

國立交通大學

資訊科學系

碩士論文

MVP — 網頁元件與行動代理人系統的整合

MVP — An Integration of Web Components and Mobile Agent System for a More Versatile Portal

研究生：林奕宇

指導教授：袁賢銘 教授

中華民國九十三年六月

MVP — 網頁元件與行動代理人系統的整合
MVP — An Integration of Web Components and Mobile Agent System
for a More Versatile Portal

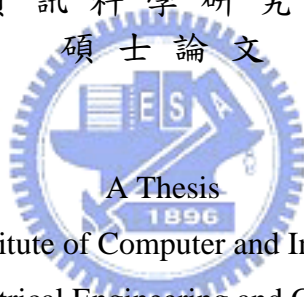
研究生：林奕宇

Student：Yi-yu Lin

指導教授：袁賢銘

Advisor：Shyan-Ming Yuan

國立交通大學
資訊科學研究所
碩士論文



Submitted to Institute of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

MVP — 網頁元件與行動代理人系統的整合

研究生：林奕宇

指導教授：袁賢銘

國立交通大學資訊科學研究所

摘要

隨著網際網路的蓬勃發展及各式各樣的網站如雨後春筍般興盛，功能多樣的網站應用與服務對數以百萬計的使用者來說已是不可或缺的。但一想到要如何有效率地使用這些應用與服務，卻是場惡夢。大部分的使用者為了要使用各網站提供的不同功能，必須要打開多個瀏覽器視窗，然後往返在數個網站之間，藉以把各項服務提供的資訊拼湊出自己所需要的結果。除此之外，現今的網站應用與服務多半是最普遍的 HTTP 通訊協定與使用者溝通，所以伺服器端的程式，不論是 CGI、PHP 或是 Java Servlet 等，其功能都會受限於必須符合 HTTP 通訊協定回應時間，而無法做出太過複雜、耗時的運算，或是具有智慧的決策。

在這篇論文中，我們將引入一個全新的架構，其中包含了客製化網站(Personal Customized Portal)以及的行動代理人(Mobile Agents)的觀念。如此一來，不但讓使用者可以輕易地把自己所需要的各項功能，以網頁元件(Web Components)的形式集中整理在一個網頁上，而且每一個網頁原件都被視為一個行動代理人。這些代理人(Agent)能夠有自主行為，而且有自己的生命週期，更可以有自己的簡易行事策略與判斷能力，經過一些簡單的設定之後，將能更有效率地為使用者達成任務。

MVP – An Integration of Web Components and Mobile Agent System for a More Versatile Portal

Student : Yi-yu Lin

Advisor : Shyan-Ming Yuan

Department of Computer and Information Science
National Chiao Tung University

Abstract

With the flourishing of Internet and the mushrooming of kinds of web sites, various web applications and services are now indispensable for millions of users. But it turns to a nightmare when someone has to deal with such a huge amount of applications and services scattered all over Internet. Most users need to launch several browser windows, gather information from each site, and analysis it in order to acquire something they expect. Additionally, these applications and services do the jobs synchronously only. This is because CGI, PHP, Java Servlets, or other dynamic webpage technologies are deliberately designed to be compatible to the most popular web protocol: HTTP. As a result, the behavior and the lifecycle of application and service programs running on the server side are both confined to the reasonable response time for users or HTTP standard. Thus, these kinds of we applications and services are not able to perform asynchronous tasks. And thus, more complicated, time-consuming computation and intellectual decision also have such a constraint on response time.

In this paper, we will introduce a new framework which integrates the concepts of Personal Customized Portal and Mobile Agents. This framework not only makes it much easier for users to gather all necessary applications and services in a single portal, but also employs them as intelligent and autonomous agents to efficiently accomplish asynchronous tasks that need more complex computation, reasoning and decisive ability.

Acknowledgements

First of all, I'd like to thank my advisor Shyan-Ming Yuan for lots of suggestions and instructions. I also want to thank those learned and experienced senior members in our laboratory, especially those who belong to Web Technology team: Ming-Chun Cheng and Chi-Huang Chiu. They gave me many invaluable ideas and guide me to complete this thesis.

I'd also like to thank all the people that have ever had contribution to this thesis. And of course, I am very grateful to my adorable parents who always strive to support me in whether physical or mental aspects. Last but not least, I must thank my friends who keep company with me all the time, especially when I felt bored or depressed in writing this thesis.



Contents

MVP — 網頁元件與行動代理人系統的整合	i
MVP — An Integration of Web Components and Mobile Agent System for a More Versatile Portal	ii
Acknowledgements	iii
Contents	iv
List of Figures	v
Chapter 1 Introduction	1
Chapter 2 System architecture of MVP	9
2.1 Fragment	11
2.1.1 Itinerary	12
2.1.2 ActionStrategy	13
2.2 Fragment Container	15
2.3 Fragment Manager	16
Chapter 3 Using Scenarios of MVP Framework	19
Chapter 4 Development based on MVP	30
4.1 Development of Portal Application	30
4.2 Development of Fragments	34
Chapter 5 Related Work	38
5.1 Portal	38
5.2 Mobile Agent	40
Chapter 6 Conclusion and Future Work	42
References	46

List of Figures

Figure 1-1: Java Servlet Execution Scenario	2
Figure 1-2: Traditional Web Portal Viewed on Mobile Devices.....	4
Figure 1-3: Fragment traveling and aggregation	5
Figure 1-4: Portable Fragments for service aggregation.....	6
Figure 1-5: Aggregation and rearrangement process for display on Mobile Device..	7
Figure 2-1: Architecture of MVP Framework.....	9
Figure 2-2: Relationship and Interaction of MVP Components	10
Figure 2-3: Fragment Behaviors	11
Figure 2-4: Behaviors of Itinerary object of Fragments	13
Figure 2-5: Inner Structure and Behaviors of ActionStrategy	14
Figure 2-6: A decision tree made of ActionStrategy objects.....	15
Figure 2-7: Behaviors of Fragment Container	15
Figure 2-8: Behaviors of Fragment Manager.....	16
Figure 3-1: Fragment Traveling from Portal A to Portal B.....	19
Figure 3-2: Fragment Migration Example	22
Figure 3-3: Traffic Distribution Example	25
Figure 3-4: Fragment Traveling according to the Itinerary and Action Strategy.....	26
Figure 3-5: Fragment Locating and Messaging.....	27
Figure 3-6: Messaging Scenario I.....	28
Figure 3-7: Messaging Scenario II.....	29
Figure 4-1: MVP Sample Portal View on a Desktop Browser.....	32
Figure 4-2: MVP Sample Portal View on Mobile Devices.....	32
Figure 4-3: Class diagram for Arrangement	34

Figure 6-1: Façade gateway scenario.....45



Chapter 1

Introduction

The computer using habit has been substantially changed since World Wide Web (WWW) emerged from early 90's. Before WWW appeared, computer users were similar to the dwellers on desolate islets and accustomed to deal with local data and limited resources on their own computers. Since early 90's, WWW has connected the computers all over the world and offers friendly user interfaces and intuitive using procedures. Therefore, it becomes very easy for everyone to access various contents and services via WWW. Most people nowadays even use computers only for access WWW! All the daily-life errands or chores, such as news reading, shopping, searching data and materials for the work, or chatting and communicating with others, can be carried out after a few mouse clicks.



By reason of that, here comes more and more portals that provide or aggregate multifarious contents and services and arrange or organize them for using convenience. It has become a growing tendency that the first thing to do, or even the only thing for somebody, after connected to Internet is visit certain popular portal. As for most users, they don't have to worry about forgetting certain URL. They only need to launch one browser window, and then almost everything can be done on this portal. On the portal vendor's view, the more contents and services he provides, the more people will visit the portal, and then the more money will come.

Besides, portals, which are called Enterprise Information Portals (EIP) or Enterprise Application Portal (EAP), have been introduced into management of

information inside enterprises. In this way, kinds of members about the enterprises, say employees, managers, or investors, can be identified after logging, and be dealt with accordingly and immediately with a variety of one-stop services and supports.

