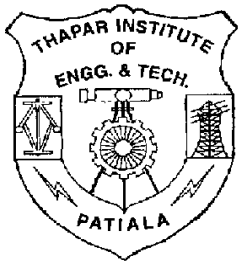
PORTLETS: USER-INTERFACE COMPONENTS TO GRID

A Thesis

Submitted in partial fulfillment of the requirement
for the award of degree of

Master of Engineering
In
Software Engineering



Under the Supervision of

Dr. Seema Bawa

Professor

Computer Science and Engineering Department
Batch 2003-2005

Submitted By
Anju Singla
(8033103)

Computer Science & Engineering Department

Thapar Institute of Engineering & Technology

(Deemed University), Patiala-147004 (India).

May 2005

ABSTRACT

Grids are becoming platforms for high-performance and distributed computing. Grids benefit users by permitting them to access heterogeneous resources, such as machines, data, people and devices that are distributed geographically and organizationally.

The advent of Portlet API offers Grid Portal vendors a common programming model that will allow developers to more rapidly plug new functionality into a Portal server, making it easily available for its consumer base. The Portlet model gives users a flexible, easy-to-use interface, and it gives Portal developers a model to create pluggable and dynamic application support. A Portlet provides a “mini-window” within a Portal page.

The main focus of this work is on” Grid Portlet”. It describes the Grid Portlets, Concept behind them, their importance in Portal design and how they interact with the Portal. It is also discussed how Portlets are different from the Servlets, Portlet standards that are available for the development of the Grid portal i.e. WSRP and JSR-168. Finally, design and implementation details of various Portlets like *Login Portlet, Our Service Portlet, Search Portlet, News Portlet, Weather Portlet, Date and Time Portlet, Calculator Portlet, Service Submission Portlet, New User Portlet, Portal Administrator Portlet, Install Portlet, Uninstall Portlet, Layout Manager Portlet, Credential Portlet, Resource Browse Portlet* etc *a*re Elaborated.

The features provided by these Portlets are:

- Common infrastructure for managing content.
- No need to reinvent login, user customization services.
- Add our own service implementation in a well-defined way.
- Content (and service user interfaces) are added in a well defined way

DECLARATION

I hereby certify that the work which is being presented in the thesis entitled, ***"Portlets: User-Interface components to Grid"*** *in* partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering at Computer Science and Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Dr. Seema Bawa.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other University.

(Anju Singla)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Seema Bawa)
Professor
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
PATIALA- 147004

Countersigned by

(Mr. R.S Salaria)
Head
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
PATIALA- 147004

(Dr. D. S. Bawa)
Dean Of Academic Affairs
Thapar Institute of Engineering and Technology
PATIALA- 147004

ACKNOWLEDGEMENT

I wish to express my deep gratitude to Dr. Seema Bawa, Professor, Computer Science and Engineering Department for providing her individual guidance and support throughout the Thesis work.

I am also thankful to Mr. Maninder Singh, Assistant Professor, Computer Science and Engineering Department for providing timely assistance and encouragement which went long way in successful completion of my thesis.

I am also thankful to Mr. R. S. Salaria, Head, and Mr. Rajesh Bhatia, P.G Coordinator, Computer Science and Engineering Department, for the motivation and inspiration that triggered me for my thesis work.

I would also like to thank all the staff members and my Co-students who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis.

I am also thankful to the authors whose works I have consulted and quoted in this work. Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Anju Singla.
(8033103).

TABLE OF CONTENTS

LIST OF FIGURES

ORGANISATION OF THESIS

The First chapter briefly introduces Grid computing, gives the history of the Grid , tells the taxonomy of the Grid and the working of the Grid.

The Second chapter of this thesis is devoted to the introduction of the Grid portals and Grid Portlets, tells us how the Portlets differ from the Servlets, how the Portlets work, how the portal and Portlets interact with each other.

The Third chapter provides a brief overview of the various available standards for the Portlet development and gives the details of the available Portlets from the different vendors.

The Fourth chapter concerns the design and implementation of the application. Finally, concluding thesis in Fifth chapter with future scope.

CHAPTER 1

INTRODUCTION

Grids are becoming platforms for high-performance and distributed computing. Grids benefit users by permitting them to access heterogeneous resources, such as machines, data, people and devices that are distributed geographically and organizationally. It benefits organizations by permitting them to offer unused resources on existing hardware and software. They allow users to execute compute intensive problems whose computational requirements cannot be satisfied by a single machine. A computational Grid environment behaves like a virtual organization consisting of distributed resources. A virtual organization is a set of individuals and institutions defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing takes place. Grid computing has emerged as the new paradigm in distributed computing and has undergone significant developments over recent years.

Grid computing is the next generation IT infrastructure that promises to transform the way organizations and individuals compute, communicate and collaborate. It offers untapped processing cycles from networks of computers spanning vast geographical boundaries.

Grid focus on ensembles of distributed heterogeneous resources used as a platform for high performance computing. It can also be used to create virtual databases that pool the content from different offices, and take on large engineering and modeling projects beyond the scope of any single computer.

1.1 HISTORY OF GRID

Many of the basic ideas behind the Grid have been around in one form or other throughout the history of computing. For example, one of the "novel" ideas of the Grid is sharing computing power.

Nowadays, where most people have more than enough computing power on their own PC, sharing is unnecessary for most purposes. But back in the sixties and seventies, sharing computer power was essential. At that time, computing was dominated by huge mainframe computers, which had to be shared by whole organizations.

In 1965 the developers of an operating system called Multics (an ancestor of Unix, which in turn is an ancestor of Linux - a popular operating system today) presented a vision of "computing as an utility" - in many ways uncannily like the Grid vision today. Access to the computing resources was envisioned to be exactly like water, gas and electricity - something which the client connects to and pays for according to the amount of use. Ironically, "utility computing" is all the rage again these days, and used more or less as a synonym for the Grid by some people.

So, yes, there is a certain amount of "reinventing the wheel" going on in developing the Grid. However, each time the wheel is reinvented; it is reinvented in a much more powerful form, because computer processors, memories and networks improve at an exponential rate which is associated with Moore's law.

Because of the huge improvements of the underlying hardware (typically more than a factor of 100x every decade), it is fair to say that reinvented wheels are qualitatively different solutions, not just small improvements on their predecess.

It all Started With Distributed Computing

In an effort to generate processing power for meeting workload challenges, computing resources were initially aggregated and allocated from across a business. The idea was to match the supply of processing cycles with the demand created by applications. This concept, known as distributed computing, is now a ubiquitous solution practiced by leading organizations around the world. It ensures continuous computing availability despite scheduled maintenance, power outages, and unexpected failures.

Enter Grid Computing

The same idea of sharing resources has paved way for Grid computing; an approach of using untapped processing cycles from across geographical boundaries with a far wider scale and scope. Grid computing, in effect, provides a global reach to distributed computing.

1.2 TAXONOMY BEHIND GRID COMPUTING

Before going into the details of Grid computing, we will discuss the taxonomy that how the Grid computing came into existence. i.e. how we moved from the standalone computing to the Grid computing

1.2.1 Standalone Computing

The standalone configuration is liked by the organizations and individuals who do local things that are confined to one location. In this type of configuration the standard program (e.g., an accounting package, image editor, or word processor) or a custom program (e.g., company-specific inventory management, actuarial table calculation, and mortgage calculator) takes input on that computer, does some calculations, stores intermediate data, and produces output. For a standalone computing system there is no network. The problem with the standalone system is that the sharing of data is very difficult i.e. we have to put it on the disk the physically transport it where we want to use that standalone or a custom program. So to overcome this limitation of standalone, there came the Transaction processing system.

1.2.2 Transaction Processing System

Transaction processing systems have a single server with custom software doing computation and data storage. Each client machine is usually a standard "smart" terminal or a personal computer running a standard program acting as one. The input and output occurs on the PC. The PC has no storage and is only connected directly to the single

server through a dedicated communications system. Part of the input sometimes comes from other sources.

This configuration let organizations do applications that involved people at diverse locations involved with a single database. It was best for applications that were worth dedicating the communications system as well as the dedicated client machines. Since the client machines were standard, their cost and maintenance could be commodities. The applications were often mission-critical to the company employing them. Interacting with computers started to be integral to the operation of an enterprise, for example at banks or airlines.

1.2.3 Distributed Computing

Computers started being connected to one another through networks so that data such as files and email could be exchanged. Over time, more and more of the computer's capabilities are being shared over the network creating the realm of distributed computing.

Distributed computing is a programming model in which processing occurs in many different places (or nodes) around a network. Processing can occur wherever it makes the most sense, whether that is on a server, Web site, personal computer, handheld device, or other smart device.

In other words we can say that Distributed computing is simply applying the two old sayings to the realm of computer resources.

"Many hands make light work" refers to the idea that you can take a task and break it down so that many different people or computers can work on it at the same time. Then it only takes a small amount of work by each computer (or human) to complete the bigger task [22].

"The whole is greater than the sum of its parts" applies to the fact that when a number of computers work in conjunction the result is something that can not be achieved by the computers working alone [22].

1.2.3.1 Benefits of Distributed Computing

- It can continue to operate even if some of its parts are missing, this resulting in much better reliability of the system than a single computer could ever provide.
- A distributed operating system takes the abstraction to a higher level, and allows hides from the application where things are. The application can use things on any of many computers just as if it were one big computer.
- A distributed operating system will also provide for some sort of security across these multiple computers, as well as control the network communication paths between them.

1.2.3.2 Services Provided by the Distributed System

- They control what application gets to use the CPU and handle switching control between multiple applications.
- They also manage use of RAM and disk storage. Controlling who has access to which resources of the computer (or computers) is another issue that the operating system handles.
- Coordinated the multiple machines. As systems grow larger handling them can be complicated by the fact that not one person controls all of the machines so the security policies on one machine may not be the same as on another.

1.2.3.3 Problems with Distributed System

Some problems can be broken down into very tiny pieces of work that can be done in parallel. Other problems are such that you need the results of step one to do step two and the results of step two to do step three and so on. We also don't want the chunk of work to be done to be too big of a chunk because then you can't spread it out over enough machines to make the thing run quickly.

A computer system with multiple processors in a single machine can handle very fine-grained problems well, while a system built from computers distributed over the Internet can handle only coarse-grained problems.

1.3 GRID COMPUTING

The same idea of sharing resources has paved way for Grid computing; an approach of using untapped processing cycles from across geographical boundaries with a far wider scale and scope. Grid computing, in effect, provides a global reach to distributed computing. It promises lowering of the total computing cost with on-demand, reliable and inexpensive access to the vast, available computing resource.

The Grid is a service for sharing computer power and data storage capacity over the Internet. The Grid goes well beyond simple communication between computers, and aims ultimately to turn the global network of computers into one vast computational network.

Grid focus on ensembles of distributed heterogeneous resources used as a platform for high performance computing. We can define Grid computing as

An ambitious and exciting global effort to develop an environment in which individual users can access computers, databases and experimental facilities simply and transparently, without having to consider where those facilities are located[24].

In general, the need for high performance **distributed computing** has driven the development of a set of **Grid computing** technologies that allows resource sharing across multiple domains.

Grid computing is defined as secure, controlled resource sharing across heterogeneous platforms within multiple administration domains creating 'virtual organizations' [2].

1.3.1 How Does Grid Computing Work?

The concept of computational Grid has been inspired by the "electric power Grid", in which a user could obtain electric power from any power station present on the electric Grid irrespective of its location, in any easy and reliable manner. When we require additional electricity we have to just plug into a power Grid to access additional electricity on demand, similarly we plug into a computer Grid to access additional

computing power on demand using the cheapest possible resource. Basically, user submits a job to the Grid through an interface on his computer that serves as a portal to the Grid. Special Grid management software accepts the job and breaks it down into hundreds or even thousands of independent tasks, then locates idle processors and distributes the tasks among them. Finally, it aggregates the works and spits out the results.

The theory behind “Grid computing” is not to buy more computational resources but to borrow the power of the computational resources you need from where it’s not being used.

1.3.2 The Ideas behind the Grid

Of course, there are many big ideas behind the Grid. And of course, some of them have been around long before the name Grid appeared. Nevertheless, if you look at where the software engineers and developers who are building the Grid are spending their time and effort, then the basic areas are

- The most important is the **sharing of resources** on a global scale. This is the very essence of the Grid.
- **Security** is a critical aspect of the Grid, since there must be a very high level of trust between resource providers and users, who will often never know who each other are. Sharing resources is, fundamentally, in conflict with the ever more conservative security policies being applied at individual computer centers and on individual PCs. So getting Grid security right is crucial.
- **Balance the load** on the resources, so that computers everywhere are used more efficiently and queues for access to advanced computing resources can be shortened.
- Communications networks have to ensure that **distance no longer matters doing** a calculation on the other side of the globe, instead of just next door should not result in any significant reduction in speed.

- Finally, **open standards**, which are needed in order to make sure that R&D worldwide can contribute in a constructive way to the development of the Grid, and that industry will be prepared to invest in developing commercial Grid services and infrastructure.

1.3.3 Key Benefits of Grid Computing

- Global reach - resource sharing for a globally scattered IT infrastructure.
- Tremendous potential for productivity and cost savings - transparent collaboration, greater storage capacity than traditional networks, tremendous processing power (available from resources across multiple locations and time zones), no expenditure on supercomputers.
- Extensive Collaboration - greater sharing of software applications and accessibility of remote databases around the world.
- Resilient IT Infrastructure - ability to respond to minor or major disasters.

1.3.4 Issues Related to Grid

- Grid must be able to quickly ascertain what resources are available on any computer that joins it -- and more to the point, the Grid shouldn't be bogged down by a slow or outdated system. Avoiding the "least common denominator" problem is somewhat high on the technical "to-do" list.
- Making applications work in a Grid environment. Right now, most applications work in server or desktop environments.
- Security is definitely an enormous requirement -- after all, you don't want just anyone getting access to Grid resources.
- We have worry about how the data is shared, chopped, sifted, moved around, secured, and managed. The user or application that requested the data needs to be the only entity that gets the data back, and it has to be intelligible.

1.3.5 Types of Grids

Grid computing can be used in a variety of ways to address various kinds of application requirements. Often, Grids are categorized by the type of solutions that they best address[2]. There are no hard boundaries between these Grid types and often Grids may be a combination of two or more of these.

Computational Grid: A computational Grid is focused on setting aside resources specifically for computing power. In this type of Grid, most of the machines are high-performance servers.

Scavenging Grid: A scavenging Grid is most commonly used with large numbers of desktop machines. Machines are scavenged for available CPU cycles and other resources. Owners of the desktop machines are usually given control over when their resources are available to participate in the Grid.

Data Grid: A data Grid is responsible for housing and providing access to data across multiple organizations. Users are not concerned with where this data is located as long as they have access to the data.

1.4 APPLICATION AREAS FOR GRID COMPUTING

Grids are persistent environments that enable software applications to integrate instruments, displays, computational and information resources that are managed by diverse organizations in widespread locations. Most of the Grid applications have their origin in applications which solve large problems where the computational resources are distributed. Some of the areas where Grid computing is applicable are [24].

- **Medical Applications**: In diagnostics huge amounts of data are generated at one place by specialized devices. These data have to be transported to the specialists, possibly located at several locations, while the patient might be at a third location. The task of a Grid in this scenario is to prepare and transport the medical data, so that they are available at the right location at the right time. Challenges within this group

of applications include security (integrity of data, and making sure that the patient's data do not fall into unwanted hands), synchronization (deliver the right data at the right time to the right destination), ...

