uPortal is a framework for producing a campus portal. We don't intend it to be an out-of-the-box or "turn key" portal "solution". Instead, uPortal is a set of Java classes and XML/XSL documents that you can use to produce a portal for use on your campus.

2. Who is developing uPortal?

JA-SIG, the Java in Administration Special Interest Group. Several JA-SIG member institutions are collaborating on the development of uPortal, and they make the code itself available as a reference implementation to any member institution of higher education at no cost.

3. What level of Java knowledge is required to use JA-SIG's uPortal framework?

This is really three different questions rolled into one, because there are three different "levels" that we can use to discuss what you have to understand in order to "use" the uPortal framework. These levels aren't totally distinct - there's considerable overlap between them. Let's consider them one at a time.

One level addresses what you have to know in order to keep an instance of the portal, customized for your campus' requirements and IT infrastructure, running under your chosen application server. I'll call that the "administrative" level, since a lot of the tasks are closely analogous to administering a DBMS or HTTP server. See the section called "Deploying, administering and maintaining your portal", below.

The second level addresses what you must know in order to do the development necessary to *produce* an instance of the portal, customized for your campus' needs and IT infrastructure, that you'll use to facilitate the delivery of content and services to end-users. I'll call that the "implementation" level. See the section called "Developing your campus portal", below.

The third level addresses what you must know in order to develop programs (which the uPortal developers call *channels*) that will work within the portal to actually deliver the content and services. I'll call that the "content development" level. See the section called "Developing content and applications for use with uPortal ", below.

4. Will uPortal run on all platforms?

uPortal will run on any platform that has a Java 2 implementation available for it. JA-SIG members are running uPortal for development and deployment purposes on a number of different platforms, including Microsoft Windows, Solaris, Linux on 3 different architectures and MacOS X.

5. What differentiates uPortal from other portal solutions?

The primary differentiator is that uPortal is focused on the requirements of higher education institutions, whereas other products are mostly focused on the requirements of large corporations, even if the product is marketed to higher education institutions.

uPortal is a framework that provides the Java programmers on your staff with the classes, interfaces, XML data files and XSL stylesheets they need to develop a customized portal for your campus. JA-SIG realizes that a portal, in order to be truly useful, can't something you take out of a box and install, it has to be something you customize for the needs of your institution.

Some portal "solutions" are actually third party packages that a vendor of a non-Java, non-portal product acquired in order to enter the portal market without doing any R & D. uPortal was designed from the ground up to provide the tools you need to build a portal and has always been is a 100% pure Java.

Still other portal "solutions" are services that are no charge to your campus, but require your campus to accept advertising in exchange for the use of the service. With uPortal, you don't have to run advertising in your portal, but if you wish to, your campus can keep the revenue produced.

Last but not least, uPortal is also available free to member universities, most commercial portal "solutions" cost upwards of $50,000 per year for a typical campus' license.

3. Deploying, administering and maintaining your portal

1. What kind of Java expertise would I need to have in order to successfully deploy and keep my campus' instance of the portal running?

In general, over and above a working knowledge of and substantial experience with your chosen application server, you really have to understand the issues related to server-side application deployment and support in a modern, web-based environment. If you understand how to start, stop, and backup your your application server and atabase server, and how to deploy a JSP 1.1/Servlet 2.2-based web application, then you know enough to administer a uPortal instance. Your application server vendor will be able to help you on this front with training and documentation if you don't.

2. What application server should I use for my campus' uPortal instance?

uPortal will run in conjunction with any application server that complies with the Java Servlet 2.2 and Java Server Pages 1.1 specifications. There are members of JA-SIG currently operating uPortal instances with BEA's WebLogic, IBM's Websphere, and Caucho's Resin, among others.

Please don't think that uPortal is somehow dependent upon Tomcat, the open-source JSP and servlet container from the Apache Jakarta project that ships with the existing "monterey" distribution and soon-to-be-released   "plymouth" distribution. You can use any JSP 1.1/Servlet 2.2-compliant container or application server, and if you want commercial quality tech support, formal training or documentation, then a commercial application server is something to consider.

3. Does uPortal support single-sign on?

A qualified "Yes". The uPortal framework does provide an authentication system that explicitly supports the notion of single sign-on, but as of this writing there are few channels that ship with the framework which take advantage of it. Of course, any campus that wants to write their own channels that will use the uPortal single sign-on capabilities are certainly able to do so.