However, several problems rise while the portals are getting more and more popular:

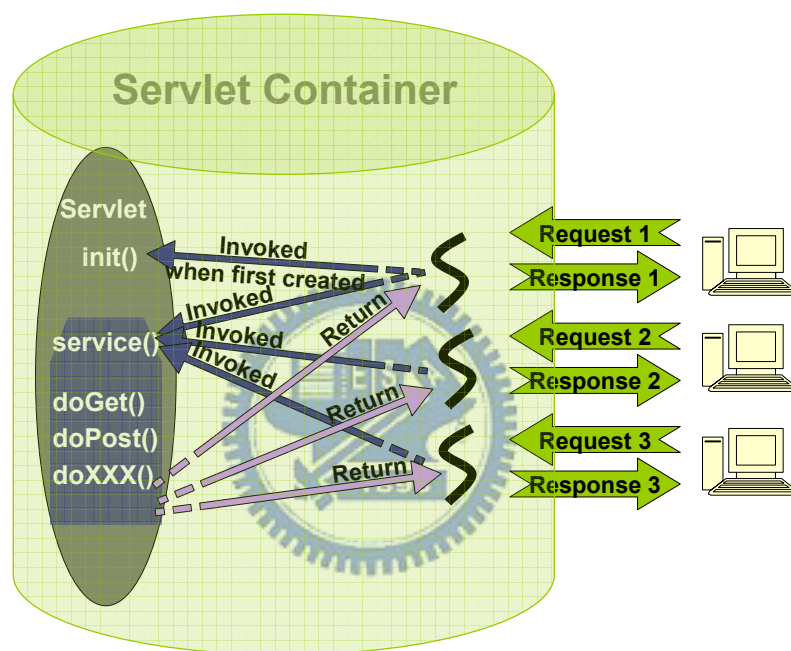


Figure 1-1: Java Servlet Execution Scenario

1. Traditional Web application cannot do jobs asynchronously. HTTP restricts the functionalities of portals whether they are for common users or used as EIP or EAP. For fear of HTTP timeout or a lengthy time for users to wait for, portals nowadays can simply offer services of querying, updating or modifying, and then display the data and information in an appropriate way. Furthermore, almost all of the web technologies for generating dynamic pages, such as Common Gateway Interface (CGI), PHP: Hypertext Preprocessor (PHP [1]), or Java Servlet [2] and

Java Server Pages (JSP [3]), are deliberately designed for the compatibility to HTTP. Take Java Servlet for an example, as shown in the Figure 1-1, what a Servlet instance can do is confined to the period between the initiation of the request and response. When a request is initiated by a client, a thread will be spawned in the Servlet container and then execute the Servlet code. The thread will invoke `init()` method in the Servlet if this Servlet is executed first time, do the tasks described in the `service()` method inclusive of `doGet()`, `doPost()`, or `doXXX()`, which are corresponding to the HTTP methods, and then produce response to the client. This process for generating dynamic web page is typical of synchronous interaction. Furthermore, it is not suitable for more complicated, time-consuming, and intellectual computation because it can only do things that must be finished in a restricted period of time.

2. As far as portal users are concerned, they have no choice but to use services fastened to the portals. For example, if someone, say user A, is a regular visitor of portal B, but now he needs some of portal C's and portal D's services, say search engine and city guide which are much powerful than those on portal B. The only way for A to do is to open several browser windows and access them respectively. Consequently, users have to remember many URLs of portals, and then continuously switch plenty of browser windows to pick the services they want. This absolutely violates the original intension of web portals.
3. General mobile devices, suchlike PDAs or mobile phones, have tiny screens compared with those of desktop computers or laptops. Users therefore will get awfully poor appearances when trying to browse common web pages, as demonstrated in Figure 1-2. For the time being, common web portals or EIP and EAP either don't take mobile devices users into consideration, or proffer a much simpler and trimmed version dedicated to fit the size of mobile device screens. It

costs much time and efforts to re-author a dedicated version for mobile devices. Nevertheless, the employment of all kinds of portals from mobile devices is not supposed to be neglected due to the fact that the number of mobile device users is getting lager and larger. Users are not supposed to be confused and waste time being adapted to different user interfaces between normal and dedicated version which are both for presenting the same content.



Figure 1-2: Traditional Web Portal Viewed on Mobile Devices

In order to solve the problems mentioned above, we propose a whole new framework: More Versatile Portals (MVP). Two concepts, web components and mobile agents, are adopted into MVP as the basis. As MVP stands for, these two concepts are going to make portals more versatile. The first one, web components, means integral parts that present fragments of a page individually. That is, several web components can constitute an entire page. This idea is quite different from the traditional one that always regards a web page as just a tree structure comprised of

HTML tags. And the second one, mobile agents, is a very popular technology and technique in Distributed Computing. The main idea of mobile agents is to consider a program or process an automatic and intellectual agent, and this agent can be transferred among hosts all over the network whenever it needs computing power or resources from other hosts.

In the MVP architecture, the concepts of web components and mobile agents are integrated into a single component. We will call it a fragment agent or a Fragment in brief in the following paragraphs and chapters. As the Figure 1-3 shows, there are several computational logics, or say Fragments, in every portal. These Fragments have autonomous behaviors and are able to accept users' instructions and do the designated jobs asynchronously. They can also travel around portals according to certain purposes. Afterwards, they are aggregated to produce a page to show the results or the user interfaces to the users.

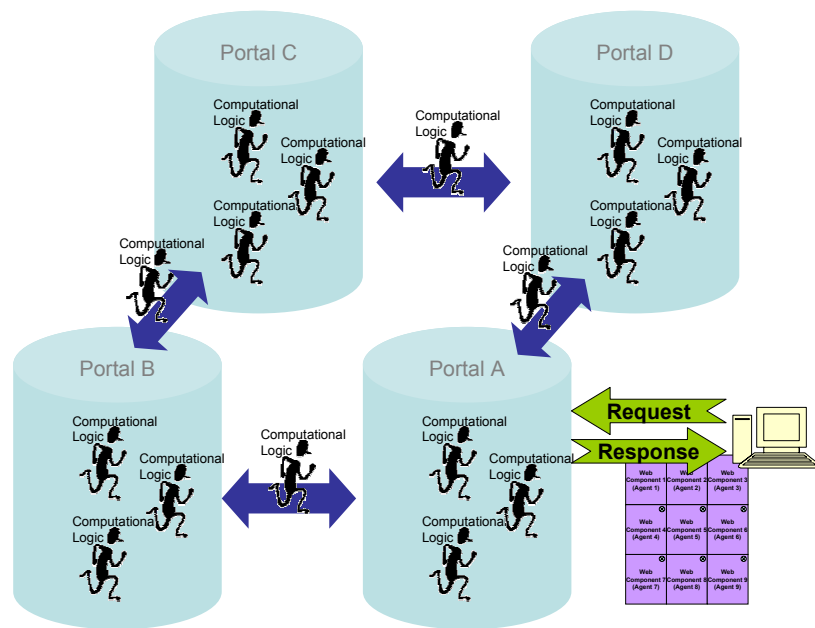


Figure 1-3: Fragment traveling and aggregation

As a result, every portal page can be composed of many Fragments, and each of which has its own functionalities and tasks to be performed. Portal users are allowed to gather these specific-purposed Fragments into a single page and designate them to do tedious and asynchronous work as automatically and intellectually as they can. People thus can do much more things through MVP than using the traditional portals that can only do very simple jobs constrained by the response time.

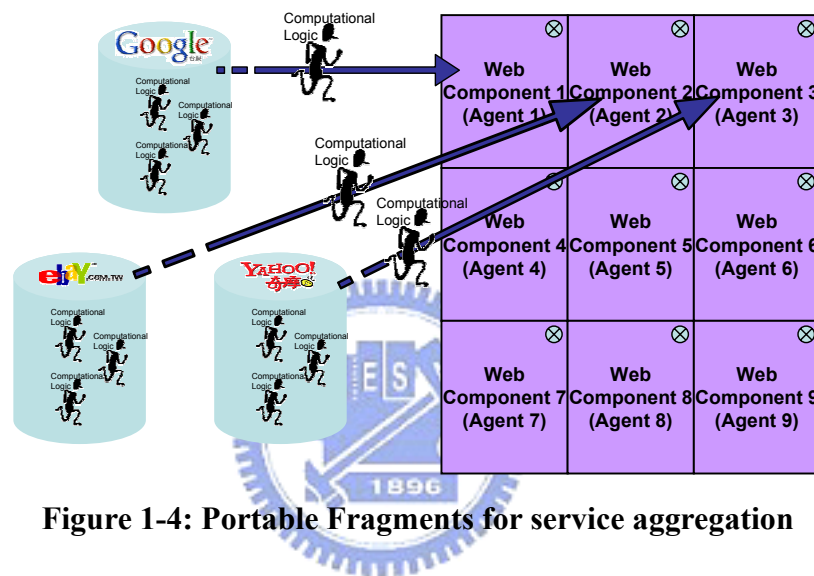


Figure 1-4: Portable Fragments for service aggregation

Moreover, if every portal is based on MVP, the Fragments of each portal can be interchanged to each other. Fragments all around the network are also able to be aggregated into one portal to serve certain users. People who are not satisfied to any of existing portals consequently have a chance to customize a specialized one for personal requirement. Figure 1-4 provides an example to illustrate this concept. If Google, eBay, and Yahoo portals are established on MVP framework, we can thus collect Google search engine, eBay shopping mall, and yahoo news into a single Fragment-style page. People who were used to open several browser windows simultaneously can now manipulate all contents and services they need in a single one.