- **Support for multinational enterprises**: Multinational enterprises work at several locations in several time zones. Data, e.g. multimedia data from inspections, must be pre-processed and forwarded to specialists who can take decisions. A Grid infrastructure can support these applications. Other applications supporting Mobility are within the same frame.

- **Multimedia Applications**: Several Multimedia Applications can make use of a Grid for processing media streams. Within multimedia QoS control is most important. Applications could include the handling of Digital Rights Management. E.g., multimedia data can be watermarked, scrambled etc. Other typical Grid applications would include indexing of media streams. Multiplayer Games would be another example within this class of applications. Applications within (scientific) visualization and computer graphics have traditionally been a candidate for distributed (and Grid) computing.

- **Intrusion Detection**: Within Security there are several applications that can be handled better with a Grid than by single computers. For intrusion detection and forensics a Grid can be enabled to find evidence within these enormous masses of data. However, in order to protect personal data information, these Grids have special security needs.

Sun Microsystems was the first large computer vendor to make Grid computing available for general-purpose commercial use [11]. It was developed to optimize utilization of workstations, servers and dedicated compute farms, an area of interest to sun.

This Chapter describes what is Grid, from where it has evolved and what are the various application areas for grid. The next chapter will discus about the Grid Portlet, their architecture, their approach and nature of portlets.

CHAPTER 2

GRID PORTLETS

Grid is type of parallel and distributed system that enables the sharing , selection, and aggregation of geographically distributed resources dynamically at run time depending on their availability, capability, performance, cost, user quality-of–self-service requirement[23].

In Simple words we can say that Grid is,

- to coordinate resources that are not subject to centralized control.
- to use standard, open, general-purpose protocols and interfaces.
- to deliver nontrivial qualities of service.

"A computational Grid is a hardware and a software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [11].

Dependable service means assurance to user that they will receive predictable, sustained, and often high levels of performance from the diverse components that constitute the Grid in the absence of such assurances; applications will not be written or used. The need for **consistency** of service is a second fundamental concern. As with electric power, we need standard services, accessible via standard interfaces, and operating within standard parameters. Without such standards, application development and pervasive use are impractical. **Pervasive** access allows us to count on services always being available, within whatever environment we expect to move. Pervasiveness does not imply that resources are everywhere or are universally accessible. We cannot access electric power in a new home until wire has been laid. Finally, an infrastructure

offers **inexpensive** (relative to income) access if it is to be broadly accepted and used. Homeowners and industrialists both make use of remote billion- dollar power plants on a daily basis because the cost to them is reasonable.

Grid enables the sharing, selection, and aggregation of geographically distributed resources dynamically at run time. So user submits a job to the Grid through an interface on his computer that serves as a portal to the Grid

Portal is web site that act as hub gateway or window to other information and service on the web. E.g. common web portal are yahoo, Netscape etc.

Grid portal Grid Portals build upon the familiar Web portal model, such as Yahoo or Amazon, to deliver the benefits of Grid computing to virtual communities of users, providing a single access point to Grid services and resources.

*"**A Portal is a concept for a web site that serves as a single gateway to a company's information and knowledge base for employees and possibly for customers, business partners, and the general public as well."***

2.1 WHY WE NEED PORTALS

- Portals offer many advantages over other software applications. First, they provide a single point of entry for employees, partners, and customers.
- Second, portals can access Web services transparently from any device in virtually any location.

2.2 GENERAL GRID PORTAL DESIGN

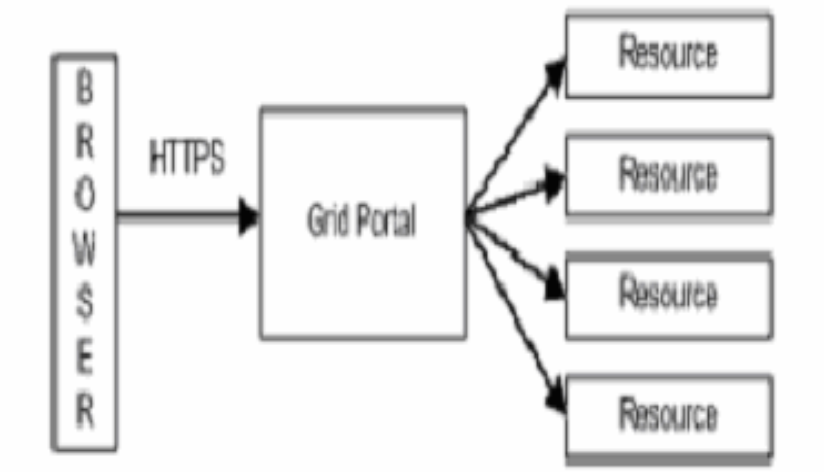


Figure 2.1 Grid Portal Design

The Grid computing architecture is supported by the above shown portal design. In this portal design the user interact with the Grid by sending its request through the Grid portal which is executed by the support of HTTPS browsers. Further these Grid interact with the required resource and performs the tasks it is asked for**.**

2.3 WHAT DO USERS WANT FROM A GRID PORTAL?

The primary requirements for a Grid portal system from a user' perspective involves access to Grid services. These include

- Security services.
- Remote File management.
- Access to information services.
- Application interfaces.
- Access to collaboration

*"**A Portal Environment (which is an integral part of a Portal) provides system-level services for supporting Portlets ("mini-applications") that reside in the Portal**"*.

Portlets is specialized content area that occupies a small window of a portal page.

E.g. Portlets are weather info. News flashes, stock tickers etc.

"Portlets are the pluggable user-interface components, to provide a presentation layer to information systems."

In other words we can say that

- The Portlet lays the foundation for a new open-standard for Web portal development frameworks.
- Portlets define an API for building atomic, composable visual interfaces to Web content or service providers.
- A Portlet provides a "mini-window" within a portal page. Multiple Portlets can be composed in a portal page.

Portlets are contained by the Portlet containers. A Portlet container runs Portlets and provides them with the required runtime environment. It manages the lifecycle of the Portlet. It provides persistent storage for Portlet preferences.

2.4 WHY DO WE NEED PORTLETS

- The Portlet model gives users a flexible, easy-to-use interface, and it gives portal developers a model to create pluggable and dynamic application support.
- Portlets, that allows portal components to be shared between projects.
- Web component easily plug into and run in enclosing web applications like portal.

- Uses existing web services to access information can also be described, wrapped and published as web services enables the sharing of Portlets.
- Facilitates both users and developers.

2.5 VALUE OF PORTLET APPROACH

- With Portlets, we have a common infrastructure for managing content.
- We don’t have to reinvent login, user customization services.
- We may add our own service implementation in a well-defined way.
- Content (and service user interfaces) are added in a well defined way.

2.6 PORTAL –PORTLET INTERACTION

This section describes how the Portlets interact with the Portals. The interaction is always started by the user. First of all, the user sends the request to the Portal. Portal further sends the user request to Portal container which is the interacting medium between the Portal and Portlets. Portal containers are one where Portlets are contained. Then the Portal container sends the request of the Portal to the Portlet. In the backend Portlet container runs Portlets and provides them with the required runtime environment. It also manages the lifecycle of the Portlet. Then Portlet processes the request, performs the required action and send back the processed outcomes to the container. The Portlet container is not responsible for aggregating the content produced by the Portlets. It just passes the content to the Portal, which is further responsible to handle the aggregation of the content. Then the aggregated results are passed to the user. As we can have multiple Portlets on a single Portal so while the user is interacting with one Portlet, Portal renders the functionality of other Portlets in parallel.

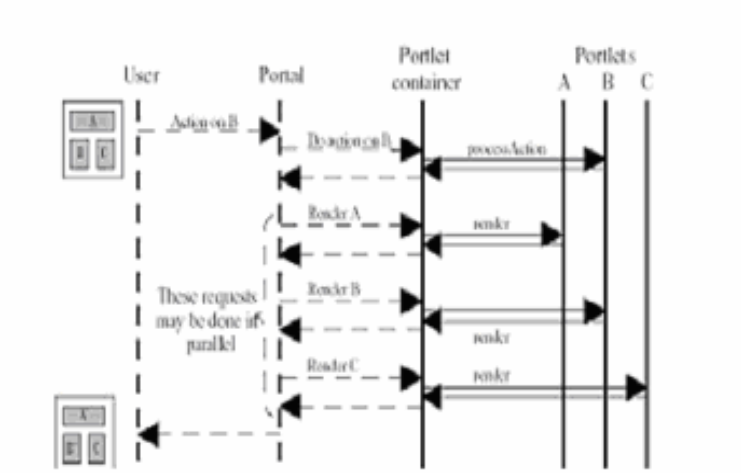


Figure 2.2: How Portlets interact with the Portal Ref [27]

2.7 HOW DOES PORTLETS WORK?

Portlets are pieces of a larger Portal; they have the ability to communicate with other Portlets and to be affected by other Portlets in a single request. This inter-Portlet communication provides a way to create a dynamic Portlet application crossing multiple Portlets on the same page.

- A Portlet is a Java technology web based component, managed by a Portlet container, which processes requests and generates dynamic content. Portlets are the pluggable user-interface components, to provide a presentation layer to information systems.

- Web clients interact with the Portlets via a request/response paradigm implemented by the portal. Normally, users interact with content produced by Portlets, for example by following links or submitting forms, resulting in Portlet actions being received by the portal, which are forwarded by it to the Portlets targeted by the user's interactions.

- The content generated by a Portlet may vary from one user to another depending on the user configuration for the Portlet.

2.8 PORTLETS AND SERVLETS

Portlets has extended the approach used by the Servlets for accessing the information. In this section we will address some of the conceptual similarities and differences between Servlets and Portlets.

2.8.1 Similarities with Servlets:

- Portlets are Java based technology based web components.
- Portlets are managed by a specialized container.
- Portlets generate dynamic content.
- Portlets interact with web client via a request/response paradigm.
- A Portlet's lifecycle is managed by a container.

2.8.2 Difference from Servlets:

- Portlets only generate markup fragments, not complete documents. The portal aggregates Portlet markup fragments into a complete portal page.
- Portlets are not directly bound to a URL.
- Web clients interact with Portlets through a portal system.
- Portlets have a more refined request handling, action request.
- Portlets have predefined Portlet modes and windows that indicate the function that the Portlet is performing and the amount of real state in the portal page.
- Portlets can exist many times in a portal page.

2.8.3 Portlets access the following extra functionality not provided by Servlets:

- Portlets have means for accessing and storing persistent configuration and customization data.
- Portlets have access to user profile information.

- Portlets can store transient data in the Portlet session in two different scopes: the application wide scope and the Portlet private scope.

2.8.4 Portlets do not access the following functionality provided by Servlets:

- Setting characters set encoding of the response.
- Setting HTTP headers on the response.
- The URL of the client request to the portal.

2.9 TYPES OF PORTLETS

The Portlets can be classified into two basic types:

- User Portlets
- Administrative Portlets

2.9.1 User Portlets

- **Login/Logout Portlet** enables user to logon/logout Pluggable authentication modules with which we provide support for database password and credential based logins. Configurable option enables new users to request an account.
- **Profile Settings Portlet** enables users to configure personal information e.g. name, email, preferences
- **layout Configuration Portlet** enables users to configure their layout including placement of Portlets within tabs, tab and subtab titles
- **Subscription Manager Portlet** enables users to select set of Portlets they would like to subscribe/unsubscribe.

2.9.2. Administrative Portlets

- **User Manager Portlet** enables administrator to create/delete/edit portal usersGroup.

- **Manager Portlet** enables administrator to add/remove users to/from Portlet Groups and also Enables administrator to select whether a group is public or private (public means anyone can join, private requires an administrator approval)
- **Portlet Manager Portlet** enables administrator to start, stop, or redeploy a Portlet application.

2.10 LAYOUT & PRESENTATION OF GRID PORTLET

- Portlets define how to construct and deliver Web content as modular components within a Web page.
- Portlets can be “maximized” or “minimized” within a Web page.
- Users can choose to which Portlets they want to be “subscribed”.

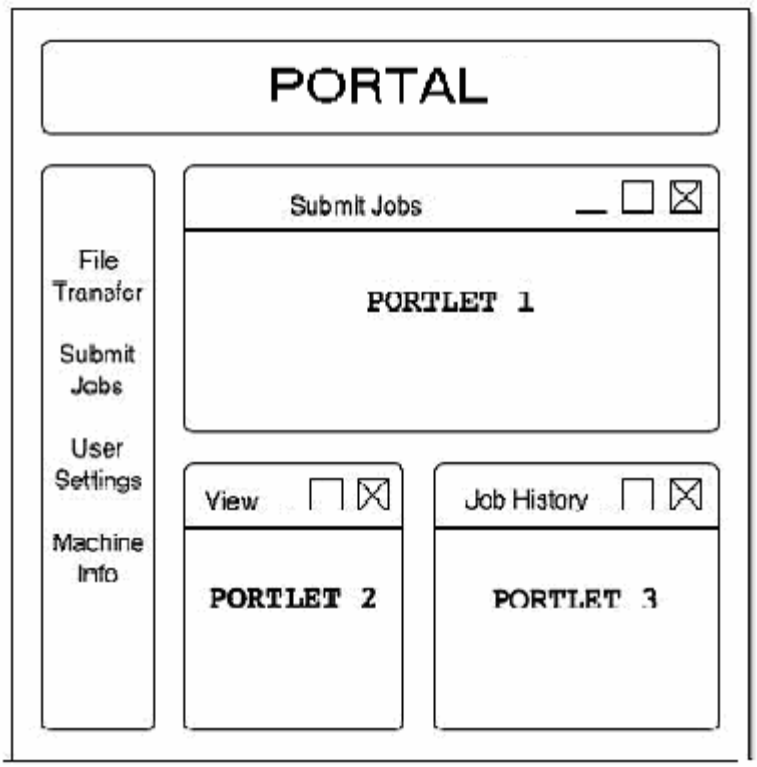


Figure 2.3 Layout and Presentation of Grid Portlets

2.11 LAYERED PORTLET MODEL

In past Portlet mechanisms, Portlets are always designed for certain applications; the implementation of basic functions and application functions are not treated respectively, which resulted in certain redundancy work and inflexibility in constructing portals. To overcome this problem comes a Layered ***Portlet Model.*** The Basic Portlet is in charge of interoperation with basic Grid services. It implements the functions similar to Basic Portal's, but in a component manner. And the Application Portlet is equivalent to the Application Portal. In Application Portlet certain interfaces are provided for Basic Portlets or some other lower-layered more specific Application Portlets. In a Portal the Basic Portlet and the Application Portlet may not necessarily come from the same Grid node. One Basic Portlet can be invoked in one or more Application Portlets, vice versa, an Application Portlet can also invoke several Basic Portlets [17].

