The University of Sydney

A CATALOG BROWSER FOR THE OPEN DIRECTORY PROJECT

ABBAS CHEBLI

BACHELOR OF SOFTWARE ENGINEERING
0011561

SUPERVISOR

DR RAFAEL A. CALVO
WEB ENGINEERING GROUP 2003

Abstract

As the web continues to grow at exponential rates, users have continued to increase their dependence on tools and services that allow some sort of organisation and categorization of the internet and its resources. Although many may not realise this, it is only evident that the best search engines and catalog functionality sites in the world are supported by some sort of main organisational catalog hierarchy that allows the most relevant resources available in the world to be returned in search queries, or allow even the smallest areas of specialization in fields to have thousands of relevant sites, documents and resources. As the world is rapidly evolving such tools are becoming a daily necessity in our busy lifestyle not only for recreational web users but also for millions of students all over the world who access such resources regularly in order to retrieve information concerning their units of study.

With the initiative of this electronic dependence on information the new E-learning solution is quickly becoming a core part of every educational institution and has been viewed as the current and future solution to speeding up changes in the existing educational systems in order to provide a much more knowledge-based society.

This thesis project aims to investigate opportunities for the development of an open source web-based solution to aid the E-Learning community in obtaining such catalog resources within their community portal. Following careful analysis of the options a dotLRN portlet was designed and developed using the OpenACS web application framework as a proposed solution to this issue, this web application achieved its set goals and provided the following:

- *A reliable and efficient cataloguing portlet*
- *A simple corresponding browser interface to access catalog hierarchy*
- *Support for all ODP data hierarchies and future imports of catalogs*
- *An original solution to this problem by being the first web application to integrate such functionality into an E-Learning platform of any kind*

Acknowledgements

Special acknowledgement must go out to my supervisor Dr Rafael Calvo for his great supervising role, he provided great help and guidance throughout the entire thesis period by dedicating much of his time to answering my queries and proposing better ideas for the development of the application.

Secondly I would like to thank Nick Carroll for his help in many of the technical issues throughout the beginning of the semester, his experience in such issues served well in helping me get started on my development. Thank you also for Ernie Ghiglione who aided me in the importing of the catalog onto the University server, this lengthy process consumed some of his time which is much appreciated. Furthermore, I am especially appreciative and grateful to the entire OpenACS community for their invaluable support with many of the architecture and software development issues, a special thanks to community member Neophytos Demetriou who spent a lot of his time aiding me in the understanding of the technical issues of the RDF files and the importing process of the catalog, his assistance sparked my understanding of the entire process and allowed it to be possible.

Thank you for all fellow group members of my thesis group, their online collaboration and discussions throughout the semester proved to be useful and informative.

I wish to also thank my family for their understanding and continued support throughout this stressful semester. Finally, I would like to thank god for keeping me patient and focused on the goals I set out to achieve and allowing me to remain healthy throughout the semester.

Table of Contents

Table of Figures

Introduction

The prospering of current available web resources has constituted many projects of its own, some of which have extended for many years and till this day are still in operation. Such an example is the well known Open Directory Project (ODP) allowing net citizens or “editors” to register to a central website and help in “the organisation of the internet”. The ODP data is indisputably the most powerful catalog available on the web today, its information forms the backbone of some of the best online tools such as Google search and Yahoo categories by providing them with the most updated data of web resources. However, this powerful tool has not been used to its full potential, it has not been provided for the millions of students throughout university’s and schools via their everyday E-learning platform, which is also proving to be the best strategy for convenience in education.

In the last few years, the increase of such electronic applications and platforms in our everyday life has profoundly and irreversibly changed our views on the development of solutions to aid this process of E-learning. Such software cannot be based on the traditional static E-learning platform but rather must be developed on open architectures that continuously change and evolve to accommodate new requirements and components of the future. Hence the decision to allow the integration of this powerful catalog functionality into OpenACS, a prospering open architecture with its corresponding open source E-learning platform “dotLRN” which is already being implemented across some of the best university’s around the world, proving that it is here to stay.

Statement of Achievements

According to my thesis study, the following achievements were made:

In terms of background and new concepts learnt:

- I learnt and understood the OpenACS Framework and many of its operations.
- I learnt and understood the dotLRN platform and its operations.
- I learnt in depth about RDF file formats and how they are implemented.
- I learnt how to program in TCL and plpgsql according to the OpenACS architecture.
- Increased my experience with PostgreSQL and the SQL query language.

In terms of the main catalog system development:

- I allowed the import of current/future ODP catalogs into the PostgreSQL database.
- I designed the data model to hold the catalog.
- Analysed, designed and developed an OpenACS package
- Analysed, designed and developed a dotLRN portlet package
- Allowed the exact depiction of catalog hierarchy listing according to professional ongoing projects such as DMOZ.
- Created a simple and powerful user interface that catered for experienced and non experienced catalog users and allowed browsing and searching of the catalog.
- Allowed the addition of resource links from the catalog into the dotLRN portlet without burdening the user of any technical tasks.
- Provided a permission system so as to customize the functionality available according to certain rights of adding and removing resources from the portlet.
- Provided add/delete/edit/comment functionality on links added to the portlet.

Abbas Chebli Date

Rafael Calvo Date

CHAPTER 1

OVERVIEW

1.1. OpenACS – A Web Application Framework

The OpenACS application framework is an open source system that was initiated by the ArsDigita Corporation. It is based on the former Arsdigita Community System (ACS) that was first initiated in 1997 by Phillip Greenspun and others as a form of solution to the common problem of “creating scalable web community systems” as well as reaching a compromising level of software reuse across website developments in communities that have similar business requirements [1].

Today, with almost 7000 community members the prospering OpenACS has been widely recognized as an advanced toolkit used for building scalable, community-oriented web applications by many software developers and organisations from all over the world. What initially started as a single project has matured and is now responsible for the creation and co-ordination of many online community projects of its own.

In general, an application framework can be defined as a “reusable, ‘semi-complete’ application that can be specialised to produce custom applications“[1], which essentially provides software developers with a number of benefits consisting of modularity, reusability, extensibility and inversion of control. The term “web application framework” relates to these framework architectures that are designed specifically to meet the needs of a particular web application domain [1].

OpenACS is a “web application framework” which enables web developers to “rapidly implement modular and extensible web applications, achieving significant levels of software reuse by utilizing the features of the core OACS packages” [1]. It is mainly implemented via TCL, as a group of applications that interact with AOLServer and a RDBMS (so far only PostgreSQL an Oracle) and can be divided into a number of core packages, services and applications, this is outlined in figure 1.

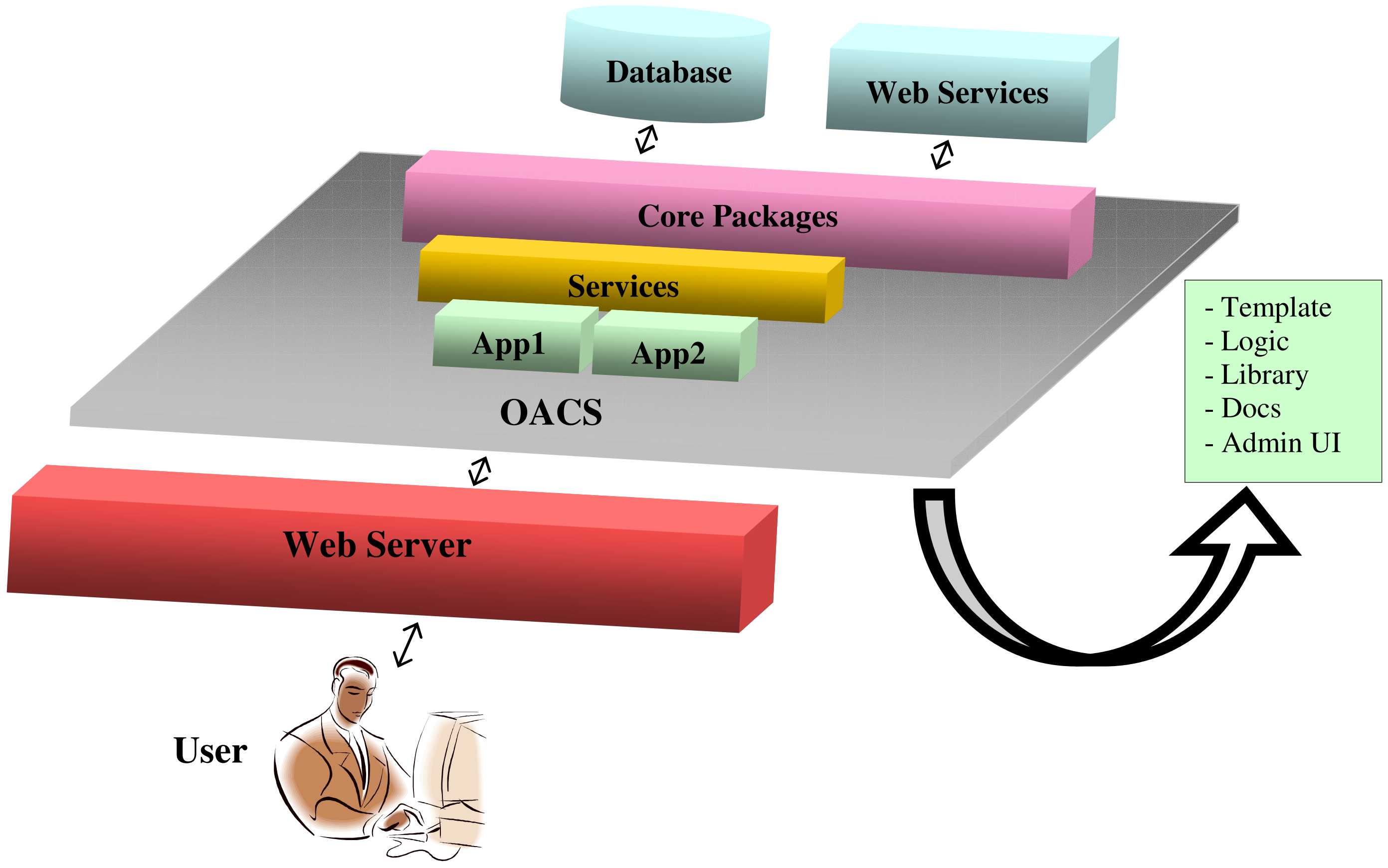


Figure 1: The OpenACS Architecture [1]

The main base layer labeled “core packages” consists of the core packages required for OACS to function, they include a kernel, package management, administration utilities (users, sitemap and other common business objects), a template system that cleanly separates the business logic from the presentation layer, and a backend mail and messaging system. Services, such as the events package are implementations of reusable logic that is not application dependant and does not require a user interface. Application packages are used to interact with the users, they usually coincide with the service packages to provide the functionality for the users. These packages have a user interface and their code is clearly divided into application and presentation logic (discussed in more detail further on in the document) [1].

Each service and application within the architecture requires a database data model that will be integrated into the framework. Through this integration we enable reusability and a number of functionalities. To help increase the reusability the framework has been designed to have an Object Oriented (OO) architecture although, since the RDBMS does not implement OO functionalities, they must be added ad-hoc by the framework. The OACS objects have their attributes stored in tables and a set of functions/methods defined as PL/SQL packages. These packages hold the procedures that make the programming interface for the data model [1].

The essential idea behind this OO architecture is that each piece of information that may be reusable should be an object. Since RDBMS are inherently not OO, data structures are integrated into the framework by being added to a table that keeps track of every OACS object. Another table defines the standard attributes stored on every object, including a system wide unique ID, an object type and auditing columns. The concept of OACS object types is equivalent to classes in OO programming languages (such as Java), regrettably since we are using a RDBMS additional work must be performed [1].

1.2. The dotLRN Design and Architecture

DotLRN is one of the major projects under the development of the OACS community, it is an open source solution that consists of an enterprise-class suite of web applications with an accompanying portal framework for supporting course management, online communities and collaboration [2]. This platform was originally built by Arsdigita for the Sloan School of management at MIT where it was based on the ArsDigita Community (ACS). This learning management system was initially called ACES until two years later where it was released with a new architecture and under the name of “dotLRN” [3].

Today dotLRN is used worldwide in places such as MIT (U.S.), University of Heidelberg (Germany), University of Cambridge (U.K.), University of Bergen (Norway), University of Sydney (Australia), Universidad Galileo (Guatemala), and University of Copenhagen (Denmark). Its popularity is due not only to the functionality it provides to users but also to its great architectural design that is built upon the existing OpenACS framework. By adhering to this design, the inheritance of all OACS functionalities, modularity and features by dotLRN was made possible as the underlying layers of the dotLRN platform refer back to the initial OACS architecture which consists of the ACS-core packages that provide an API for common functionality between packages and services, this is described in figure 2 below.

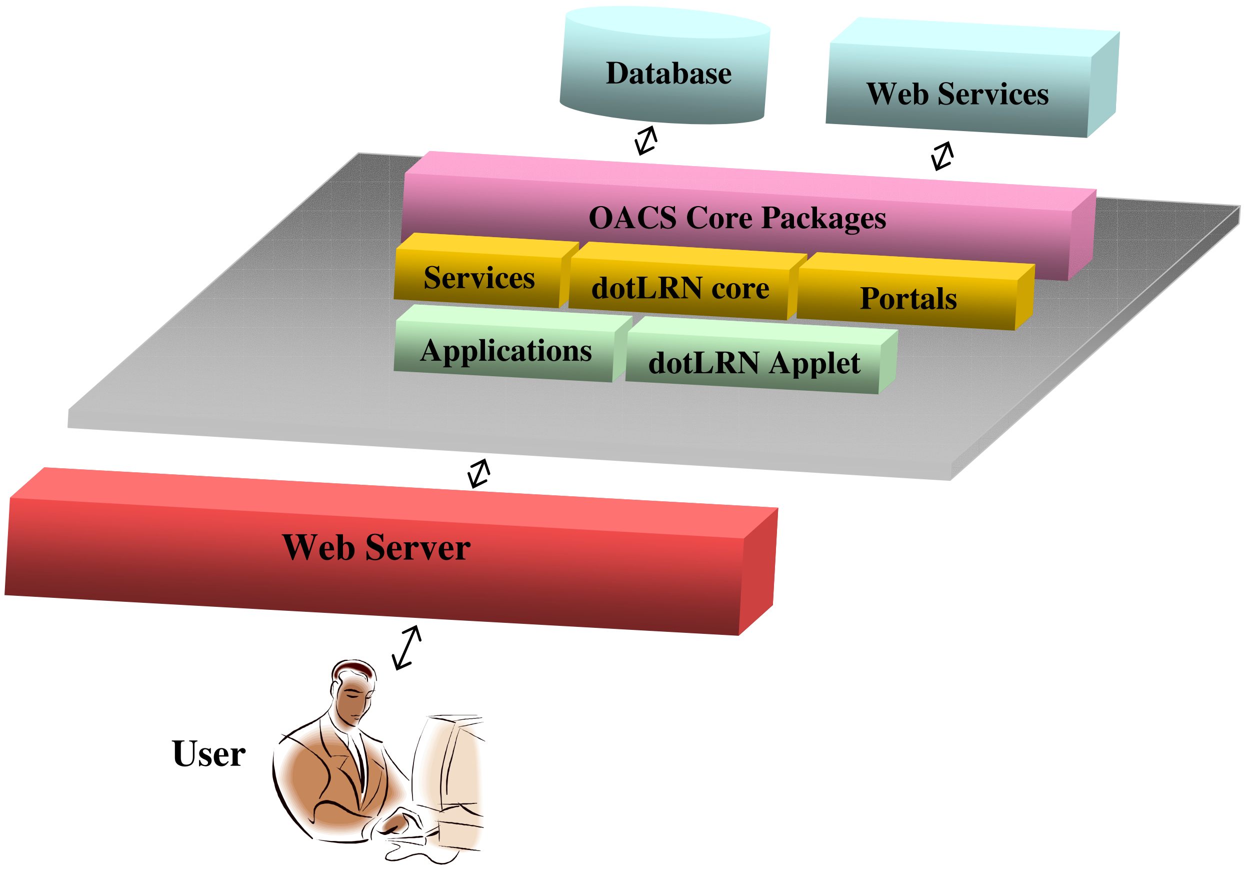


Figure 2: OACS and dotLRN Architectures [3]

From the diagram we note that the OACS core packages, services and Applications layers are exactly as they were described in figure 1, that is they constitute the same functionality that they perform within the OACS architecture. The new layers visible in this diagram consist of:

dotLRN Core: This refers to the main packages that make up the dotLRN library, they provide an API that all other packages in the dotLRN architecture follow and use, you will also note that this layer lays above the main OpenACS core packages base layer which is the main API for OACS development, hence its reason for inheriting all of its functionality and properties [3].

Portals: A Portal is a user interface that presents information collected from many sources or packages within the architecture to the end user. Portals can belong to either groups/communities or a single user, they will be further discussed in the next section to help indicate their importance in the design of the proposed system [4].

dotLRN applet: A dotLRN applet is simply the "glue" that responds to specific requests. Any dotLRN package is an applet according to the dotLRN platform, which essentially means that an applet is some sort of "small application" that must provide a certain API which allows some basic functionality of the dotLRN platform. What is meant here is the ability of this API to allow the dotLRN core to dispatch calls to each applet when certain events happen. In particular there are 4 types of events that the applet must be able to respond to these are [5]:

- Addition of this applet to a community
- Removal of this applet from a community
- User is added to a community that has enabled this applet
- User is removed from a community that has enabled this applet

There are also other main architectural aspects of dotLRN that are not illustrated in figure 2 since they are more related to the way groups/users and content is managed within a portal these consist of:

DotLRN communities: These dotLRN communities were created in order to follow the realistic way people learn within a physical institution, that is through the courses they take and the communities they interact in. Each community is liable to its own templates, applications, groups, permissions and much more. Multiple communities may be setup under different sub-sites, as well as each community can be organised into "classes" or "clubs" both of which provide different collaboration tools, roles for users and default templates [3].

Portlets: A portlet is basically the interface of the applet, it is a "view" of a package or piece of content visible within a certain group or community's portal, one such example is the calendar portlet which is presented as a small "window" in the main dotLRN community portal page, a user can add and remove selected portlets from the page to customize it to his needs and liking [3].

The most significant point to realize about the dotLRN platform is that it is there to increase the amount of learning for students while maintaining a minimal workload for the teachers by allowing or encouraging students to learn from each other and depend less on their teachers. This includes online collaboration by students in forums, chat, file sharing and many other activities all joined together in one main “online community” which is used to assist the traditional management of an educational course or subject being held at an educational institute.

1.3. Portlet Architecture

From the previous section’s discussion the architecture of OACS and dotLRN shall be fairly clear, although one main point that was not elaborated on is why this portal architecture was chosen as a solution to implement the dotLRN functionality?

From the origin of portals back in 1999 when the developer Aurelius Prozhacka built the first portal system, it was realized that this architecture helped gather information from different places and then organise or present this data in a customized way to allow usability by different users or groups of users. During this time the portal system was designed for a web site to collect data from several departmental servers and then present this data to different classes of users ranging from students to professors.

Today after the improvements on the portal system by Arsdigita (and then OACS) who basically made it a standard platform for the education solution, the portal system has become the best architecture for web based E-learning solution since it solves two of the major problems for this environment. The first is being able to aggregate information or content from the organisations internal and external data sources into the one main portal, the second is allowing this content to be customized and personalized as well as laid out accordingly to the group or personal users.

This new Portal architecture is used in dotLRN to allow the mentioned dotLRN applets from figure 2 to provide the users of their portal with an interface to a certain package. These dotLRN applets can just simply add themselves to an existing portal page via a portlet, since they are independent of this dotLRN-architecture in the sense that when they are added to a portal, but the portlet content usually relies on dotLRN functionality [5].

1.4. The ODP Directory

Those who have heard of the Open Directory Project (ODP) can only appreciate what it has contributed to modern website services and tools, as it has been the largest , most comprehensive human-edited directory of the Web. Constructed and maintained by a global community of volunteer editors, ODP is following the footsteps of the Oxford dictionary, which became the definitive word on words through efforts of volunteers, although its aim is to become the definitive catalog of the web [6].

With over 3.8 million websites categorised in one of 460 000 categories ODP has provided a means for the internet to organise itself, by allowing each net-citizen to contribute and select a small portion of the web to categorise, today the number of contributors is almost 59 000 and still growing, each of which remove any bad or useless links and categorise the best ones from the category they are given. Being 100 % free the ODP has been become a standard for many major websites all over the world. Famous names such as Google, Netscape search, AOL search, Lycos, Hotbot and hundreds more operate on the ODP to increase their search speeds as well as quality for their services. Although, it must not be mistaken, the ODP is a web directory not a search engine, that is its main aim is to list and categorise web sites in one of its categories that seem best suited – and it is through this that allows these search engines to improve their retrieval speed of results [6].

The ODP data or catalog is made available via files called RDF (Resource Description Framework) “dumps” which is basically an XML format, that can be parsed and inserted into a selected Database and data model (all of which are discussed further later in the document).

1.5. Browser System Introduction

From the previous two sections, two main points must be stressed. The first being the importance of the new portal architecture to the E-learning platform that is growing to be one of the largest in the world-dotLRN. The second is realising the capabilities of such a tool as the ODP catalog and how it can be used to provide many services for web citizens. These are the two main pivotal points that the proposed catalog browser system is based on.

Please note: *The proposed system developed in this thesis named "catalog browser system for the ODP project" will be referred to as the "browser system" throughout the document for the sake of brevity.*

The browser system is to combine these two factors of great web sources being classified and categorised and the new E-learning architecture with its portal design to provide one tool for dotLRN communities that will allow catalog functionality to be used in an E-learning environment. This refers to certain relevant resources being made available mainly to students of a course by their lecturer/staff who would have added a personalized list of resources they recommend according to certain assessments/projects within the school term.

The browser system consists of two main components, the first of which is a general OACS package that will allow the ODP catalog (both structure and content) to be imported and viewed through a great, easy to use UI which will support a personalized list of resources to be selected. The second is to provide a dotLRN portlet package that can display the personalized list within a portlet for a certain course/community portal to its students who can access these links/categories directly from their personalized community portal.

1.6. Competitive Analysis

In terms of competitive analysis, although there are hundreds of sites all over the world that provide catalog browsing functionality with the ODP data, not one of them have integrated it into an E-learning system/platform or any kind of E-learning website, hence this thesis study was completely original, in fact this was one of the main reasons for my supervisor Dr Rafael to propose this great idea as a thesis project. Hence to find actual competitors for any similar work is impossible, rather I have chosen to discuss one of the main prospering catalog software programs on the market currently, named “Senga” to give the reader an indication of how the browser package can be expanded in future work to include such functionalities.

Senga

Found at www.senga.org the Senga catalog is a perl program that implements a MYSQL database and allows the user to create, maintain and display directories, similar to the ones available at www.Yahoo.com. The project is based on the main idea that creating a catalog is mainly a matter of organizing objects in a structured tree, where each object is a record in the table of the database [7]. Senga catalog currently has many features and functionalities that it provides for its users, some of its main characteristics are listed below [7]:

Senga Characteristics
• Display structured trees of resources. • Display chronologically ordered resources and associated calendar. • Display alphabetical indexes of resources. • Allow full text search in the resources and the category names. • Powerful HTML based administration of catalogs. • On-Line editing of resource records. • Handle an arbitrary number of catalogs. • High performances using mod_perl and Apache. • Easy customization of the user view using HTML template files. • It is possible to define more than one view of the same catalog. • Load and unload a thematic catalog in XML. • Create an HTML dump of a structured tree to publish a static version. • 100 % free – Open source under GNU license

Senga is an ongoing project, that has been in operation for years, hence in terms of catalog functionality it cannot be compared to the browser system, but rather can be used as a guide for future work to include all these functionalities into the browser system.

1.7. Historical Considerations

In 2002 two students by the name of Mitsunori Habadera and Roger To from the University of Sydney worked on a catalog package for OACS. This package named “Collaborative Bibliographic Catalog” aimed to provide an online bibliographic system, that would allow users to access the catalog and update it with their literature of interest. The system allowed resources to be gathered and organised into a central database of references that could be edited and commented upon. It supported a type of XML importing , although this was very limited as it only could insert XML documents following an exact format that suited their data structure, and this format was not a well know standard such as RDF.

Due to the lack of time available they did not complete one of their extra functionalities, which was to import the ODP catalog into their database since this would have introduced a totally new design of the importing functionality, as the ODP RDF structure had a tag structure of its own, consisting of over 30 tags – each of which were inter-related to other category, subcategory and link tags.