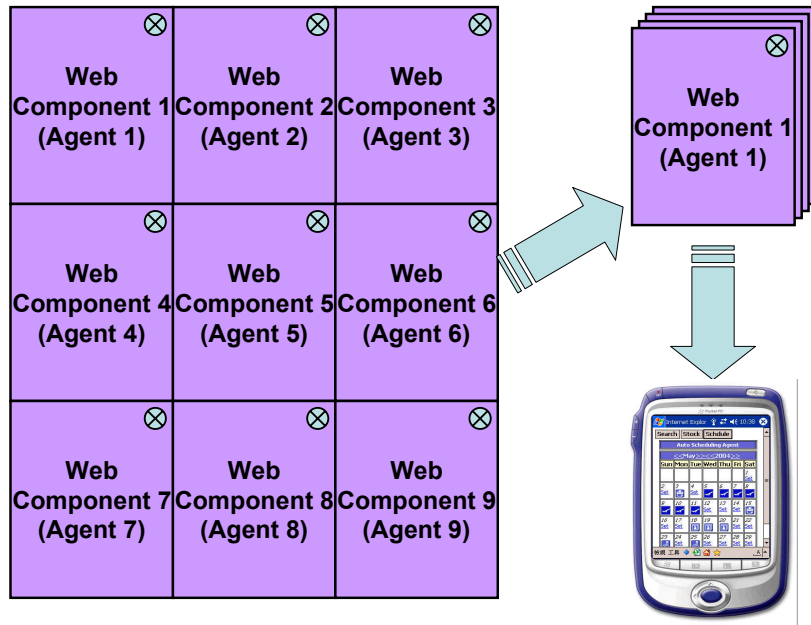


Figure 1-5: Aggregation and rearrangement process for display on Mobile Device

Mobile device users are also taken into account. Fragments in MVP framework are born to present parts of pages respectively. When mobile clients send requests to MVP-based portals, the portals can rearrange the Fragments to form an appropriate display. As shown in Figure 1-5, all Fragments are considered cards. When the portals want to display for the mobile devices, they will be rearranged as a deck of cards and sent back to the client side. Users may choose the card they want at a time in the display, or they can only download the cards they want to deal with according to their personal configuration or connection reliability of the mobile devices. Therefore portals which exploit MVP are capable of displaying suitable pages for mobile devices. It is not necessary for portal vendors to re-author dedicated pages for mobile devices.

The rest parts of this thesis are organized as follows: Chapter 2 explains the system architecture of MVP; Chapter3 illustrates several useful using scenarios for

the MVP framework; Chapter 4 indicates how to develop Fragments and Portals based on MVP framework, and give some development examples; Chapter 5 introduces some related work; Chapter 6 give the conclusion and propose plans about future work.



Chapter 2

System architecture of MVP

MVP is based on Java programming language [4]. Servlet specification of J2EE, Java Portlet Specification from JSR 168 [5] [6] are also referenced. MVP framework also adopts concepts of common mobile agents [7] [8] and the design of other implementation of mobile agent system, such as Aglet [9].

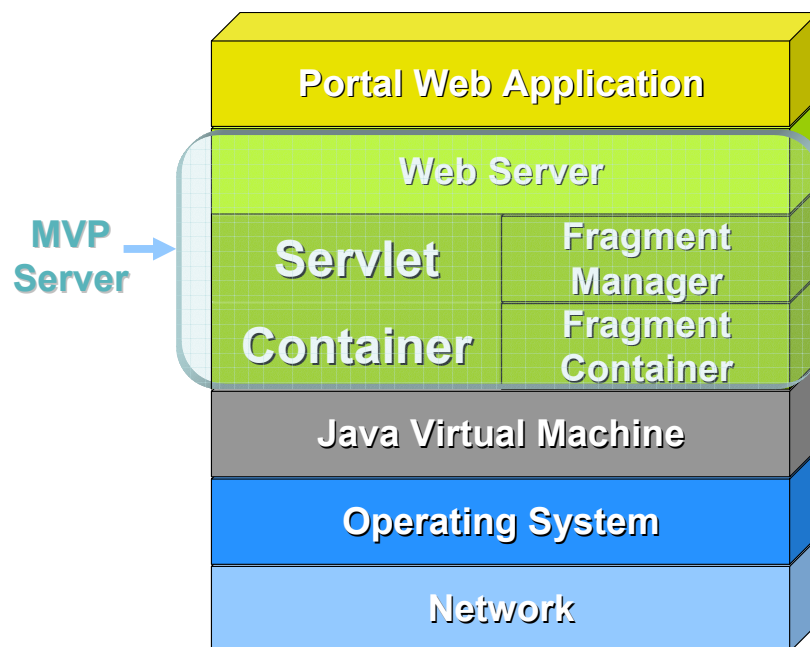


Figure 2-1: Architecture of MVP Framework

Figure 2-1 shows the system architecture of MVP. MVP server is established on the Java Virtual Machine (JVM). It consists of four components: Web Server, Servlet container, Fragment Container, and Fragment Manager. Web Server and Servlet container provide an execution environment for certain functionalities of the portal web application; Fragment Container and Fragment Manager store and manage the necessary Fragment instances for users. Finally, a portal web application runs on the

Web Server. Portal web application exploits the resources and power of Web Server and Servlet Container to generate friendly user interfaces, handle user requests, arrange Fragments, and generate appropriate display for users.

As Figure 2-2 demonstrates, when a request is sent by a user, portal web application will process it and decide the Fragments that comprise an entire page. The Fragment Manager is afterwards asked to retrieve required Fragment instances from Fragment Container, or produce Fragment instances that are required but doesn't exist yet. After Fragments generating contents according to the user requests, portal web application can load these Fragment contents and aggregate their contents to form a whole page. Consequently, the complete page will be sent to the client who sent the request before.

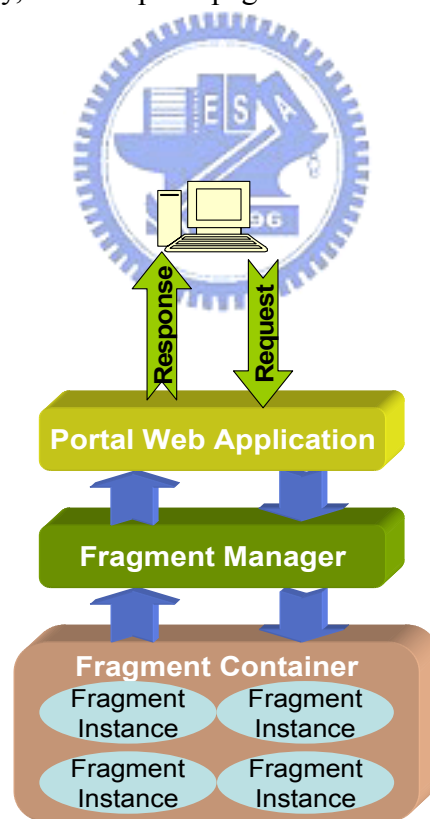


Figure 2-2: Relationship and Interaction of MVP Components

In MVP server, there are no designated distributions or implementations for Web

Server and Servlet Container. That is, all kinds of web servers and Servlet containers are allowed to be used as long as they can correctly process HTTP protocol and follow the Servlet standard in J2EE. The following three sections are going to explain respectively the design concepts and internal operation of the rest parts of MVP: Fragment, Fragment Container, and Fragment Manager.

2.1 Fragment

Fragments play the role of web components when it comes to generating pages that display user interfaces and execution result. On the other hand, Fragments are considered agents that are specialized for certain tasks. Therefore, Fragments need to have their own lifecycles to accomplish their individual missions. There are corresponding designs for both web components and mobile agents.

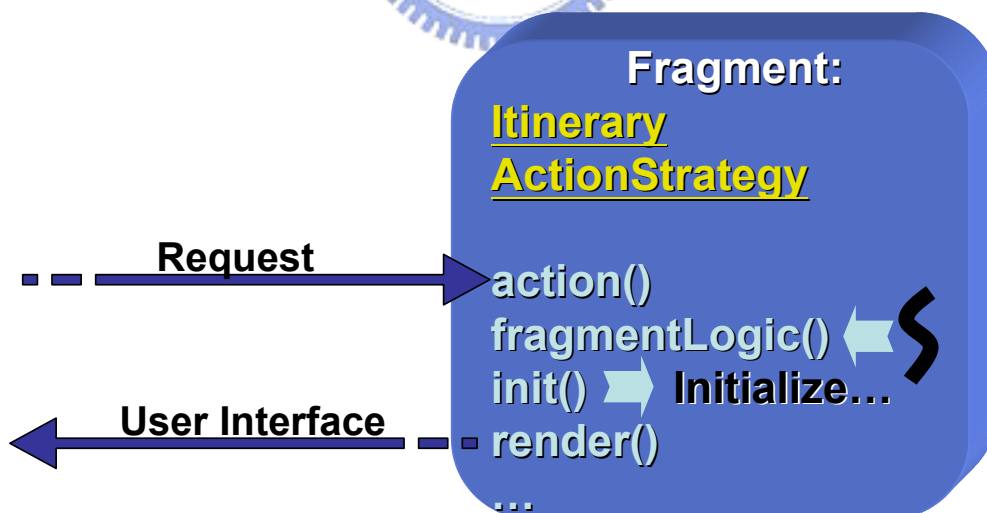


Figure 2-3: Fragment Behaviors

As shown in Figure 2-3, `action()` method in a Fragment will receive the request passed by the web portal application and act as the request said, and

`render()` method is responsible for generating the user interface and display the execution result belonging to this Fragment. Besides, acting as a thread, a Fragment is therefore able to have its own lifecycle. The `fragmentLogic()` method contains the code for the Fragment routine behavior, and the Fragment thread will execute the `fragmentLogic()` method until the task finishes. Users certainly can request to change the status and lifecycle of this Fragment thread by setting some attributes. For instance, if the `suspend` attribute is set as `true`, the Fragment is asked to be suspended.