4. Does uPortal support LDAP authentication?

Yes. The problem, however, is that there are dozens of possible ways to do an LDAP authentication for uPortal and hence what is available now may not be useful for your campus. But there are a few Universities now who are using uPortal in conjunction with an LDAP server to authenticate users. There is a reference implementation available now for uPortal which handles authentication of users to the portal from information stored in an LDAP server.

5. Why doesn't uPortal use JAAS for authentication and/or authorization?

JAAS enables services to authenticate and enforce access controls upon users, but as currently specified by Sun, it makes many assumptions about how the services which use it will operate for effective use in a portal. JAAS allows authentication modules to be "plugged in" but doesn't allow services to modify the behavior of the system as a whole.

uPortal doesn't operate the way that JAAS assumes because a portal, any portal, must aggregate content and applications from disparate sources. uPortal must have an authentication mechanism that behaves differently than JAAS does because a portal must support single sign-on to applications which have their own, disparate authentication and/or authorization requirements.

The bottom line is that the uPortal developers think that authentication and authorization are currently too tightly coupled within JAAS, and would therefore delay the implementation of single sign-on in uPortal until Sun publishes a later revision of the JAAS specification which decouples them. Perhaps in the future JAAS could provide uPortal with security services, but in it's current state it does not have the flexibility nor the functionality to be incorporated well into uPortal.

4. Developing your campus portal

1. What kind of Java expertise is necessary to develop a campus portal using uPortal?

In general, to use the uPortal framework to develop a campus portal (the "instance" of the portal for use with your application server), over and above the level of expertise necessary to administer your application server (see above), you must be an experienced Java programmer, familiar with the JDK 1.2 tools for compiling, packaging, deploying and running server-side Java programs, and conversant with most of the technologies that comprise the J2EE specification.

Specifically, at a *minimum*, you must have experience with:

- XML. The uPortal framework uses XML and XSLT extensively to ensure the platform independence of data and to separate that data from it's presentation. To develop a portal with uPortal, you will have to create/edit XML files and create/edit XSL stylesheets.

  If you're not fluent with the XML-related APIs in Java, [McLaughlin2000] provides a comprehensive tutorial on writing Java programs that can create and manipulate XML documents. The uPortal developers highly recommend [Kay2000] to those looking for a tutorial and in-depth reference guide to XSLT. [Bradley2000] provides a guide to the overall XSL standard.

- JSP: The portal framework uses Java Server Pages to facilitate the intermixing of markup language code required by a web browser and programming language code that leverages the underlying Java Runtime environment. If you want to change the look and feel of the portal substantially, you'll have to edit Java Server Pages, so you should understand how to use JSP syntax correctly.

  For those looking for a tutorial on JSP, [Manning] is an excellent choice.

- Servlets. JSP works as an abstraction layer on top of the Java Servlets API and because of this, experience writing servlets will come in handy. Several programming 'idioms' in the portal code originated with Servlet programming.

  Although now a little dated, [Hunter1998] is a great tutorial on Java Servlets. The publisher expects to release a new edition soon.

- SQL. The portal requires the use of a relational database to store some user, authentication, and channel subscription information, and the standard way to issue requests to a relational database server is using SQL, the Structured Query Language. You must therefore be familiar with the syntax and proper use of SQL in order to work with the uPortal framework.

  [Gulutzan1999] offers comprehensive coverage of the SQL standard.

I say these are minimums because if you want to deliver something over and above the minimum functionality (for example authenticating via an LDAP server) then you'll have to be familiar with other J2EE technologies and/or APIs (for example, JNDI).

2. You keep writing that uPortal is a framework. Isn't that just an empty yuppie marketing word?

No. I'm using the word 'framework' in it's widely-accepted, Software Engineering sense[Wirfs-Brock1991]. A framework is a reusable design, expressed as a set of classes, that can serve as a solution to a family of related problems and support reuse at a larger granularity than classes. A mature framework allows components to be reused as "black boxes", that is, a programmer can incorporate them into a system under construction without knowing their implementations [Johnson1998].

Most frameworks in widespread use today are for constructing graphical user interfaces (GUIs) for traditional desktop applications. uPortal is different in that it's a framework for developing a web portal and developing content for display within that portal, but the underlying principal of both kinds of frameworks is the same: the framework provides programmers with an infrastructure that supports a coherent architectural model, allowing developers to concentrate on applying their expertise to the problem domain. In the case of uPortal, the framework takes care of the common functionality that every portal needs, so that you can implement the parts that are important and specific to your campus.