The students’ main aim was dedicated to creating a catalog that could add, delete, edit resources as well as provide search engine functionality for the existing data in the database in which they succeeded. Their package did not integrate into any part of dotLRN to provide the users with another additional E-learning opportunity, it was solely based on being an OACS package.

CHAPTER 2

REQUIREMENTS

2.1. System Requirements

Here we shall discuss the expected requirements of the browser system (both browser package and portlet) which revolve around two main types, the first being functional requirements which represent the requirements that provide tangible services for the users, the second being the non functional requirements which are either constraints or expected non tangible functionalities that are imposed on the system by the user. The complete list of functionalities is mentioned here although due to time restrictions some of the functionalities may not be implemented in the final browser system.

2.1.1. Functional Requirements

- ***1.0: The ODP Catalog***

  1.1: Import structure of ODP catalog into PostgreSQL.

  1.2: Import content of ODP catalog into PostgreSQL.

  1.3: Provide interface for ODP data browsing.

  1.4. Support generic RDF import of any future ODP catalog

- ***2.0: Browser package users (Staff and students)***

  2.1: User must have the ability to browse the catalog.

  - ***2.1.1:*** User has the ability to browse categories.
  - ***2.1.2:*** User has the ability to browse sub-categories.
  - ***2.1.3:*** User has the ability to browse links.

  2.2: User has the ability to acquire a description of the current browsing category.

  2.3: User has the ability to visit a selected link from the category.

- ***3.0: Managing the portlet list of the Browser package (Staff only)***

  3.1: Staff member can add links to a personal portlet list.

  - ***3.1.1:*** Staff member has the ability to add a single link from the catalog.
  - ***3.1.2:*** Staff member has the ability to add a single subcategory from the catalog.
  - ***3.1.3:*** Staff member can add a custom link.

  3.2: Staff member has the ability to delete links from the portlet list.

  3.3: Staff member has the ability to edit links from the portlet list.

  3.4: Staff member has the ability to view current links in the portlet list.

- ***4.0: Browser Portlet package Operations***

  4.1: Browser Portlet is able to be added to a community portal.

  4.2: Browser Portlet is able to be removed from a community portal.

  4.3: Browser Portlet is able to be minimized within a community portal.

  4.4: Browser Portlet is able to be maximized within a community portal.

- ***5.0: Browser portlet Users (Staff and Students)***

  5.1: User has the ability to view current links and their comments in the portlet window.

  5.2: User has the ability to view current categories in the portlet window.

  5.3: User has the ability to access each link within the portlet.

  5.4: User has the ability to search the ODP catalog from the portlet window.

- ***6.0: Browser System Administration***

  6.1: The administrator has the ability to add and remove the portlet from a community

  6.2: The administrator has the ability to grant administration rights to staff members, so as to allow the modifying of the portlet window.

  6.3: The administrator has the ability to view any part of the catalog

2.1.2. Non Functional Requirements

- ***7.0: Usability***

  7.1: The browser package and portlet must provide an easy to use user interface.

  7.2: All catalog hierarchy is to be clearly presented on the browse screen of the interface.

  7.3: The catalog browser interface must clearly identify the tag structure of the ODP RDF tags.

  7.4: The portlet window must simply allow the clear view of the currently added links and their comments.

- ***8.0: Portability***

  8.1: The browser package must be able to be viewed in all browsers under all OS platforms such as Internet explorer for windows and Mozilla and Netscape for Linux.

- ***9.0: Reusability***

  9.1**:** The browser package must be easily mounted and installed into OpenACS/dotLRN.

9.2: The browser package must be easily removed from OpenACS/dotLRN.

9.3: The browser system with both packages must be made freely available to all community members to view, modify and improve since it is an open source system.

- ***10.0: Availability***

  10.1: The Browser system must be available and in operation most of the time with minimum downtime for all OpenACS/dotLRN Users after being installed and mounted.

- ***11.0: Reliability***

  11.1: The Browser system must provide all users' requests at all times as well as serve users performed tasks on the portlet list (adding/deleting/modifying).

- ***12.0: Security***

  12.1: The Browser system must assure that only appropriate users with certain permissions can add links/resources to the dotLRN portlet list.

- ***13.0: Scalability***

  13.1: The Browser system must be able to handle the large ODP data and future ODP catalog imports, as well as handle a large demand of users without affecting any performance aspects of the system.

- ***14.0: Standards Compliance***

  14.1: The Browser package must be standard compliant to OpenACS architecture framework and engineering standards.

  14.2: The DotLRN portlet package must be standard compliant to DotLRN architecture framework and engineering standards.

2.1.3. Additional Functionalities

Apart from the main functionalities that were named above, there are also some other extra functionalities that can and may be added to provide extra operations for the users of the Browser system. Please note, the number of these implemented is due to the amount of spare time available , any other functionalities that are not completed are also listed to give some indication of future work opportunities.

- ***15.0: User has the ability to search the ODP catalog with an entered query.***

  15.1: User has the ability to search with the "Title" field. **[Implemented]**

  15.2: User has the ability to search with the "Category" field. **[Implemented]**

  15.3: User has the ability to search with the "All" field. **[Implemented]**

  15.4: User has the ability to search with the "urls" field. **[Implemented]**

  15.5: User has the ability to search from the portlet window. **[Implemented]**

  15.6: Implement advanced searching algorithms and techniques.

- ***16.0: User has the ability to comment on links***

  16.1: Staff member has the ability to comment on current links in the portlet list. **[Implemented]**

- ***17.0: Browser System Notifies Users of changes***

  17.1: Browser system emails student members of the community (or appropriate course) to inform of changes made to portlet links.

- ***18.0 Browser system management of editors***

  18.1 Allow editors to register.

  18.2 Allow editors to help categorise sections of chosen categories.

2.2. Use Cases and User Scenarios

2.2.1. Actors

Within the proposed browser system, many actors were involved that interacted with certain functionalities according to the permissions they have been granted by the administrator. The main actors for the system can be stated as follows:

- ***Browser System Administrator*** **–** At the top of the user hierarchy, the administrator is the one responsible for the typical administration duties of an OpenACS package. He has the right to edit and grant rights to specific groups/users within the community that the portlet is mounted. The Administrator initially manages the applet configuration of the community to include the browser portlet and then from there can make it viewable to certain groups/users within that community.

- ***Catalog Staff*** **–** Usually depicted by a staff member, or a senior tutor of a certain course in the dotLRN community, catalog staff members will have total control of the viewable contents of their particular community's portlet, by being able to modify the portlet's list in the main browser package through the user interface. These members would need to have been granted administration rights on the portlet by the Administrator for their particular community/group.

- ***Registered Portal Students*** **–** This group of actors mainly concerns the registered users of the dotLRN portal/community containing the portlet, for simplicity we will assume that in most cases these are registered students of a certain course. Their main role is to be able to view the links added to the portlet by the staff as well as access these links directly from the portlet window.

- ***Browser System*** **–** This represents the system behind the scenes of the browser package that controls the architectural protocols and communication that exists between the browser and the dotLRN portlet to apply changes (also includes database which stores the portlet list).

- **Unregistered Users –** This group consists of individual users of the browser system that have installed the browser catalog package, and use it for individual purposes. These group members cannot access the dotLRN portlet, they simply use the browser package as a normal OpenACS package.

2.2.2. Use Case Scenarios

Some of the most common use case scenarios are outlined below to give a general idea of the functionalities and the exact procedures that must be followed in order to achieve the required goals of the use case.

USE CASE 1	**Browse the catalog for entries**	
Goal In Context	For the user to manually browse through the catalog for links or categories.	
Preconditions	The user is an existing OACS member and is logged in to OACS.	
Success End Condition	User can view all categories, sub-categories and links of the ODP catalog hierarchy.	
Failed End Condition	User cannot view the full contents of the catalog hierarchy of the ODP data.	
Actors	Unregistered Users, Staff , Registered students, administrator, system	
Description	**Step**	**Action**
	1	The use case starts when the user visits the /browser sub-directory on his/her server.
	2	The system loads the index page with top ODP categories listed.
	3	User clicks on categories recursively until the required subcategory/link is found.
	4	The system loads all the sub-categories or links for the selected category/link and the use case ends.
Extensions	**Step**	**Branching Action**
	2a	If the index page does not load, an OACS error page is generated due to improper installation of the package. User must check the installation details.

USE CASE 2	**Add link from the catalog to the portlet list**	
Goal In Context	For the user to add a link to the portlet list.	
Preconditions	- The user is an existing dotLRN staff member of a certain community. - The portlet has been mounted into this user's community.	
Success End Condition	User successfully adds a link to the portlet list via the browser package UI and gets a confirmation.	
Failed End Condition	User does not successfully add the selected link to the portlet list, and does not get the confirmation.	
Actors	Staff , administrator, system	
Description	**Step**	**Action**
	1	The use case starts when the user browses into the correct sub-category for the link.
	2	The user clicks "add" beside the required category/link.
	3	The system checks if the link already exists in the portlet list.
	4	The system adds the link to the portlet list.
	5	The system loads a page confirming the addition of the selected link to the list and the use case ends.
Extensions	**Step**	**Branching Action**
	3a	If the link is already in the list, a warning page is loaded indicating that this link has already been added.

USE CASE 3	**View current portlet list links**	
Goal In Context	For the user to view the portlet list contents using the browser package UI	
Preconditions	- The user is an existing dotLRN staff member of a certain community. - The portlet has been mounted into this user's community.	
Success End Condition	User successfully views the current links in the portlet list.	
Failed End Condition	User does not successfully view the portlet list contents.	
Actors	Staff , administrator, system	
Description	**Step**	**Action**
	1	The use case starts when the user clicks on the "My current urls" link under the "My portlet list" menu.
	2	The system checks the current links in the list.
	3	The system loads a page with the links listed in a table, each of which has a title and comment field.
	4	The user views this list and the use case ends.
Extensions	**Step**	**Branching Action**
	2a	If the list is empty a message will appear, making it clear that no links have been added to the portlet list.

USE CASE 4	Comment on current portlet links	
Goal In Context	For the user to add a comment on a link in the portlet list.	
Preconditions	- The user is an existing dotLRN staff member of a certain community. - The portlet has been mounted into this user's community.	
Success End Condition	Comment made is viewable in the portlet list under the selected link.	
Failed End Condition	Comment is not added correctly for the selected link of the portlet list.	
Actors	Staff , administrator, system	
Description	Step	Action
	1	The use case starts when the user clicks on the "edit list" link under the "My portlet list" taskbar.
	2	The system loads the current link information that is in the list.
	3	The user selects the link to add a comment to and clicks on "edit" link beside the link title.
	4	The system loads the current title and comment fields of this link in a form that can be edited.
	5	The user types in a comment into the given form space, under the "comment" field.
	6	The user clicks "add to my personal list" to confirm entry.
	7	The system updates the link information and redirects the user back to the listing of the portlet list contents.
Extensions	Step	Branching Action
	2a	If the list is empty a message will appear, making it clear that no links have been added to the portlet list.

USE CASE 5	Delete an entry from the portlet link list	
Goal In Context	For the user to delete a link from the portlet list.	
Preconditions	The user list must contain at least one link, i.e. it can not be empty.	
Success End Condition	Selected link is successfully deleted from the portlet list.	
Failed End Condition	Selected link is not removed from the portlet list, no modifications are made to the list.	
Actors	Staff , administrator, system	
Description	Step	Action
	1	The use case starts when the user clicks on the "edit list" link under the "My portlet list" taskbar.
	2	The system loads the current link information that is in the list.
	3	The user selects the link to delete and clicks on "delete" beside it.
	4	The system loads a confirmation page asking to confirm the delete.
	5	The user clicks on the "confirm deletion" button.
	6	The system updates the list and redirects to the list page.
Extensions	Step	Branching Action
	5a	The user cancels the confirmation and the link is not deleted.

USE CASE 6	**Search the catalog for entries.**	
Goal In Context	For the user to search for entries in the catalog.	
Preconditions	The use must entry a query string in the search form.	
Success End Condition	Matches are listed, matching the entered query.	
Failed End Condition	No matches are listed for the query entered.	
Actors	Catalog Staff ,Students, unregistered users, administrator, system	
Description	**Step**	**Action**
	1	The use case starts when the user is at the main page.
	2	The user enters a query within the form text box under "Search Open Directory".
	3	The user clicks on the drop down menu and selects "all".
	4	The user clicks on the "Go" button.
	5	The system returns the matches for the entered query and the use case ends.
Extensions	**Step**	**Branching Action**
	3a	The user selects "title" to search by title.
	3b	The user selects "topic" to search by topic.
	3c	The user selects "urls" to search by url and url description.
	5a	If there are no matches in the catalog, the system generates a "sorry, no results found" page and the use case ends.

USE CASE 7	**View category description**	
Goal In Context	For the user to view the description of the current category.	
Preconditions	The user must have clicked into a subcategory at least one level deep into the "Top" categories.	
Success End Condition	Current category description is displayed successfully.	
Failed End Condition	Category description does not get displayed.	
Actors	Catalog Staff ,Students, unregistered users, administrator, system	
Description	**Step**	**Action**
	1	The use case starts when the user is at the main page.
	2	The clicks into a current category, until he/she is in the current category listing.
	3	The user clicks on the "click here for category description" link.
	4	The system loads the category description for that category.
	5	The user views the description and the use case ends.
Extensions	**Step**	**Branching Action**
	4a	If the current category has no description, the system prints out "sorry no description available for this category".

USE CASE 8	**Access the links from the portlet window**	
Goal In Context	To simply access the portlet links from the portal page.	
Preconditions	- The user must be logged in to dotLRN - The portlet must be installed in his community and added to his personalized page.	
Success End Condition	The user can access and view all the links added into the portlet list.	
Failed End Condition	The user cannot access the portlet or view the links.	
Actors	Catalog Staff ,Registered students, administrator, system	
Description	**Step**	**Action**
	1	The use case starts when the user clicks on his community link off the main dotLRN page.
	2	The system then directs the user to the "Community Home" page within the selected community portal.
	3	The user scrolls down the page and views the portlet.
	4	The user then clicks on a link to access that category/URL.
	5	The system redirects the user to the correct category within the catalog and the use case ends.
Extensions	**Step**	**Branching Action**
	5a	If the selected link is a URL, the user is directed to that webpage.

CHAPTER 3

DESIGN

3.1. System Design Overview

3.1.1. Catalog Portlet System Design Overview

The dependence on web resources within educational institutes is increasing day by day where the majority of course materials are provided online as well as the idea of online tutorials or quizzes being held for the students. Some institutes have reached the level of developing full online courses where the lectures and tutorials are held purely online in order to provide convenience for students from home. This expansion that we are talking about here is what “E-learning” is all about, and for that matter, what dotLRN is based on and trying to improve.

The browser system was designed to increase the functionality of the dotLRN platform by providing access to a central repository of the ODP catalog from the dotLRN portal where staff and students can utilise this data as an online resource for coursework as well as a form of collaboration to increase E-learning within a course or a unit of study in a school/university.

In order to provide such functionality the system had to be designed in such a way so as to conceal the structure complexities of such a large catalog and only portray what the user wanted to see, or what the user could use as a resource. Hence, users of the browser system can now without having any prior knowledge of internal system components use a web server to access this web based application with a simple, easy to use User Interface very similar to many directory project websites available such as Yahoo and DMOZ. Furthermore, staff members of the online E-learning community can not only browse the catalog but can also personalize links that are viewable for their students by adding them and their corresponding comments to a portlet list that will be displayed in the browser portlet. All of this is achieved without any html coding of any kind by the staff member or any form of technical burden which was the case to insert web resources into dotLRN prior to the browser system.

From here, once the browser portlet is accessed the appropriate database information is read by the portlet to display the current portlet list which will become accessible by the students or other portal members. If the link in the portlet is a http link then the student will be automatically redirected to that web page upon clicking on the link. On the other hand, if the link in the portlet is a category within the ODP data the student will be directed to that particular category listing within the browser package (UI).

Although it may appear that by redirecting the user back into the browser portlet we are opening paths for students to access and misuse permissions to add links to the portlet, it is quite the opposite as one of the great advantages of the OpenACS framework is its permissions system which allows the personalization of the user interface of the package according to the set permissions for the current user. Thus, staff admin operations are automatically hidden from the user interface if the user is a student (or does not have admin rights).

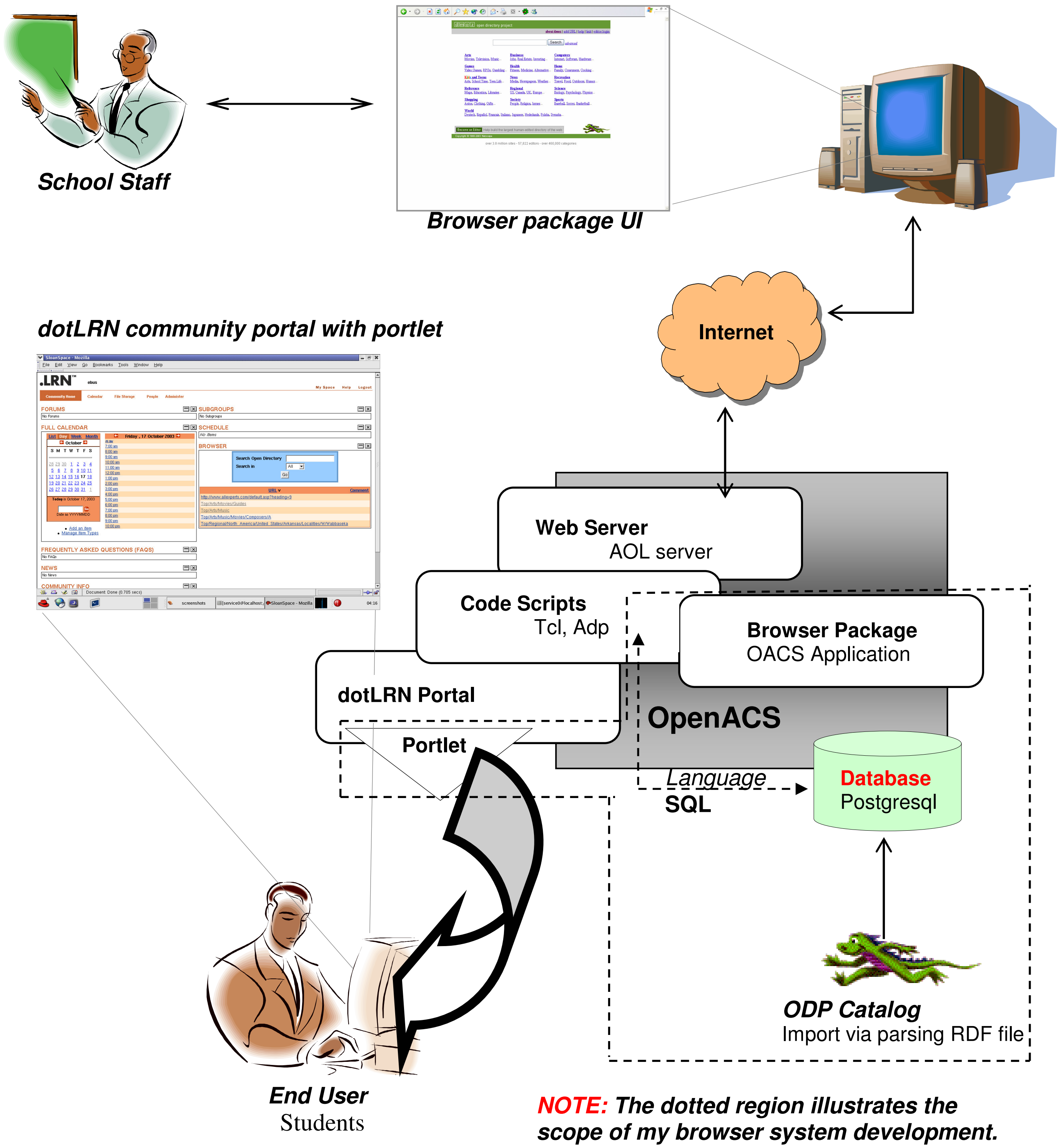
Apart from the advantage of reusability and extensibility of such framework aspects there was another contributing factor that pushed towards an open source solution for the system, the cost. Due to the large costs of many existing E-learning platforms, the browser system was designed on dotLRN and OACS in order to provide a free solution to the problem of adding web resources to an E-learning portal. This would allow a more significant number of institutes to implement this new system under an existing open source platform without any financial hurdles that can range up to hundreds of thousands of dollars per institute.

As mentioned in Chapter 1 the OACS platform is based on 3 main software packages, they are AOLServer, PostgreSQL (a RDBMS) and TCL. The first of which is the web/application server that will host the browser system and process requests from the user's web browser before they are passed into the system to perform the required transaction. The second "PostgreSQL" is what was used to access and maintain the ODP data which was stored with an appropriate data model into this database. The last "TCL" is the scripting language that implements all the functions of the browser system with its own method of handling database calls and html output (discussed further in chapter 4).

This system layout is shown in figure 2 below alongside all of the necessary entities involved and the way in which they communicate with each other.

3.1.2. Catalog Portlet System Layout Overview

Figure 3: An overview of the catalog portlet System Layout



3.1.3. Baseline Architecture

Throughout the development of this browser system, out of the main requirements listed in chapter 2, the development was successful in including all of the baseline functionality which summed together to provide a professional and versatile application both in dotLRN (portlet package) and in OACS (Browser package). These main functionalities provided the following features:

- The ability of importing current ODP catalogs into the system.
- The ability of importing future ODP catalogs into the system.
- The ability of browsing the entire hierarchy of the catalog via a simple UI.
- The ability of retrieving information about certain categories/links in the catalog.
- The ability to add/edit/delete links to/from a personalized portlet list that will appear in the portlet.
- The ability of creating the portlet with the expected operations such as adding and removing to/from a community as well as maximizing and minimizing.
- The ability of administrating the portlet contents by appropriate staff members.
- The ability of student users to access links/categories links from the portlet window.

Once these mentioned baseline features were fulfilled, the development continued in implementing additional features. These features included:

- The implementation of a search engine that allowed searches to be made in categories, sub-categories and links of the catalog via different attributes.
- The ability to allow the search engine to be queried from the portlet.
- The ability to allow custom URL links to be added to the system.
- The ability to add comments to the links in the portlet list.

Due to the time constraints of approximately 12 weeks some of the extra functionality was not implemented, these are listed below:

- Implement advanced searching algorithms.
- Send notifications to students about updates of the portlet contents.
- Allow editors to register and help organise a selected category to increase the catalog.

3.2. Data Model Discussion

The data model represents the way in which the browser system was stored in the OACS database (in PostgreSQL), this includes tables , indexes and key relations all of which sum up to determine how the data of the catalog and the custom list were accessed, manipulated and stored internally. It is clear that this plays a major role in determining the efficiency and reliability of the entire system since it is the pivot point of serving the users with timely operations, thus the careful design of this model was absolutely critical since any unnecessary fields or tables could cause major delays or issues due to the large amount of data being held which is over two gigabytes in size (DB size). At the same time the data model had to be kept simple in order to allow additional work in the future and not confuse developers with unnecessary fields/tables.

This section will be devoted to the explanation of the data models that were reviewed as well as the final one that was implemented for the browser system. To be exact there were two main data models, the first of which was the previous catalog system DM that was implemented by the two students mentioned in the historical considerations Mitsunori and Roger from the USYD, the second is the one that was implemented as the best for this thesis study in terms of the browser system goals. Both of these will be mentioned in the next two sections and then a final discussion on the decisions made will be elaborated on in order to justify these decisions in section 3.2.3.