Fragments also require some autonomous and intellectual properties for most of the tasks. Whenever a Fragment needs to travel around the hosts or make some decisions, it will act according to what its `Itinerary` and `ActionStrategy` attributes said. `Itinerary` holds the necessary route for the task, while `ActionStrategy` is a Java interface that defines a set of methods which make decisions. The following two subsections will explain the design of `Itinerary` and `ActionStrategy`. Besides, Chapter 3 will show that how a programmer use `Fragment` class to develop his own Fragment agent, and chapter 4 will compare `Fragment` with the component of other related or similar research.

2.1.1 Itinerary

Just imagine that your mother ask you to buy her today's newspaper and something for breakfast. To accomplish what she wants, you have to determine your own itinerary. For example, you may go to a newsstand near your home first, and then buy sandwiches, bread and milk from a food stand or supermarket. Finally, you go back home and enjoy breakfast with your family. Likewise, a Fragment which is assigned to perform specified jobs also needs its own itinerary. Before the Fragment is sent for designated tasks, it must be informed of that where to go and what to do.

The `Itinerary` object of a `Fragment` is designed for this purpose. It contains address of all stops while the `Fragment` is traveling around the network. Whenever a `Fragment` is dispatched to do jobs, its `Itinerary` object should be configured correctly. Figure 2-4 illustrates details of `Itinerary`. We can see from the figure that the `Itinerary` object also provides several methods for the sake of configuration:

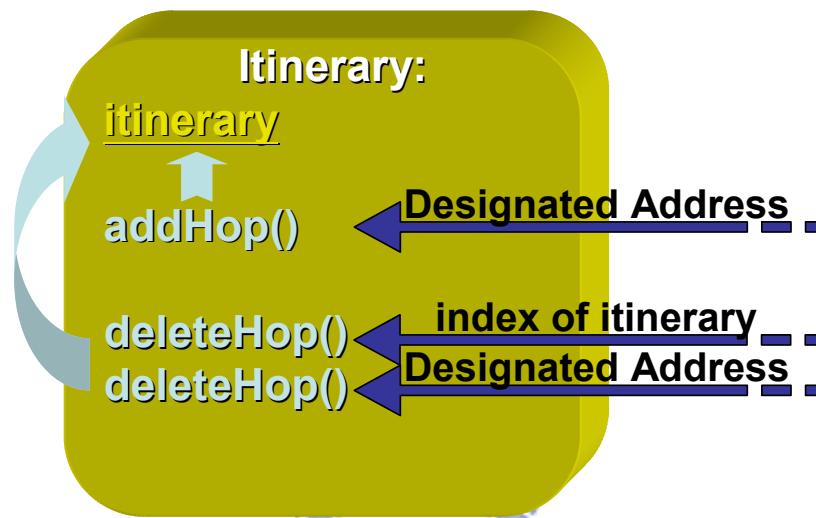


Figure 2-4: Behaviors of `Itinerary` object of `Fragments`

2.1.2 `ActionStrategy`

A `Fragment` is always assigned to do different jobs on different hosts or portals. Whenever a `Fragment` arrives at a host, it needs to fulfill what it was assigned to do. In some situation, however, the job to be fulfilled is not very simple so that the `Fragment` needs some rules or criteria to help it. For example, if a `Fragment` is sent to bid for something such as antiques, it is important to have a smart and effective strategy. Thus, `ActionStrategy` is designed for this necessity.

The design of `ActionStrategy` is similar to a traditional solution of Artificial

Intelligence: decision tree. Figure 2-5 and Figure 2-6 show the inner structure and behaviors of `ActionStrategy` and collaboration of `ActionStrategy` instances, respectively. Figure 2-5 tells that an `ActionStrategy` object has a `decide()` method. This method contains main computational logic for a certain rule or behavior. After doing things according to the rule, `decide()` method produces a result which determines what to do in the next step. Assume that there is a `Fragment` which was assigned to bid for a digital camera and dedicated accessories for it, for example, storage media and batteries, in an auction. Then if the `Fragment` gets the batteries that are designed for a certain brand, it is supposed to bid for the camera of the specified brand in the next step.

For a sequence of dependent decisions, we need a data structure that is similar to a decision tree to solve this problem. As Figure 2-6 depicts, we consider an `ActionStrategy` object a node of a decision tree. Each `ActionStrategy` object will try to accomplish certain task and then decide which node should be executed next. By traversing an entire decision tree, a `Fragment` can handle all conditions whenever a decision is need to be made. In this way, a `Fragment` can autonomously do a series of jobs while traveling around the network as long as the decision tree was configured appropriately in advance.

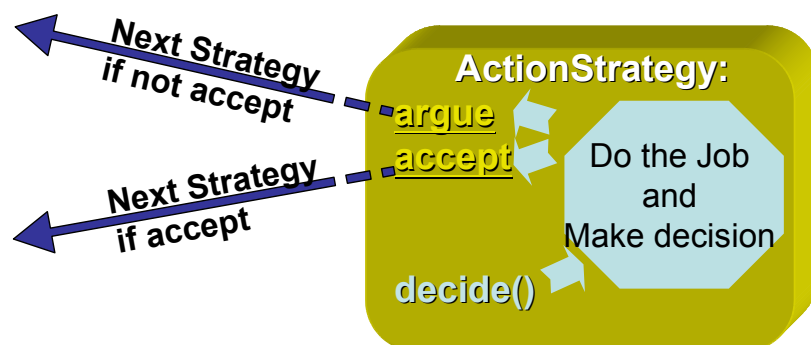


Figure 2-5: Inner Structure and Behaviors of `ActionStrategy`

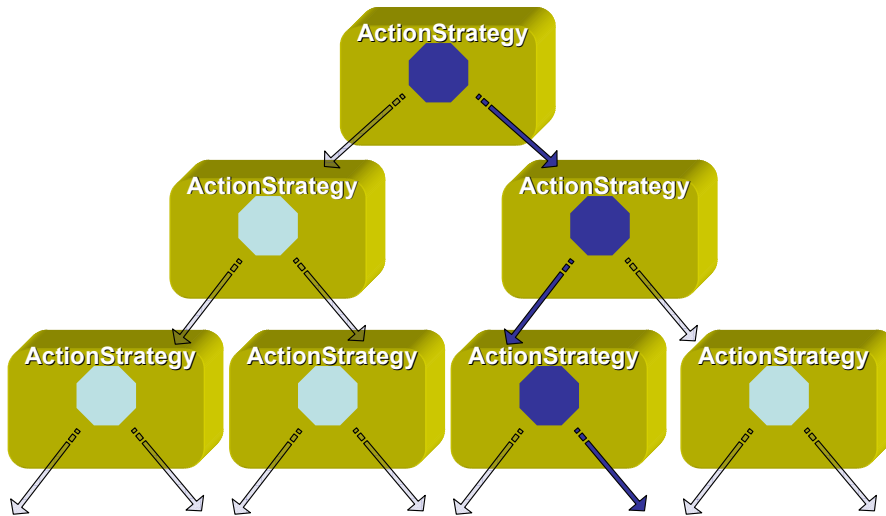


Figure 2-6: A decision tree made of ActionStrategy objects

2.2 Fragment Container

Fragment Container is just like storage of kinds of MVP data, and there is another component, Fragment Manager, that is designed as the stowage keeper. Whenever the data stored in Fragment Container are need to be modified, Fragment Manager will be informed and do the modification. This can guarantee the correctness and consistency of the modification.