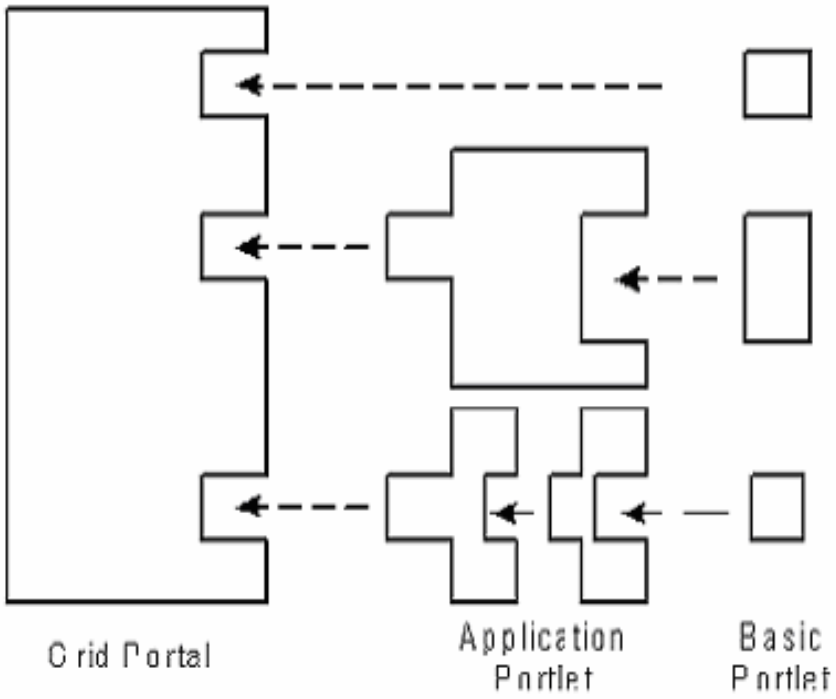


Figure 2.4 Layered Portlet Model Ref [17]

2.11.1 Characters of Layered Portlet

- **Each part be developed independently**: The Grid Portal designer only need to consider how to integrate multiple Portlets into a user-oriented interface on the front end, the Grid Node service provider only need to encapsulate their services into various Basic Portlets and Application Portlets. Finally, the middleware developers only need to deal with the Application level logic based on those low-level Portlets [24].

- **Portlets components promote reusability and reduce the Complexity of application development.**

- **Blazes a possible economical market for Portal construction**

In a market, every Portlet has its own weight, and any Portal and Portlet prefer to select components with high Performance/Cost ratio. And the competition is based on the dynamic binding of Portlets. For instance, for an Application Portlet that bases on several lower-levered Application Portlets, when it finds a new lower-levered Application Portlet with the same function but better performance, it can dynamically binds to that Application Portlet immediately [24].

2.12 PORTLET LIFE CYCLE

A Portlet is managed through well defined life cycle which has 3 phases in it. The phases are as following

1. Init
2. Render
3. Destroy

At the time of Portlet creation, a Portlet's init() method is called. The init method is only called once, during the Portlet creation phase. The lifetime of a Portlet is controlled by how long a Portlet stays in the cache. When a Portlet is removed from the cache, the Portlet is effectively removed from memory and is no longer accessible. If a Portlet is removed from the cache, and then requested again, the Portlet will have its init method

called again, since a new instance is created of the Portlet. Both the system administrator and programmer can control the lifetime of a Portlet by controlling how long it lives in the cache.

Figure 2.5 Portlet Life Cycle Ref [4]

The below shown figure gives us the more detailed description of the Portlet life cycle in two dimensions. It shows that the handle request handles the requests like the login and logout request. Logout and login are just the example of services, in reality their can be number of other services.

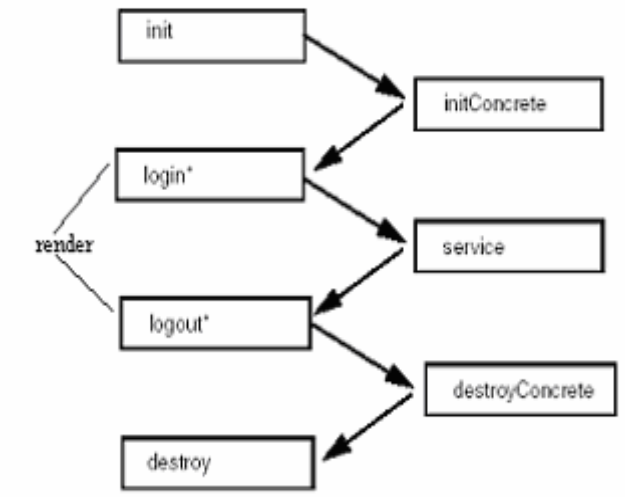


Figure 2.5 Portlet Life Cycle

2.13 PORTLET STATES

Portlet states allow users to change how the Portlet window is displayed within the portal. In a browser, users invoke these states with icons in the title bar in the same way that Windows applications are manipulated. Portlet states are maintained in the PortletWindow.State object with a Boolean value.

• **Normal** When a Portlet is initially constructed on the portal page, it is displayed in its normal state – arranged on the page along with other Portlets.

• **Maximized** When a Portlet is maximized, it is displayed in the entire body of the portal, replacing the view of other Portlets.

• **Minimized** When a Portlet is minimized, only the Portlet title bar is displayed on the portal page.

This chapter, describes the Portals and Portlets details along with their need, design, how they interact with each other, type of Portlets and its life cycle. The next chapter describes the various standards available for the Portlets and the various existing Portlets.

CHAPTER 3

EXISTING PORTLETS STANDARD

The arguments for portal standards reflect the benefits historically derived from broad agreements on technology infrastructure. They can deliver value in terms of flexibility, cost reduction, investment protection, and innovation foundation.

The most important benefit may well be flexibility for both producers and consumers of Portlet technology. Standards help reduce the conflict between best-of-breed versus single-provider acquisitions. They clarify business perspectives for producers by solidifying borders between layers of the whole product stack. In a technology environment where rapid improvement is the norm, they make it easier for consumers and producers to swap components and whole-configuration strategies in and out to quickly meet new business requirements.

Cost-reduction is a predictable consequence of standards adoption. Standards reduce expensive customization efforts across at least the portion of the delivered service environment where they come into play. Since standards also remove or commoditize whole sets of development efforts from software companies, they tend to have a downward influence on the level of enterprise license pricing as well.

Investment protection is another critical area. In information technology lack of interoperability is often the norm, and writing off significant earlier investments in technology is all too common. So standards provide real value by offering a foundation for innovation for new applications. Think of what font standards provided for desktop publishing, what Windows provided for the entire landscape of desktop applications, or what HTML provided for the Web.

3.1 THE STANDARDS EFFORTS FOR PORTLETS

WSRP AND JSR-168

Two standards efforts are now under way to support portal interoperability.

The first is Web Services for Remote Portals (WSRP), which began at Epicentric in 2001 and is currently under the management of Organization for the Advancement of Structured Information Standards (OASIS). A representative of IBM chairs the WSRP technical committee, with active participation from a broad range of software firms active in portal technology, including Sun Microsystems, Plumtree, Oracle, and others. Leading consumer firms of the technology include information services firms Reed Elsevier and Factiva, B2B leader CommerceOne, France Telecom, and the U.S. Navy.

The second effort, begun in 2002, is a project of the Java Community Process (JCP). Dubbed JSR for Java Specification Request (JSR-168), this is a complementary specification to WSRP focused on portal interoperability for the Java development environment. The JCP, the body guiding the evolution of Java technology standards, has attracted leading technology companies, open-source contributors, individual developers, and academics to these efforts. Co-led by individuals from Sun and IBM, the group responsible for drafting the specification comprises representatives from Apache, ATG, BEA, Boeing, Borland, BroadVision, Citrix, Epicentric/Vignette, Fujitsu, Hitachi, HP, IONA, Oracle, SAP, SAS, SilverStream, and Sybase.

These two groups are in ongoing communication, view their efforts as complementary, and are actively working to make their approaches consistent. The main differences lie in WSRP's focus on XML and Web services deployment and on WSRP's inclusion of interchange with remote Portlets as part of the specification. The WSRP specification anticipates the interoperability of Portlet services developed either in the Java environment or in the Microsoft .NET environment. JSR-168 is directed at the immediate needs of the Java developer community, and the ability to share JSR-168 Portlets between any of the J2EE portal environments, such as IBM Websphere Portal and BEA WebLogic.

Both groups are in the process of specification drafting and developing proof-of-concept applications within their respective working groups.

The absence of a common Portlet API standard for portals have the problem that the application providers who provide Portlets typically select a few major portal platforms and yet have to implement more than one version of a particular Portlet. From a portal server perspective, only a subset of Portlets runs on each individual portal server platform. Portal customers may end up in situations where their required combination of Portlets is not available on their selected portal server, as shown in the below figure.

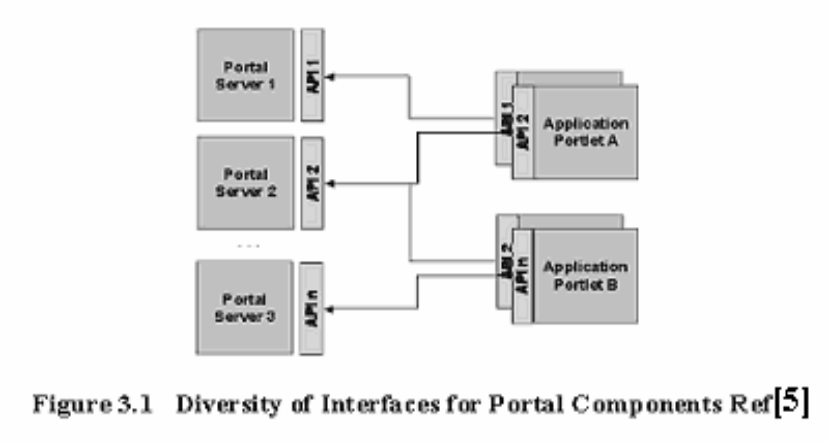


Figure 3.1 Diversity of Interfaces for Portal Components Ref[5]

The above said problem was overcome by establishing interoperability between Portlets and portals. This was done by using a common API portelt standards i.e JSR-168 and WSRP. As depicted in Figure below, all Portlets written to the Portlet API will run on all compliant portal servers.

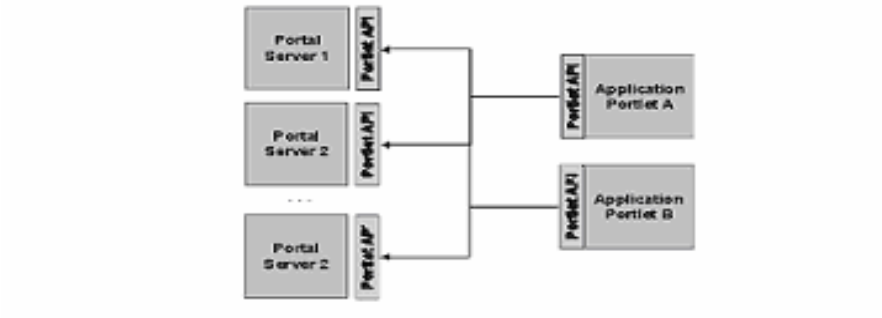


Fig 3.2 Common Portlet API for Portal Components Ref [5]

3.2 WEB SERVICES FOR REMOTE PORTLETS (WSRP)

The OASIS Web Services for Remote Portlets (WSRP) standard simplifies integration of remote applications/content into portals so that portal administrators can pick from a rich choice of services and integrate it in their portal without programming effort. As a result, WSRP becomes the means for content and application providers to provide their services to organizations running portals in a very easily consumable form.

It provides the Dynamic plug-ins for portal pages. It allows portal or application owners to easily embed a web service from a third party into a section of a portal page (a 'Portlet'). The Portlet then displays interactive content and services that are dynamically updated from the provider's own servers. Formerly known as Web Services for Remote Portals, WSRP is closely allied with WSIA (Web Services Interactive Applications).

3.2.1 WSRP Goals

- Enable interactive, presentation-oriented web services to be easily plugged into standards-compliant portals
- Ensure concepts and data exchanged are aligned with other standards in both the portal and web service arenas.
- Make the Internet a marketplace of visual web services, ready to be integrated into portals.

3.2.2 Key Capabilities of WSRP

- Allow portals to publish Portlets so that other portals can consume them without programming.
- Make the Internet a marketplace of visual Web services, ready to be integrated into portals.
- Allow anybody to create and publish his or her content and applications as user-facing Web services.
- Allow line-of-business portal administrators to browse directories for WSRP services to plug into their portals without programming effort.

Enable interactive, user-facing Web services to be easily plugged into standards-compliant portals.

3.2.3 WSRP Benefits

- Flexible Architecture - decouples Portal aggregation from Portlet execution.
- Scalable - allows Portals to use/include Portlets running somewhere else.
- Plug and Play Solution - by standardizing on interface (WSDL)
- Interoperability
- Portability
- Options for deployment
- Support by large players in the industry

3.2.4 Why Use WSRP?

WSRP is an attractive option for web development for three main reasons:

- **It decouples the deployment and delivery of applications.**

WSRP Decouples the Deployment and Delivery of Applications You can surface new applications on your portal, independent of release schedule and where and when the code is physically deployed. For example, perhaps you have a portal on machine X and another on machine Y. To get a Portlet from machine X to machine Y, currently your only method of doing so is to copy the Portlet's code, JSPs, and so on, from machine X to the destination machine (Y). By using WSRP, you can access and display that Portlet on machine Y simply by referencing it through the Producer's Web Service Description Language identifier (WSDL).

- **It delivers both data and that data's presentation logic.**

WSRP Delivers both Data and its Presentation Logic. As a "user-facing" web service, WSRP Portlets provide both application and presentation logic. This is different from standard web services, or data-oriented web services, which contain business logic but lack presentation logic and thus require that every client implement that logic on its own. While the data-oriented approach works well in many implementations, it is not well suited for dynamically integrating business applications. For example, to integrate an

order status web service into a commerce portal, you would need to write code to display the results of the status services into the portal. Using WSRP, with the presentation logic included in the web service, you can achieve the aggregation of applications and services dynamically. You no longer need to develop the presentation logic in order to do the integration; you can simply request the order status service to show up as a Portlet inside the commerce portal at a predetermined location.

- **Its implementation requires little or no programming.**

BEA's Implementation of WSRP Requires Little or No Programming. You don't have to do a lot of programming to make a Portlet remote. In a non-WSRP compliant implementation, integrating remote content and application logic into an end-user presentation usually requires a significant custom programming effort. Typically, vendors of aggregating applications, such as a portal, write special adapters for applications and content providers to accommodate the variety of different interfaces and protocols those providers use. WebLogic Workshop 8.1 SP3 provides tools that allow you to pick from a rich choice of compliant remote content and application providers, and integrate them with just a few mouse clicks—without writing a line of code. Additionally, applications created with WebLogic Workshop 8.1 SP3 are, by default, WSRP-compliant, which means they can be leveraged into other user's Portlets with little or no additional programming required on your part.

3.2.5 How WSRP works?

WSRP introduces the concepts of Producers and Consumers. By using WSRP, you can aggregate application functionality by integrating WSRP-compliant Producers into WebLogic Portal as a Consumer. Your end users thus will be able to interface with Consumers to view the integrated applications

Producers

Producers host Portlets and provide such services as self-description, mark up, registration, and Portlet management. Producers can optionally manage the registration of Consumers and require them to pre-register prior to interacting with Portlets. A registration establishes a relationship between Consumers and Producers

Consumers

A Consumer is an intermediary system that communicates with presentation-oriented web services (i.e. Producers and the Portlets they host) on behalf of its users. It gathers and aggregates the markup delivered by the Portlets and presents the aggregation to the end-user. Because of this intermediary role, Consumers are often compared to “message switches” that route messages between various parties. One typical Consumer is a portal, which mediates the markup and the interaction with this markup between End-Users and presentation-oriented web services. Another typical Consumer is an e-Commerce application that aggregates manufacturer-provided content into its own pages. Since the Consumer is an intermediary, aggregating system, the markup sent for display to the End-User and most interactions with that markup flow through the Consumer. This often results in situations where the End-User implicitly trusts the Consumer to respect their privacy and security concerns with regards to this information flow. Consumers aggegate information from Producers and surface it in other portals. Consumers route requests from users to the appropriate Producer, which, in turn processes the request and sends results back to the Consumer. The Consumer aggregates the results coming from various Producers and send the final result back to the user. Consumers provide separation of the traffic flowing between them and the Producers. They also ensure that all interactions are kept private to that specific user during the sessions.