3.2.1. Previous Catalog System Data Model

This section will provide a quick summary on the data model that was implemented in the study conducted by these two students last year. The summary will mention the tables that were used as well as their properties and purpose within the bibliographic catalog system [8]. The data model consisted of 8 main tables:

1. Categories Table

This table holds all the categories that are available within the bibliographic system it holds the category id, name, description as well as the creation date.

```
create table cat_categories (
        category_id         integer references acs_objects(object_id) primary key,
        category_name       varchar(100),
        description         text,
        creation_date       timestamp
);
```

2. Categories map Table

This table is used to build a recursive tree of categories by using the cat_categories table. It stores the category id, a parent_id which maintains a link to the category's parent, and a child_id which maintains a relationship with its children.

```
create table cat_categories_map (
        category_id     integer not null references cat_categories(category_id),
        parent_id       integer not null,
        child_id        integer not null
);
```

3. Categories entry types table

This table is used to store new types of references such as CD's, books, URL's and any other new category to the system. It stores the unique id of the new category, its title as well as its description.

```
create table cat_entry_types (
        type_id         integer references acs_objects(object_id) primary key,
        title           varchar(100) not null, --name of particular type of reference e.g. book, article etc.
        description     text
);
```

4. Category Entry References table

This table is used to hold all unique references within the catalog, the references here are similar to url's in the ODP catalog where each row of this table represents a new reference in the system. The table stores entry_id the unique id of the reference entered into the catalog, its title, author as well as its description.

```
create table cat_entry_references (
        entry_id            integer references acs_objects(object_id) primary key,
        entry_title         varchar(100) not null,
        entry_author        varchar(100) not null,
        description         text
);
```

5. Category Entry map table

This table is used to maintain a mapping between each unique reference entry with its respective category and type. This table stores the category_id of the entry's category as well as the entry_id for the reference in the cat_entry_references table and its type_id which represents a unique id for the type of entry in the cat_entry_types table.

```
create table cat_entries_map (
        category_id        integer not null references cat_categories(category_id),
        entry_id           integer not null references cat_entry_references(entry_id),
        type_id            integer not null references cat_entry_types(type_id)
);
```

6. Category field types table

This table maintains a list of all the different fields of information possible. An example of this would be that a book may have an author and publisher while a CD may have a producer and composer. For each type a new field will be created and then stored in this table. It stores the field_id which is a unique id representing the field type and the field_name which is the name of the field.

```
create table cat_field_types (
        field_id        integer not null primary key,
        field_name      varchar(100)
);
```

7. Category fields map table

This table maps an entry to all the corresponding fields that it has in its reference where each field that the entry has would require a new row. An example would be if a CD had a producer, artist and composer it would occupy 3 rows within this table. This table stores the unique entry_id which points to a unique entry that is being stored in the database, also field_id which is the unique type of field that the entry contains and field_description which represents the entry's particular details for that field.

```
create table cat_fields_map (
        entry_id            integer not null references cat_entry_references(entry_id),
        field_id            integer not null references cat_field_types(field_id),
        field_description   text
);
```

8. Category type fields map table

This table is used for the correlation of a type and the fields that can be selected for a particular entry if it were a specific type. It stores the type_id which references that of the cat_entry_types table that is it points to a particular type to which the field belongs to.

```
create table cat_type_fields_map (
            type_id             integer not null references cat_entry_types(type_id),
            field_id integer not null references cat_field_types(field_id)
);
```

Below is the original entity-relationship diagram for the bibliographic catalog.

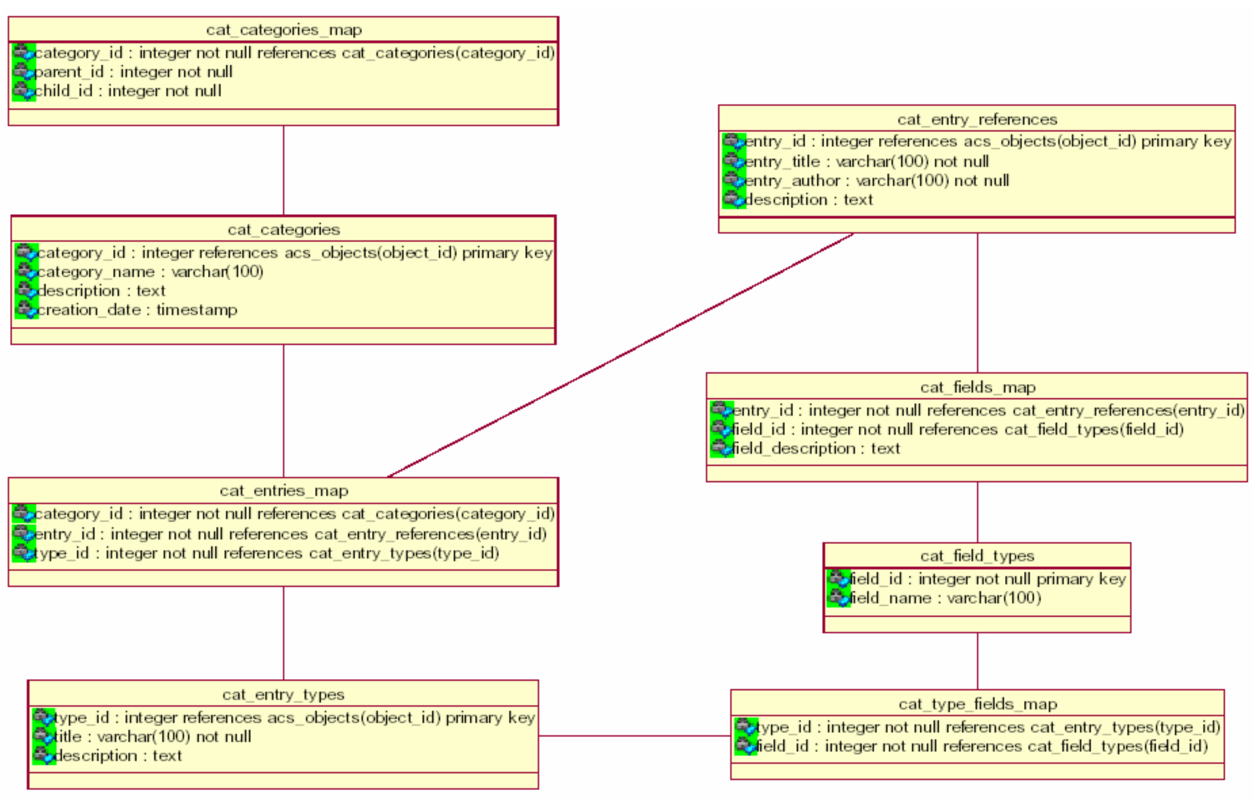

cat_categories_map
category_id : integer not null references cat_categories(category_id)
parent_id : integer not null
child_id : integer not null
cat_entry_references
entry_id : integer references acs_objects(object_id) primary key
entry_title : varchar(100) not null
entry_author : varchar(100) not null
description : text
cat_categories
category_id : integer references acs_objects(object_id) primary key
category_name : varchar(100)
description : text
creation_date : timestamp
cat_fields_map
entry_id : integer not null references cat_entry_references(entry_id)
field_id : integer not null references cat_field_types(field_id)
field_description : text
cat_entries_map
category_id : integer not null references cat_categories(category_id)
entry_id : integer not null references cat_entry_references(entry_id)
type_id : integer not null references cat_entry_types(type_id)
cat_field_types
field_id : integer not null primary key
field_name : varchar(100)
cat_entry_types
type_id : integer references acs_objects(object_id) primary key
title : varchar(100) not null
description : text
cat_type_fields_map
type_id : integer not null references cat_entry_types(type_id)
field_id : integer not null references cat_field_types(field_id)

Figure 4: The entity-relationship diagram for the bibliographic catalog data model [8]

3.2.2. Implemented Browser System Data Model

Following on from our Data model discussion this section will outline the data model that was implemented for the browser system. The data model contains four main tables, three of which were required to hold the ODP catalog including its hierarchy and contents, the final table was a single table implemented to store the portlet list contents.

Please Note: *A better understanding of the data model storing the ODP data will be clearer after reading chapter 4 since it will discuss in detail how the tables were used while the importing of the ODP catalog took place, for now an outline of the data model is sufficient.*

1. The Structure Table

This table is the first of the 3 main tables that hold the ODP catalog. It represents the structure for all the categories, in other words it contains the skeleton of the catalog holding the most complex part of the ODP data - its hierarchy, containing over 460 000 categories. You will note that the table holds catid which is the unique primary key for each catalog in the directory, it also stores the topic which is the full name or path for the category within the hierarchy such as “Top/Arts”, next we have the title field which is just the name of the category without the path, for example “Arts”. The description field contains a brief description of the content of this category, the final field in the table lastupdate is just a timestamp of the last date and time the catalog was edited by the current online ODP editors.

```
CREATE TABLE "structure" (
          catid INT NOT NULL DEFAULT '0' PRIMARY KEY,
          topic TEXT NOT NULL,
          title TEXT,
          description TEXT,
          lastupdate TIMESTAMP
);
```

2. The Resources Table

The resources table is the second table for the ODP data which stores each category’s relations with other categories/sub-categories. Due to the large number of categories within the ODP data , the relations between the categories are very important in allowing the complete retrieval of relevant resources for queries on the information. Hence this table manages all the types of relations for each category or subcategory based on the relation types within the RDF format (discussed later). It stores the catid which is the id of the category obviously this is not unique since each category in the structure table can have multiple subcategories within this table that have the same id. The other two fields contain the rtype which is the name of the relation type with the resource field , this resource field is the path for the subcategory that the current relation is based on.

```
CREATE TABLE "resources" (
          catid int NOT NULL DEFAULT '0',
          rtype TEXT,
          resource TEXT
);
```

3.The URL's Table

The third table that aids the storing of the ODP data is the xurls table. This is responsible for holding the "content" of the catalog, as in it is responsible for filling in the skeleton that is set by the "structure" table. Each field in this table is a url, it is represented by a catid which is the category id of the category this URL belongs to, we then have the priority field which is an integer representing the priority of the url, the age field stores a description of the age group expected for the viewing of this site , the mediadate field only appears if a certain link contains an article in a news paper, the url field is the actual http address to this website , the title is the name of the category it belongs to and finally the description field is the description of the content available on this URL.

```
CREATE TABLE "xurls" (
        catid INT,
        priority INT,
        ages VARCHAR(32),
        mediadate VARCHAR(32),
        url VARCHAR(1024),
        title VARCHAR(1024),
        description VARCHAR(8192)
);
```

4.The browserurl table

This table although part of the data model, is not related to the previous 3 tables, rather it is a single table that is used to store the links (to a URL or category) that are selected by the user to appear in the portlet window. The table stores three main fields about each link , url_id is the unique id for each link added into the portlet list , the title field carries the actual http link or path into a category while the body carries the comment that can be added to each link as an optional field to appear in the portlet. You will notice that the url_id references the acs_objects table , this is essential in allowing each link in the portlet window to be stored as an object – since this allows the gaining of the permissions model provided by the OACS architecture.

```
create table "browserurl" (
        url_id      integer     constraint browserurl_fk
                                 references acs_objects(object_id)
                                 constraint browserurl_pk
                                 primary key,
        title       varchar(255)
                                  constraint browserurl_title_nn
                                   not null,
        body         varchar(1024)
);
```

Below is the entity relationship diagram for the data model implemented for the browser system, all relations are shown as well as table attributes. You will note that as explained above the browserurl table has a separate concern to the tables that store the ODP data and hence is not related to them.

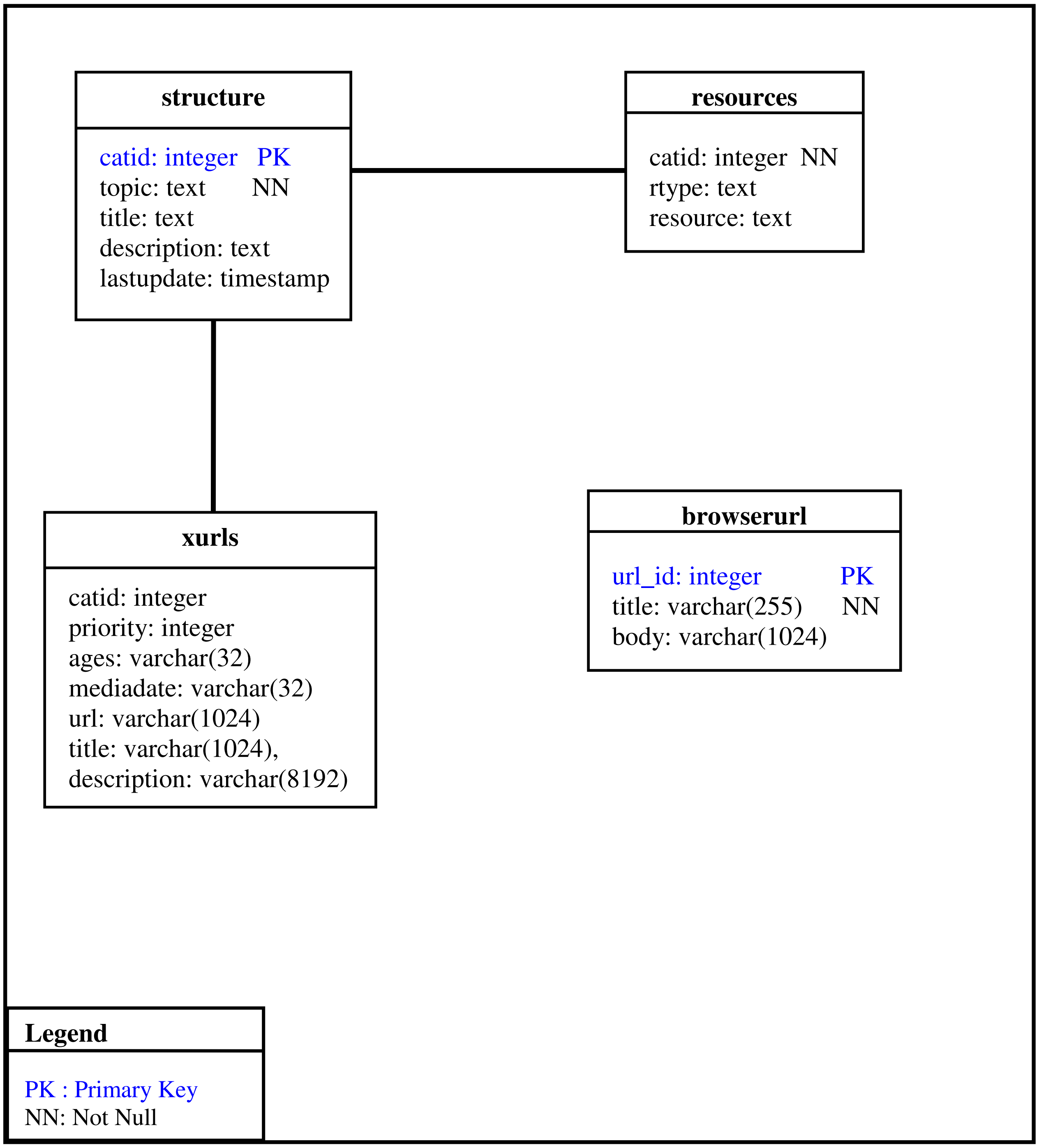


Figure 5: The entity-relationship diagram for the browser system

3.2.3. Data Model Comparisons

In the previous two sections both data models were outlined, originally the idea of this thesis study was to continue on and import the ODP catalog into the data model that was developed by the two students from the WEG group last year. Although after the careful assessment of this data model it was not too long before it was decided that that this would cause a great delay in the development of the browser system as many amendments would have had to be made since the original catalog had different goals to reach apart from its general approach to the importing of data which was not adhering to the standards of ODP dumps i.e. RDF format. To be more specific these problems along with other reasons for the selection of the second data model to be implemented are listed below:

- First of all, the students worked on a data model that was developed before the input format of their import function was decided upon, this is one major flaw since it forced them to base the import file format according to their data model which limited the type of file formats that could be imported and did not take into account any popular standard file formats such as RDF, hence automatically disallowing the import of any large catalogs available on the web since the majority of them are based on these standards.

- For the goals of the browser package the Bibliographic catalog system's DM was unnecessarily complex and could have been simplified much further, for example, the cat_entry_types table was not needed since this could have been modeled as an extra field within the categories table and served the same purpose, in fact this is what was used in the browser system DM since it is more efficient. As for the first data model each new category would have a new row in this table and with over 460 000 categories in the ODP catalog, this would create a large table structure that was not needed. Instead the browser package defines the category type by a unique string field name in the structure table that allows the RDF to do the work in terms of classifying all the different types of categories within the data. This same problem also applies to some of the other tables that were used in the bibliographic system such as cat_entries_map which maps each single reference to its details, which again is unnecessary as the detail fields could have just been added as fields in the table that holds the references that is the cat_entry_references. This method was used throughout the model , unnecessary tables being created to store information that could have been represented via text fields in another table, hence in terms of the browser package it could not afford to store such tables to carry irrelevant data since the 4 million URL links and 460 000 categories would have caused over 4.5 million fields in tables that were unnecessary – and this is not taking into account the growth of the catalog which would increase a few hundred thousand URL's/categories per year. All of this would have caused greater overhead time for the retrieval of data from the catalog and hence was not included in the final DM.

- Apart from this it can be noted that another problem existed, the tables in the Bibliographic DM were also linked to the acs_objects table for cat_entry_references, cat_categories and cat_entry_types, which caused every single reference/category in the catalog to be an object, again for the browser system the creation of over 4 million objects would have been very inefficient, in fact it would have been a critical issue in terms of retrieval since object creation can be very costly with such large numbers (storage space) which would not only slow down the retrieval of the browser catalog data but also the entire OACS database as each object would require an entry in the acs_objects table.

To sum up, we can see that although the first data model operated smoothly for the bibliographic system, it would have caused an increase in inefficiency for the browser system, mainly due to the fact that originally this DM was not expected to hold such large amounts of data (approximately 2 GB of DB space) and hence the DM did not cater for such future ideas. Although the creation of objects for each reference/category stored would have seemed to be a good idea at the start in terms of inheriting certain reusability features of the OACS framework such as the permissions system, it would have been the major downfall of the browser system in terms of efficiency. Furthermore, another note that must be made is that the direct permissions system on different categories within the catalog is unnecessary as the ODP is public to all, but what had to be controlled was the browserurl table which stored the links to be displayed in the portlet and hence this is where the OACS permissions system was applied directly (as well as the UI) in order to control the rights for different users such as the staff and students.

3.3. User Interface layout and Design

This section will concentrate on the visual design aspects of the browser system that were considered mainly in the browser package User Interface but also on how the catalog portlet was to appear within the dotLRN portal.

3.3.1. Catalog Browser Package UI

The main idea behind the selected design for layout of the interface of the portlet browser package is based on the OpenACS template system. This has become one of the significant factors in increasing the popularity of OpenACS in regards to web development, since it allows a quick and efficient method of creating an elegant user interface for a website.

The general concept of the templating system is to help in the "separation of concerns" that is, it allows the business logic of the application to be separated from the presentation layer of the website in order to allow both programmers and web designers to work simultaneously without any problematic compatibility issues.

The system allows this by splitting the work into two different files, the first being the .tcl file which is responsible for the business logic i.e. performing database queries and setting particular variables while the .adp file acts as a companion to the .tcl file by filling in the presentation details of the web page. This .adp file is called from the .tcl script (in the last line) and contains mainly HTML code for the webpage alongside some calls to variables from the tcl file.

The final point to note about this templating system is that it avoids the compatibility issues by setting up a "page contract", this is basically a list of HTTP parameters that the .tcl file expects to receive and then a list of tcl variables (data sources) that will be available in the .adp page in its display [9]. A typical template layout of an OpenACS webpage is listed below in figure 6 below.

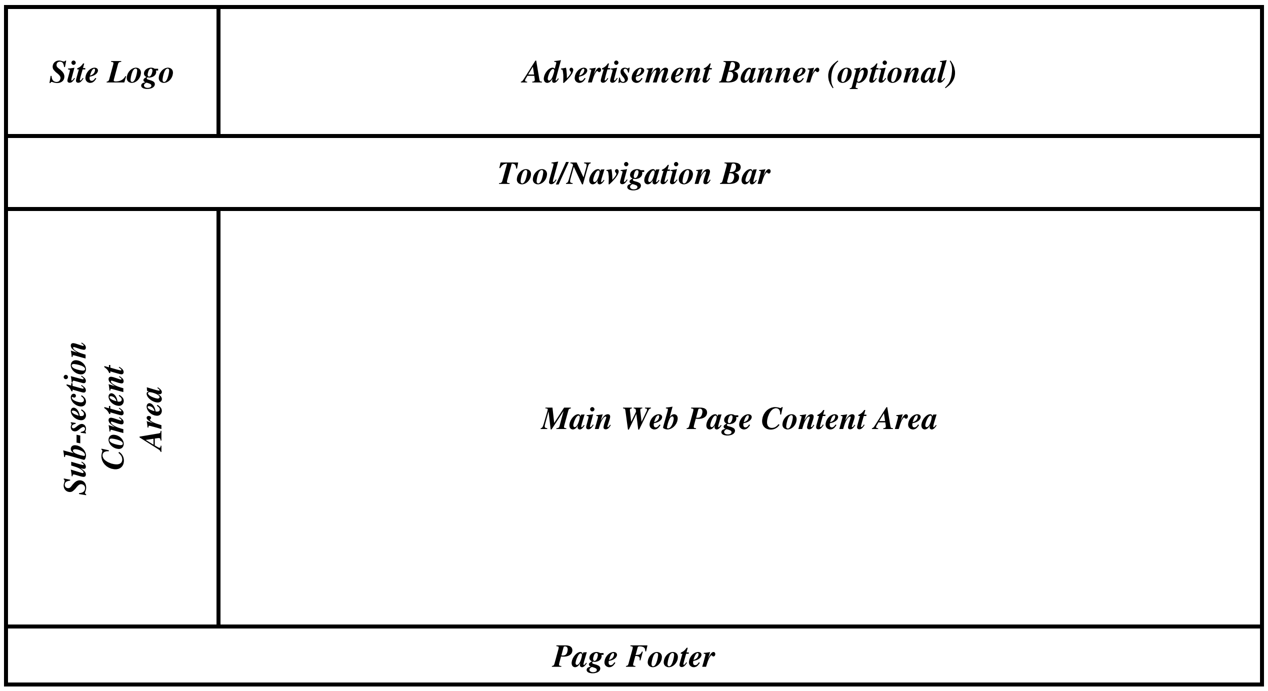
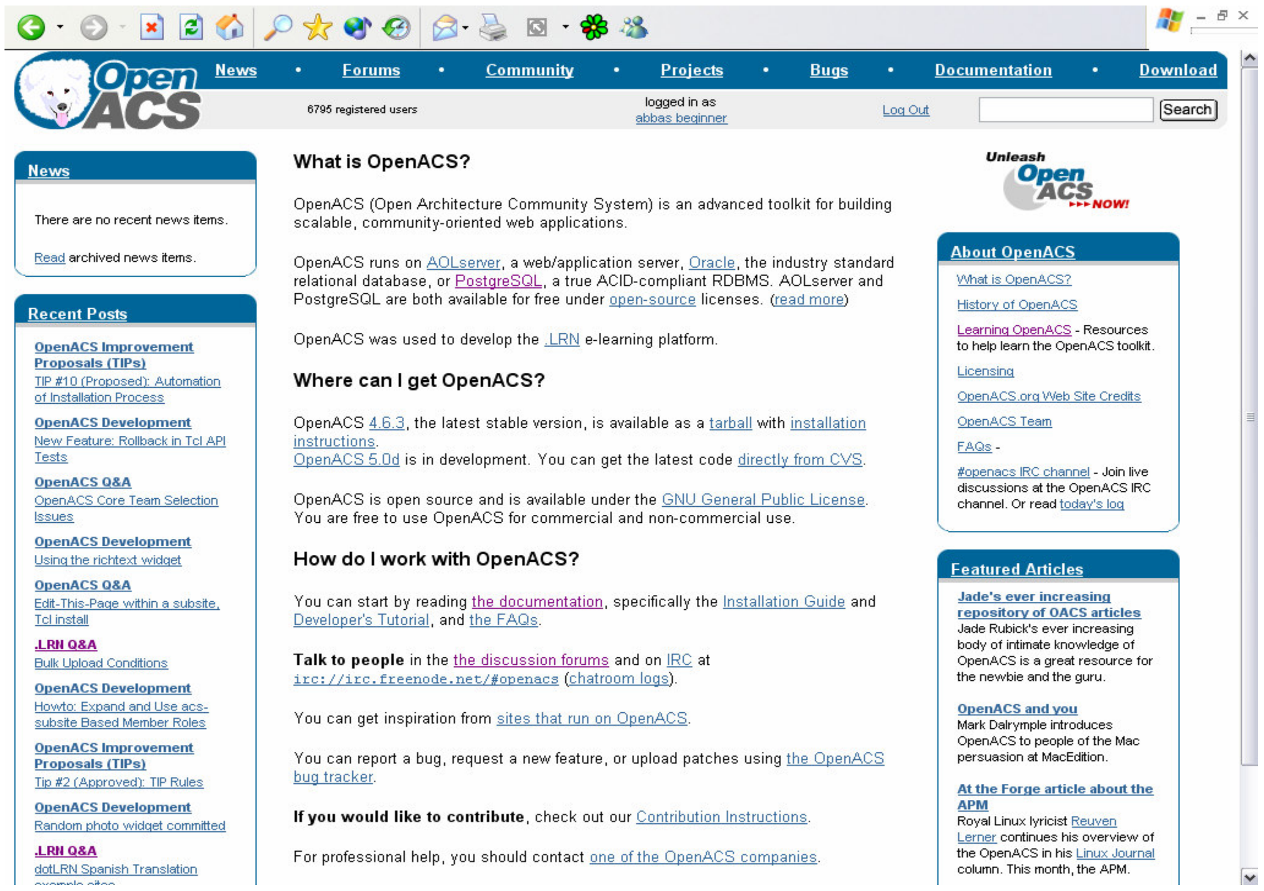


Figure 6: Typical webpage layout corresponding to the OpenACS template system


News • Forums • Community • Projects • Bugs • Documentation • Download
Search
Unleash OpenACS NOW!