5. Developing content and applications for use with uPortal

1. What kind of expertise is necessary to develop content and/or applications for use with uPortal?

It depends upon what kind of content you want to deliver through the portal.

For simple content provision, you only have to know how to produce an XML document. One of the defined kinds of channels for use with uPortal is an RSS (Rich Site Summary) channel, the same format that Netscape uses for their Internet portal. If you create an XML document which adheres to this standard , the uPortal framework will handle formatting and presenting the output for you.

To deliver content which requires complex user interaction, you'll have to create a complete web application which uses a channel as its user interface. This approach requires that you have the XML, Servlets and SQL skills that someone developing a campus portal instance using uPortal has (see above) as well as experience with:

- JDBC. It's likely you will want your application to store information in and retrieve information from a database, so the ability to use the Java Database Connectivity API will be essential. The portal itself requires a relational database to store information related to users and their preferences, and the standard way for your application to access this information is by using JDBC.

If you're not familiar with the JDBC API, [White1999] is a comprehensive tutorial, reference and source of sample code

- EJB. If your application requires distributed data access, transactions and persistence, the Enterprise Java Beans distributed component model will probably become an important part of your architecture.

  For EJB beginners, [Monson-Haefel2000] provides a complete reference.

To get an idea of what you're in for when you begin developing channnels, carefully read Michael Oltz' tutorial .

2. My university has a large number of legacy applications that we want to provide access to from the portal. Can I do this?

Maybe. It depends upon the application. Did you write the application (and have the source code) or did you purchase a binary package from a vendor?

If you have the source code, its relatively easy to create an alternate user interface to the application that operates within the portal. in uPortal parlance, this would be called a "channel". Your channel would display itself to the user in the portal and feed the input from that user to your application. Your legacy application would then process the data and send the output back to the channel, and the channel would in turn display the results to the user inside the portal.

If you purchased an application from a vendor, things are a bit more complicated, but the high-level approach is the same. You'd have to write an adaptor that would wrap the output of the legacy application in XML so that it can be displayed inside the portal. When you have this, you can then write a channel for use with the portal that interacts with your adaptor and actually facilitates the translation of the XML so that the portal can display it.

6. Colophon

1. How was this FAQ produced?

The FAQ document was primarily written with GNU EMACS, version 20.3.1, on a no-name 100 Mhz Pentium PC clone running RedHat Linux 6.2.

The source code is an XML document using tags defined in The DocBook XML DTD. DocBook is an open, public, *de jure* industry-standard DTD for software documentation created and promulgated by OASIS, The Organization for the Advancement of Structured Information Standards . Using DocBook allows us to be platform and application independent, as well as facilitate the distribution of the FAQ in multiple formats, most notably PDF and HTML .

In order to produce these secondary formats, we use The Apache Group's Xalan-J Java-based XSLT processor in concert with Norm Walsh's DocBook XSL stylesheets to transform the XML source into the target formats. In JA-SIG, we drink the Kool-Aid!

2. Why can't you just use Microsoft Word .doc files?

Three reasons: Microsoft Word .doc files are not portable, the file format is proprietary, and the format changes often and in highly undocumented and incompatible ways.

Microsoft Word .doc files are inherently non-portable because they are only designed to be read and modified by Microsoft Word or other programs in the Microsoft Office Suite. Since Word is not available for all the platforms supported by uPortal, we choose not to use it so as to not hinder the use and deployment of uPortal on platforms which do not have a version of MS Word available for them. A sizeable portion of the members of JA-SIG use platforms that do not have a version of Word available for them as their primary computing platform, for both development and deployment.

The Word file format is proprietary to Microsoft and certain aspects of it are patented, which would prevent JA-SIG developers from developing and distributing tools and applications that use it free of charge. One of JA-SIG's objectives is to deliver uPortal to member institutions free of charge. Microsoft does make libraries for reading and writing .doc files available to third parties, but those libraries are only available for Microsoft operating systems and compilers and thus are inherently non-portable.

The .doc file format changes rapidly, sometimes with every new version of Microsoft Office and/or Word. Some versions aren't even completely portable across different versions of MS Word and to non-Windows operating systems with a version of MS Word available. Steering clear of .doc files completely allows us to avoid dictating which specific version of MS Word readers of our documents must use.