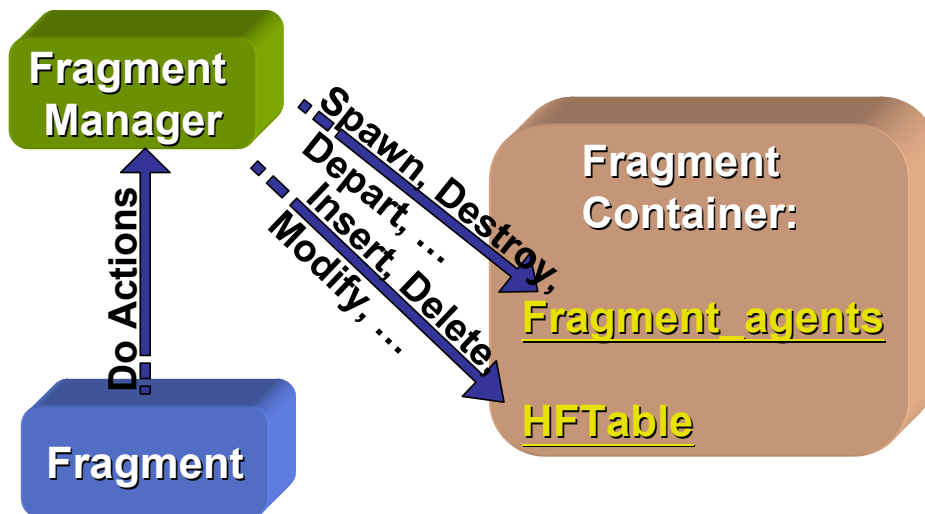


Figure 2-7: Behaviors of Fragment Container

Fragment Container includes two major parts: `Fragment_agents` and `HFTable`. `Fragment_agents` is stowage of Fragment instances while `HFTable` keeps track of the current position of Fragment instances. The content of `Fragment_agents` and `HFTable` should be modified by Fragment Manager whenever a new Fragment instance is spawn, destroyed, or a Fragment instance has finished his work on a host and has to travel to the next stop along the determined route. For example, as demonstrated in Figure 2-7, when a new Fragment instance is needed to be spawned, Fragment Manager will be notified to create a new Fragment instance, and store it into the Fragment Container. At the same time, Fragment Manager inserts a new entry into `HFTable` for the new Fragment instance. This entry is a registry indicating the present location of this Fragment instance. Then if this Fragment instance is going to be destroyed, Fragment Manager will delete the related entry in `HFTable` of Fragment Container.



2.3 Fragment Manager

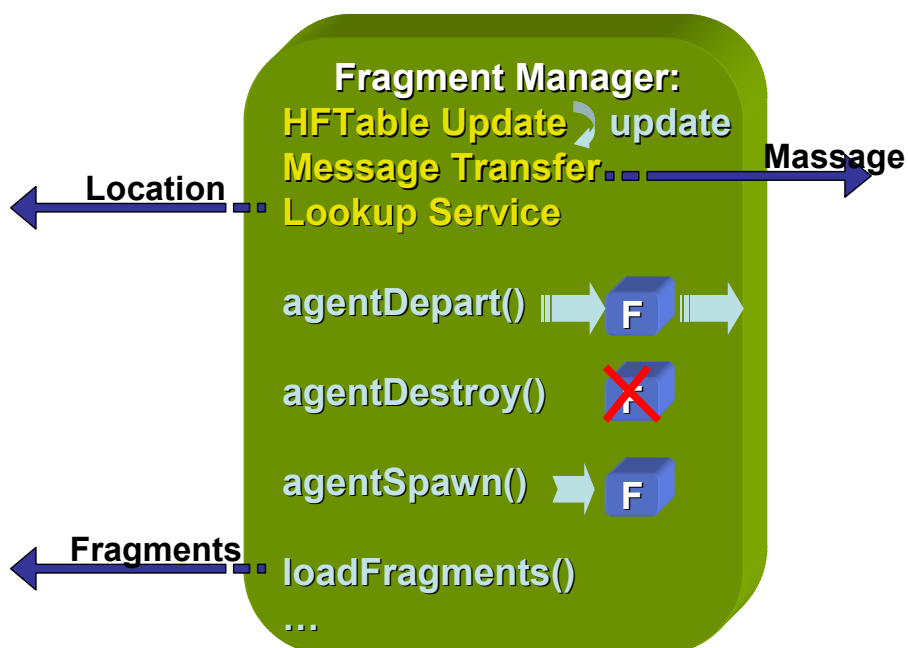


Figure 2-8: Behaviors of Fragment Manager

As mentioned in section 2.2, Fragment Manager is the stowage keeper of the Fragment Container. Thus, Fragment Manager provides managing methods for several purposes. Figure 2-8 shows that `loadFragments()` obtains the required Fragment instances in the Fragment Container. While `agentSpawn()` creates a new Fragment instance and store it into Fragment Container, `agentDestroy()` destroys the perishing Fragment instances in the Fragment instance. In chapter 3, there will be more detailed explanation for these methods.

Besides, several services are provided in order for the advanced manipulation to facilitate Fragment behaviors:

1. Agent Departure:

You may regard this service as an airport for Fragment to go abroad. When a Fragment is going to depart for some purpose, it can invoke agent departure service, and then this service will transfer the Fragment to the designated destination.

2. HFTable Update:

HFTable update service in Fragment Manager is responsible for modifying entries in the HFTable of Fragment Container. As changing current location, a Fragment should inform the HFTable in its birthplace of the destination address, and then HFTable update service in its birthplace will update the corresponding entry with the latest information.

3. Message Transfer:

Fragments might need to collaborate for a certain purpose. As a result, they have to exchange data and information in order to work together. Message Transfer service can help fragments to interchange information and data by means of sending messages to each other.

4. Lookup Service:

Lookup service of Fragment Manager is designed to obtain the latest location of a Fragment. Whenever a Fragment or a service is about to find another Fragment, the lookup service in the birthplace of the target Fragment will be invoked and return the current position of the Fragment to the requester.

We will see some of the popular using scenarios of MVP framework in the next chapter.



Chapter 3

Using Scenarios of MVP Framework

The specialized services mentioned in the former chapter compose an infrastructure that facilitates various behaviors of Fragments. We will show some using scenarios for demonstration in the following paragraphs:

1. A Fragment is appointed to migrate from a portal to another, as Figure 3-1. A user employs HTTP protocol to issue a request to ask some Fragment leave for another portal. Fragment Manager then transfers the requested Fragment to the destination.