6795 registered users

logged in as abbas beginner

Log Out

News

There are no recent news items.

Read archived news items.

Recent Posts

OpenACS Improvement Proposals (TIPs)
TIP #10 (Proposed): Automation of Installation Process

OpenACS Development
New Feature: Rollback in Tcl API Tests

OpenACS Q&A
OpenACS Core Team Selection Issues

OpenACS Development
Using the richtext widget

OpenACS Q&A
Edit-This-Page within a subsite, Tcl install

.LRN Q&A
Bulk Upload Conditions

OpenACS Development
Howto: Expand and Use acs-subsite Based Member Roles

OpenACS Improvement Proposals (TIPs)
Tip #2 (Approved): TIP Rules

OpenACS Development
Random photo widget committed

.LRN Q&A
dotLRN Spanish Translation

What is OpenACS?

OpenACS (Open Architecture Community System) is an advanced toolkit for building scalable, community-oriented web applications.

OpenACS runs on AOLserver, a web/application server, Oracle, the industry standard relational database, or PostgreSQL, a true ACID-compliant RDBMS. AOLserver and PostgreSQL are both available for free under open-source licenses. (read more)

OpenACS was used to develop the .LRN e-learning platform.

Where can I get OpenACS?

OpenACS 4.6.3, the latest stable version, is available as a tarball with installation instructions.
OpenACS 5.0d is in development. You can get the latest code directly from CVS.

OpenACS is open source and is available under the GNU General Public License. You are free to use OpenACS for commercial and non-commercial use.

How do I work with OpenACS?

You can start by reading the documentation, specifically the Installation Guide and Developer's Tutorial, and the FAQs.

Talk to people in the the discussion forums and on IRC at `irc://irc.freenode.net/#openacs` (chatroom logs).

You can get inspiration from sites that run on OpenACS.

You can report a bug, request a new feature, or upload patches using the OpenACS bug tracker.

If you would like to contribute, check out our Contribution Instructions.

For professional help, you should contact one of the OpenACS companies.

About OpenACS

What is OpenACS?

History of OpenACS

Learning OpenACS - Resources to help learn the OpenACS toolkit.

Licensing

OpenACS.org Web Site Credits

OpenACS Team

FAQs -

#openacs IRC channel - Join live discussions at the OpenACS IRC channel. Or read today's log

Featured Articles

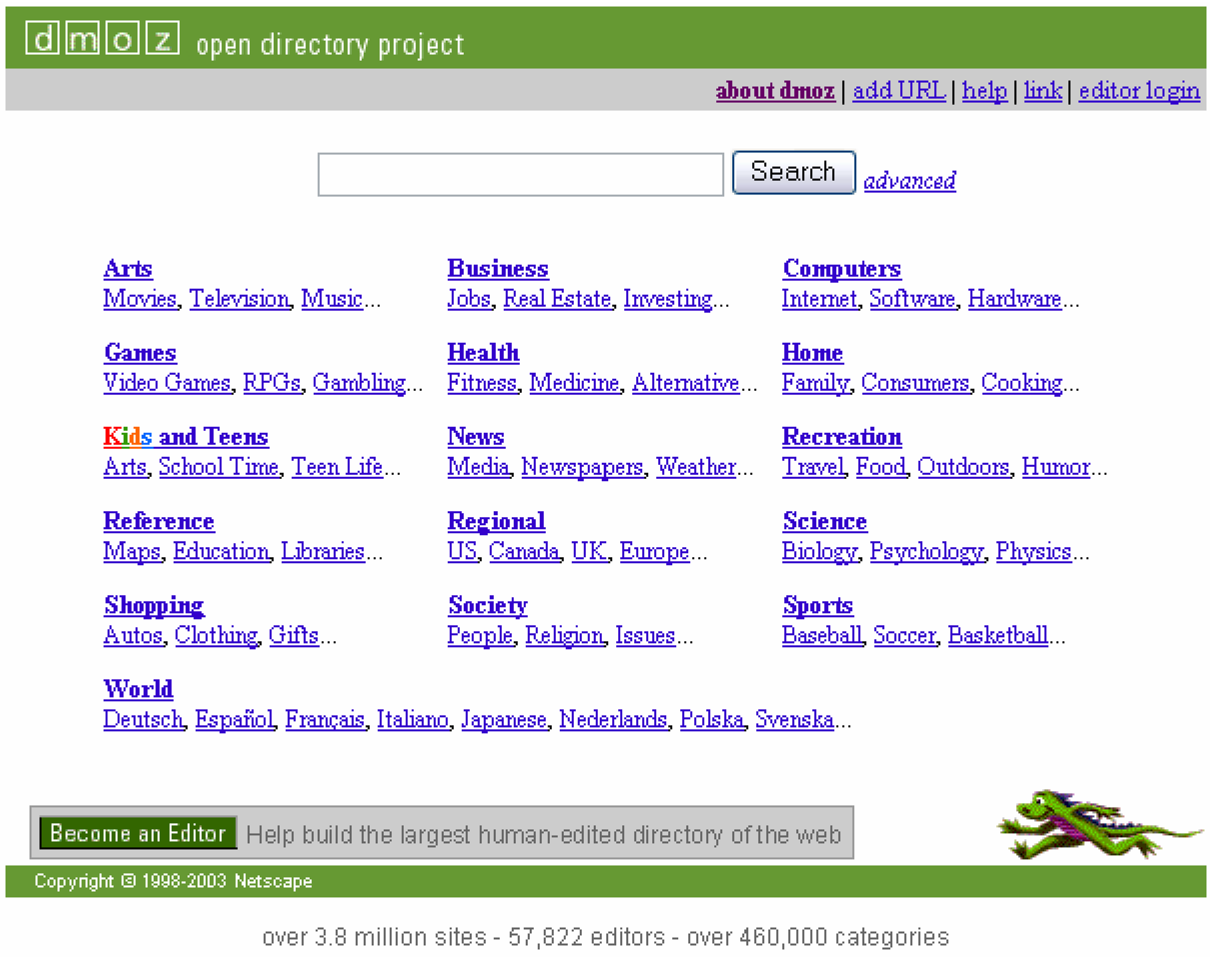
Jade's ever increasing repository of OACS articles
Jade Rubick's ever increasing body of intimate knowledge of OpenACS is a great resource for the newbie and the guru.

OpenACS and you
Mark Dalrymple introduces OpenACS to people of the Mac persuasion at MacEdition.

At the Forge article about the APM
Royal Linux lyricist Reuven Lerner continues his overview of the OpenACS in his Linux Journal column. This month, the APM.

Figure 7: Real life example of an OpenACS website and User Interface

The original proposed Browser package design appeared fairly similar in structure to the template diagram of figure 6 above and still had the same properties such as the navigation bar, main content, logo as well as the footer. The main idea for the content section of the interface was gathered from the www.dmoz.org website shown in figure 8 below. This was chosen since it had a simple and clear interface and because it displays the same catalog data that is to be available in the browser application's content area.


dmoz open directory project
about dmoz | add URL | help | link | editor login
Search
advanced
Become an Editor

Arts Movies, Television, Music...
Business Jobs, Real Estate, Investing...
Computers Internet, Software, Hardware...
Games Video Games, RPGs, Gambling...
Health Fitness, Medicine, Alternative...
Home Family, Consumers, Cooking...
Kids and Teens Arts, School Time, Teen Life...
News Media, Newspapers, Weather...
Recreation Travel, Food, Outdoors, Humor...
Reference Maps, Education, Libraries...
Regional US, Canada, UK, Europe...
Science Biology, Psychology, Physics...
Shopping Autos, Clothing, Gifts...
Society People, Religion, Issues...
Sports Baseball, Soccer, Basketball...
World Deutsch, Español, Français, Italiano, Japanese, Nederlands, Polska, Svenska...

Help build the largest human-edited directory of the web

Copyright © 1998-2003 Netscape

over 3.8 million sites - 57,822 editors - over 460,000 categories

Figure 8: DMOZ User Interface of web catalog

Below is a diagram of the original proposed browser UI layout design with the DMOZ website content within the content area.

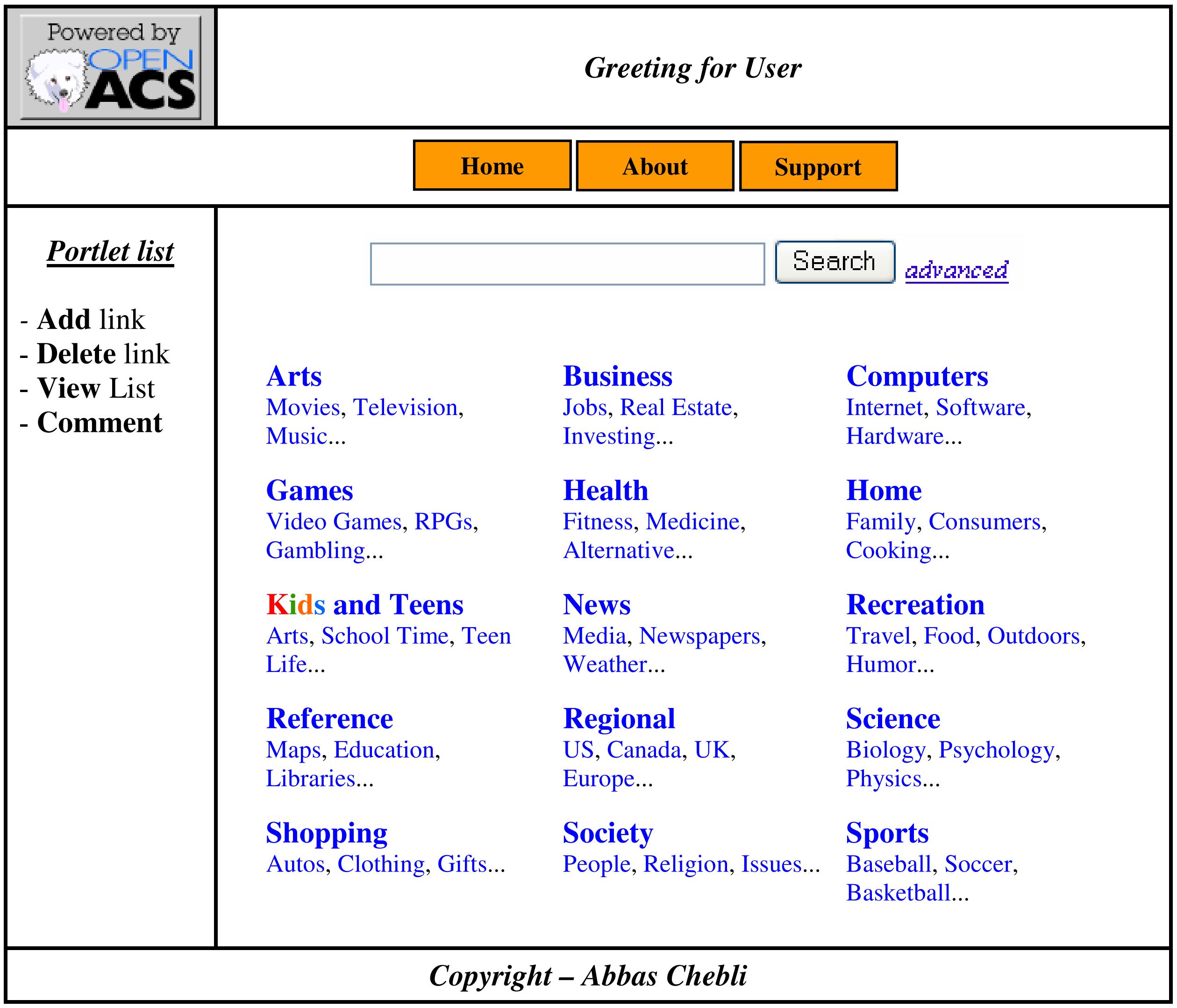
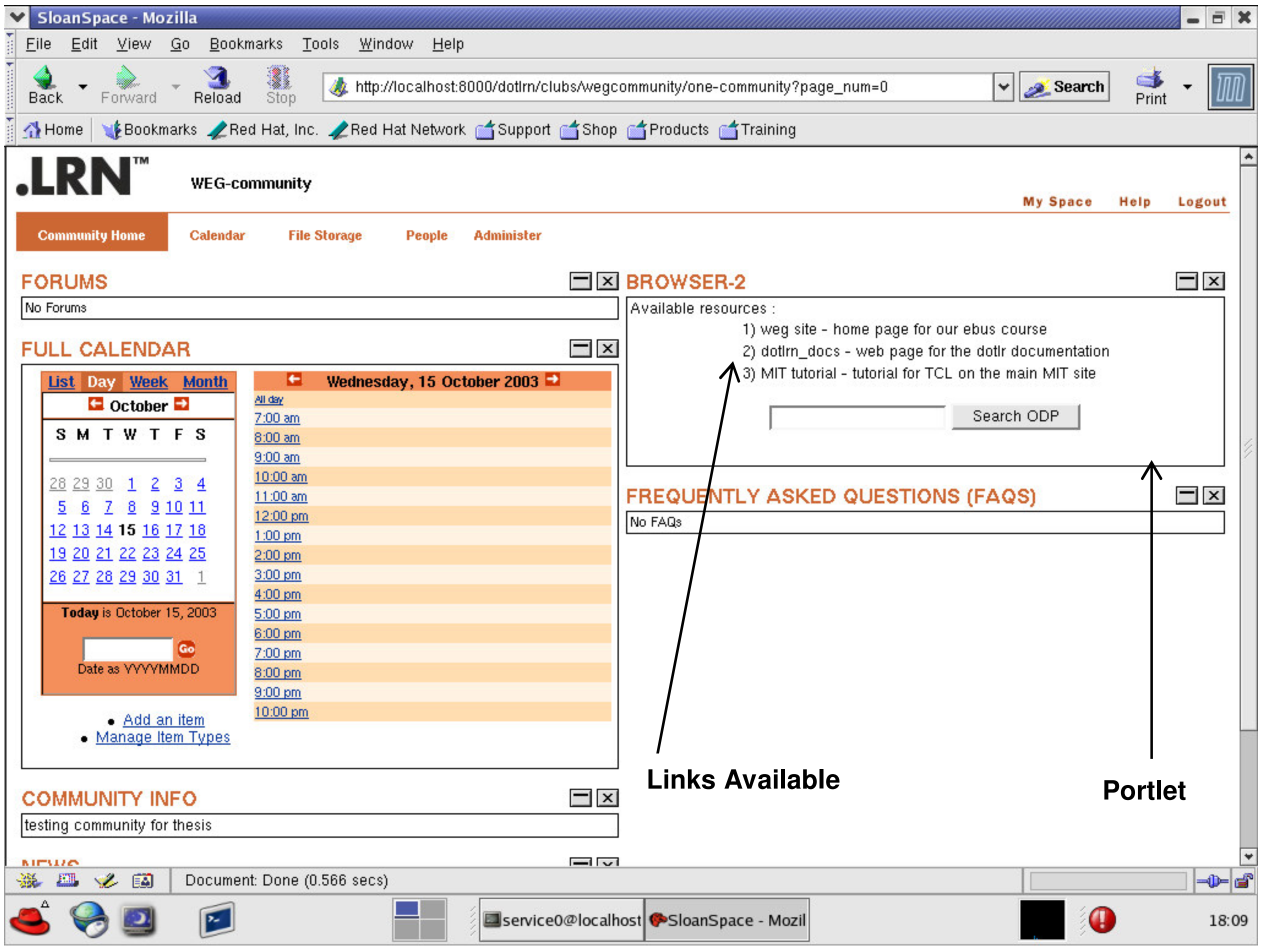

Powered by OPEN ACS
Greeting for User
Home
About
Support
Portlet list
- Add link
- Delete link
- View List
- Comment
Search
advanced
Arts
Movies, Television, Music...
Business
Jobs, Real Estate, Investing...
Computers
Internet, Software, Hardware...
Games
Video Games, RPGs, Gambling...
Health
Fitness, Medicine, Alternative...
Home
Family, Consumers, Cooking...
Kids and Teens
Arts, School Time, Teen Life...
News
Media, Newspapers, Weather...
Recreation
Travel, Food, Outdoors, Humor...
Reference
Maps, Education, Libraries...
Regional
US, Canada, UK, Europe...
Science
Biology, Psychology, Physics...
Shopping
Autos, Clothing, Gifts...
Society
People, Religion, Issues...
Sports
Baseball, Soccer, Basketball...
Copyright – Abbas Chebli

Figure 9: The original proposed browser package UI design

3.3.2. Catalog Portlet Interface

The catalog portlet User interface also adheres to the template system for OACS that was discussed in the previous section, the only difference here is that the browser portlet UI appears smaller since it is constricted to a certain small window within the dotLRN portal. The portlet's original proposed UI was designed accordingly to the portlet window views already available in the dotLRN community for many other packages such as Calendar, homework and news. In figure 10 below is the picture of the original proposed UI design for the portlet.



No Forums

Available resources :
1) weg site - home page for our ebus course
2) dotlrn_docs - web page for the dotlr documentation
3) MIT tutorial - tutorial for TCL on the main MIT site

No FAQs

Today is October 15, 2003

testing community for thesis

Document: Done (0.566 secs)

Figure 10: The original proposed browser portlet UI design for dotLRN

CHAPTER 4

IMPLEMENTATION

4.1. The Importing of ODP into PostGRESQL

The first stage of the implementation phase was attempting to import the ODP hierarchy and content into the designed data model for the browser package that was discussed in section 3.2.2. For the sake of clarity this section will not discuss the exact steps taken to import the catalog into PostgreSQL as the instructions have been attached to Appendix B, but rather it will explain how the data was parsed accordingly to the standards of the RDF and how this data was then coordinated with the data model for PostgreSQL. This section will give a brief definition of the RDF standard before delving into its complex tag structure in order to supply sufficient knowledge in understanding the following sections of the implementation process.

4.1.1. RDF and the ODP Import files

The Resource Description Framework is a foundation for processing metadata and is the source that provides interoperability between applications that exchange information on the web [10]. RDF can and has been used in a variety of applications but with respect to this thesis study its main use was in terms of cataloguing since it is the standard by which the ODP data is “dumped” in. What is meant here is that sites like the popular dmoz.org provide updates for their ODP catalog weekly by exporting it into an RDF file format which is essentially a type of XML file. This is essentially where the original information of the ODP directory can be accessed from since it is a source free to all.

The ODP data is dumped into two main files, the first of which is the structure.rdf (approx 500 MG) file which is a dump of the category hierarchy (skeleton) for the entire catalog. The second file called content.rdf (approx 1.2 GB) contains the catalog content mainly consisting of URL’s and other resources available on the web and their descriptions each of which is categorised into one of the categories in the structure hierarchy. In the two figures below are small fragments of the structure.rdf file and content.rdf file , although they only show one or two different tags , they are listed in order to give an idea of the structure that the RDF dumps follow for the ODP catalog.

```
  - <Topic r:id="Top">
      <catid>1</catid>
      <d:Title>Top</d:Title>
      <lastUpdate>2003-08-10 12:35:01</lastUpdate>
      <narrow r:resource="Top/Arts" />
      <narrow r:resource="Top/Shopping" />
      <narrow r:resource="Top/Science" />
      <narrow r:resource="Top/Games" />
      <narrow r:resource="Top/Business" />
      <narrow r:resource="Top/Computers" />
      <narrow r:resource="Top/Health" />
      <narrow r:resource="Top/Sports" />
      <narrow r:resource="Top/World" />
      <narrow r:resource="Top/Test" />
    </Topic>
```

Figure 11: A fragment of the structure.rdf file listing some of the "Top" categories.

```
<Topic r:id="Top/Home">
  <tag catid="7"/>
  <d:Title>Home</d:Title>
  <link r:resource="http://www.homeideas.com/"/>
  <link r:resource="http://www.housenet.com/"/>
  <link r:resource="http://members.aol.com/yodapro/index.html"/>
  <link r:resource="http://www.marthastewart.com"/>
  <link r:resource="http://www.mfgsupply.com"/>
  <link r:resource="http://www.bccondo.com"/>
  <link r:resource="http://www.contractornet.com"/>
  <link r:resource="http://www.kcweb.com/fl/freelance.htm"/>
</Topic>

<ExternalPage about="http://www.homeideas.com/">
  <d:Title>Home Ideas: Home Improvement Ideas for Kitchen, Bath, Yard and
Garden, etc.</d:Title>
  <d:Description>The ultimate resource for home projects. Research projects
using past issues of Today's Homeowner magazine, request free product
literature, and link to over 700 industry websites.</d:Description>
</ExternalPage>
```

Figure 12: A fragment of the content.rdf file listing some URL's for the Home category

4.1.2. Parsing the RDF Files and Tags

The parsing of the two files that were discussed in the previous section was executed by two perl scripts (included with the attached interactive CD), these scripts simply cleaned out all the tags from the RDF files and stored the appropriate fields of the categories and links in the data model for the browser system. More specifically the structure file was parsed and stored into the “structure” and “resources” table since this file contains the relations for each category as well as the category structure/hierarchy, while the content file was parsed and stored into the “xurls” table listing each entry as a single URL. It must be noted that the RDF structure contains many tags within each dumped file, but only some of these tags are necessary depending on the scope they are being applied to. The first file used for the browser package “structure.rdf” contained the tags listed in figure 13 below although only the shaded tags were needed to be stored into the browser DM.

Alias	-
Target	-
Topic	Represents the topic field in the structure table.
altlang	* a category that is in another language
catid	Represents catid field in structure table.
d:Description	Represents description field in structure table.
d:Title	Represents title field in structure table.
d:charset	-
editor	-
lastUpdate	Represents lastupdate field in structure table.
letterbar	* category that is categorised in letters or numbers i.e. A-Z or 0-1000
narrow	* represents a subcategory of the lowest level – level 0
narrow1	* represents a subcategory of the second highest level – level 1
narrow2	* represents a subcategory of the highest level – level 2
newsGroup	-
related	* a category that may be related to the current category
symbolic	* another link in the catalog that can be a level 0 subcategory
symbolic1	* another link in the catalog that can be a level 1 subcategory
symbolic2	* another link in the catalog that can be a level 2 subcategory

Figure 13: Structure.rdf tag list

***Very Important Notes about the Structure tags*:**

- All rows marked with a * are types of the attribute “rtype” in the resources table from the Data model. This field is vital in collecting the correct categories from the data since it defines the way a category is related to its sub categories and is used in a recursive manner from within the resources table of the DM to collect all the subcategories of a current category and each subcategory’s subcategories and so on (until it reaches the links level) via its relational types and then lists them in order from the highest level to the lowest level, this is exactly what is used on websites such as Yahoo and DMOZ to create the hierarchical appearance of the data, and hence was also implemented in the browser package to give this effect.

- **Narrow** tags are subcategories of the given category and are organised into three levels narrow2, narrow1 and narrow.

- **Symbolic** categories are links to other categories within the catalog that can also be returned as a subcategory to a given category and are organised into three levels symbolic2, symbolic1 and symbolic.

- The **narrow** and **symbolic** levels are grouped together when the categories are returned on the browser UI, this is a common act in displaying the ODP data on any web site i.e. narrow2 and symbolic2 appear together etc.

- **Related** categories can also be subcategories since they may be related in some way to the category, this does not have levels and it appears under the “See also:” heading once the browser catalog has made a listing on the web page (Discussed later in the document).