3. What's so great about DocBook?

There are at least three reasons for us to use DocBook: The document files are portable, the files interoperate with tools from multiple vendors and the process for changing the set of defined tags is open and well-publicized.

DocBook files are portable because they are simply normal text files with tags marking up the data or prose. Any operating system that supports the ISO standards for text, including Unicode, can successfully host a DocBook file.

We can create, display and modify DocBook files using any tool that can process XML. In addition to the myriad of commercial tools available on the market for processing

XML, there are several free and/or open-source tools available for use with XML documents. An important thing to keep in mind is that these tools interoperate, that is, you can use one tool to create a DocBook file, use a tool from a completely different source to process it and use yet another to display it, with no loss of fidelity due to translation.

The DocBook tag set changes with time, but OASIS publicizes them in advance so that users and vendors can prepare for them. The specific rules for making such changes reflect industry consensus. For example, minor releases of the DocBook can add to the markup model, but not change it in a backward-incompatible way. Major releases can introduce backwardly-incompatible changes, but those changes must be announced in the previous release, which must pre-date the new version by at least one year. Finally, OASIS makes every version of the DocBook Document Type Definition available for free on the World-Wide Web.

One of the primary benefits of using DocBook is common to any structured markup language: separation of semantics and presentation, which facilitates software translation of the document into multiple formats. For uPortal, we're currently using HTML for the web and PDF for both on-line viewing and support for high-quality printing. We generate both formats from exactly the same set of DocBook source files, and if we choose to support another format in the future, say, `.rtf` or some format that hasn't yet been invented, we can do so without changing the original source documents.

4. But wait...I can't find the PDF version of the FAQ!

You're probably looking in a directory structure produced by a direct checkout from the source code repository. Since PDF files can get large, we only produce them when we issue a release of uPortal. The latest source XML files, however, are always available from the repository. If you'd like to look at PDF versions of the uPortal docs in their development state, you have to build the .pdf file from source using the tools described in the README file in the docs/ directory.

7. Appendix

References

Books

[1] [Bradley2000] Neil Bradley. Copyright © 2000 Neil Bradley. 0-201-67487-4. *The XSL Companion*. Addison-Wesley Publishing Company.

[2] [Gulutzan1999] Peter Gulutzan1999 and Trudy Pelzer. Copyright © 1999. 0-879-30568-1. *SQL-99 Complete, Really*. CMP Books.

[3] [Harold1999] Elliotte Rusty Harold. Copyright © 1999. 0-764532-36-7. *XML Bible*. IDG Books.

[4] [Hunter1998] Jason Hunter and William Crawford. Copyright © 1998. 1-565923-91-X. *Java Servlet Programming*. O'Reilly & Associates.

[5] [Kay2000] Michael Kay. Copyright © 2000 Wrox Press. 1-861003-12-9. *XSLT Programmer's Reference*. Wrox Press.

[6] [Fields2000] Duane K. Fields and Mark A. Kolb. Copyright © 2000. 1-884777-99-6. *Web Development with JavaServer Pages*. Manning Publications Company.

[7] [McLaughlin2000] Brett McLaughlin. Copyright © 2000 O'Reilly & Associates. 0-596-00016-2. *Java and XML*. O'Reilly & Associates.

[8] [Maruyama1999] Hiroshi Maruyama, Kento Tamura, and Naohiko Uramoto. Copyright © 1999. 0-201-48543-5. *XML and Java : Developing Web Applications*. Addison-Wesley Publishing Company.

[9] [Monson-Haefel2000] Richard Monson-Haefel. Copyright © 2000 O'Reilly & Associates. 1-56592-869-5. *Enterprise JavaBeans*. O'Reilly & Associates.

[10] [White1999] Seth White, Maydene Fisher, Rick Cattell, and Graham Hamilton. Copyright © 1999. 0-2014-3328-1. *JDBC API Tutorial and Reference, Second Edition: Universal Data Access for the Java 2 Platform*. O'Reilly & Associates.

Articles

[1] [Bosak1999] Jon Bosak and Tim Bray. *XML and the Second-Generation Web*. *Scientific American*. 16-25. May 1999.

[2] [Johnson1998] Ralph Johnson. *Designing Reusable Classes*. *Journal of Object-Oriented Programming*. 19-45. June/July 1998.

[3] [Wirfs-Brock1991] Rebecca J. Wirfs-Brock. *Object-Oriented Frameworks*. *American Programmer*. 21-29. October 1991.

$Revision: 1.1.2.8 $