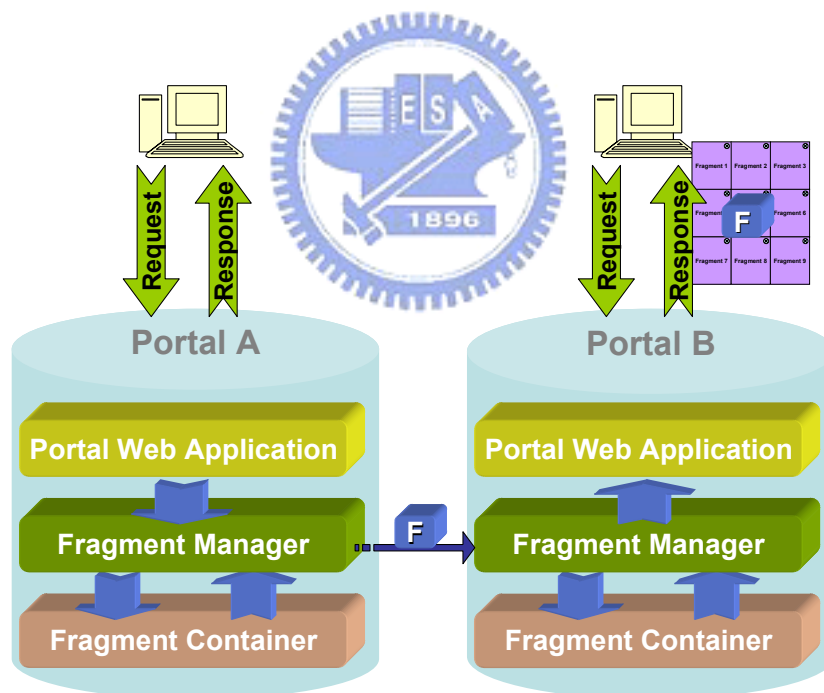


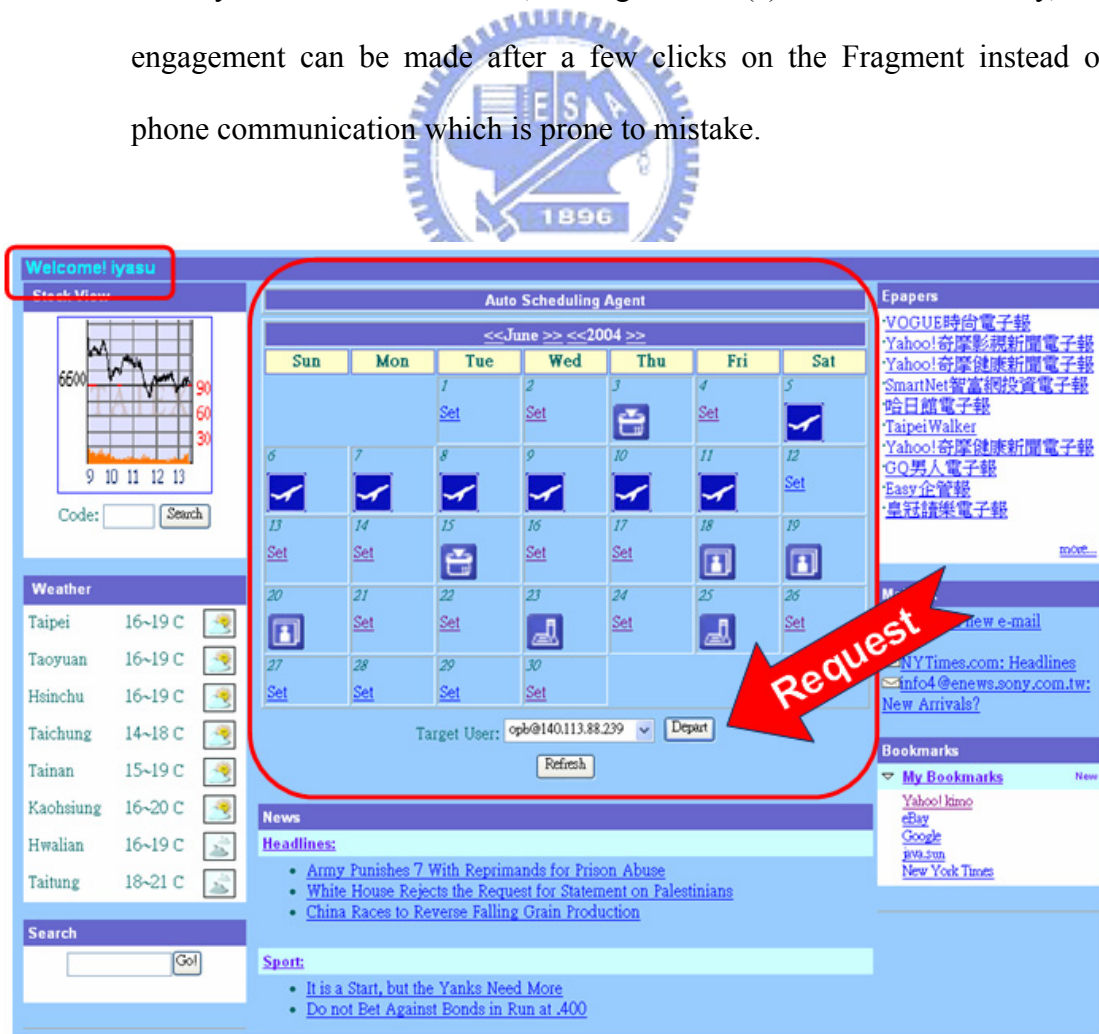
Figure 3-1: Fragment Traveling from Portal A to Portal B

Use case:

- I. Suppose there are several portals that are all based on MVP framework. A user A needs some services provided by these portals respectively. For

using convenience, user A might want to aggregate all services he needs from different portals into a single portal. Thus user A can send request to the Fragments that are responsible for the services, and direct them to certain portal for assembling a page full of desired services.

II. User A might need to negotiate with user B who is registered on a different portal for some matters or business. For example, as Figure 3-2 shows, a user opb is going to make an appointment with iyasu. In Figure 3-2 (a), we can see that iyasu can directly send his scheduler to opb. User opb thus can understand the schedule of each month after a glance. And then, in Figure 3-2 (d), opb can communicate with the coming Fragment and choose the days he wants for a date, as Figure 3-2 (e) shows. In this way, the engagement can be made after a few clicks on the Fragment instead of phone communication which is prone to mistake.



(a): User iyasu sends an auto scheduler Fragment to user opb

Welcome! iyasu

Refresh

Stock View

Code: [] Search

News

Headlines:

- Army Punishes 7 With Reprimands for Prison Abuse
- White House Rejects the Request for Statement on Palestinians
- China Races to Reverse Falling Grain Production

Sport:

- It is a Start, but the Yanks Need More
- Do not Bet Against Bonds in Run at .400

Astrology

Gemini

Fortune Index: ☆

Wealth: Decrease unnecessary outgo.

Mood: Social engagements such as dinner party should be avoided.

Business: As busy as it could be

Love: Good communication skills are needed.

Weather

Taipei 16~19 C

Taoyuan 16~19 C

Hsinchu 16~19 C

Taichung 14~18 C

Tainan 15~19 C

Kaohsiung 16~20 C

Hwalian 16~19 C

Taitung 18~21 C

Epapers

- VOGUE時尚電子報
- Yahoo!奇摩影視新聞電子報
- Yahoo!奇摩健康新聞電子報
- SmartNet智富網投資電子報
- 哈日館電子報
- TaipeiWalker
- Yahoo!奇摩健康新聞電子報
- GO男人電子報
- Easy企管報
- 皇冠娛樂電子報

more...

Mailbox

You have 2 new e-mail

- NYTimes.com: Headlines
- info4@enews.sony.com.tw: New Arrivals?

Bookmarks

My Bookmarks

- Yahoo! kimo
- eBay
- Google
- gva.sun
- New York Times

Search

[] Go!

(b): The Fragment has left

Welcome! opb

Search

[] Go!

Weather

Taipei 16~19 C

Taoyuan 16~19 C

Hsinchu 16~19 C

Taichung 14~18 C

Tainan 15~19 C

Kaohsiung 16~20 C

Hwalian 16~19 C

Taitung 18~21 C

Stock View

Code: [] Search

News

Headlines:

- Army Punishes 7 With Reprimands for Prison Abuse
- White House Rejects the Request for Statement on Palestinians
- China Races to Reverse Falling Grain Production

Sport:

- It is a Start, but the Yanks Need More
- Do not Bet Against Bonds in Run at .400

Astrology

Gemini

Fortune Index: ☆

Wealth: Decrease unnecessary outgo.

Mood: Social engagements such as dinner party should be avoided.

Business: As busy as it could be

Love: Good communication skills are needed.

Epapers

- VOGUE時尚電子報
- Yahoo!奇摩影視新聞電子報
- Yahoo!奇摩健康新聞電子報
- SmartNet智富網投資電子報
- 哈日館電子報
- TaipeiWalker
- Yahoo!奇摩健康新聞電子報
- GO男人電子報
- Easy企管報
- 皇冠娛樂電子報

more...

Mailbox

You have 2 new e-mail

- NYTimes.com: Headlines
- info4@enews.sony.com.tw: New Arrivals?

Bookmarks

My Bookmarks

- Yahoo! kimo
- eBay
- Google
- gva.sun
- New York Times

(c): The original page of user opb

Welcomel opb

Search Go!

Weather

Taipei 16~19 C

Taoyuan 16~19 C

Hsinchu 16~19 C

Taichung 14~18 C

Tainan 15~19 C

Kaohsiung 16~20 C

Hwalian 16~19 C

Taitung 18~21 C

Stock View

Code: Search

Auto Scheduling Agent

<<June >> <<2004 >>

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1 Set	2 Set	3 	4 Set	5
6 	7 	8 	9 	10 	11 	12 Set
13 Set	14 Set	15 	16 Set	17 Set	18 	19
20 	21 Set	22 Set	23 	24 Set	25 	26 Set
27 Set	28 Set	29 Set	30 Set			

Target User: tyasu@140.113.88.245

News

Headlines:

- Army Punishes 7 With Reprimands for Prison Abuse
- White House Rejects the Request for Statement on Palestinians
- China Races to Reverse Falling Grain Production

Sport:

- It is a Start, but the Yanks Need More
- Do not Bet Against Bonds in Run at .400

Epapers

- VOGUE時尚電子報
- Yahoo!奇摩影視新聞電子報
- Yahoo!奇摩健康新聞電子報
- SmartNet智富網投資電子報
- 哈日酷電子報
- TaipeiWalker
- Yahoo!奇摩健康新聞電子報
- GO男人電子報
- Easy企管報
- 皇冠娛樂電子報

Mailbox

You have 2 new e-mail

- NYTimes.com: Headlines
- info4@enews.sony.com.tw: New Arrivals?

Bookmarks

My Bookmarks New

- Yahoo! kimo
- eBay
- Google
- gva.sun
- New York Times

(d): The Fragment has arrived at the page of user opb

Welcomel opb

Search Go!

Weather

Taipei 16~19 C

Taoyuan 16~19 C

Hsinchu 16~19 C

Taichung 14~18 C

Tainan 15~19 C

Kaohsiung 16~20 C

Hwalian 16~19 C

Taitung 18~21 C

Stock View

Code: Search

Auto Scheduling Agent

<<June >> <<2004 >>

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1 Set	2 Set	3 	4 Set	5
6 	7 	8 	9 	10 	11 	12 Set
13 Set	14 Set	15 	16 DATE 	17 Set	18 	19
20 	21 Set	22 Set	23 	24 Set	25 	26 Set
27 Set	28 Set	29 DATE 	30 Set			

Target User: tyasu@140.113.88.245

News

Headlines:

- Army Punishes 7 With Reprimands for Prison Abuse
- White House Rejects the Request for Statement on Palestinians
- China Races to Reverse Falling Grain Production

Sport:

- It is a Start, but the Yanks Need More
- Do not Bet Against Bonds in Run at .400

Epapers

- VOGUE時尚電子報
- Yahoo!奇摩影視新聞電子報
- Yahoo!奇摩健康新聞電子報
- SmartNet智富網投資電子報
- 哈日酷電子報
- TaipeiWalker
- Yahoo!奇摩健康新聞電子報
- GO男人電子報
- Easy企管報
- 皇冠娛樂電子報

Mailbox

You have 2 new e-mail

- NYTimes.com: Headlines
- info4@enews.sony.com.tw: New Arrivals?