- **altlang** categories are the categories that are listed in the “See this category in other languages:” heading once the browser catalog has made a listing on the web page, they represent the current category in different languages.

Below is the tag structure for the content.rdf file, again similar to the structure file only the important tags were selected to be stored in the data model. The content tag structure is more self explanatory since the fields refer directly to the attributes of the xurls table.

ExternalPage	Represents the url field in the xurls table.
Topic	-
Ages	Represents the ages field in the xurls table
Catid	Represents the catid field in the xurls table
d:Description	Represents the description field in the xurls table
d:Title	Represents the title field in the xurls table
link	-
link1	-
mediadate	Represents the mediadate field in the xurls table
priority	Represents the priority field in the xurls table

Figure 14: Content.rdf tag list

NOTE: A step by step listing of the importing instructions is provided in appendix B for the use of OACS and browser package developers.

4.2. The OpenACS Browser Package

OpenACS software development adheres to certain software principles that must be followed in order to add a new web application to the existing framework. Its method consists of a set package structure that must be followed so as to allow the web application to be installed, maintained and removed by the central package manager for OACS called the ACS Package Manager (APM). This APM is used to manage all packages within OACS and is responsible for the following tasks [11]:

- Package registration
- Automatic installation of a package, which includes loading data models, code libraries etc.
- Checking what packages depend on other packages before they are installed.
- Storing information on the package including ownership and a file list.

To ensure that all packages can be maintained correctly each package must store its files according to the package file structure depicted in the diagram below, this was followed for the development of the browser package.

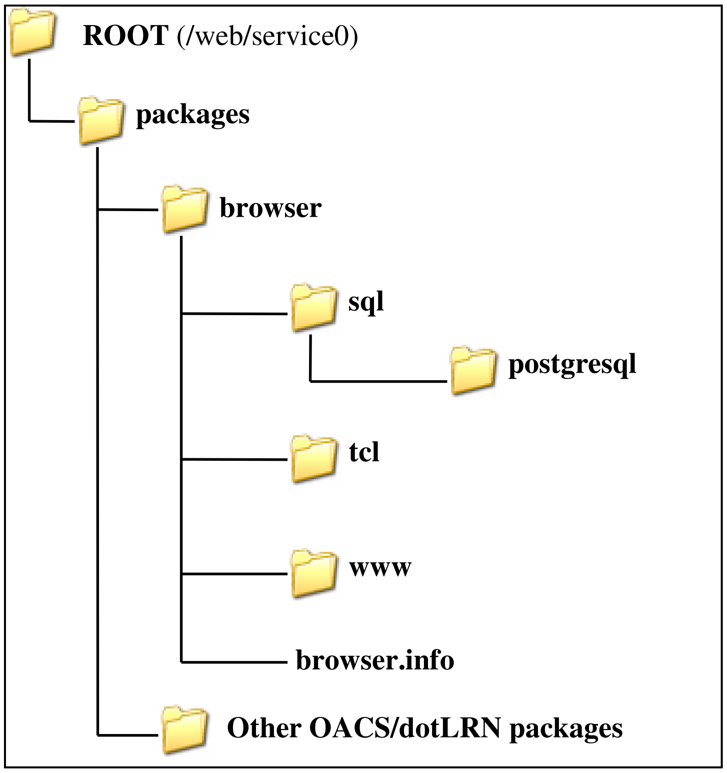


Figure 15: OACS Package Structure

As noted from the diagram each package including the browser package has three main folders (sql, tcl, www) as well as the package's corresponding .info file (browser.info) which specifies information about the package such as the name of each file it contains as well as the package parameters. Each of the three folders within the browser package will be discussed below.

4.2.1. SQL Directory

The main contents of this directory are directly related to the data model of the package. Within this directory there must be a sub-directory by the name of the database that the model will be installed into, for the browser package this is postgresql since the package supports only the postgresql database. Within this sub-directory are a number of SQL files that have the main aim of creating and dropping the data model as well as any SQL functions used to access the database. The actual file names must correspond to the package name , for example the create file for the DM is called browser-create.sql while the drop file is called browser-drop.sql as these are automatically called from the APM upon installation. Within this directory are another two files for the browserurl table one of which creates the table and the second which creates the functions that access that table in the DB their names are browserurl-table-create.sql and browserurl-functions-create.sql respectively. These files are called from the original create file for the browser package and hence the APM only needs to call that file to create the data model.

4.2.2. TCL Directory

The TCL directory contains library files that contain TCL methods (procs) that can be called from any file within the package. For the browser package this consists of a single file called browser-procs.tcl that contains a number of methods that are called from some of the other files in the package. This file serves as a form of API for the package, which increases code re-use by allowing the central repository of generic functions for the package rather than copying these functions into each file separately.

4.2.3. WWW Directory

This directory contains all the web pages that are viewable by browser package users, as mentioned earlier through the discussion of the OACS template system these web pages have a “separation of concerns” methodology of their own. That is each web page has a *.tcl, *.adp and a *.xql file that is stored in this directory. This helps keep the web page logic apart from the html design so as to allow a designer and programmer to work simultaneously. The .tcl files are the scripts that perform the logic of a web page, that is generating non static data or results from the database or even setting variables that are to be used on the html page. The .xql file stores any DB queries that this file uses, so as to allow a neater approach to code layout by keeping the queries separate from the scripting code. The .adp files correspond to the templates and web pages (html) that the user sees in the presentation layer of the UI. For the browser package there were a large number of files in this directory as we well as some sub-directories that were HTML related such as “css” and “images”, a complete list of these files is listed in Appendix B.

NOTE: In order to view the application under the current installation of OACS/dotLRN a new subfolder had to be mounted via the APM manager to the root of the site. That is, the APM will create an instance of the browser package at the /browser directory of the site unless specified otherwise. Hence the home page for my application will be under http://server/browser.

4.3. The dotLRN Browser Portlet Package

If we recall from our discussion in chapter 1 about the dotLRN platform we learnt that every single dotLRN package is an "applet" – not a java applet, but rather a term referring to "small application" [5]. We also discussed how these applets must provide an API under the acs-service-contract mechanism so as to allow dotLRN core packages to dispatch calls to them on the occurrence of certain events such as removing/adding the applet to/from the community or adding/removing a user to/from a community that has this applet enabled.

The dotlrn-browser applet package was developed under the exact same file structure that was discussed in the previous section concerning the browser package hence for the sake of clarity this will not be repeated. The only main difference that has to be mentioned is that the dotLRN applet package had a number of procs that were included in the tcl directory alongside some SQL functions that were included in the postgresql directory in order to provide for the function calls of the dotLRN core packages. Apart from community activity some of the procs catered for functions like minimizing the portlet window, maximizing the portlet window and allowing correct permissions for the portlet so that only administrators could view the portlet admin page. All of these functionalities will be displayed in chapter 5 as part of the final UI results that were achieved by the final browser system.

4.4. Browser System Operation

This section details an in depth discussion on how the functionalities of the final browser system were implemented and developed. It is mainly coordinated with the proposed data model and the achieved baseline architecture (as well as some extra functionalities) that were listed in the design – Chapter 3.

4.4.1. Browser Database Interactions

In developing the browser system most of the user interactions were via the browser package interface, which interacted with the stored ODP catalog in the database in order to achieve a well set out hierarchy of results for display on the UI of the application. The database interactions that were used in order to retrieve these particular results were collected recursively by calling the main script of the browser package (sub-cat.tcl) as soon as a category was clicked upon in the interface. What is meant by recursively is that each time a category was clicked upon this category's path was fed into the script as a parameter and executed all the resource fields of the resources table for a particular type of relation of this parameter, and then each of the returned subcategories could also be clicked upon. For example if we take a look below:

```
SELECT resource FROM resources
WHERE catid = (SELECT catid from structure where topic = 'Top/Arts') AND
(rtype = 'narrow2')
Order By resource
```

In this example query we have selected all the different resource (path of subcategory) entries of the catalog that are of the category 'Top/Arts' (which was clicked upon in the UI) and that are related to this category by the 'narrow2' type of relation , which in other words means we have collected the highest level of subcategories for the category of 'Top/Arts' or arts. Although it may be difficult to understand the first time around , it should be re-read to ensure that this simple example is understood as it is the basis of the database interactions that take place on any current website listing ODP catalog's or any other RDF imported catalogs. This example lists one of the types of relations to a clicked upon category and hence similar queries are executed to return the remaining different relations , these were listed in Figure 13 (marked by a *).

The final result is the listing of all the possible subcategories and links of a clicked upon category on the UI.

4.4.2. Browser Hierarchy listing

Now that we understand how the database operations were used in order to retrieve the relevant categories and links from the database, the format in which this data is listed on the interface must be explained. The subcategories are listed in a certain hierarchy followed by the URL links of that category, that is, if there are links for that category since it is possible for categories not to have URL links and just have subcategories. The main idea for the listing was collected from the DMOZ website as mentioned in the design since they represented the same ODP catalog very clearly and simply. The hierarchy format is depicted in figure 16 below according to the way it was arranged in the browser UI.

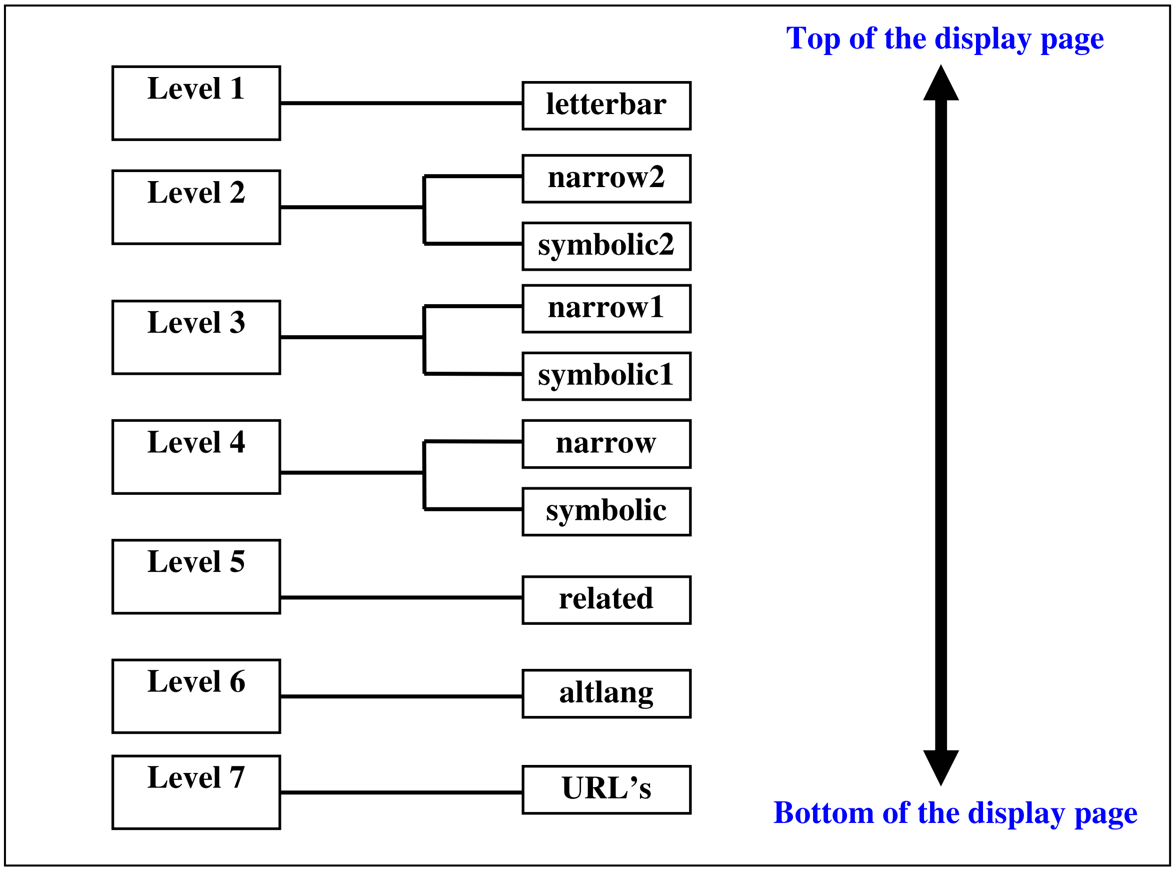


Figure 16: Hierarchy of the catalog output onto the screen

4.4.3. The Browser Permissions System

According to the design, the browser system had to be implemented in such a way so as only to allow people with administrator permissions, that is only staff members of the community to access certain areas and operations of the browser UI and the portlet display.

This was successfully implemented and proved to be a key aspect of the system since it controlled the users that could add and edit information from the portlet list. Based on the permissions system of dotLRN, the package only allowed the access of the portlet link list by people who had administration rights granted by the administrator for the particular community that the portlet was to appear in. The way this access was controlled is that each time the browser UI was accessed checks upon the access rights of the user (that is if he/she had admin rights) were conducted in order to determine if the portlet list navigation bar would appear on the interface. If the user was a staff member with administration rights the portlet navigation bar was generated on the user interface allowing the portlet list to be edited, also allowing an “add” link beside each subcategory and url in the UI to appear so as to provide quick adding to the portlet list. Whereas when a student was either directed from the portlet or was browsing through the site individually the browser UI would recognize this students access rights and not display the portlet list navigation bar or the “add” links beside each category/link in the Interface. Hence if the user logged into the browser system was a student the interface would be presented just like a normal catalog listing website similar to yahoo.com or dmoz.org by allowing the browsing and accessing of the categories and URL’s directly from the UI.

4.4.4. Adding links to the portlet list

The adding of links to the portlet list was implemented successfully. Each staff member had the ability to add new links from the catalog listing into the portlet list so as to appear in the portlet within their community. The addition of links supported both adding of subcategories and URL’s, since after the top categories each link in the catalog was either one of these. Once the link was added by the user the link and its details were automatically inserted into the browserurl table in the DB, if the link existed in the list already a message would appear informing the user of this and asking him/her to add another link.

The implementation of the adding functionality also included an extra adding option that could allow a custom url to be added into the portlet list. The user could add a custom url by typing in the details manually, from here on the link was added in the same way as if it was selected from the catalog listing in the UI.

4.4.5. Deleting Links

The deleting functionality was implemented as planned. Although it had to be carefully designed in order to allow the staff users with a confirmation page in order to ensure the user wanted to remove the link that was selected. The user could delete any URL/subcategory from the portlet list successfully with confirmation. Once the user had confirmed the delete action the link would be removed from the browserurl table from the database and hence the portlet contents would also mirror the change once the portal page was refreshed. Both the links and the subcategories could be removed from the portlet list.

4.4.6. Adding Comments to Links

One of the extra functions added, the commenting ability on links was implemented according to the requirements specified. It only permits staff members of the community to add comments on existing links in the portlet list. The user is able to choose any of the current links in the portlet list and add a new comment to them that would appear beside the link in the portlet window. The user could add comments to subcategories and URL’s , this was implemented by inserting the comment into the browserurl table for the chosen link , hence allowing the portlet to show this comment once the page was refreshed.

4.4.7. Editing Links

This was also successfully implemented in that it allowed the correct group of users (staff) to edit current link information from the portlet list, each change was applied to the database entry for that particular row within the browserurl table and hence all changes would appear in the portlet window once refreshed without any problematic issues.

4.4.8. Searching the Catalog

This was the final functionality added to the browser package in order to add extra functionality to the browser system. It was successfully implemented via queries that were made on both the structure and the urls of the ODP data according to a user selected attribute and a user entered query. This was developed by searching through the selected column contents of certain fields in the tables for the entered query. The user can enter any string field into the search form and it would return appropriate results or it would return a message that the query could not be matched. The search mainly applied to the structure and xurls tables, according to the particular search option that was chosen. There were 4 main types of searches that could be executed:

- The "urls" option search was implemented by searching for the entered query in the url and description fields of the xurls table in the database.
- The "title" option search was implemented by searching for the entered query in the titles of all subcategories in the structure table.
- The "topic" option search was implemented by searching for the entered query in all the subcategories topics in the structure table.
- The "All" option search was implemented by searching for the entered query anywhere in the structure table so as to return subcategories.

The search could have been implemented with many other search options, but it was decided that these four would be the significant searches that would be provided. Although my first aim was also to provide advanced searching algorithms, due to the time constraints it could not be implemented as part of the browser system.

4.4.9. Portlet Accessibility

This operation of the browser accessibility is one of the main features of the browser system. It was implemented successfully according to all the set requirements in terms of catalog functionality and accessibility as well as all dotLRN standards in terms of integrating with the community portal. With respect to the integration factor with the dotLRN community portal the final browser portlet supports:

- The addition to a dotLRN community portal.
- The removal from a dotLRN community portal.
- The addition and removal of a user to/from a community that has the browser applet enabled.
- Minimization, maximization from the portal page.
- Removal from the "portal layout" personalization page for the community.
- Different access levels – only members with administration rights can view the portlet administration page.

In terms of the catalog functionality the final browser portlet implementation included:

- The ability for users (both students and staff) to view the added links by the staff members of the community, this was implemented by allowing the portlet/applet package to read the contents of the portlet list from the browserurl table from the database. This table consists of the entire list that the staff selected from the browser UI and hence is displayed in the portlet window. Each link alongside its comment is shown.
- Category links can be accessed, if the link is a subcategory once clicked upon by the portal member it will direct them into the appropriate catalog listing for that subcategory. This was implemented via calling the same script that is used in the browser UI to generate the hierarchy listing for that category.
- URL links can be accessed, if the link is a URL link once clicked upon by the portal member it will redirect them automatically to the corresponding website. This was implemented by storing the http links in the portlet list with their title as the url value of the website , so they can be generated into the portlet as actual <a href> tags in order to be accessed.

- An extra functionality added to the browser portlet was the search ability, this permits the same search functions as the search in the browser UI to be accessed via the portlet window, once a search is submitted from the portlet the user would be redirected to the search results page in the browser UI for his query. This was implemented via the same search method used in the browser package, by mirroring the search form in the portlet and allowing its access directly from there to the students. A main point that must be noted is once a search query was entered and submitted the search results would not appear with “add” links beside them if the user who submitted the query was a student, again this was controlled by the permission system of the browser application to ensure no loose holes were present in the system that would allow students to modify links to the portlet list.

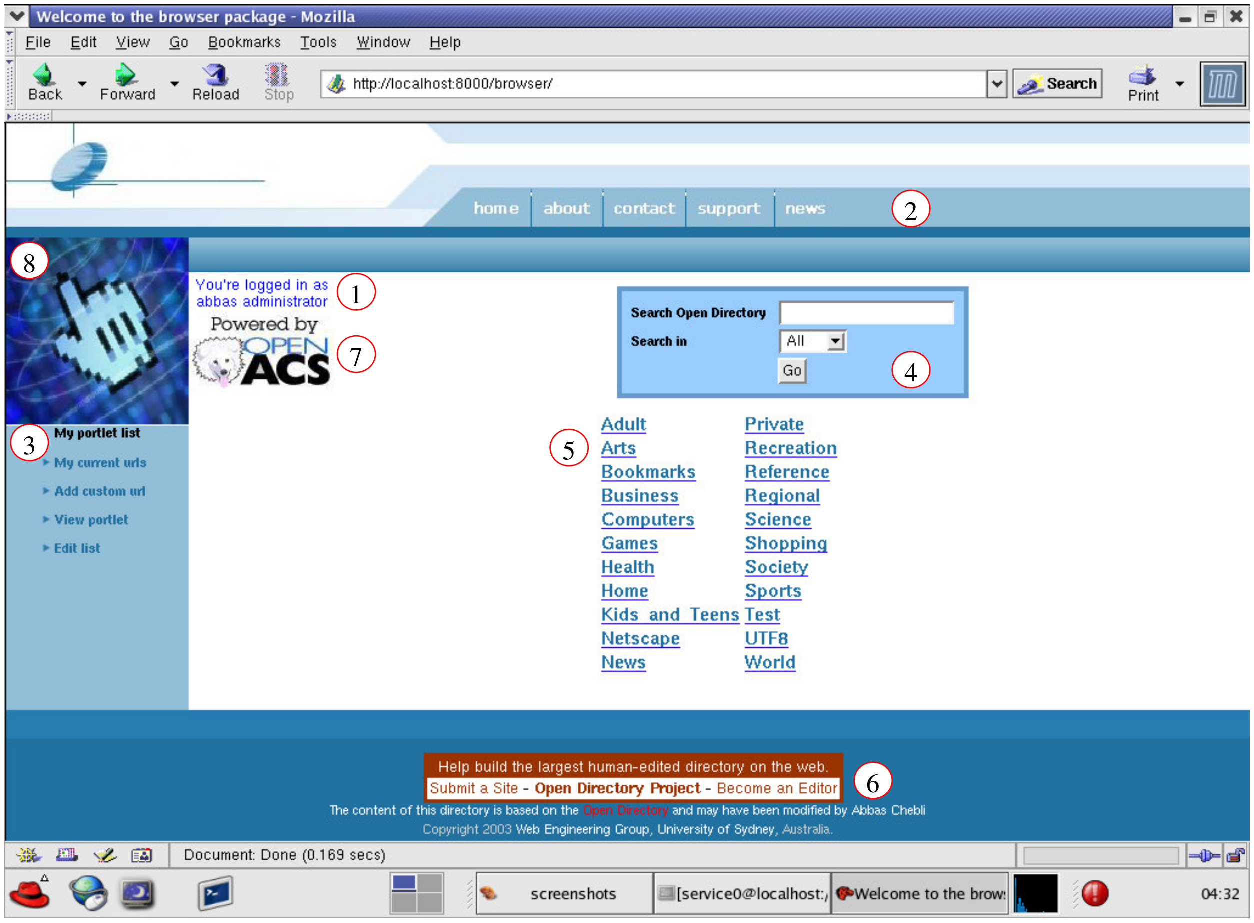
CHAPTER 5

RESULTS

By this stage we have successfully analysed the entire design and implementation phases of the browser system, hence a detailed understanding of the core back-end principles of this web application is clear. In this Chapter the main focus is to discuss and walk through the User Interface of the system in order to help visualize the discussed functionality available for the user in the application. Both screen shots of the browser application and portlet will be displayed exactly as they would appear from the user's perspective.

5.1. Final User Interface

The browser application had a very simple and clear looking interface. As planned it was to depict many of the popular ODP catalog web sites available such as "Yahoo" in order to adhere to the user's cognitive model of such hierarchy listings and allow the same usability features to flow through the use of this interface. The main screen of the final browser UI is shown below in figure 17:



You're logged in as abbas administrator

Help build the largest human-edited directory on the web.
Submit a Site - **Open Directory Project** - Become an Editor

The content of this directory is based on the Open Directory and may have been modified by Abbas Chebli
Copyright 2003 Web Engineering Group, University of Sydney, Australia.

Figure 17: Browser package home page for staff (screenshot)

As you can see the interface used an appropriate layout of color that would be comfortable to the user's eye. It provides a good combination of images and content without over clogging the screen with much complex data that would confuse new users to the system. The system consists of a number of core aspects each of which is labeled in the diagram and explained below.

Please note: This is the description of the staff member browser package since the portlet list navigation bar is visible.

1. **User Greeting:** This is a common standard of OACS User Interfaces greeting the user that is logged into the application.
2. **Main Navigation bar:** Provides links to less task oriented sections of the application such as support and contact and news which links back to the OACS website.
3. **Portlet List Navigation bar:** This is the navigation bar that allows portlet list accessibility as well as links to certain portlet list related tasks.
4. **Search form:** Search query form for access to the browser search engine.
5. **Top Category Listing:** Listing of the highest categories of the ODP data, this is where all sub-categories branch from.
6. **Open Directory links:** For copyright purposes this was included to indicate that the ODP data was used, it provides links to the main website www.dmoz.org.
7. **OACS Logo:** Another commodity of the OACS interface, the main logo to indicate the application is running on OACS.
8. **Browser Logo:** The browser system logo, added to provide a more professional appearance to the UI.

In the next few sections the different screens displayed in the interface will be explained, these are all staff pages since the access list for the portlet is available, a student perspective of this interface will be displayed later in this chapter in order to show the UI differences for non staff users.

5.1.1. Hierarchy Listing

As discussed in section 4.2.2. the main hierarchy is listed in a particular order as we view it down the page, this symbolises the different sublevels and relations of the categories. Since not all categories once clicked upon will have every single different level of the hierarchy the screen shots below show different category listings that outline all the available levels of the ODP catalog. The figure below will outline the new features of the UI that have changed since the home page depicted in figure 17.