End-User

The main purpose of a Consumer that aggregates content from various Producers/Portlets is the preparation and presentation of markup to an End-User. In addition, the Consumer needs to manage the processing of interactions with that markup in order to properly correlate the interactions with the, potentially stateful, environment that produced the markup.

In this section we had discuss the details of WSRP and in the next section we will discuss the JSR-168. the working of WSRP can better understood from the below shown Figure 5.1

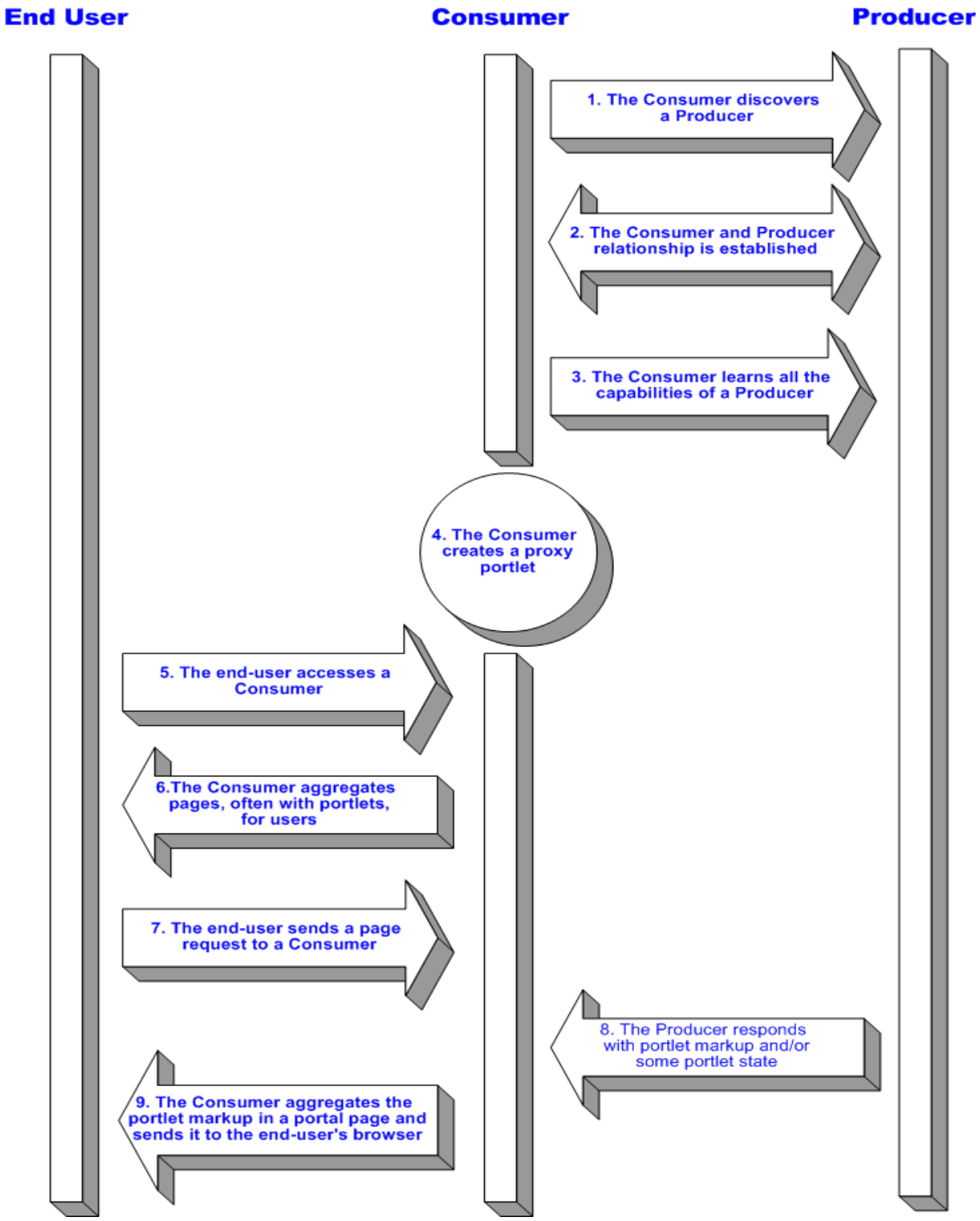


Figure 3.3 Working of WSRP Ref[9]

3.3 JSR: A PORTLET STANDARD

The release of the JSR-000168 Portlet API has been accompanied by announcements for a Sun ONE Studio Portlet Builder beta and a pre-release of Oracle9iAS Portal's WSRP Portal. The JSR-000168 Portlet Specification has been distributed as a PDF Portlet Specification with code and reference in Portlet API Specification Interface Classes and Portlet API Javadoc Documentation. The Portlet Specification defines a proposed standard for the Java Portlet API as outlined in the JSR 168 Java Specification Request, designed to "enable interoperability between Portlets and portals by defining a set of APIs for Portal computing addressing the areas of aggregation, personalization, presentation and security."

- It is java specification request.
- It is proposed by SUN and IBM.
- Released in Oct 2003.
- It is way to standardize java related technologies.
- Define the APIs that standardize the development of Portlets.

3.3.1 JSR 168 Portlet API Specification

The *Portlet Specification* defines a proposed standard for the Java Portlet API as outlined in the JSR 168 Java Specification Request, designed to "enable interoperability between Portlets and portals by defining a set of APIs for Portal computing addressing the areas of aggregation, personalization, presentation and security."

A ***Portal*** in terms of this specification is a web based application that "provides personalization, single sign on, content aggregation from different sources and hosts the presentation layer of Information Systems. Aggregation is the action of integrating content from different sources within a web page. A portal may have sophisticated personalization features to provide customized content to users, and portal pages may have different set of Portlets creating content for different users."

A ***Portlet*** is "a Java technology based web component, managed by a Portlet container, which processes requests and generates dynamic content. Portlets are used by portals as

pluggable user interface components that provide a presentation layer to Information Systems."

The typical sequence of events that are initiated by users to access their portal page:

- A client (*e.g.*, a web browser) after being authenticated makes an HTTP request to the portal.
- The request is received by the portal.
- The portal determines if the request contains an action targeted to any of the Portlets associated with the portal page.
- If there is an action targeted to a Portlet, the portal requests the Portlet container to invoke the Portlet to process the action.
- A portal invokes Portlets, through the Portlet container, to obtain content fragments that can be included in the resulting portal page.
- The portal aggregates the output of the Portlets in the portal page and sends the portal page back to the client"

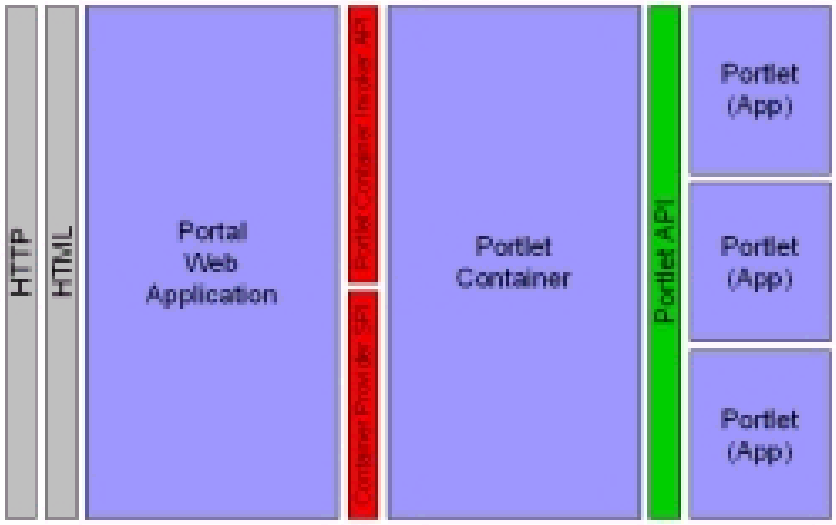


Fig 3.4: Sequence of events for Grid Portlets Ref[18]

3.3.2 Elements of a Portal Page

A Portlet generates markup fragments. A portal normally adds a title, control buttons and other decorations to the markup fragment generated by the Portlet, this new fragment is called a Portlet window. Then the portal aggregates Portlet windows into a complete document, the portal page. Portal Page Creation, Portlets run within a Portlet container. The Portlet container receives the content generated by the Portlets. Typically, the Portlet container hands the Portlet content to a portal. The portal server creates the portal page with the content generated by the Portlets and sends it to the client device (*i.e.*, a browser) where it is displayed to the user. [In the] Portal Page Request Sequence, users access a portal by using a client device such as an HTML browser or a web-enabled phone. Upon receiving the request, the portal determines the list of Portlets that need to be executed to satisfy the request. The portal, through the Portlet container, invokes the Portlets. The portal creates the portal page with the fragments generated by the Portlets and the page is returned to the client where it is presented to the user."

3.3.3 Major task performed by JSR 168

- Define a standard Java technology-based web component model for Portlets in portal server that are build on Java platform.
- Enables portability and vendor independence.
- It does not define interoperability of remote Portlets.
- Portlets that are built according to JSR168 model works with all Java tech. based Portlet servers.
- It enables interoperability among portal and Portlets defines a set of application programming interface (APIs) for Portlets and addresses standardization for preferences, user information Portlets request and responses, deployment packaging and security.

3.3.4 Benefits of JSR 168

Before JSR 168 it is difficult to deploy one Portlet developed for certain Portlet server into some other Portlet server. E.g. you can not deploy a Portlet developed for IBM web

sphere into BEA web logic Portlet after JSR 168 standardization, all Portlet vendors will support standard API resulting in interoperability.

3.3.5 JSR 168 Portlet specific capabilities

- **Portlets mode**: Three mode of Portlets interactions are:
  - **View**: View existing user profile data (User interface), and groups a user is in. When a Portlet is initially constructed on the portal page for a user, it is displayed in its view mode. This is the Portlet's normal mode of operation
  - **Edit**: Allows user to edit profile including User attributes, and groups a user is in . the Portlet provides a page for users to customize the Portlet for their own needs. For example, a Portlet can provide a page for users to specify their location for obtaining local weather and events.
  - **Help:** the Portlet provides a help page for users to obtain more information about the Portlet.
- User profile attributes
- Window state
- Security
- Portlet persistence

3.3.6 A Critique of JSR 168

- There is no way to share data/objects between Portlet applications. So all Grid Portlets would have to be in the same Portlet app.
- There is no way to extend the Portlet API to add such services.
- There is a lack of general purpose Portlets. Right now, you make specific extensions to Generic Portlet for each Portlet you develop.
- JSR 168's MVC approach is incompatible with Turbine, Struts.
- No defined way to integrate with portal-specific services (i.e. logins).
- No inter-Portlet communication.

3.4 CHALLENGES FOR EXISTING STANDARDS

We are clearly bullish on the potential contribution of these standards efforts, but there are some challenges.

The first has to do with **timing**: Will the standards arrive before it's too late? While the portal standards groups are off to a comparative jackrabbit start, if they take too long to create workable specifications and demonstration platforms the market may pass them by. That means a repeat of something like the decade-long struggle around UNIX. In that case protracted disagreements within and among multiple standards bodies prevented the goal of interoperability from being achieved, and the industry has had to live with a state of incomplete convergence.

The second challenge revolves around **politics**. In the software business, the Windows versus OS/2 conflict. The risk here is that major vendors choose to develop proprietary product thrusts designed to corner the market.

The third challenge is **whether the standards will go far enough**. If they don't provide an adequate breadth of scope to address the needs of tomorrow's composite business applications, they will be viewed as flawed and largely irrelevant. Advanced software vendors could build out functionality ahead of the standards due to frustration. Early adopting companies will then deploy proprietary solutions to cut down the time-to-benefit for new business applications, while continuing to incur the costs of this approach for at least one more cycle of the standards process.

Last, will these standards attract a critical mass of **industry support**? In the Java vendor community the two standards bodies already enjoy wide support. However, if the Java customer community finds the implementations cumbersome or otherwise lacking in value, they will pass on the standards-based approach. Another significant risk involves Microsoft's plans for the future of portal interoperability. Thus, the value of the Java community standards may be compromised for the many enterprises with heterogeneous Microsoft-Java Deployments?

3.5 EXISTING PORTLETS

Currently, many portal vendors are supporting Portlets including IBM WebSphere, Sun One Portal Server, Oracle 9iAS, and the Jakarta Jetspeed project.

3.5.1 Portlets by Sun One Portal Server

- **My computing jobs Portlet**

This Portlet allows creating, configuring and submitting computing jobs to the Grid, monitor the job execution, and download and visualize the results. It communicates with the GSP Job Submission and Application Management services and the DMS services.

- **Applications Portlet**

This Portlet allows to manage the application repository. It allows adding and removing, to prepare multiple configurations of applications, and to prepare virtual applications. It communicates with the Application Management Service and the DMS services.

- **My data Portlet**

This Portlet allows to manage the user Grid data file space. The users have a possibility to browse the folders of the Data Managements System, upload and download files from/to DMS. The Portlets cooperates with the DMS services.

- **Gaussian Portlet**

This Portlet was the first specialized application Portlet implemented. It is aimed to deliver a popular application from the chemistry area, Gaussian. The "Gaussian" Portlet served as the proof of concept for the idea of specialized application Portlets and the Progress Portlet Framework mechanisms.

- **DNA Assembly Portlet**

This Portlet is aimed to facilitate the use of the DNA Assembly application that has been prepared as an additional task within the project. This Portlet not only allows to utilize the application itself, but is also capable of adding additional data format converters to the job if necessary, thus creating workflow jobs. In the same time the workflow structure of such a job is hidden from the user: this is very useful for non-advanced users who just want to have their data delivered in their favorite data format analyzed: they do not need to create these workflows by themselves, the Portlet does it automatically.

- **My News Portlet**

This Portlet is aimed to deliver news service within portals and other websites. It served as the proof of concept for utilization of a GSP service within multiple independent portals/websites. It also served as the proof of concept for customization mechanisms and features of the PROGRESS Portlet Framework: we used these mechanisms and features to quickly (one day) prepare customized Servlets on the base of the "My news" Portlet for delivery of the Short News Service.

- **Management Portlet**

This Portlet is currently aimed to deliver information stored by the Provider Management Service to the Grid-portal environment administrators. It also allows to manage the list of accepted resource requirements for the computing jobs. We plan to make this Portlet more functional and sophisticated in the future, adding some more possibilities for configuration of various aspects of the underlying Grid-portal infrastructure.

3.5.2 Portlets by IBM Websphere

- **Report Portlet** enables users to display any MicroStrategy Report within IBM WebSphere Portal. Customers can point this Portlet to an existing MicroStrategy report and instantly leverage all the formatting and end user interactivity.
- **Folder Portlet** enables users to display any MicroStrategy Folder within IBM WebSphere Portal. Customers can point to any MicroStrategy folder, click on a report and execute it within the context of a portal.
- **Search Portlet** enables users to search for a MicroStrategy Report within MicroStrategy metadata. Customers can conduct a basic or advanced search, click on a report from the search results, and execute the report within the context of a portal.
- **History List Portlet** enables users to execute a report that has been previously run from its cache within IBM WebSphere Portal.
- **Subscription Portlet** enables users to subscribe to any MicroStrategy report of their choosing within IBM WebSphere Portal. Users can set their own schedule for report delivery through an intuitive self-subscription interface.