Bookmarks

My Bookmarks New

- Yahoo! kimo
- eBay
- Google
- gva.sun
- New York Times

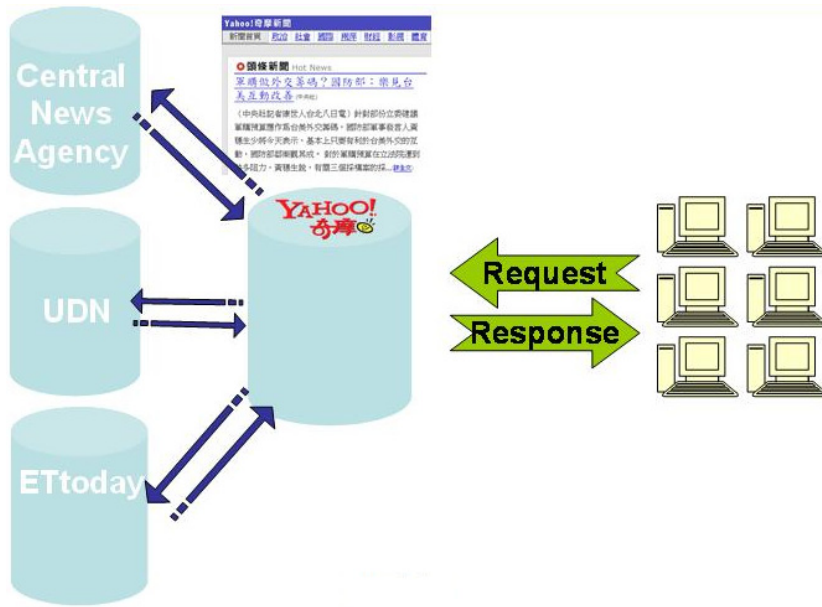
(e): User opb selects two days for the date

Figure 3-2: Fragment Migration Example

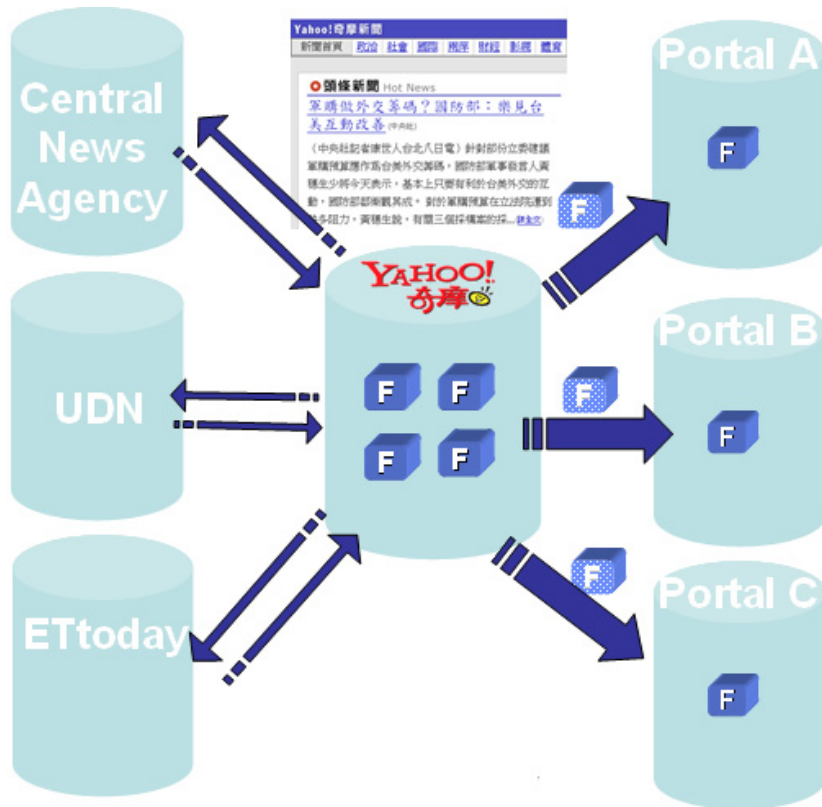
This situation may also happen in enterprise management when using EIP and EAP. For example, investors and managers of the company use individual accounts to login the EIP or EAP for accessing and maintaining personal data. When a manager has to negotiate with an investor or employer, he can send a Fragment which carries only the necessary information and data. Thus, the investor or employer will only see the related information that the manager wants to display.

III. Fragment migration also helps traffic distribution. Figure 3-3 shows a good example for this. Yahoo News aggregates all kinds of News from many News sources to help users read news conveniently. Thus Yahoo becomes the traffic bottleneck when users read news on Yahoo, as illustrated in Figure 3-3 (a). Everyone reads news from Yahoo News though these kinds of news are all from other news providers.

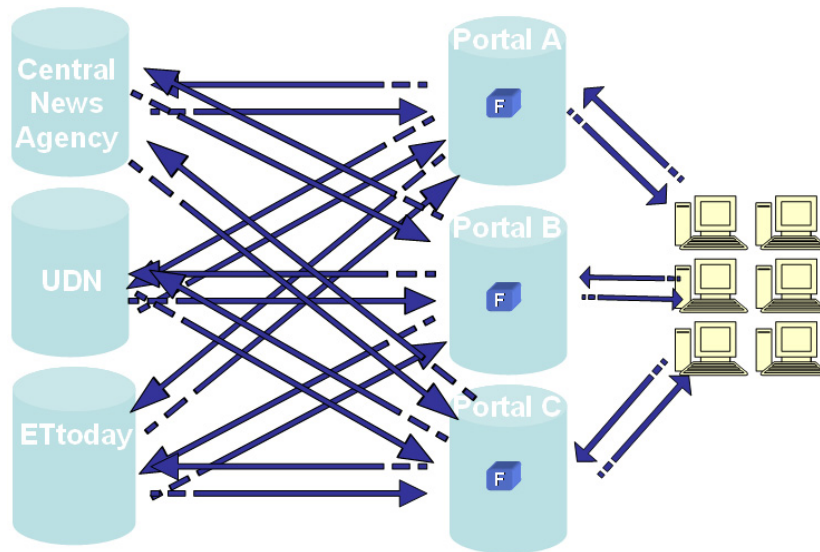
If Yahoo is established upon MVP framework, the Yahoo News Service can be distributed to other collaborative MVP-based Portals, as Figure 3-3 (b). In this way, all of these portals can serve users with Yahoo News, as depicted in Figure 3-3 (c). Therefore, users who want to access Yahoo News service can choose to connect to other collaborative portals that also have Yahoo News Fragment. And then the network traffic is thus distributed.



(a): Traffic bottleneck on Yahoo News



(b): Distribute News Fragment to several portals



(c): Network Traffic after News Fragment distribution

Figure 3-3: Traffic Distribution Example

2. A Fragment travels around many hosts on the network for the sake of automatically accomplishing the assignments. Figure 3-4 indicates that a Fragment on portal A have to travel among four portals. After `Itinerary` and `ActionStrategy` of the Fragment being configured, the Fragment can decide where to go and what to do by its own.

Use case:

- I. Fragments can be utilized to conduct tasks that ought to follow some determined rules and procedures. For example, a user may need to send a Fragment with official documents that need several staffs or officers to sign in a certain order. First, the user has to configure the `Itinerary` for this task, and set the `ActionStrategy` to handle some predicable situations. In this case, the Fragment must deliver the documents and make sure the signature order. Some of the officers may not be satisfied with the content of documents and reject them. Thus the Fragment is supposed to go back to the former stop and help the staffs there to modify them. After the

configuration of *Itinerary* and *ActionStrategy*, the *Fragment* can travel among portals, communicate with the appointees, and do the assignments autonomously.