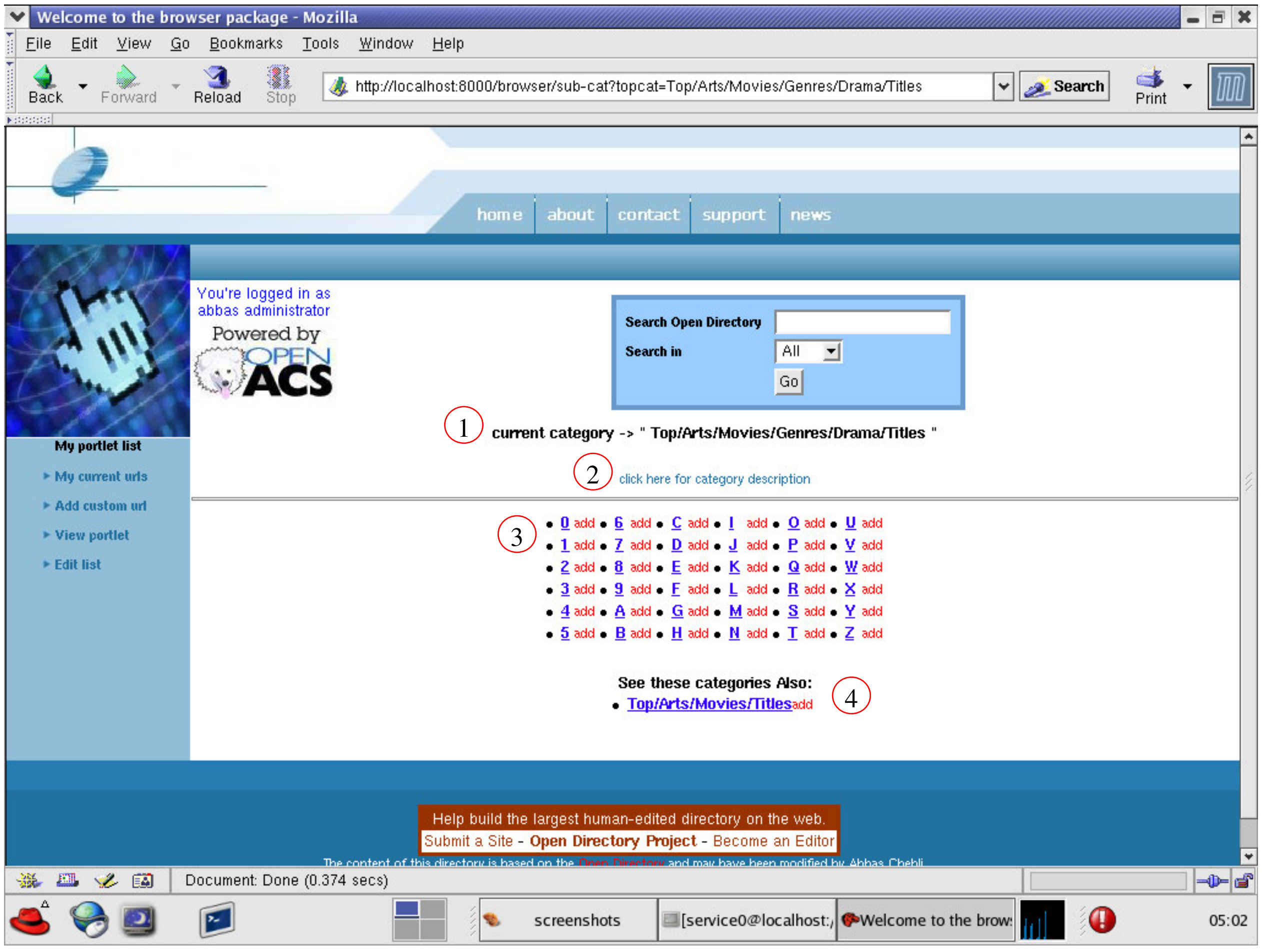

Welcome to the browser package - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8000/browser/sub-cat?topcat=Top/Arts/Movies/Genres/Drama/Titles
Search Print
home about contact support news
You're logged in as abbas administrator
Powered by OPEN ACS
Search Open Directory
Search in All
Go
My portlet list
My current urls
Add custom url
View portlet
Edit list
1
current category -> " Top/Arts/Movies/Genres/Drama/Titles "
2
click here for category description
3
0 add 6 add C add I add O add U add
1 add 7 add D add J add P add V add
2 add 8 add E add K add Q add W add
3 add 9 add F add L add R add X add
4 add A add G add M add S add Y add
5 add B add H add N add T add Z add
See these categories Also:
Top/Arts/Movies/Titlesadd
4
Help build the largest human-edited directory on the web.
Submit a Site - Open Directory Project - Become an Editor
The content of this directory is based on the Open Directory and may have been modified by Abbas Chebli
Document: Done (0.374 secs)
screenshots
[service0@localhost:/
Welcome to the brow:
05:02

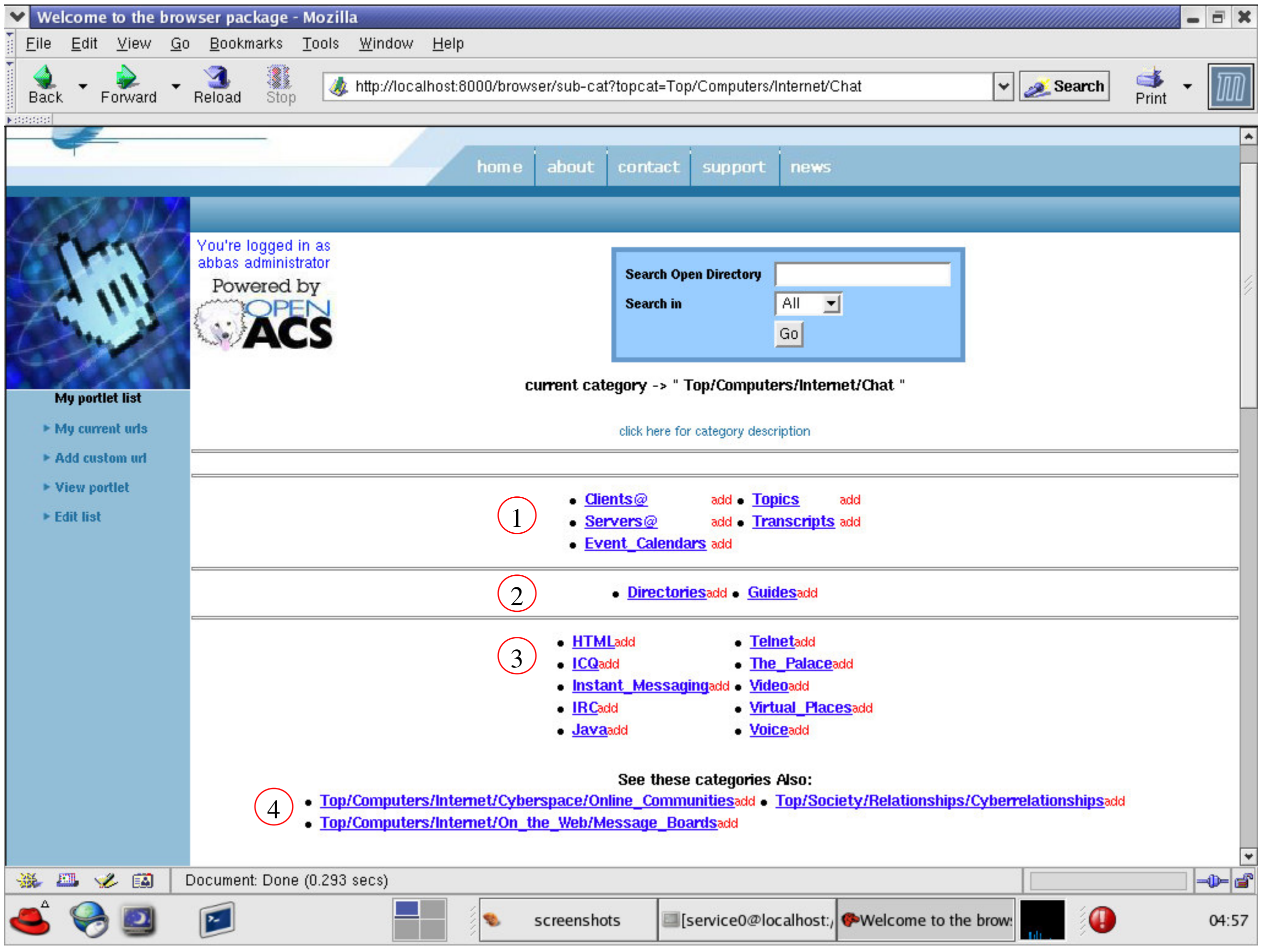
Figure 18: Browser hierarchy listing of the letterbar subcategory (screenshot)

From the diagram:

1. **Current category indicator:** Indicates the current category path in this example "Top/Arts/Movies/Genres/Drama/Titles".
2. **Current Category Description link:** link to the current category description.
3. **LetterBar subcategory Listing:** Level 1 of the screen hierarchy displays each subcategory with its "add" link to its right.

4. **Related Category Listing:** level 5 of screen hierarchy display, since all other levels are not available for this category they are omitted and the next available subcategory is listed under the letterbar listing which is the related subcategories.

Below are some other hierarchy listing screens of the remaining levels of the ODP relations.



http://localhost:8000/browser/sub-cat?topcat=Top/Computers/Internet/Chat

You're logged in as abbas administrator

current category -> " Top/Computers/Internet/Chat "

click here for category description

1
- Clients@ add
- Servers@ add
- Event_Calendars add
- Topics add
- Transcripts add

2
- Directoriesadd
- Guidesadd

3
- HTMLadd
- ICQadd
- Instant_Messagingadd
- IRCadd
- Javaadd
- Telnetadd
- The_Palaceadd
- Videoadd
- Virtual_Placesadd
- Voiceadd

See these categories Also:

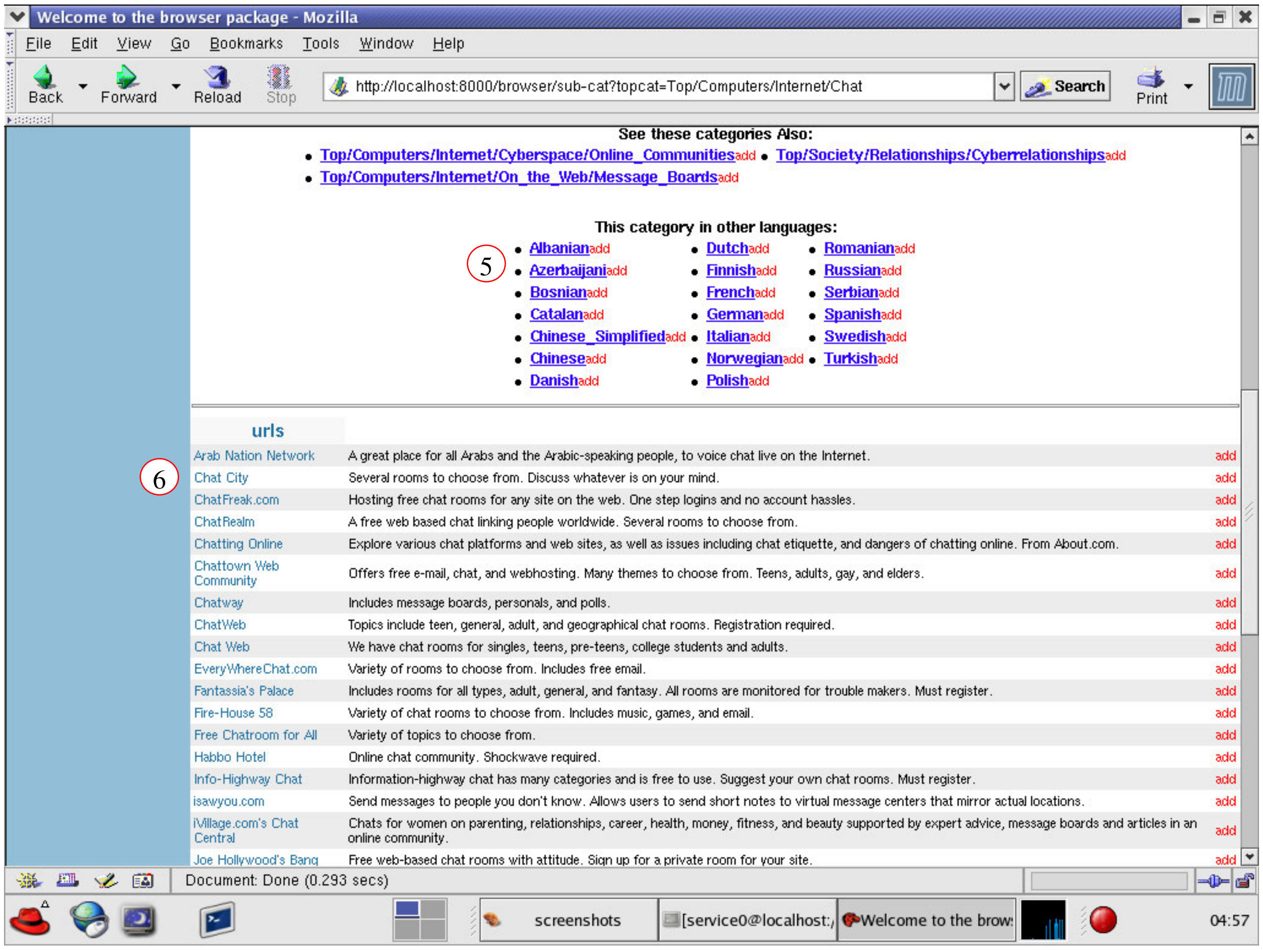
4
- Top/Computers/Internet/Cyberspace/Online_Communitiesadd
- Top/Society/Relationships/Cyberrelationshipsadd
- Top/Computers/Internet/On_the_Web/Message_Boardsadd

Document: Done (0.293 secs)

Figure 19: Browser hierarchy listing 2 (screenshot)

1. **Level 2 subcategory Listing:** Level 2 of screen hierarchy contains narrow2 and symbolic2 links.
2. **Level 3 subcategory Listing:** Level 3 of screen hierarchy contains narrow1 and symbolic1 links.
3. **Level 4 subcategory Listing:** Level 4 of screen hierarchy contains narrow and symbolic links.
4. **Level 5 subcategory Listing:** Level 5 of screen hierarchy contains related links.

The screenshot above is continued below with the level 6 and level 7 of the hierarchy.


Welcome to the browser package - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8000/browser/sub-cat?topcat=Top/Computers/Internet/Chat
Search Print
5
6
Document: Done (0.293 secs)
screenshots
[service0@localhost:
Welcome to the brow:
04:57

See these categories Also:

- Top/Computers/Internet/Cyberspace/Online_Communitiesadd • Top/Society/Relationships/Cyberrelationshipsadd
- Top/Computers/Internet/On_the_Web/Message_Boardsadd

This category in other languages:

- Albanianadd
- Azerbaijaniadd
- Bosnianadd
- Catalanadd
- Chinese_Simplifiedadd
- Chineseadd
- Danishadd
- Dutchadd
- Finnishadd
- Frenchadd
- Germanadd
- Italianadd
- Norwegianadd
- Polishadd
- Romanianadd
- Russianadd
- Serbianadd
- Spanishadd
- Swedishadd
- Turkishadd

urls		
Arab Nation Network	A great place for all Arabs and the Arabic-speaking people, to voice chat live on the Internet.	add
Chat City	Several rooms to choose from. Discuss whatever is on your mind.	add
ChatFreak.com	Hosting free chat rooms for any site on the web. One step logins and no account hassles.	add
ChatRealm	A free web based chat linking people worldwide. Several rooms to choose from.	add
Chatting Online	Explore various chat platforms and web sites, as well as issues including chat etiquette, and dangers of chatting online. From About.com.	add
Chattown Web Community	Offers free e-mail, chat, and webhosting. Many themes to choose from. Teens, adults, gay, and elders.	add
Chatway	Includes message boards, personals, and polls.	add
ChatWeb	Topics include teen, general, adult, and geographical chat rooms. Registration required.	add
Chat Web	We have chat rooms for singles, teens, pre-teens, college students and adults.	add
EveryWhereChat.com	Variety of rooms to choose from. Includes free email.	add
Fantassia's Palace	Includes rooms for all types, adult, general, and fantasy. All rooms are monitored for trouble makers. Must register.	add
Fire-House 58	Variety of chat rooms to choose from. Includes music, games, and email.	add
Free Chatroom for All	Variety of topics to choose from.	add
Habbo Hotel	Online chat community. Shockwave required.	add
Info-Highway Chat	Information-highway chat has many categories and is free to use. Suggest your own chat rooms. Must register.	add
isawyou.com	Send messages to people you don't know. Allows users to send short notes to virtual message centers that mirror actual locations.	add
iVillage.com's Chat Central	Chats for women on parenting, relationships, career, health, money, fitness, and beauty supported by expert advice, message boards and articles in an online community.	add
Joe Hollywood's Bang	Free web-based chat rooms with attitude. Sign up for a private room for your site.	add

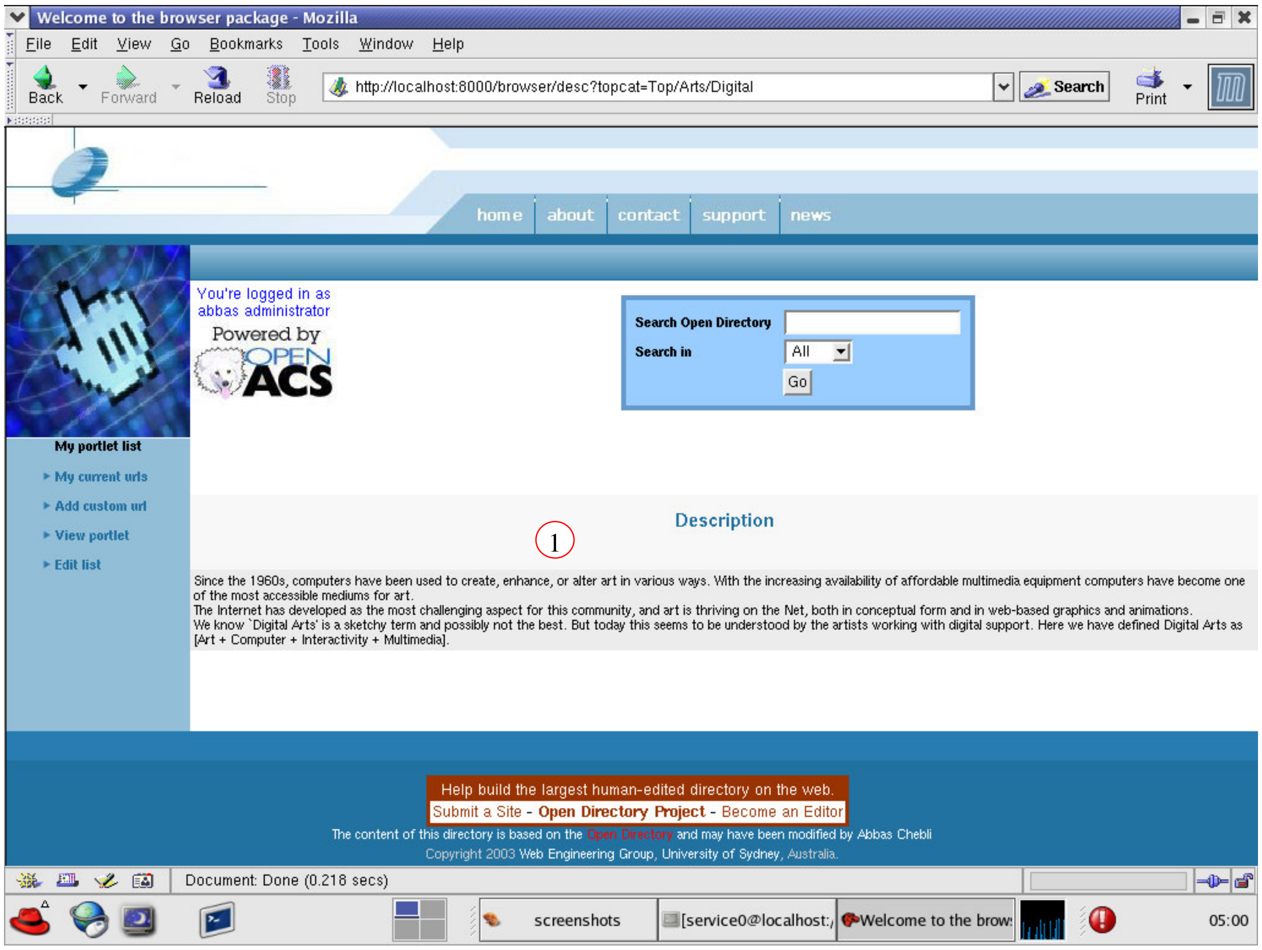
Figure 20: Browser hierarchy listing 3 (screenshot)

5. **Level 6 subcategory Listing:** Level 6 of screen hierarchy contains the altlang links which represents this category in other languages.
6. **Level 7 subcategory Listing:** Level 7 of screen hierarchy contains the url links which are the URL's available for this category.

The above screenshots have displayed all of the 7 levels of the hierarchy that were discussed in the implementation (4.2.2.), please note that each level is able to be added to the portlet list whether it is a subcategory or a link , this can be done via the corresponding "add" link to the right of the link.

5.1.2. Category Description

Once the user clicks on the “click here for category description” link from a listing page such as the one in figure 18, the following screen is shown


Welcome to the browser package - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8000/browser/desc?topcat=Top/Arts/Digital
Search Print
home about contact support news
You're logged in as abbas administrator
Powered by OPEN ACS
Search Open Directory
Search in All
Go
My portlet list
My current urls
Add custom url
View portlet
Edit list
1
Document: Done (0.218 secs)
screenshots
[service0@localhost:/
Welcome to the brow
05:00

Description

Since the 1960s, computers have been used to create, enhance, or alter art in various ways. With the increasing availability of affordable multimedia equipment computers have become one of the most accessible mediums for art.
The Internet has developed as the most challenging aspect for this community, and art is thriving on the Net, both in conceptual form and in web-based graphics and animations.
We know `Digital Arts' is a sketchy term and possibly not the best. But today this seems to be understood by the artists working with digital support. Here we have defined Digital Arts as [Art + Computer + Interactivity + Multimedia].

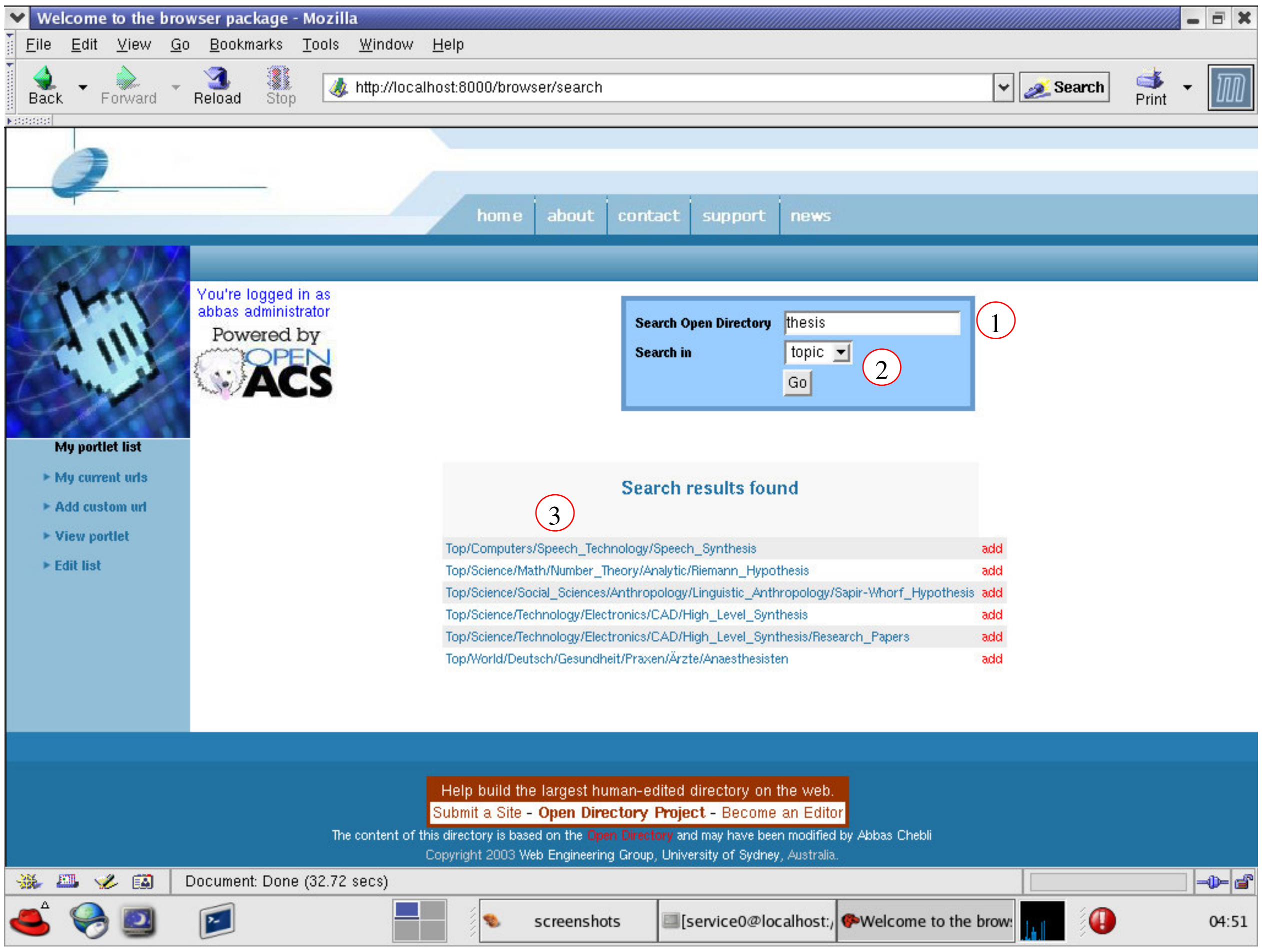
Help build the largest human-edited directory on the web.
Submit a Site - **Open Directory Project** - Become an Editor
The content of this directory is based on the Open Directory and may have been modified by Abbas Chebli
Copyright 2003 Web Engineering Group, University of Sydney, Australia.