- **Report Services Portlet** enables users to link or embed any Report Services document within IBM WebSphere Portal.
- **Web Portlet** enables users to showcase any or all of the Web functionality within IBM WebSphere Portal. Customer can browse through various projects or various folders and execute a report within the context of a portal.
- **Document Portlet** enables users to run an HTML document containing multiple reports and dashboards within IBM WebSphere Portal.

3.5.3 Portlets by Oracle 9.1AS

- **Administration Portlets**

External Applications: List links to your external applications.

Login: Log on directly from a page.

SSO Server Administration: Administer users, configure partner and external applications for authentication through the Single Sign-On Server, and edit the Single Sign-On Server configuration

User: Create or edit users.

Group: Create or edit groups of users.

People Search: Search for a user

Portal Group Profile: Edit a group's portal preferences and global privileges

Portal User Profile: Edit a user's portal preferences and global privileges

- **Database Portlets**

Database Navigator: Navigate database objects. The Navigator Portlets are specifically designed for use on the Navigator pages. If you add them to pages outside of the Navigator, you may get unexpected behavior. Schemas: Create, edit, or navigate to a database schema.

Roles: Create or edit a database role.

Database Information: List statistics about database settings and parameters

Database Memory Consumption, Transactions and Locks: List statistics about database memory consumption, transactions, and locks.

Database Storage: List statistics about database storage

Batch Results: List status and results for background applications.

- **Portal setting Portlets**

Services: Edit portal settings.

Transport Set: Edit transport sets for exporting, import transport sets, and view information about existing transport sets.

- **Oracle9iAS Portal Community Portlets**

Product News: Display news about Oracle9*i*AS Portal.
Developer News: Display technical news about Oracle9*i*AS Portal.
Oracle9iAS Portal Community: Provides links to the Oracle9*i*AS Portal community.
Documentation News: Display news about Oracle9*i*AS Portal documentation

CHAPTER 4

PORTLETS DESIGN N IMPLEMENTATION

4.1 PROBLEM UNDERTAKEN

Portlets are the pluggable user-interface components to a grid Portal i.e. Portlets are the mini windows on the portal. The project “TIET-CSED GRID” requires the design and implementation of the various Grid Portlets for the “TIET-CSED GRID PORTAL”. The Portlets we are to design:

- Login Portlet
- About TIET-CSED Portal
- Our Service Portlet
- Search Portlet
- News Portlet
- Date and time
- Calculator
- Weather forecast
- Service submission
- New User
- Portal Administrator
- Install Portlet
- Uninstall Portlet
- Layout Manager
- New Credential
- View Credential
- Credential list
- Resource Submission
- Resource Browser List

4.2 DESIGNING THE PORTLET

From a technical standpoint, the Portlet architecture is an extension of the Java Servlet architecture. It is a small piece of an application that is plugged into or assembled in a portal to create a single enterprise application. Portlets are also assembled into portal pages, which form a portal implementation.

In general, A Portlet includes two required components, a title bar and content area, and several optional components including the banner, header, footer, edit URL, alternate header, alternate footer, maximized URL, and help URL etc.

We can have more than one Portlet on a single portal, depending upon our requirements as shown in the figure 4.1 below.

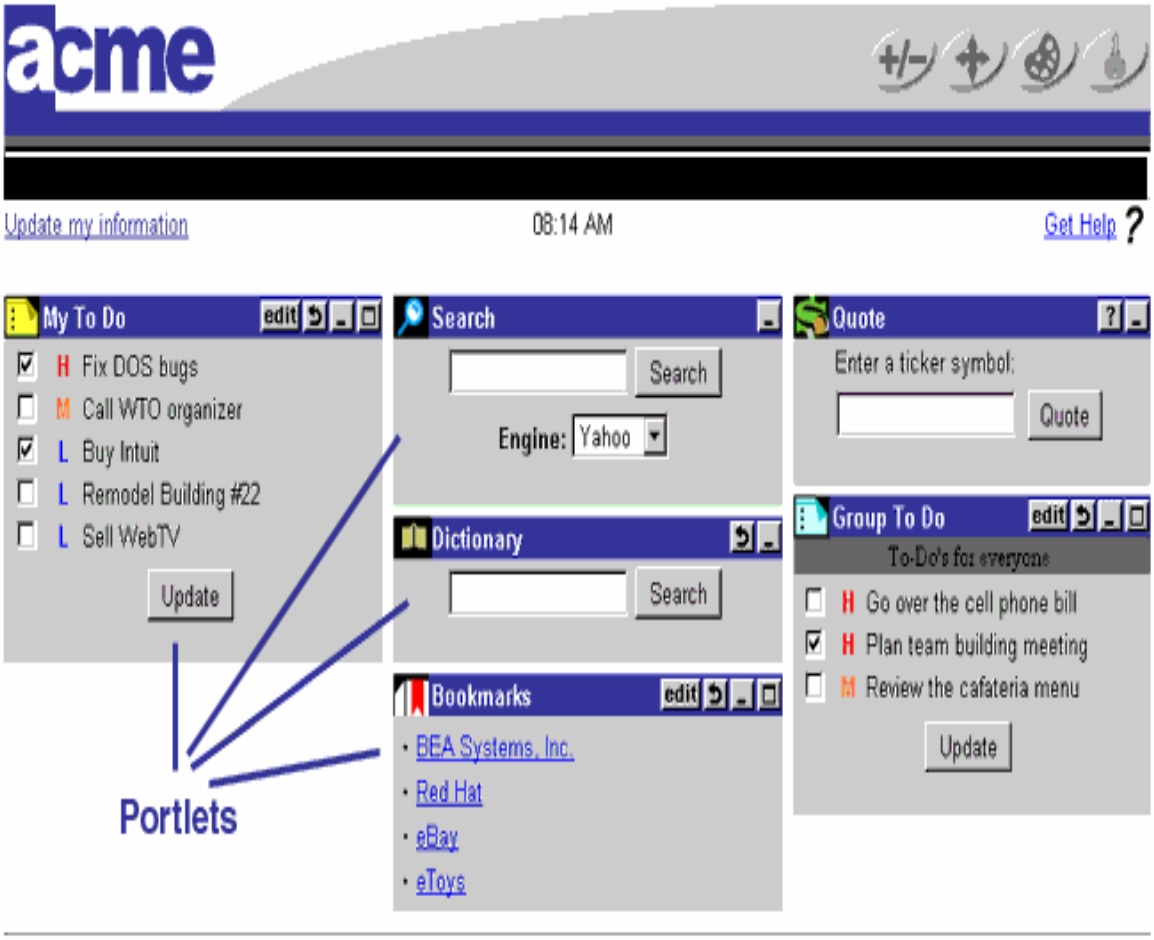


Figure4.1: How Portlets are placed on a portal Ref [29]

Description of the TIET-CSED GRID Portlets

- **Login Portlet**

Login Portlet is the Pluggable authentication modules with provide support to users to access the database password and credential based logins. It also enables new users to request an account.

Login Portlets consists of Login Name and the Passwords. If the user is an existing user on the Grid he needs to enter its user details i.e. login name and password. Then the system compare its details from the data base, if it matches allows the user to access the Grid otherwise gives a message invalid password and ask to try again. And if the user is new to the Grid it needs to register using the New User link. When the new user register the data is stored in the database so that next time whenever it tries to login to the Grid its identity should be confirmed from the existing database.

- **About TIET-CSED Portal**

This Portlet gives the details that what sorts of services are provided by our Grid and who all can be the member of our Grid.

- **Our Services Portlet**

This Portlet tells us that what different types of services available on our Grid. For this timing the services available with our TIET GRID are calculator, Date and time, News, weather forecast.

- **Search Portlet**

This Portlet allows us to search the database and tells us that where the resource are available. This Portlet provide us the facility to select the different search engine depending upon our requirements

- **News Portlet**

The news Portlet is aimed to deliver updated news about the TIET gird within the portal. Like what events are currently taking place and what are events that are going to take place in the near future. This Portlet allows users to read the news along with their details.

- **Date and Time Portlet**

This Portlet gives us the details about the day of the month, year , shows the calendar , shows the time clock and also provide us the current time zone.

- **Calculator Portlet**

The calculator Portlet help us to perform the basic mathematic operations like addition, multiplication, subtraction, division etc.

- **Weather forecast**

As it is clear from the name Weather Portlet gives the weather report depending upon the zip code we enter. In this Portlet we need to just enter our zip code and we will get the weather information.

- **Service submission**

This Portlet helps us to submit a new service. For submitting a new service we need to enter its Service ID, description, Job type, Resource, status and date of creation.

- **New User Portlet**

New User Portlet is a type of Profile Settings Portlet which enables users to configure personal information. A new user to the Grid needs to register him. He needs to click the new user link on the login Portlet which then opens a new Portlet i.e. New user Portlet. This New User Portlet ask the details of the user like User name- First name, last name; login name; password; name of organization; name of department; role in organization, email , operating system then stores the details in the database for further references. This Portlet also allows the existing users to update their profile.

- **Portal Administrator**

This Portlet helps us to administrate the various portal activities. This Portlet handles the other Portlet like the install Portlet, uninstall and layout manager Portlet.

- **Install Portlet**

As the name implies, this Portlet helps us to install a new Portlet in our portal page. The various Portlets are managed by the Portal administrator.

- **Uninstall Portlet**

This allows the user to delete the Portlet which is not required by him.

- **Layout Manager**

Layout Manager helps us setting the page layout of the various Portlets on the portal page. This Portlet allows us to select the skin of the Portal and also the number of columns i.e. number Portlet to be displayed on the portal page.

- **New Credential**

Through this Portlet we can enter the new credential to our Grid along with the various information like the label, username, credential name, pass phrase, use portal credential and single sign on. The use portal Credential and single sign on are always checked (by default) so that we can use the portal credential whenever we retrieve the new portal.

- **View Credential**

In this Portlet, we can view the credential information i.e. username, credential name, pass phrase, credential name, use portal credential and single sign on. It also provides us the information about the status of the credential i.e. when it has been created, when it has been last retrieved and whether it’s a active credential or not.

- **Resource Browser List**

It provides us the list of the various resources along with their hostname, platform, type and IPaddress.

4.3 THINGS FOCUSED WHILE DESIGNING THE GRID PORTLET.

We should take care of number of Portlet design practices so that it will support overall portal benchmarks for flexibility, scalability, extensibility, and performance [17]. Some of them are

- **Design Your User Interface (UI), Model Your Portlet:**

By this we mean, Good Portlet design does not equal good Portlet UI design. An effective design supports Portlet reuse and enables multipage rendering without modifications.

- **Portlet design must follow Simplicity:**

Developers should limit the information displayed to include only content required for task completion. If it requires the display of large amounts of page information, developers should break it into standalone "atomic" Portlets.

- **Familiarize Portlets Into One Family:**

Portal support a browser-independent method for establishing a uniform Portlet look and feel.

- **Model Your Data, Control Your View:**

The Portlet API provides many features for an MVC-based design, including tag libraries. Portlet APIs also support Java Community Process standard JSR 168 features, such as coordination of user action and action sequences.

- **Promote Diversity Through Integration:**

A portal typically integrates diverse sets of information into a single user-friendly view. Depending on the data source, integration can become a complex multiphase exercise.

4.4 PREREQUISITES FOR GT3

Before setting up or installing GT3 we need to set the various variables like the environment variables and setting up the path of the various variables. So here are the variables which we had set before installing GT3

- Set Environment Variables
  - GLOBUS_ROOT = c:\Grid
  - ANT_HOME = %GLOBUS_ROOT%\apache-ant-1.5.4
  - JAVA_HOME=c:\j2sdk1.4.2_02
  - GLOBUS_LOCATION=%GLOBUS_ROOT%\ogsa-3.0.2
  - CATALINA_HOME=%GLOBUS_ROOT%\Tomcat4.1

- Add to PATH Environment Variable
  - %JAVA_HOME%\bin
  - %ANT_HOME%\bin
  - %GLOBUS_LOCATION%\bin
  - %GLOBUS_ROOT\bin

- Create GLOBUS Directory Structure
  - Create GLOBUS_ROOT directory
  - Create GLOBUS_ROOT\download directory
  - Create GLOBUS_ROOT\bin directory

4.4.1 Setting the GT3 Environment

For the GT3 environment setting we need to install the various tools like the java JSDK, Apache Ant, Apache tomcat , python HSQL database etc. these tools are needed to be installed in the proper sequence. As given below in the form of a flow chart.

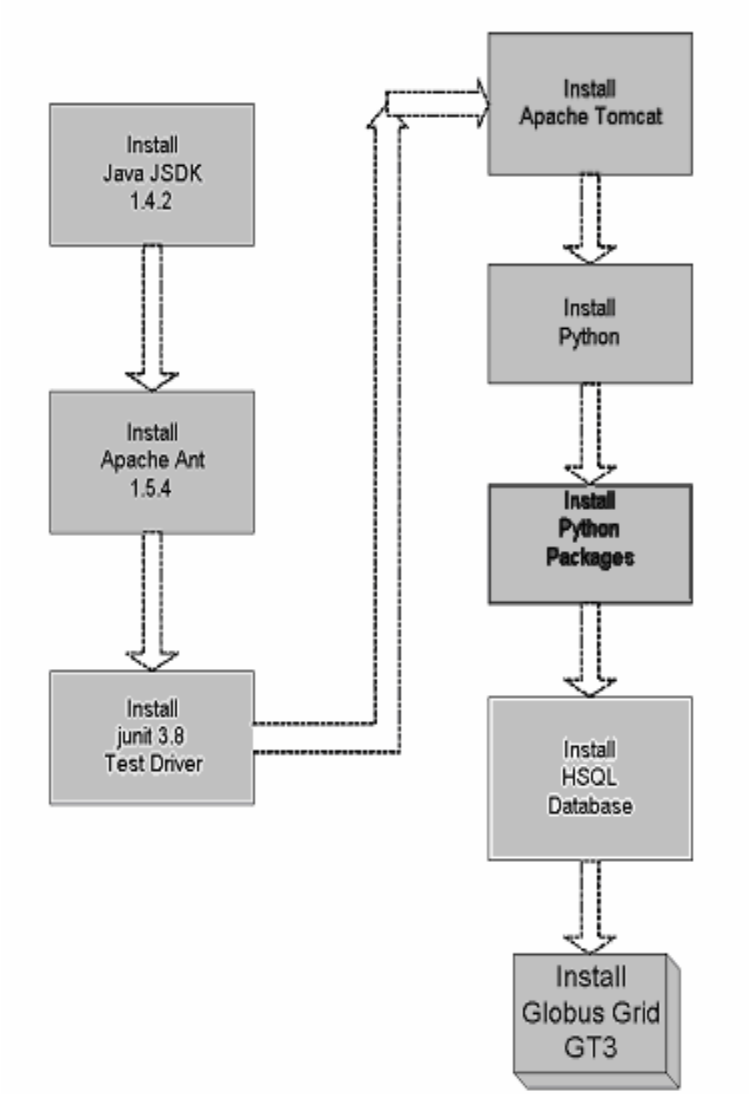


Fig 4.2: Supporting Tools to GT3

4.4.2 Installing the GT3

After installing various tools for the GT3, we need to install and configure the toolkit itself. for the configuring the Globus toolkit we need to download the Grid toolkit, binary toolkit and then deploying the GT3 to tomcat. And then finally executing our application i.e. Portlet. The below shown flowchart gives us the sequence of the various steps that are performed.