Figure 21: Category Description (screenshot)

The above screen lists:

1. **Category Description:** This description represents a brief mentioning of what this category is about.

5.1.3. Search Engine


Welcome to the browser package - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8000/browser/search
Search Print
home about contact support news
My portlet list
My current urls
Add custom url
View portlet
Edit list
1
2
3
Document: Done (32.72 secs)
screenshots
[service0@localhost:/
Welcome to the brow:
04:51

You're logged in as abbas administrator

Powered by OPEN ACS

Search Open Directory: thesis

Search in: topic

Go

Search results found

Top/Computers/Speech_Technology/Speech_Synthesis	add
Top/Science/Math/Number_Theory/Analytic/Riemann_Hypothesis	add
Top/Science/Social_Sciences/Anthropology/Linguistic_Anthropology/Sapir-Whorf_Hypothesis	add
Top/Science/Technology/Electronics/CAD/High_Level_Synthesis	add
Top/Science/Technology/Electronics/CAD/High_Level_Synthesis/Research_Papers	add
Top/World/Deutsch/Gesundheit/Praxen/Ärzte/Anaesthesisten	add

Help build the largest human-edited directory on the web.
Submit a Site - **Open Directory Project** - Become an Editor

The content of this directory is based on the Open Directory and may have been modified by Abbas Chebli
Copyright 2003 Web Engineering Group, University of Sydney, Australia.

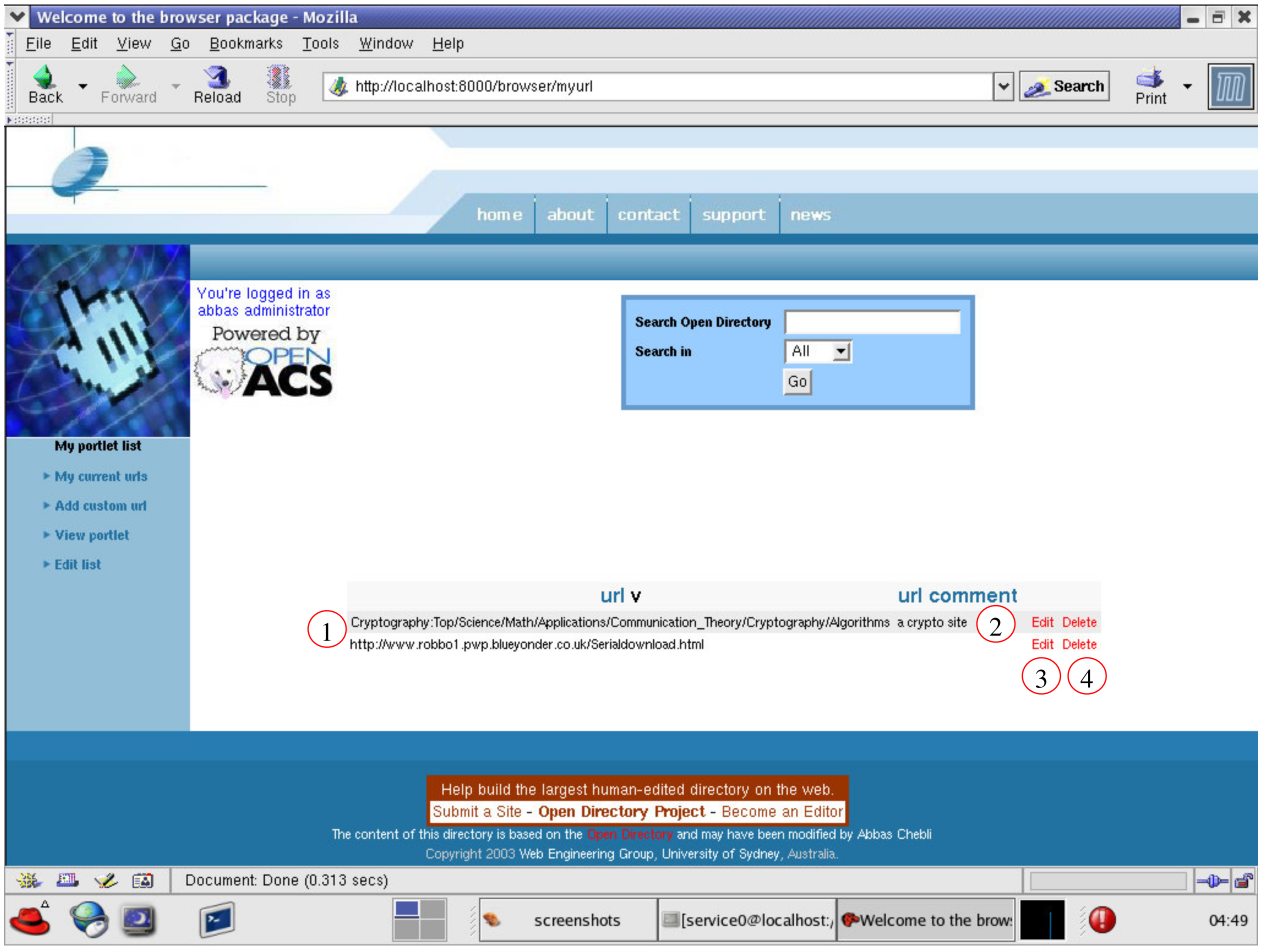
Figure 22: Browser search (screenshot)

The above screen shows the search engine being queried.

1. **Query field:** Query field where the user must enter the desired search query.
2. **Search type:** The user can select the search type to use.
3. **Search Results Display:** Once the search query is executed the displayed search results are listed here. Each search result can be added to the portlet list.

5.1.4. Portlet List Navigation Bar

As discussed earlier the portlet list labeled as 3 in the main UI diagram (Figure 17) allows the staff members to access the portlet list links. The diagram below depicts the main portlet list listing as well as the operations that can be applied to this list.


Welcome to the browser package - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8000/browser/myurl
Search Print
home about contact support news
You're logged in as abbas administrator
Powered by OPEN ACS
Search Open Directory
Search in All
Go
My portlet list
My current urls
Add custom url
View portlet
Edit list
1
2
3
4
Document: Done (0.313 secs)
screenshots
[service0@localhost:
Welcome to the brow:
04:49

url v	url comment		
Cryptography:Top/Science/Math/Applications/Communication_Theory/Cryptography/Algorithms	a crypto site	Edit	Delete
http://www.robbo1.pwp.blueyonder.co.uk/Serialdownload.html		Edit	Delete

Help build the largest human-edited directory on the web.
Submit a Site - Open Directory Project - Become an Editor
The content of this directory is based on the Open Directory and may have been modified by Abbas Chebli
Copyright 2003 Web Engineering Group, University of Sydney, Australia.

Figure 23: Portlet List Contents (screenshot)

From the diagram above:

1. **Current URL listings:** The available links in the portlet list that have been added by the staff members.
2. **Comment:** This labels the comment entered for this particular link.
3. **Edit link:** This link is to allow the staff user to edit the link as well as its comment**.**
4. **Delete Link:** This link is to allow the staff user to delete the link from the portlet list.

Upto this point all of the browser package functionality has been outlined apart from the one link on the portlet list navigation bar, the "view portlet" link. This link redirects the user to the dotLRN community portal where the user can enter his community and view the portlet's contents. In this next section the focus will be on the portlet appearance and functionality within this portal, outlining the users perception of the operations that can be executed on the portlet.

5.2. Portlet Interface and Appearance

The portlet interface adhered to the typical portlet layout of dotLRN allowing the portlet list contents to be displayed clearly as links that could be clicked upon by members of the community. Each link would either lead back to the catalog in the case of a category link, otherwise the user would be redirected to that particular website if the link was URL. As we discussed earlier if a user is forwarded from a category link to the browser UI, the "add" would disappear alongside the portlet list navigation bar since this student does not have any permissions to edit list preferences , he/she can only browse through the catalog. A screenshot of such a case will be depicted in this section also.

5.2.1. Student Portlet Display

The following diagram depicts the browser portlet as a student would see it.

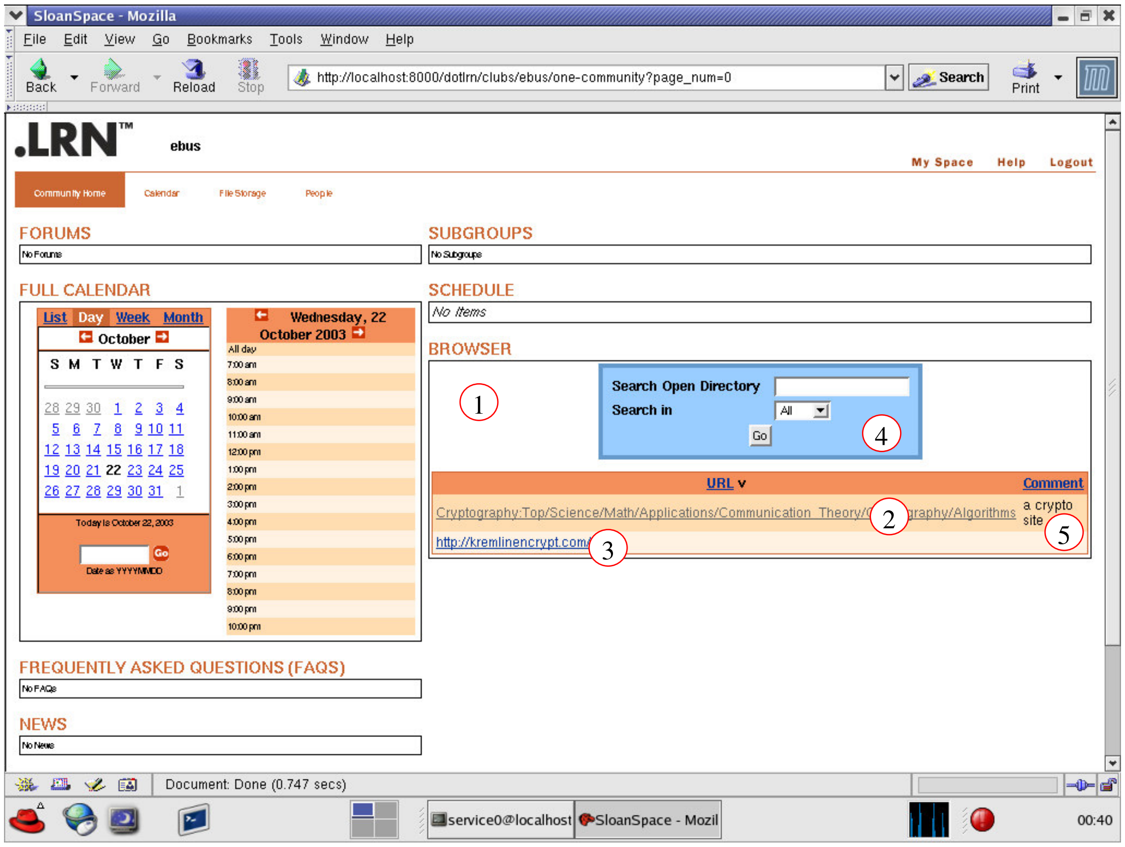
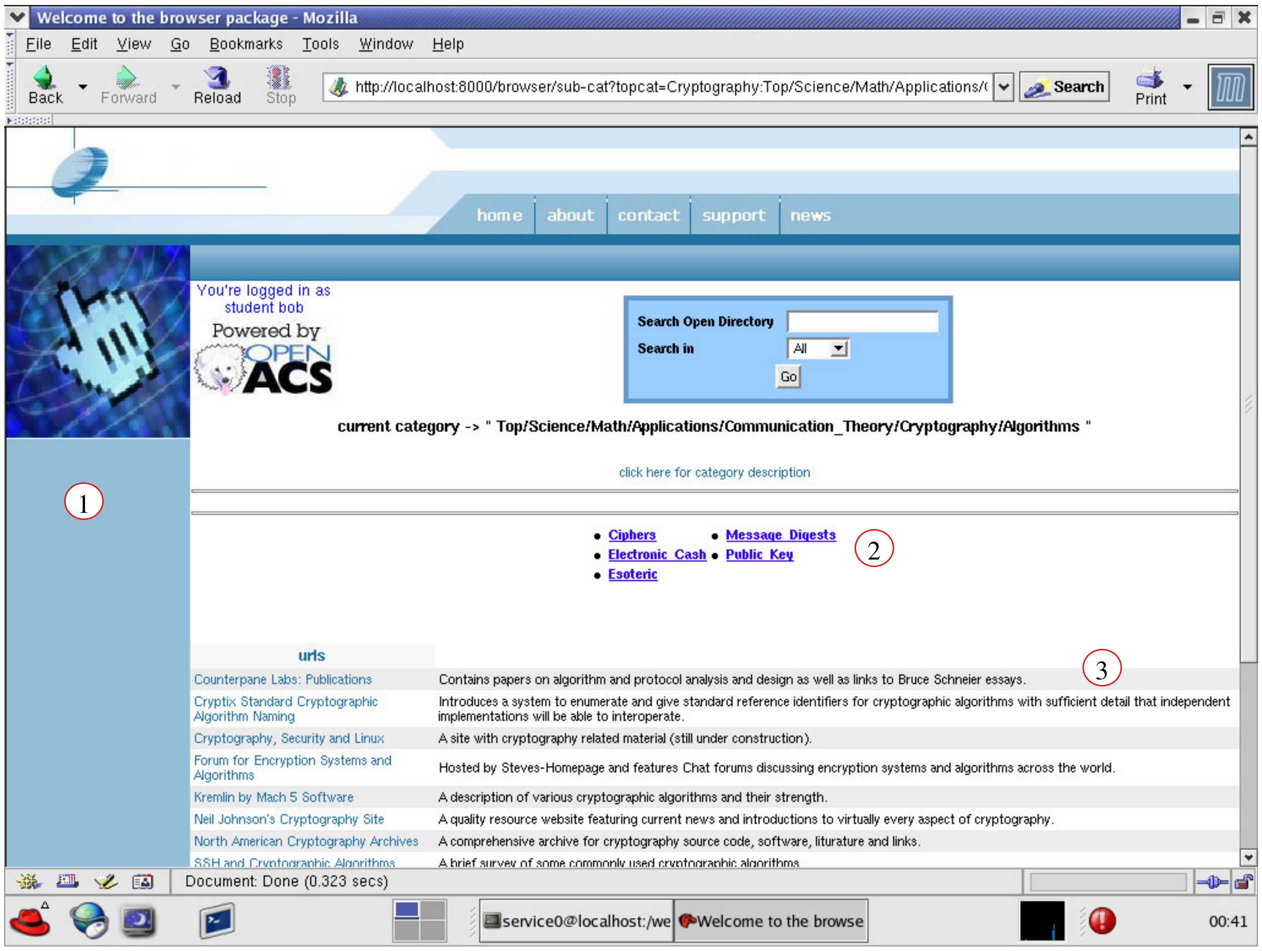

SloanSpace - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop Search Print
http://localhost:8000/dotlrn/clubs/ebus/one-community?page_num=0
.LRN™ ebus
My Space Help Logout
Community Home Calendar File Storage People
FORUMS
No Forums
SUBGROUPS
No Subgroups
FULL CALENDAR
List Day Week Month
October
Wednesday, 22 October 2003
SCHEDULE
No Items
BROWSER
Search Open Directory
Search in All
Go
URL
Comment
Cryptography:Top/Science/Math/Applications/Communication_Theory/Cryptography/Algorithms
a crypto site
http://kremlinencrypt.com/
Today is October 22, 2003
Date as YYYYMMDD
FREQUENTLY ASKED QUESTIONS (FAQS)
No FAQs
NEWS
No News
Document: Done (0.747 secs)
service0@localhost
SloanSpace - Mozil
00:40
1
2
3
4
5

Figure 24: Student Portlet View (screenshot)

From this diagram we note 5 main aspects:

1. **Browser portlet:** Actual portlet window for the browser package in the portal.
2. **Category link in the portlet:** A category link that links back to the browser UI.
3. **URL link in the portlet:** A URL link in the portlet that links to the corresponding website, once the user clicks on this he is automatically forwarded to the web link listed.
4. **Search Form:** Depicts the exact same search functionality in the browser UI.
5. **Comment:** Comment made on the category link displayed.

Please note once the student clicks on the category he will be redirected to the browser UI with the interface shown below in figure 25. We must carefully note how the portlet list operations usually depicted in place number 1 (on the diagram) have been removed since the user has no permissions to access these operations. Similarly the "add" links have been omitted that were usually depicted beside each link on the right, usually at 2 and 3 in the diagram.


Welcome to the browser package - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8000/browser/sub-cat?topcat=Cryptography:Top/Science/Math/Applications/(
Search Print
home about contact support news
1
2
3
Search Open Directory
Search in All
Go
service0@localhost:/we
Welcome to the browse
00:41

You're logged in as student bob

Powered by OPEN ACS

current category -> " Top/Science/Math/Applications/Communication_Theory/Cryptography/Algorithms "

click here for category description

- Ciphers
- Message Digests
- Electronic Cash
- Public Key
- Esoteric

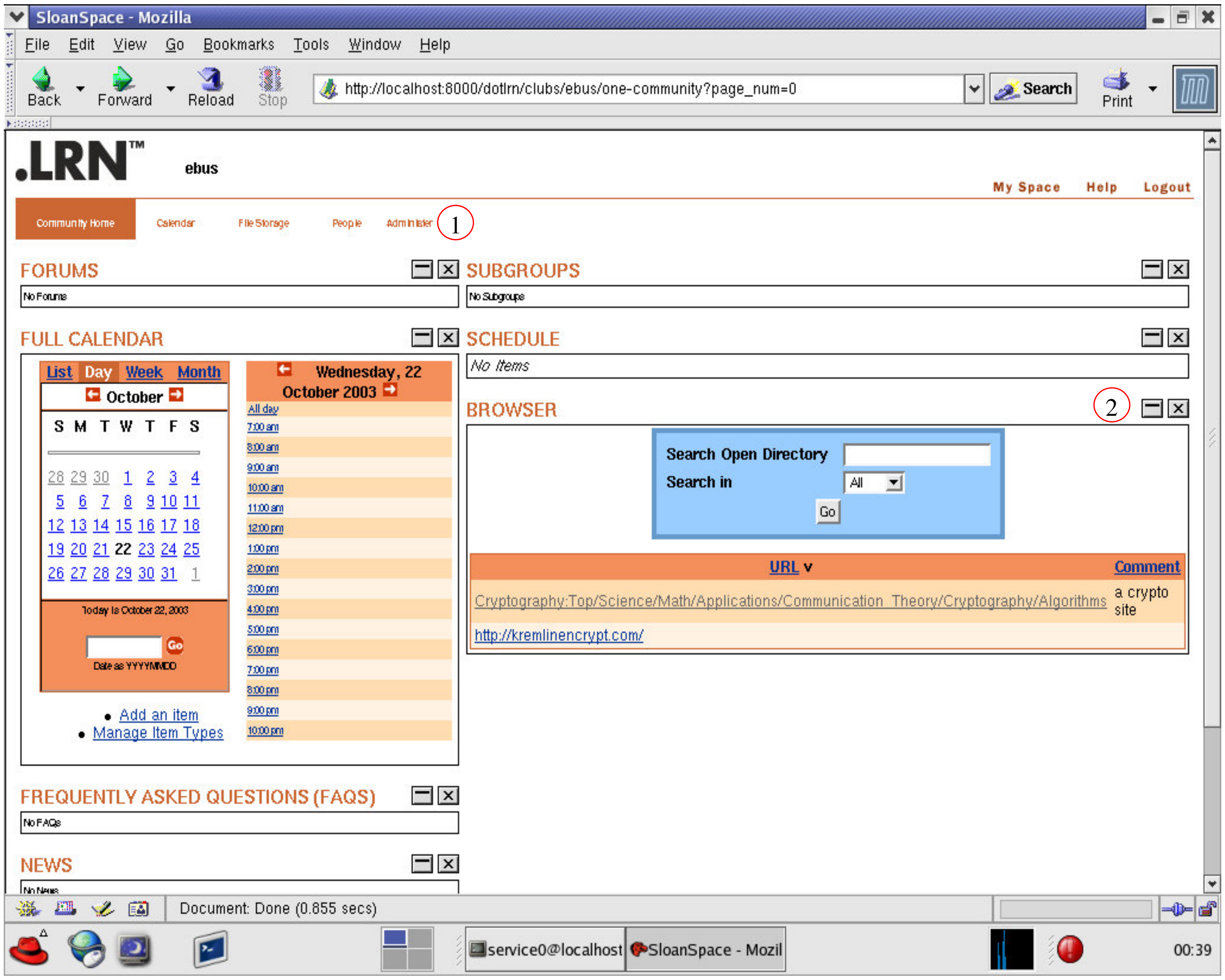
urls	
Counterpane Labs: Publications	Contains papers on algorithm and protocol analysis and design as well as links to Bruce Schneier essays.
Cryptix Standard Cryptographic Algorithm Naming	Introduces a system to enumerate and give standard reference identifiers for cryptographic algorithms with sufficient detail that independent implementations will be able to interoperate.
Cryptography, Security and Linux	A site with cryptography related material (still under construction).
Forum for Encryption Systems and Algorithms	Hosted by Steves-Homepage and features Chat forums discussing encryption systems and algorithms across the world.
Kremlin by Mach 5 Software	A description of various cryptographic algorithms and their strength.
Neil Johnson's Cryptography Site	A quality resource website featuring current news and introductions to virtually every aspect of cryptography.
North American Cryptography Archives	A comprehensive archive for cryptography source code, software, liturature and links.
SSH and Cryptographic Algorithms	A brief survey of some commonly used cryptographic algorithms

Document: Done (0.323 secs)

Figure 25: Student redirect to browser package (screenshot)

5.2.2. Staff Portlet Display

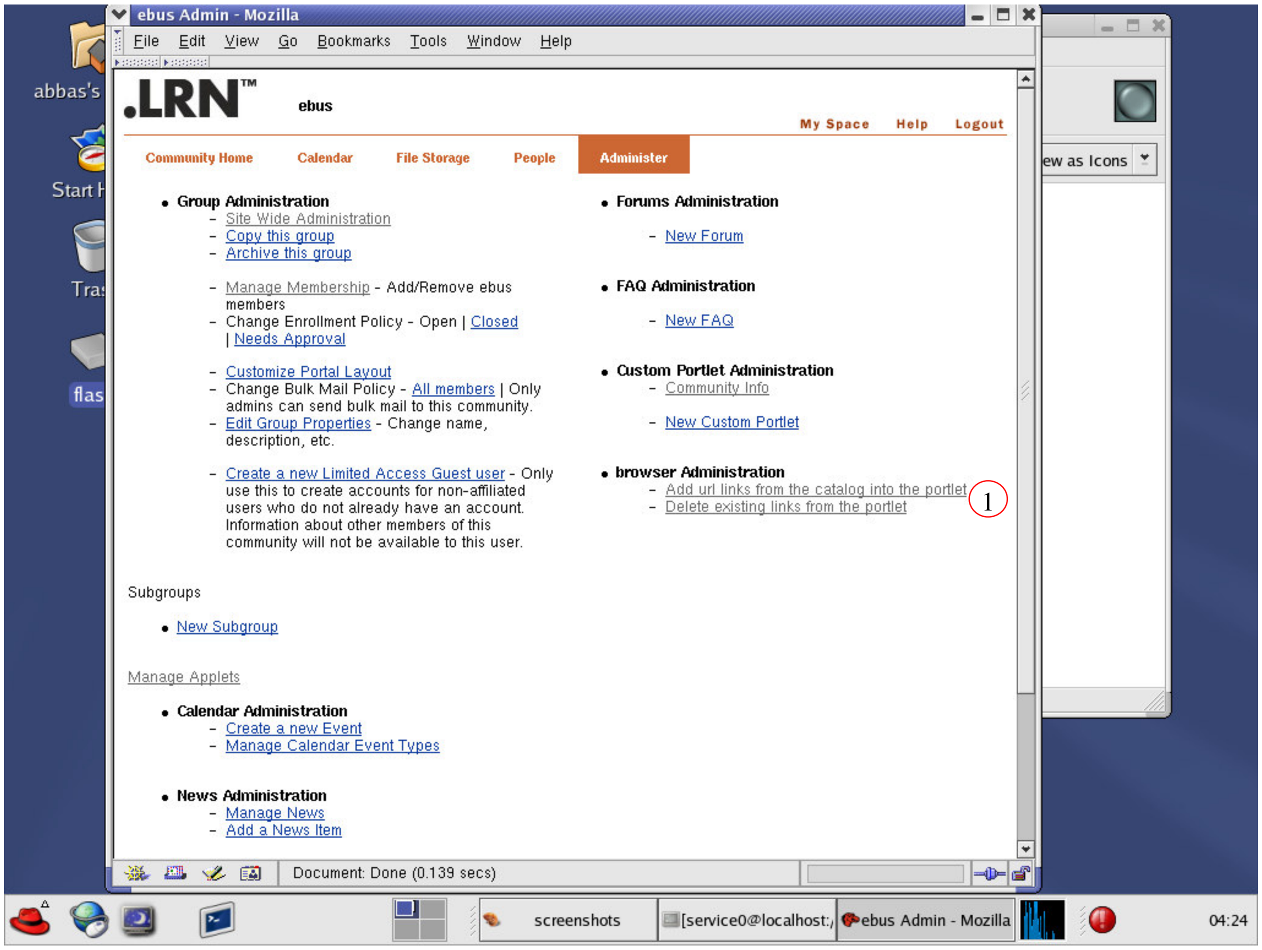
The following diagram depicts the browser portlet as the staff members would see it, it is clear that the only differences in this diagram to the student portlet is the administrator option labeled 1 and the minimization option labeled 2 allowing the staff member to administer and minimize the portlet respectively. These two screenshots are shown in the following two pages.