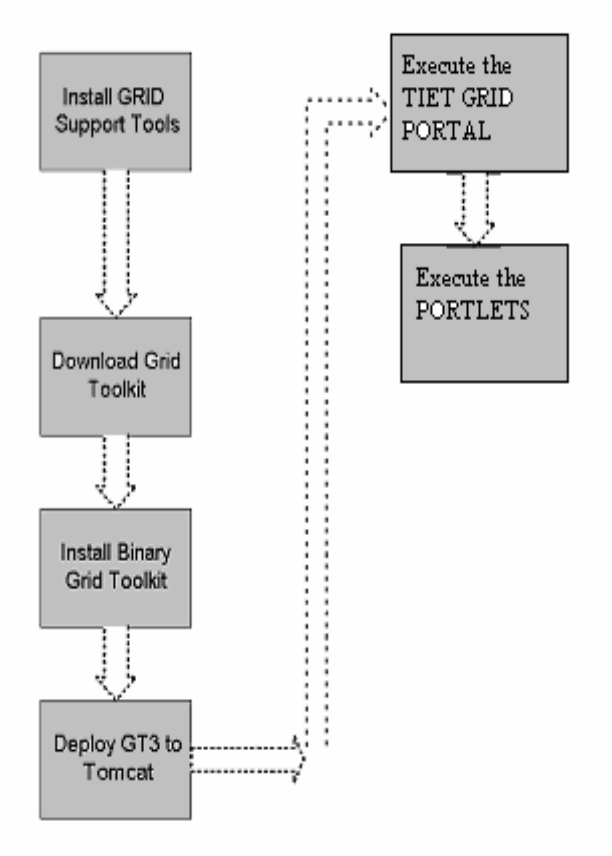


Fig 4.3: Installation of GT3 environment

As we have now set up the environment for the Grid. We need to develop our Portlets which will we placed on the portal page for our TIET-CSED GRID, accessible to the external world.

4.5 IMPLEMENTATION OF A GRID PORTLET

we have build own Portlets using the Grid development environment GT3 and its available tools. The following are the steps for developing a Portlet.

Step 1: Writing the Portlet code
Step 2: Compiling Java source
Step 3: Creating the JAR file
Step 4: Writing the Portlet descriptors
Step 5: Setting up the GAR file directory structure
Step 6: Packaging and deploying Portlets

Step 1: Writing a Portlet code

Before writing the code for the Portlet we should first of all understand the requirements of the Portlet i.e. what the Portlet should do after identifying all the requirements we should develop the high level design of the Portlet which will tell us that how the Portlet looks like and where it will be placed in the portal page. Then go for the detailed design which tell us that how the Portlet work , what is logic behind the Portlet, how the different interactions taking places like the interaction between the different elements in the Portlet, between different Portlets, and how will the Portlet interact with the Portal. Then finally we will code our Portlet.

Step 2: Compiling Java source

Use the JDK provided by GT3 to compile your Java source files. Before you compile your Java source, set the CLASSPATH for the compiler to find the JAR files for any Portlet packages that your Portlet uses.

C:/ javac –classpath ./build/classes/%CLASSPATH% <path of the java file to be compiled.

Step 3: Creating a jar file

The Portlet must be packaged in the JAR file format.

jar cf *jar-file input-file(s)*

Here,

- The c option indicates that you want to *create* a JAR file.
- The f option indicates that you want the output to go to a *file*
- jar-file is the name that you want the resulting JAR file to have.
- The input-file(s) argument is a space-separated list of one or more files that you want to be placed in your JAR file.

Step 4: Writing the Portlet descriptors

One of the key components of the deployment phase is a file called the *deployment descriptor*. It's the file that tells the Grid server how it should publish our Grid Portlet The deployment descriptor is written in WSDD format (Web Service Deployment Descriptor).

The following properties should be set to correspond to the Portlet descriptor:

- <description/> describes the Portlet application.
- <display-name/> indicates the Portlet application name.
- <security-role/> indicates the Portlet application security role mapping. we Omitted this tag as our Portlets does not use this feature

Step5: Setting up the GAR file directory structure

Before you package your Portlet, the class files and resources must be arranged in the GAR file directory structure described here. A Portlet application exists as a structured hierarchy of directories.

This GAR file is a single file which contains all the files and information the Grid Portlet container need to *deploy* our Portlet and make it available to the whole world.
However, creating a GAR file is a pretty complex task which involves the following [2]:

• Converting the GWSDL into WSDL
• Creating the stub classes from the WSDL
• Compiling the stubs classes
• Compiling the Portlet implementation
• Organize all the files into a very specific directory structure

This all can be done in the single step using a very useful tool called Ant which is provided by the GT3.Ant, an Apache Software Foundation project, is a Java *build tool*. It allows programmers to forget about the individual steps involved in obtaining an executable from the source files, which will be taken care of by Ant. The build file directs Ant on what it should compile, how it should compile it, and in what order.

Ant –Djava.interface = true
-Dpackage = counter.core.factorty
-Dinterface.name = counter
-Dpackage.dir = counter/core/factory
-Dservices.namespace = Factory.core.counter

Step 6: Packaging and Deploying Portlets

To deploy a Portlet and run it on the server, it must be packaged in the form of a Grid application ARchive or GAR file. The GAR file format contains the Java classes and resources that make up one or more Portlets in a Portlet application. The resources can be images, JSP files, Portlet descriptors, and property files containing translated message text. Packaging Portlet classes, resources, and descriptive information in a single file makes distribution and deployment of Portlets easier.

The GAR file, as mentioned above, contains all the files and information the web server needs to deploy the Grid Service. Deployment is also done with the Ant tool, which unpacks the GAR file and copies the files into key locations in the GT3 directory tree. It also reads our deployment descriptor and configures the web server to take our new Grid Portlet into account [2].

ant deploy -Dgar.name=<***full path of GAR file***>

4.6 RUNNING THE GRID PORTLET

In the above section we described how the Portlets are implemented using the Globus toolkit (GT3). Now, in this section we will discuss how the Portlets run. The Portlets are handled by the Portlet container. So we need to start the container which will handle our Portlet and it is done in the Globus toolkit by issuing the command

globus-start-container

This not only helps us in running our container but also give us the list of the various services that are running on our Grid.

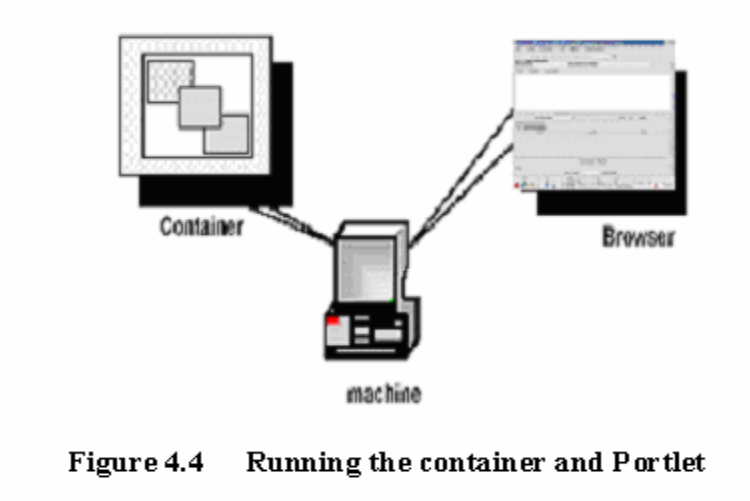


Figure 4.4 Running the container and Portlet

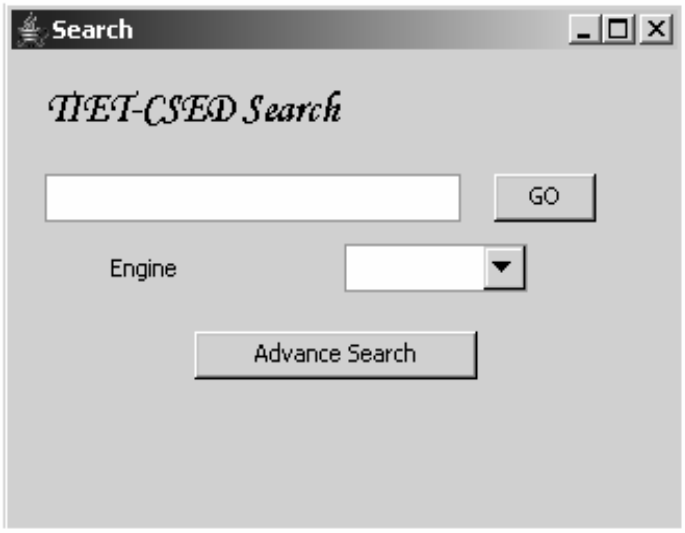
Activities we performed while running a Portlet:

- Start default container
- Start the service browser
- Create instance with the service browser
- Use the counter service using the service browser
- Create a new service instance with the command line
- Destroy the service

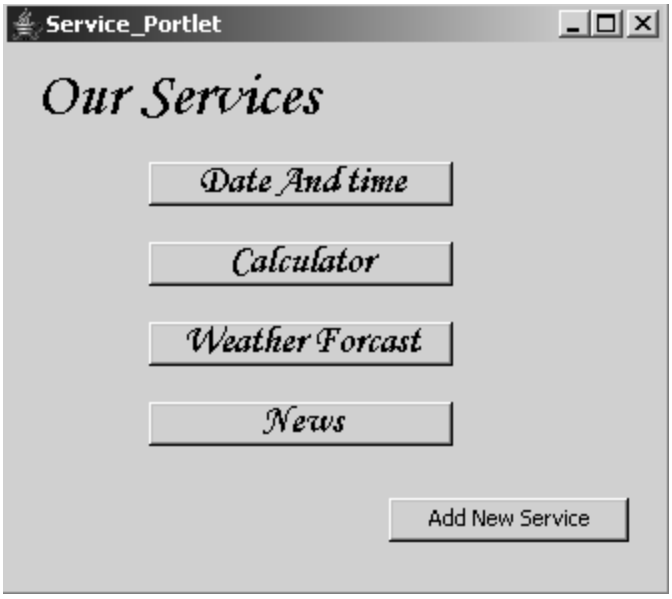

About TIET_CSED Portal
Welcome to our Grid
@ You can now access all the resources like computational, Data,Software etc available in the Grid.
@ You can make your resources available in the Grid so that other members can use them.
@ You will be able to work in a collaboration environment.
Login

Screen#1

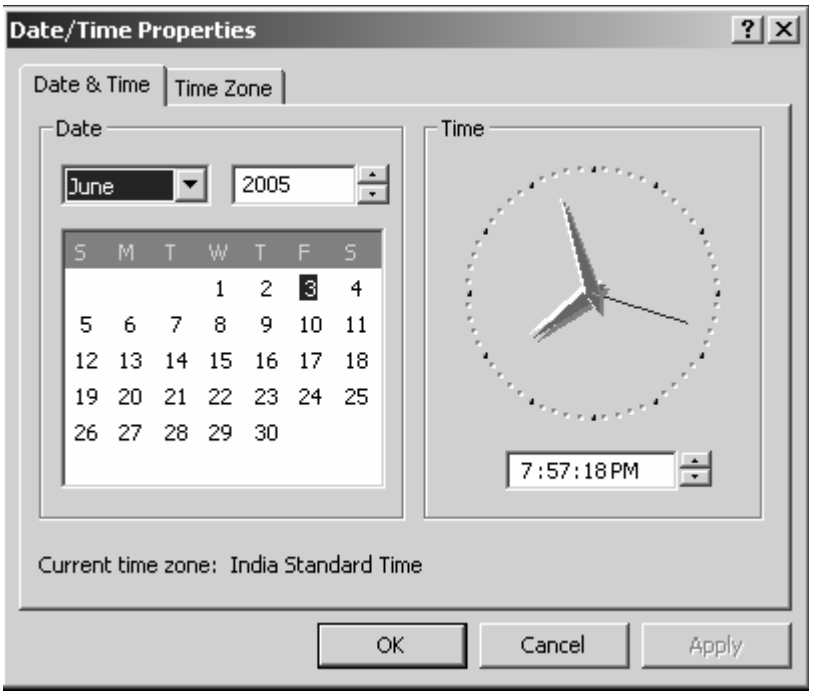
Screen #2



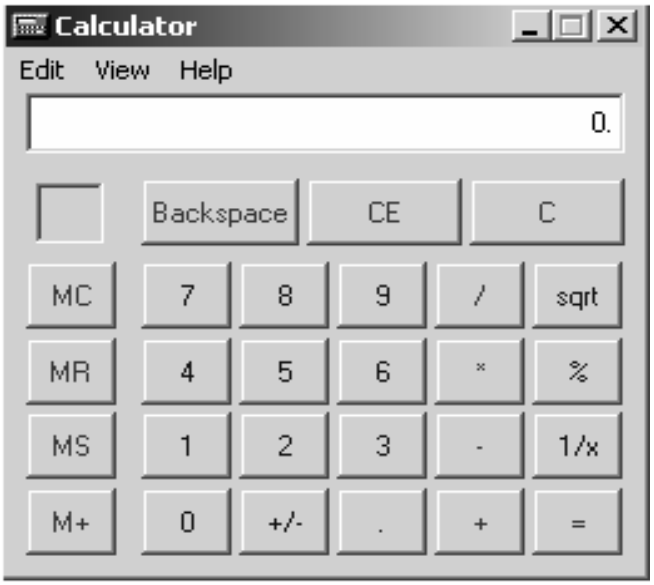
Screen#3



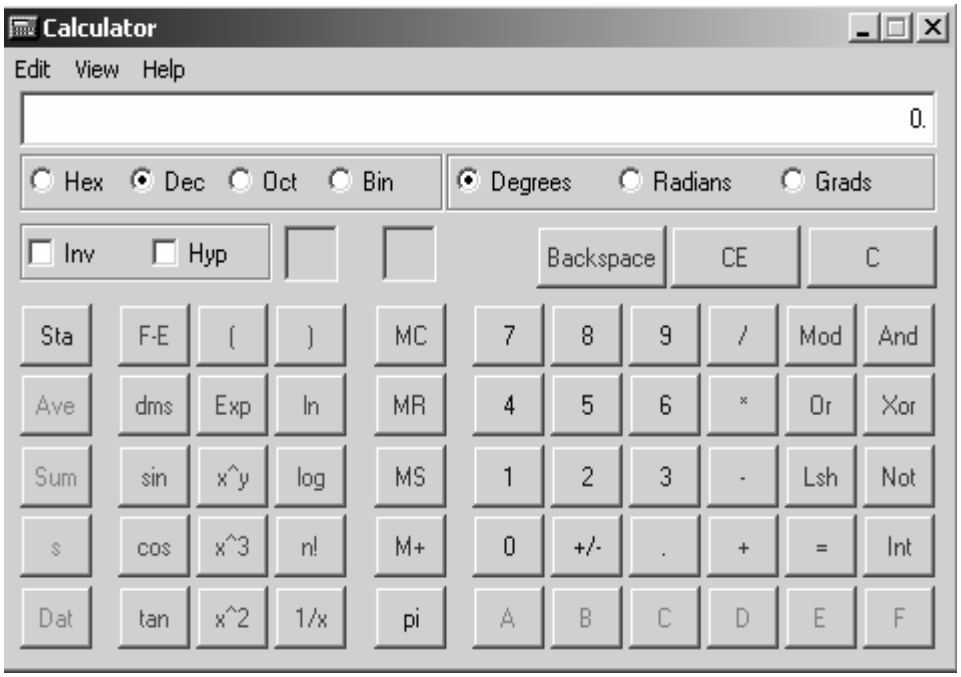
Screen#4



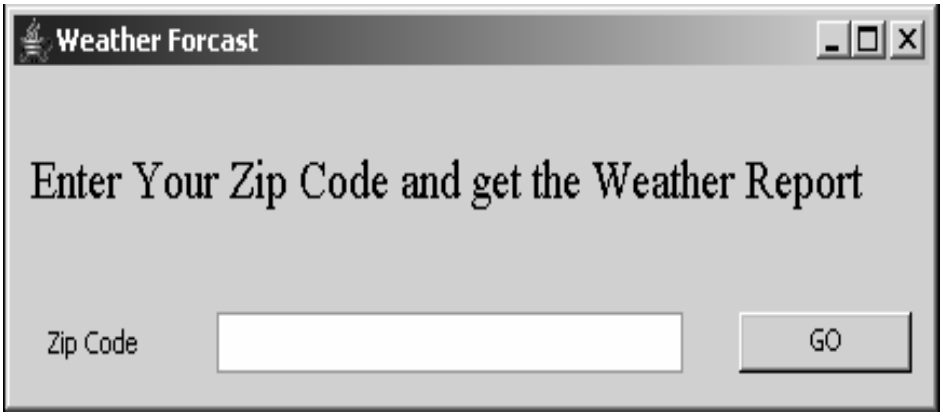
Screen#5



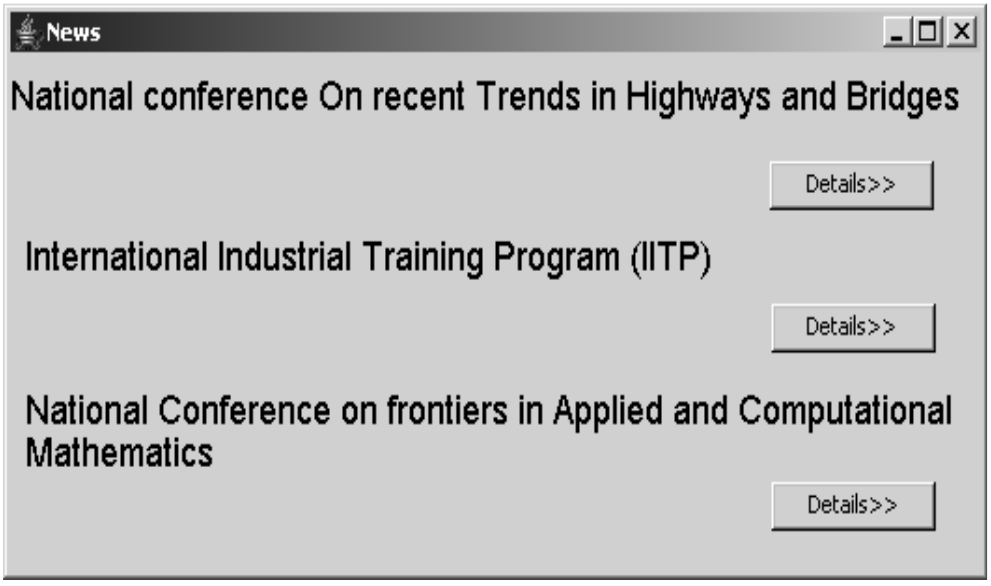
Screen#6



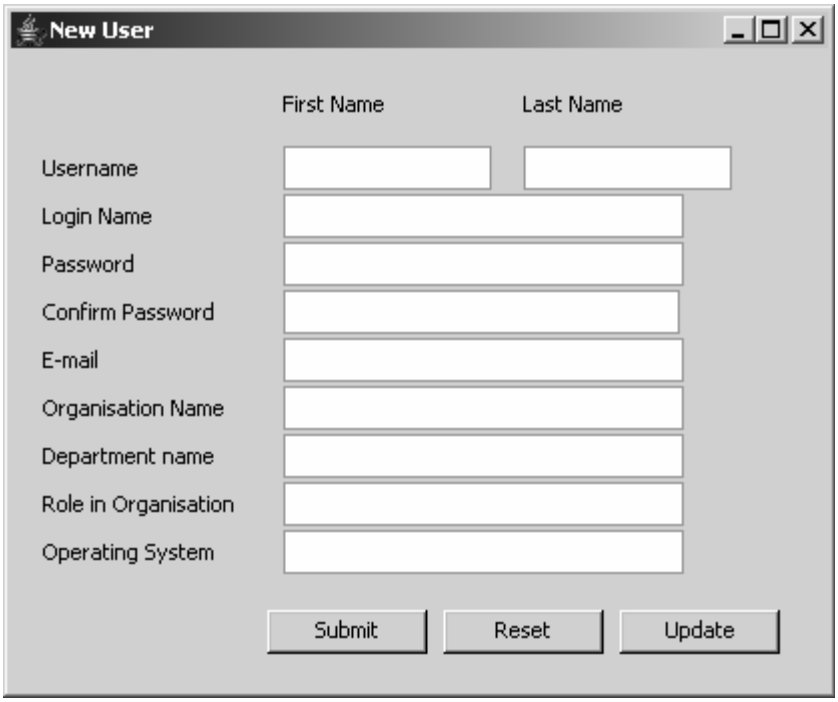
Screen#7



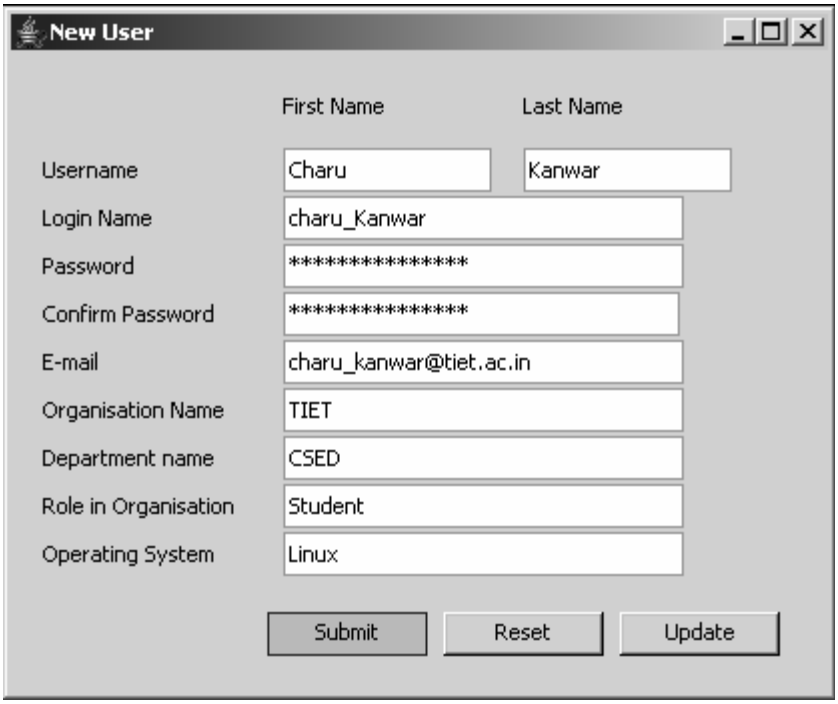
Screen#8



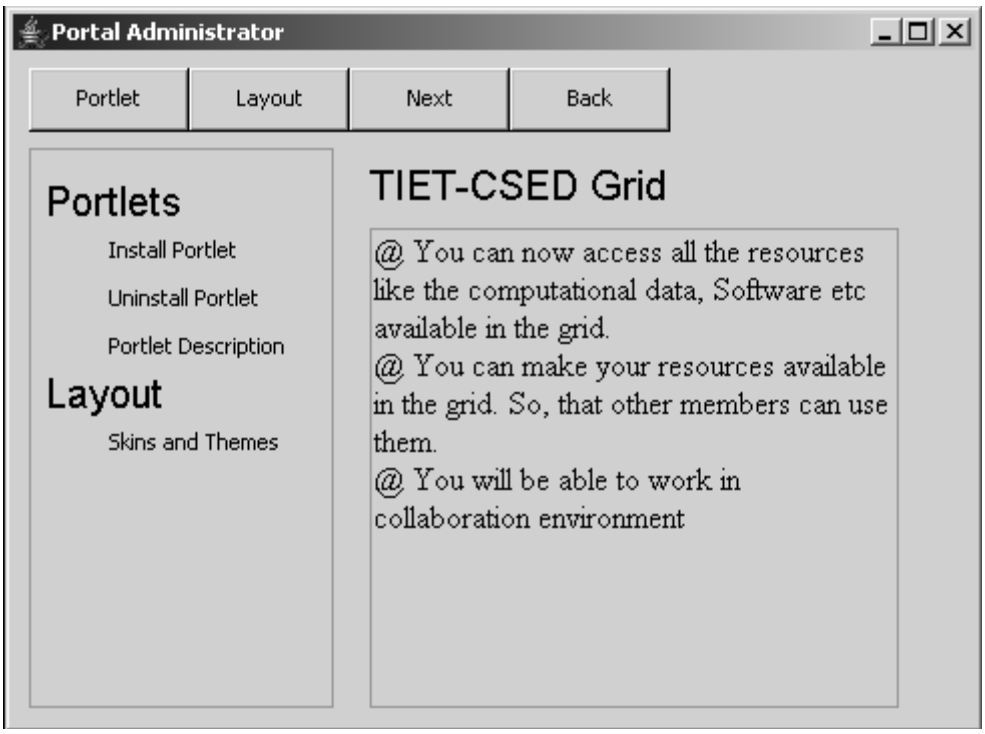
Screen#9



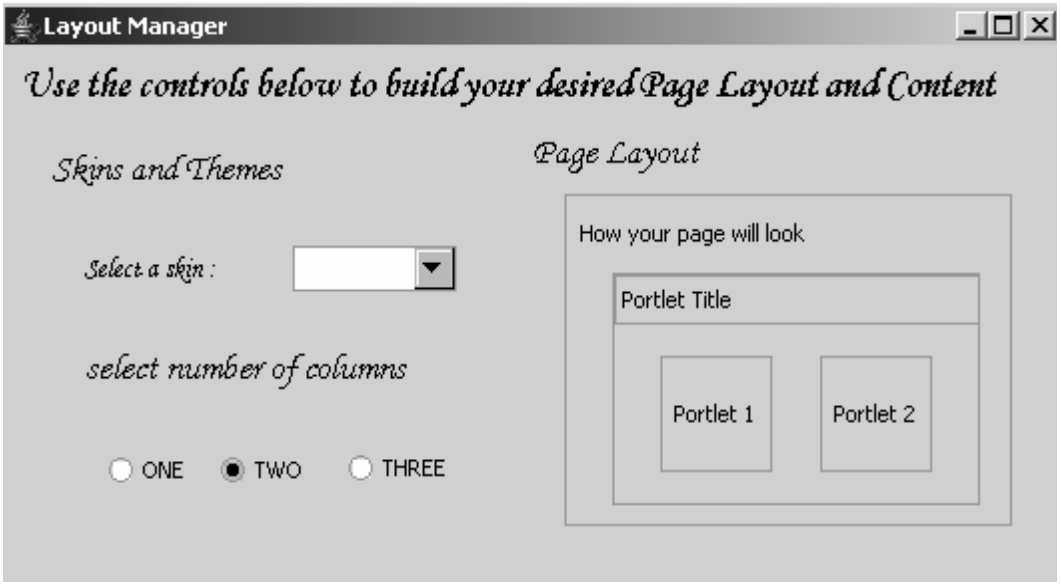
Screen#10



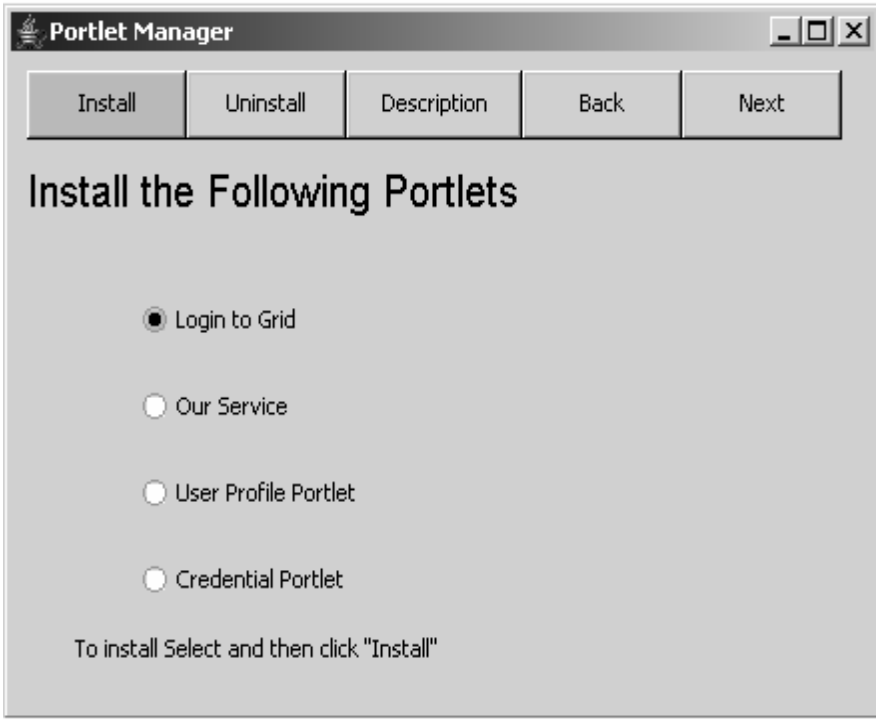
Screen#11



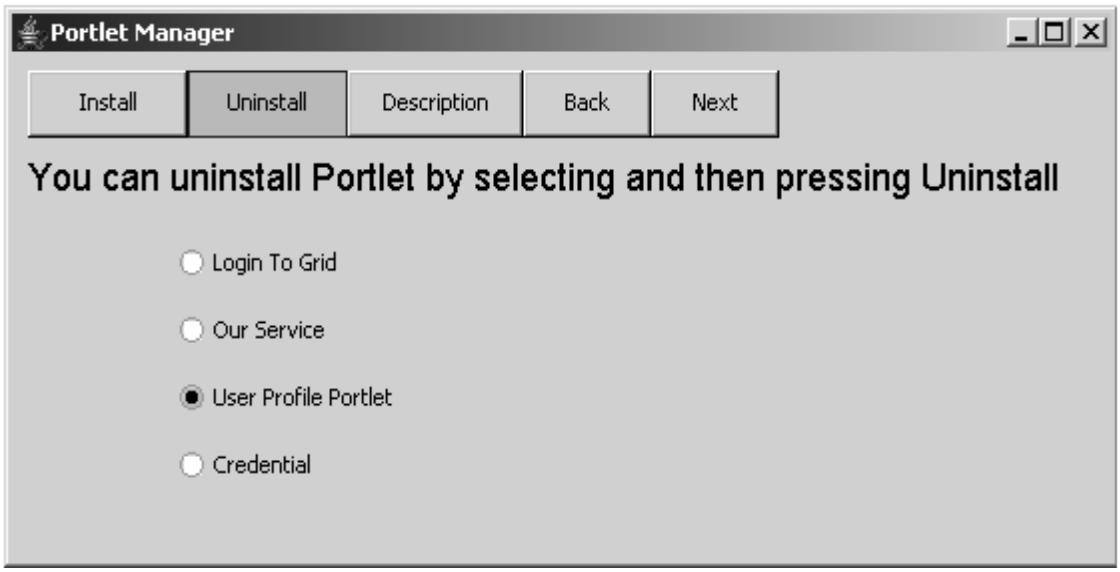
Screen#12



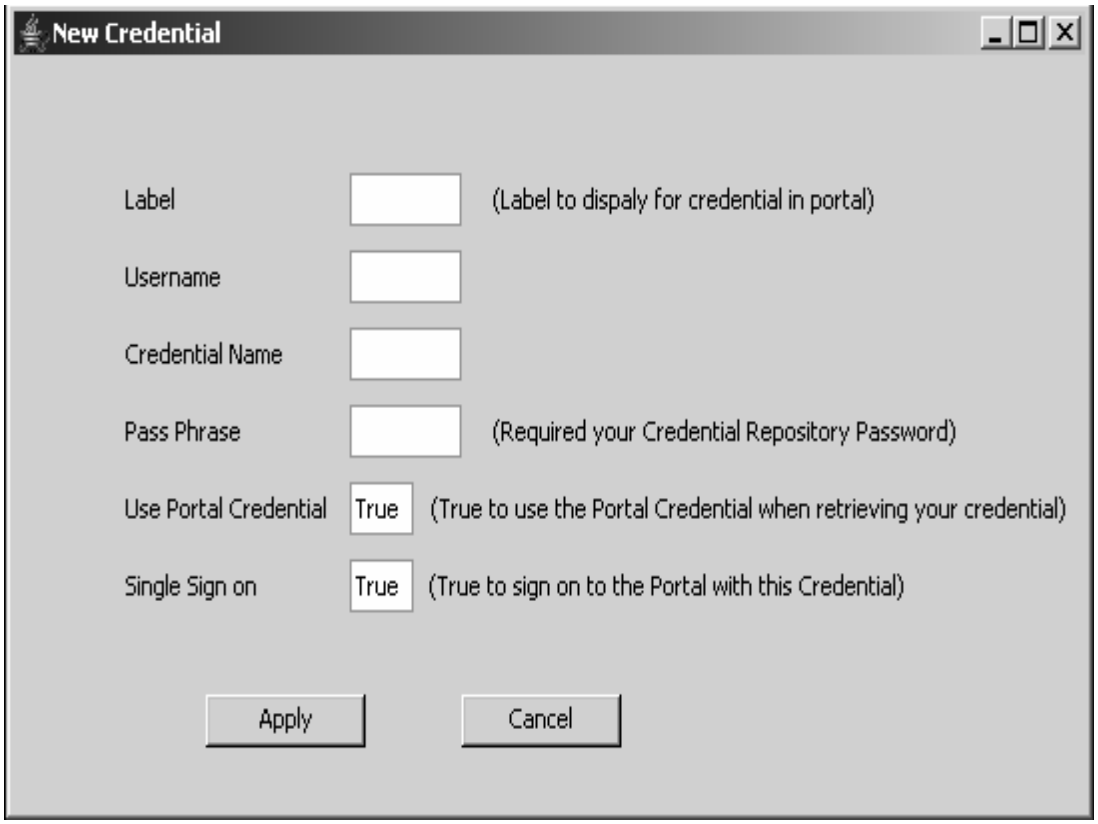
Screen#13



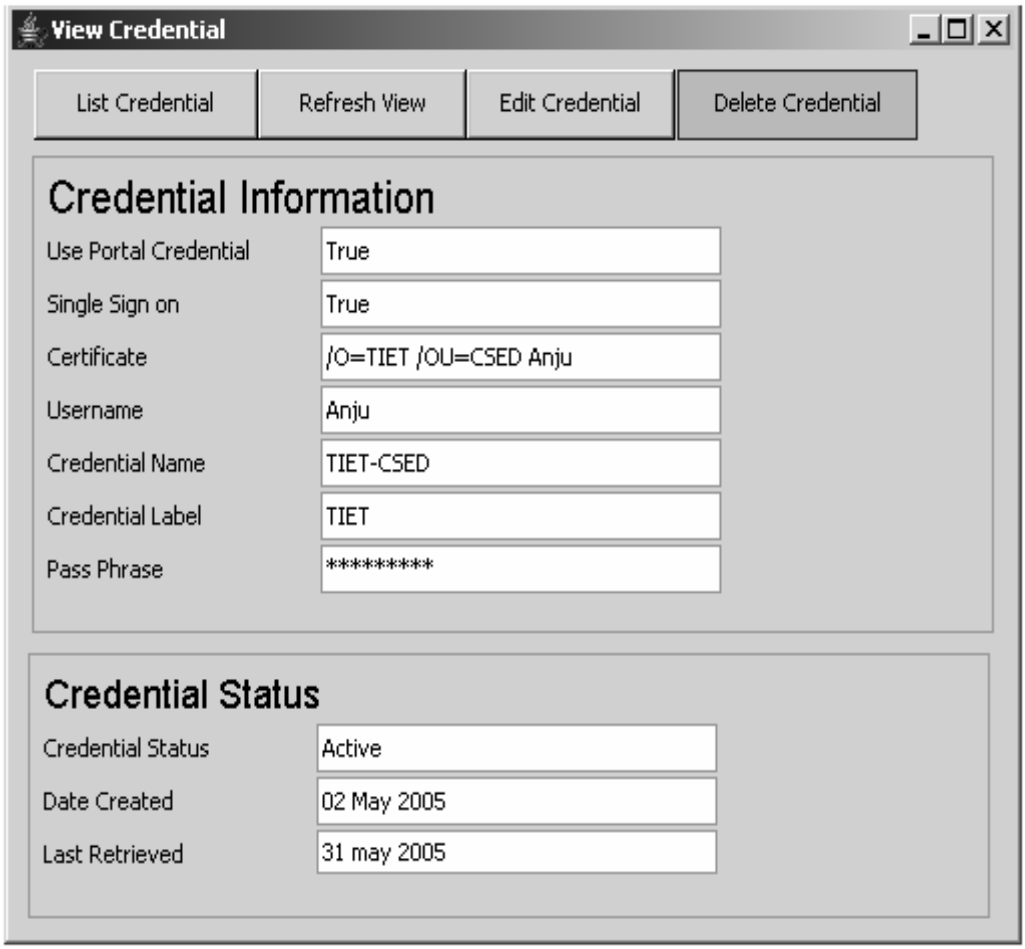
Screen#14



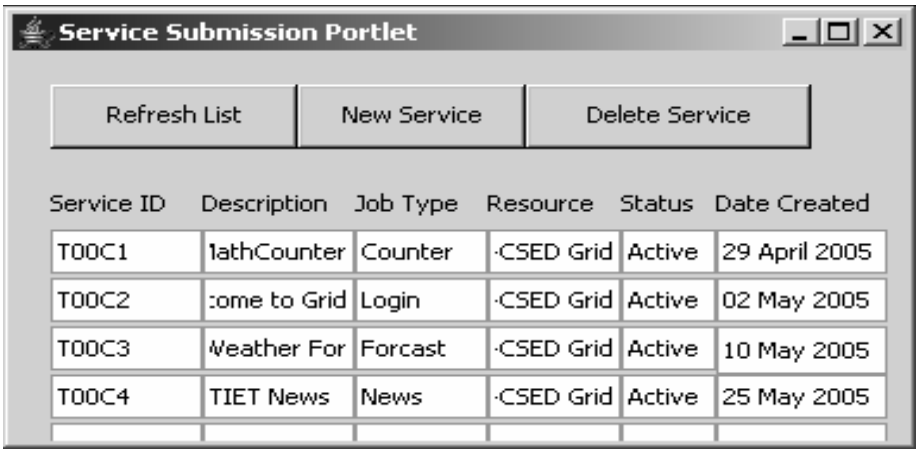
Screen#15


New Credential
Label
(Label to dispaly for credential in portal)
Username
Credential Name
Pass Phrase
(Required your Credential Repository Password)
Use Portal Credential
True
(True to use the Portal Credential when retrieving your credential)
Single Sign on
True
(True to sign on to the Portal with this Credential)
Apply
Cancel

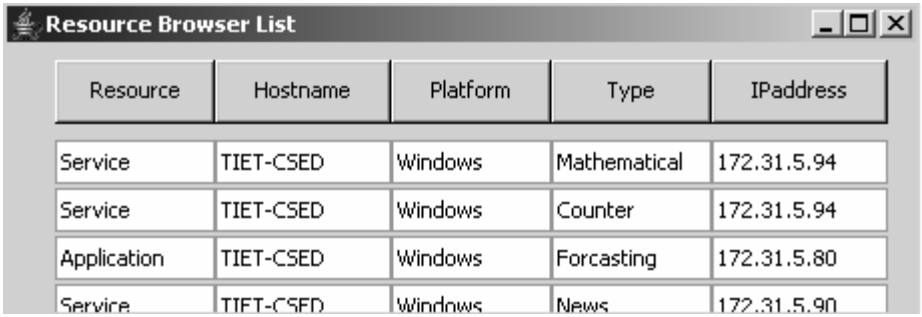
Screen#16


View Credential
List Credential
Refresh View
Edit Credential
Delete Credential
Credential Information
Use Portal Credential
True
Single Sign on
True
Certificate
/O=TIET /OU=CSED Anju
Username
Anju
Credential Name
TIET-CSED
Credential Label
TIET
Pass Phrase
****
Credential Status
Credential Status
Active
Date Created
02 May 2005
Last Retrieved
31 may 2005

Screen#17



Screen#18



Screen#19

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

“Design is not for philosophy - it's for life.”

- Issey Miyake.

The thesis provides the introduction to the Grid computing, it emphasizes the usage of Grid Portlets: user interface components to Grid applications, and discusses in detail the design and development aspects of portlets. A good Portlet design can be used for the life of the application and extends the life of the application. The Portlets Architecture provide us with number of advantages like

- Makes component approach very compatible with the Grid service model.
- Make it easy to add new Grid services.
- Many different Groups can contribute to design a portal.
- User can select and configure Portlets he/she wishes to use – Selection becomes part of persistent context.

Portlet technology benefits both the Producers and Consumers as it provide Flexibility. The Standards help to reduce the conflicts between best-of-breed versus single-provider acquisitions.

Various Portlets have been implemented in this work to demonstrate the effectiveness of Portlet for a Grid.

5.2 FUTURE

Grids have attracted a lot of interest by large companies that can use it to connect geographically diverse offices and unify the supply chain. Portlets, in simple terms, provide a way to do this. In industrial sector, IBM is putting in the most effort to this point, but certainly HP, Oracle and Sun are all very engaged. Good numbers of academic institutions like IITs are also adding to their grid computing resources.

From the above details we can conclude that the future of Grid services, based on the array of current technologies, is very healthy. As more companies provide Grid Portlets that are easy and flexible to use, more people make use of Grid Portlet standards and provide Grid portals and services of their own. The ability to closely monitor the execution of components for individual clients will also appeal to companies that want to make their existing computing power available for other Grid projects. Therefore, good Portlet design, development and deployment are critical and necessary. Further to this work, Portlets can be developed which have the access to the various functionalities that only Servlets supports.

REFERENCES

Books:

[1]. J. Joseph, C. Fellenstein, “Grid Computing”, Prentice Hall/IBM Press, Edition 2004

Tutorials and Manuals:

[2]. Borja Sotomayor, The Globus Toolkit 3 Programmer's Tutorial, Edition-2004

[3]. Grid Service Development Tools Guide from Globus

[4]. Globus ToolKit 3.0 Quick start from IBM

[5]. Java Programmer’s Guide Core Framework from Globus

Journals:

[6]. Foster, C. Kesselman. and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications , 15 (3). 200-222. 2001.

Papers:

[7]. Bill Nicholls,“Grids and Portals in Your Future”, Whitepaper,9Mar2001

[8]. Christian Antognini,”Is Oracle Database Moving Toward Grid Computing?” at www.trivadis.ch/Images/ grid_computing_en_tcm18-11759.pdf

[9]. Foster, C.Kesselman, S.Tuecke,”The Anatomy of the GridEnabling Scalable Virtual organizations, 2004” www.globus.org/research/papers/anatomy.pdf

[10]. Gregor Von Laszewski, “Grid Computing: Enabling a Vision for Collborative Research Argonne National Laboratory, 2002” www.mcs.anl.gov/~gregor/papers/vonLaszewski--para4.pdf

[11]. “*Introduction to JSR 168-The Java Portlet Specification*”,whitepaper at http://developer.sun.com

[12]. JAVA Portlets by Java™ Technology Evangelist Sun Microsystems, Inc. at www.javapassion.com/j2eeadvanced/Portlet4.pdf

[13]. Kropp,C. Leue, R. Thompson, “Web Services for Remote Portlets Specification”,www.oasis-open.org/committees/download.php/3343/oasis 200304-wsrp-specification-1.0.pdf, version 1.0, March 9, 2003

[14]. M.Wiley, *“The Java Portlet API(JSR 168)” 2001,* at media.wiley.com/product_data/ excerpt/13/04714695/0471469513.pdf

[15]. “Portals and portlets 2003” from OASIS, April 14, 2004 at http://tyne.dl.ac.uk/Papers/Portals/portals.pdf

[16]. “Portlet development from IBM WebSphere Portal for Multiplatforms”, Version 5.1 information center, Jan 2005, www.ibm.com/software/ - genservers/portal/library/pdf/ws_portal_51_G224-9144-00.pdf

[17]. “Portlets and Portals Design Overview”, websphere journal available at websphere.sys-con.com/read/45404.htm

[18]. Thomas Schaeck and Richard Thompson,"Web Services for Remote Portlets (WSRP)", Whitepaper, 28 may 2003

[19]. Ge He Zhiwei XU,Design and Implementation of a Web-Based Computational Grid Portal, International Conference on Web Intelligence,IEEE-2003

Thesis:

[20]. Anil. Kumar, "Data Grid", submitted at Computer Science and Department, Thapar Institute of Engineering and Technology,Patiala ,2003

[21]. B. Saxena, "Grid Computing,", submitted at Computer Science and Department, Thapar Institute of Engineering and Technology, Patiala ,2003

Internet links

[22]. Daniel Oelke,"Overview of Distributed Computing", Edition 2000 available at http://www.mithral.com/projects/cosm/ch-02.html

[23]. Developing Portlets from BEA WebLogic Personalization server 3.1.1 e-docs.bea.com/wlcs/docs31/p13ndev/portlets.htm

[24]. Globus Grid GT3 Install/Setup on Windows 2000 at www.bigdogsoftware.org

[25]. Grid Computing at www.grid.org

[26]. Grid Computing by Department of Applied Research in Information Technology, available at http://www2.nr.no/dart/projects/emerging/Grid.html

[27]. Thomas Schaeck and Stefan Hepper "Portal Standards", Edition 2002 available at http://websphere.sys-con.com/read/43168.htm

Others:

[28]. Michael Gerndt, “Grid Computing - An Infrastructure for Large Scale Simulations”,Institut fur Informatik, LRR Technische Universitat Munchen.

[29]. Jennifer M. Schopf, “*Grids: The Top Ten Questions*”, Mathematics and Computer Science Division, Argonne National Lab, Department of Computer Science, Northwestern University. www.**m**cs.anl.gov/~j**m**s/CV/cv.pdf

[30]. “Jportal”, by Apache Software Foundation,Edition April 2005

[31]. Chen Xin ,”Portlet Research in Grid Environment”,CS Department,Network Information Center of USTC 2002

PAPERS COMMUNICATED

[1]. Anju Singla and Seema Bawa " **Design and Development of Portlets:User-Interface Components to Grid Applications**" at Tencon'05, International Technical Conference sponsored by IEEE Region 10 to be held in Melbourne , Australia from November 21-24, 2005 **[Communicated].**