URL v	Comment
Cryptography:Top/Science/Math/Applications/Communication_Theory/Cryptography/Algorithms	a crypto site
http://kremlinencrypt.com/	

Figure 26: Staff Portlet View (screenshot)

The administration page is generic for all the dotLRN portlets, it displays the administration functions for each portlet individually. As you can see the browser portlet administration allows the staff member to access the adding and deleting of links (1 in the diagram) by redirecting them to the main browser UI displaying the current portlet list.


ebus Admin - Mozilla
File Edit View Go Bookmarks Tools Window Help
.LRN™
ebus
My Space Help Logout
Community Home Calendar File Storage People Administer
Document: Done (0.139 secs)
screenshots
[service0@localhost:/
ebus Admin - Mozilla
04:24
1

- **Group Administration**
  - Site Wide Administration
  - Copy this group
  - Archive this group
  - Manage Membership - Add/Remove ebus members
  - Change Enrollment Policy - Open | Closed | Needs Approval
  - Customize Portal Layout
  - Change Bulk Mail Policy - All members | Only admins can send bulk mail to this community.
  - Edit Group Properties - Change name, description, etc.
  - Create a new Limited Access Guest user - Only use this to create accounts for non-affiliated users who do not already have an account. Information about other members of this community will not be available to this user.

Subgroups

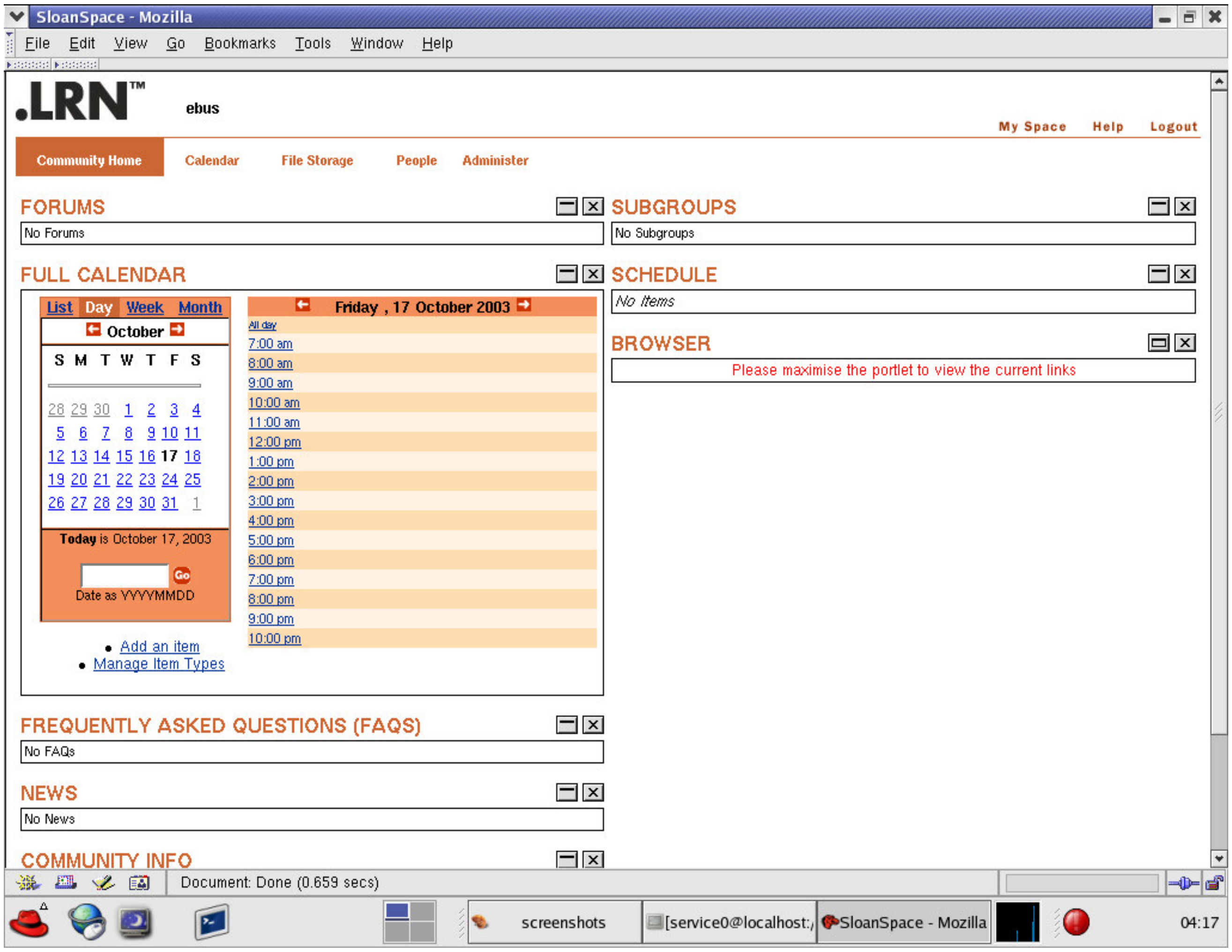
- New Subgroup

Manage Applets

- **Calendar Administration**
  - Create a new Event
  - Manage Calendar Event Types
- **News Administration**
  - Manage News
  - Add a News Item
- **Forums Administration**
  - New Forum
- **FAQ Administration**
  - New FAQ
- **Custom Portlet Administration**
  - Community Info
  - New Custom Portlet
- **browser Administration**
  - Add url links from the catalog into the portlet
  - Delete existing links from the portlet

Figure 27: Portlet Administration page (screenshot)

Below is the depiction of the minimized portlet, the portlet can only be minimized by staff members.



No Forums

No Subgroups

No Items

Please maximise the portlet to view the current links

Today is October 17, 2003

Date as YYYYMMDD

No FAQs

No News

Document: Done (0.659 secs)

Figure 28: Minimised Portlet Display (screenshot)

CHAPTER 6

DISCUSSION

6.1. System Evaluation

In order to get some feedback on both the user interface design and the usability of the browser system it was decided that certain testing had to be conducted. Since this was to be mirrored in a usual everyday university life, it was decided that a group of fellow students from the university would test the browser package and portlet giving their insights on what they thought of the user interface as well as the general usability of the browser package and portlet.

A special thanks to these students who aided me in simple testing conducted:

- Mauricio Henriquez (Bachelor of software engineering (honors) -USYD)
- Katrina Rokhlin (Computer Science – Masters in architectural design - USYD)
- Alex Taylor (Computer Science - USYD)

In order to avoid bias decisions and conclusions it was also decided to test users from other fields of education, that is those of which had a less technical computer knowledge in order to get their viewpoints, hence a few external university friends from the fields of education and dentistry also participated in the evaluation of the browser system.

In general the students noted a number of points about the system:

- The students mentioned that the interface was clear, simple and concise and allowed a good browsing interface.
- The portlet was noted as being “convenient” since the students had not seen this in any of their current E-learning platform (WebCT).
- Some of the students mentioned that the search engine could be improved if it was quicker in returning the results of the entered query.
- Some students mentioned that advanced functionality would be added if each student could personalize his own link list for personal use.

In general, the student’s evaluation was positive in terms of the goals set for the cataloging portlet system, the added views on extra functionality and faster search engine results were also positive in that they helped indicate some future work that could be added upon the existing core functionality.

6.2. Future Work

Since the browser system developed was the first of its kind for such set goals, the majority of this thesis study consisted of researching the ODP catalog and its hierarchy as well as allowing its importing and display. Hence all the additional features that were planned to be contained in the application could not be implemented in this version of the system. However, due to its development on such an evolving open architecture as OACS it has inherited the discussed reusability and modularity features and hence provided a solid base for the work of implementing catalog functionality into dotLRN.

The browser system has sparked the connection between the greatest catalog available on the web today and dotLRN, if in the future this is seen as an opportunity that can be built upon so as to allow the system to become a permanent component of dotLRN, there will be many extra opportunistic developments that can extend the system, some of these include:

- Implementation of advanced searching algorithms and techniques to improve efficiency of searches.
- Allow students to receive notifications via email about changes made to the portlet links by their staff.
- Provide mechanisms to permit students to keep personal link lists in their portal.
- Allow editors to register with the system and help categorise and expand the catalog, similar to the DMOZ project.
- Allow the integration of the portlet with a multilingual dotLRN platform to serve for all non-English speaking students.

Conclusion

This thesis paper discusses the phases in which the ODP browser and portlet package were developed and implemented on top of the existing OpenACS framework. We commenced with a brief overview on this main software architecture and the dotLRN platform that the system was to be developed upon as well as some historical considerations within this field of development. This also encompassed a brief mention of some competitive analysis on existing catalog software currently available. Following this we discussed the deep software engineering requirements that the system had to include as well as the data model and user interface design of both of the packages. Next we continued by looking even further into the development of the browser system by discussing the implementation details of the catalog importing as well as hierarchy display for the ODP data. We concluded by discussing the User Interface details and each of its available functionalities as well as on overall discussion on the results of the proposed study.

Overall it seems that the introduction of this catalog functionality into the dotLRN platform allowed a number of benefits in accessing web based resources within an E-learning community. The final results were successful in adhering to the original set goals for the system in providing:

- A Mechanism allowing the addition of the ODP data into a PostgreSQL database.
- An OACS package that allows the browsing of this catalog’s hierarchy.
- A portlet for the dotLRN platform to allow community members to access catalog resources via their E-learning portal.

Hence through this discussion and study it is clear that there is now a solution for allowing students and staff to access and integrate internet resources into their everyday E-learning portals in order to provide more convenient methods of online collaboration with such powerful tools as the ODP catalog.

References

[1] Calvo, R.A. and Peterson, D. H. (2002) OACS-A Framework For Web Applications [online], Available: **http://ausweb.scu.edu.au/aw02/papers/refereed/calvo/paper.html**. [September 2003]

[2] dotLRN Home (2003) **http://dotlrn.org/** [last accessed online September 2003]

[3] Calvo, R. A., (2003) The OpenACS E-learning infrastructure [online],
Available: **http://ausweb.scu.edu.au/aw03/papers/calvo2/paper.html** [August 2003].

[4] Alatovic, Tarik, (2001) Portals Design Documentation [online],
Available**: http://eveander.com/arsdigita/doc/design/portals.html** [August 2003]

[5] Adida, Ben, (2003) "Writing a dotLRN Package"[online],
Available**: http://dotlrn.collaboraid.net/dotlrn/doc/writing-a-dotlrn-package** [August 2003]

[6] **http://www.dmoz.org/about.html** [last accessed online October 2003]

[7] Senga information Retrieval software (2002), **www.senga.org** [last accessed online September 2003]

[8] To, Roger, (2002) A collaborative Bibliographic Catalog [October 2003]
Thesis Document - Available: University of Sydney

[9] Lerner, M. Reuven, (2003) OpenACS Templates [online],
Available: **http://www.linuxjournal.com** [September 2003]

[10] Lassila, Ora and Swick, Ralph (1999) RDF Model and Syntax Specification [online]
Available: **http://www.w3.org/TR/REC-rdf-syntax/#intro** [October 2003]

[11] Su, Pete and Quinn, Bryan, OpenACS 4.6.2 Packages [online]
Available: **http://aufrecht.org/doc/packages.html** [October 2003]

[12] dotLRN (2003) **http://www.collaboraid.biz/products/dotlrn.** [last accessed online September 2003]

[13] OpenACS (2003) **www.openACS.org** [last accessed online October 2003]

[14] PostgreSQL (2003) **www.postgresql.org** [last accessed online October 2003]

[15] Habadera, Mitsunori, (2002) <u>A collaborative Bibliographic Catalog</u> [October 2003]
Thesis Document - Available: University of Sydney

APPENDIX A

ACRONYMS

Some of the acronyms that have been referred to in this thesis document are:

Acronyms	***Definition***
ACS	ArsDigita Community System
AOL	America Online
API	Application Program Interface
APM	ACS Package Manager
DB	DataBase
DM	Data Model
GB	GigaByte
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
MIT	Massachusetts Institute of Technology .
OACS	Open Architecture Community System
ODP	Open Directory Project
OO	Object Oriented
OpenACS	Open Architecture Community System
RDBMS	Relational DataBase Management System
RDF	Resource Description Framework
SQL	Simple Query Language
TCL	Tool Command Language
UI	User Interface
URL	Uniform Resource Locator
USYD	University of Sydney (Australia)
WEG	Web Enginnering Group (USYD)
WWW	World Wide Web
XML	eXtensible Markup Language

APPENDIX B

DEVELOPER DOCS

B1. Code File Hierarchy

The browser package hierarchy is listed below as the files are organised within the package:

Path	File type	Database support
browser.info	package_spec	
sql/		
postgresql/		
browser-create.sql	data_model_create	postgresql
browser-drop.sql	data_model_drop	postgresql
browserurl-functions-create.sql	data_model	postgresql
browserurl-table-create.sql	data_model	postgresql
tcl/		
browser-procs.tcl	tcl_procs	
www/		
about.adp	content_page	
add-to-portlet.adp	content_page	
add-to-portlet.tcl	content_page	
add-to-portlet.xql	query_file	
contact.adp	content_page	
css/		
styles.css	content_page	
desc.adp	content_page	
desc.tcl	content_page	
desc.xql	query_file	
doc/		
design.html	documentation	
images/		
Thumbs.db	content_page	
arrow.jpg	content_page	
basebg1.jpg	content_page	
basebg2.jpg	content_page	
btn_about.jpg	content_page	
btn_about_dn.jpg	content_page	
btn_contact.jpg	content_page	
btn_contact_dn.jpg	content_page	
btn_home.jpg	content_page	
btn_home_dn.jpg	content_page	
btn_news.jpg	content_page	
btn_news_dn.jpg	content_page	
btn_products.jpg	content_page	
btn_products_dn.jpg	content_page	
btn_support.jpg	content_page	
btn_support_dn.jpg	content_page	
jscript.js	content_page	

lizard2a.gif	content_page
mainpic1.jpg	content_page
navbasebg.jpg	content_page
navspacer.jpg	content_page
oacs2.jpg	content_page
oacs3.jpg	content_page
pointerphoto.jpg	content_page
snap1.jpg	content_page
snap2.jpg	content_page
spacer.gif	content_page
topbg.jpg	content_page
toplogo.jpg	content_page
topmidspace.jpg	content_page
topnavbg.jpg	content_page
index.adp	content_page
index.tcl	content_page
index.xql	query_file
master.adp	content_page
master.tcl	content_page
myurl.adp	content_page
myurl.tcl	content_page
myurl.xql	query_file
search.adp	content_page
search.tcl	content_page
search.xql	query_file
sub-cat.adp	content_page
sub-cat.tcl	content_page
sub-cat.xql	query_file
support.adp	content_page
url-delete.adp	content_page
url-delete.tcl	content_page
url-delete.xql	query_file
url-edit.adp	content_page
url-edit.tcl	content_page
url-edit.xql	query_file

The dotlrn-portlet package hierarchy is listed below as the files are organised within the package:

Path	**File type**	**Database support**
dotlrn-browser.info	package_spec	
sql/		
postgresql/		
dotlrn-browser-applet-create.sql	data_model	postgresql
dotlrn-browser-applet-drop.sql	data_model	postgresql
dotlrn-browser-create.sql	data_model_create	postgresql
dotlrn-browser-drop.sql	data_model_drop	postgresql
dotlrn-browser-portlet-create.sql	data_model	postgresql
dotlrn-browser-portlet-drop.sql	data_model	postgresql
tcl/		
dotlrn-browser-applet-procs.tcl	tcl_procs	
dotlrn-browser-applet-procs.xql	query_file	
dotlrn-browser-portlet-procs.tcl	tcl_procs	
dotlrn-browser-procs-postgresql.xql	query_file	postgresql
www/		
css/		
styles.css	content_page	
doc/		
design.html	documentation	
dotlrn-browser-admin-portlet.adp	content_page	
dotlrn-browser-admin-portlet.tcl	content_page	
dotlrn-browser-portlet.adp	content_page	
dotlrn-browser-portlet.tcl	content_page	
dotlrn-browser-portlet.xql	query_file	
master.adp	content_page	
master.tcl	content_page	

B2. Installation Instructions

Versions to run browser system on :

- dotlrn 1.0
- Postgresql 7.3.4 (as it has the ltree module needed for the Database) MUST HAVE ENABLED UNICODE SUPPORT FOR POSTGRESQL by adding the flags:
  `--enable-locale` and `--enable-multibyte`.

- AOL server 3.3.1
- Red Hat 8.0
- Perl – version 5.005_03 or later (default version with RH 8)

Extra needed software / files

- structurerdf.u8 file included on CD
- xmlclean script , included on CD
- content2db script , included on CD
- structure2db script, included on CD
- DBI module for Perl – for your database server
- DBD module for Perl – for your database server
- Standard iconv utility – usually included in all OS

Requirements

- must have installed the browser package , since this installs the data structure for the Data
- also must have installed the dotLRN-browser applet , if the portlet is to be used

Instructions after the installations of the above have been made

1. Install Parser::module for perl (Should be available with Red Hat 8 +)
2. Install the DBI module for perl – follow instructions in README file - you will need root permission for this
3. Install the DBD module for perl – follow instructions in README file -you will need root permission for this

4. Copy all the files on the CD to a directory , then CD into that directory:

5. ODP exports i.e. RDF files are known to be non-standard files, they contain many errors such as UTF-8 encoding errors, XML errors and RDF errors. Hence before we insert the RDF data into the data base we must clean up these errors, by using the iconv utility. The first time we use this will be on the structurerdf.u8 file, to clean up the file for the structure:

a. clean up all the invalid UTF-8 sequences :

COMMAND :

```
iconv -c -f UTF-8 -t UTF-8 structurerdf.u8 > valid-structure.u8
```

Please note : This will take a few minutes depending on the machine being used.

- now we have directed the clean output of the RDF to the valid-structure.u8 file

b. clean up all the invalid XML characters in the file:

COMMAND:

```
Cat valid-structure.u8 | ./xmlclean.pl > valid-structure.clean
```

c. You can now remove unnecessary files with the following commands

COMMANDS:

```
rm structurerdf.u8 valid-structure.u8
```

6.a. Must edit the structure2db.pl to log into the current OACS/dotLRN Database.

COMMANDS:

```
Gedit structure2db.pl
```

- now scroll down to the comment "connect to database:"
- change "service0" string to the name of your database

- also do the same for the password change "password" to password of the database owner , if no password exists leave it as "password"

b. save the file structure2db.pl and exit the editor

c. run the structure2db.pl script

COMMAND: **./ structure2db.pl**

Note : it should start printing like the text below :

```
Loading structure records:
Record: 1000
Record: 2000
Record: 3000
  .
  .
  .
  .
Record: 568000
```

568585 loaded (if a newer structure RDF is used other than the given one on the CD there may be more records than this)

7. Repeat the process for the content.rdf.u8 file

a. clean up all the invalid UTF-8 sequences :

COMMAND :

```
iconv -c -f UTF-8 -t UTF-8 content.rdf.u8 > valid-content.u8
```

Please note : This will take a few minutes depending on the machine being used.

- now we have directed the clean output of the RDF to the valid-content.u8 file

b. clean up all the invalid XML characters in the file:

COMMAND:

```
Cat valid-content.u8 | ./xmlclean.pl > valid-content.clean
```

Please note : This will take up to a couple of hours depending on the machine being used.

c. You can now remove unnecessary files with the following commands

COMMANDS:

```
rm content.rdf.u8 valid-content.u8
```

d. Must edit the content2db.pl to log into the current OACS/dotLRN Database.

COMMANDS:

```
Gedit content2db.pl
```

- now scroll down to the comment “connect to database:”
- change “service0” string to the name of your database

- also do the same for the password change “password” to password of the database owner , if no password exists leave it as “password”

e. save the file content2db.pl and exit the editor
f. run the content2db.pl script

COMMAND: **./ content2db.pl**

Note : it should start printing like the text below :

```
Loading structure records:
Record: 1000
Record: 2000
Record: 3000
  .
  .
```

g. After this is completed , the ODP should be available in the database.

APPENDIX C

INTERACTIVE CD

C1. CD Contents

Attached to this thesis document is a CD containing a number of useful resources to help the understanding of the browser system as well as the installing and configuration of the ODP catalog into PostgreSQL. It contains

Software – All software versions used in the development and installation of the browser system that are recommended for this version of the browser system

.

- **DotLRN 1.0 –** dotLRN-1.0.tar
- **AOLServer 3.3 –** aolserver3.3oacs1.tar
- **PostgreSQL 7.3.4 –** postgresql-7.3.4.tar

Import – In this directory is contained the perl modules needed to import the ODP into postgresql, as well the scripts that will perform this process. Please note the structure.rdf.u8 file is included on this CD but the content.rdf.u8 is not due to its size, this will have to be downloaded from dmoz.org.

- **DBI perl module -** DBI-1.38.tar
- **DBD perl module -** DBD-Pg-1.22.tar
- **xmlclean.pl –** script to clean the RDF files
- **content2db.pl script –** script to import the ODP content into PG
- **structure2db.pl script –** script to import the content into PG
- **structurerdf.u8 –** ODP data hierarchy structure

Browser System Code - The browser system packages.

- **browser package**
- **dotLRN-browser package**