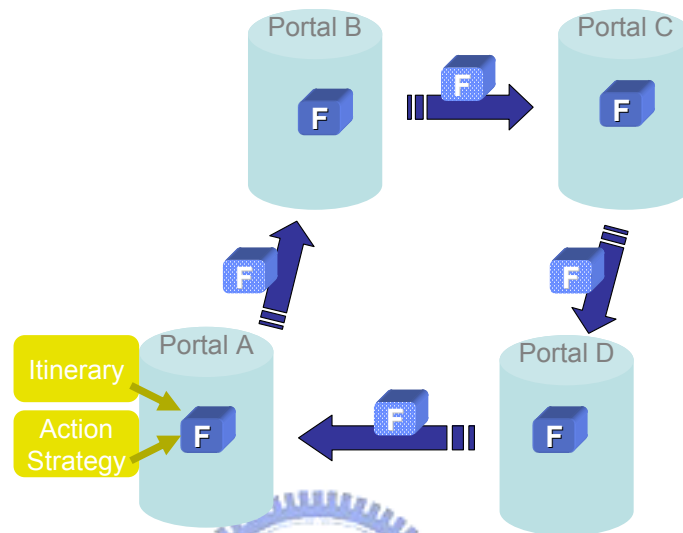


Figure 3-4: Fragment Traveling according to the Itinerary and Action Strategy

- II. On the other hand, *Fragment* may need tremendous data and information distributed in many different portals. Under such circumstances, the *Fragment* with the computational logics can be sent to travel the portals. The only thing that migrates from one portal to another is the code for computation in the *Fragment*. This using strategy will prevent network from enormous data traffic.
3. Collaboration among *Fragments* is required when we divide a complicated task into several much simpler ones. In this condition, it is necessary for *Fragments* to negotiate and communicate with each other. MVP framework provides an infrastructure much similar to the one of telephony. Wherever a *Fragment* is, it can invoke the message transfer service in the local host, which is

just like to give a phone call. Then the messaging infrastructure will try to find the receiver according to the provided Fragment address.

Figure 3-5 depicts that how the messaging infrastructure facilitates the communication among Fragments. Suppose that a Fragment F is on the portal C. When F is going to communicate with Fragment A which was born on portal A, F will invoke the message transfer service on the portal C. Message transfer service will first ask portal A for the current location of Fragment A. After portal A tells portal C where Fragment A is, say portal B, the message transfer service of portal C will send the message to portal B for Fragment A.

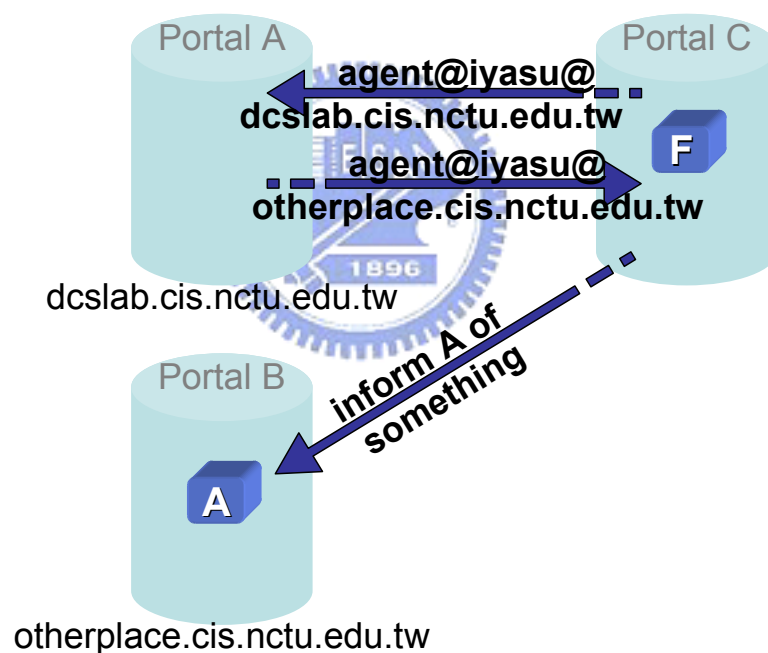


Figure 3-5: Fragment Locating and Messaging

Use case:

- I. In addition to collaboration among several Fragments, this messaging infrastructure can also be applied for other purposes. Suppose there are two users who are behind two firewalls individually. They are allowed to use the

network only via port 80. Thus they can use HTTP to control two remote representative Fragments to communicate with each other, as shown in Figure 3-6.

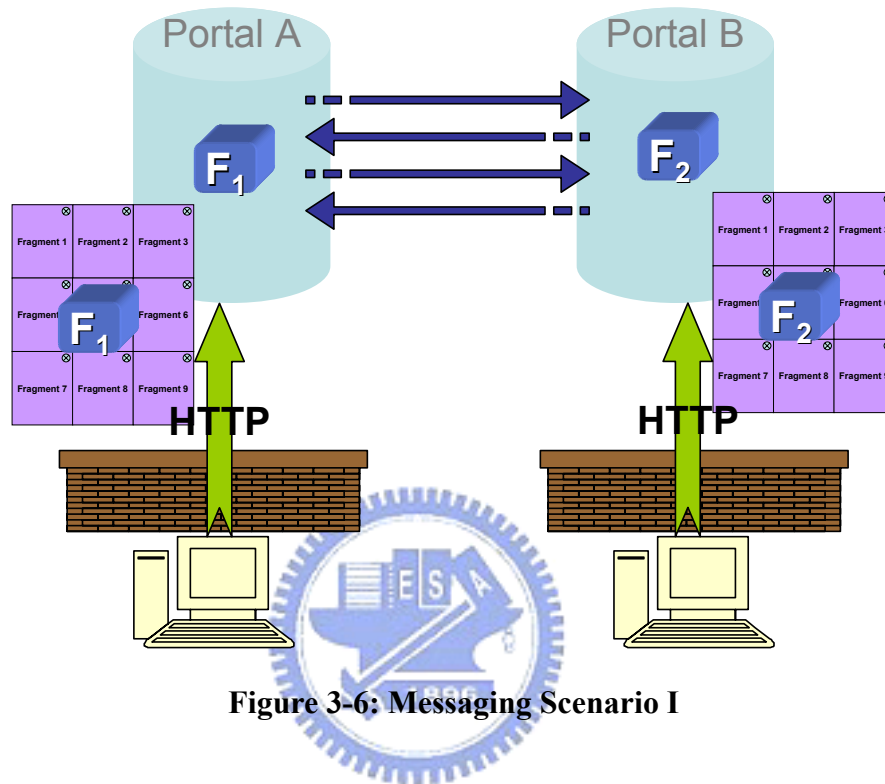
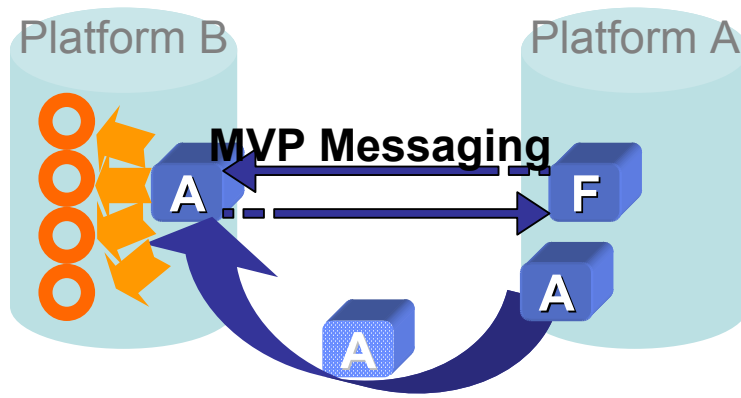


Figure 3-6: Messaging Scenario I

II. Besides, messaging infrastructure also facilitates communication among heterogeneous platforms. In spite of proprietary protocol on each independent platform, platform A can send a Fragment to platform B, and then the Fragment will be remotely controlled by platform A to invoke necessary processes or services. Figure 3-7 illustrates such a condition.



○: Processes or services on Platform B

➡: Invoke processes or services

Figure 3-7: Messaging Scenario II




Chapter 4

Development based on MVP

To develop a portal on MVP framework, there are two things to do. First, write a portal application that is directly access by the end user. Second, develop Fragments which will be employed in the written portal application. In this chapter we will explain that how to develop a portal application and Fragments using MVP framework.

4.1 Development of Portal Application



It is very simple to develop a portal application based on MVP framework. As mentioned above, there is a Java Servlet Container embedded in the MVP framework. Programmers thus can use Java Servlet to write their own portal application according their own demands. Besides, programmers need to design their own patterns to arrange the Fragments' appearances in a user-friendly way, or use a default pattern offered by MVP framework: three-column arrangement. Each of the Fragments has an attribute to indicate the position it should appear at, and thus the arrangement patterns can make use of this attribute to make up appropriate appearance of a portal page. Table 4-1 shows a usual program template to form a portal application. As depicted in the table, `myPortal` is the sample portal application. Similar to common portal applications, `myPortal` extends `HttpServlet` class from a standard Java Servlet package. Then the programmers may write the code in the `doPost()` or `doGet()` method to arrange Fragments' appearances properly and compose their portal pages.

```

public class myPortal extends HttpServlet{
public void doPost (.....) {
    //code for determining fragments that constitutes a page
    fragments = FragmentManager.loadfragments();
    fragments.someAgent.
    action(request);
    .....
    //Other user-dependant code
    .....
    this.arrange (fragments);
}
private void arrange (ArrayList fragments) {
    //get UI generated by render() of fragments
    //arrange fragments in certain order
    ThreeColumnPattern.arrange (fragments);
    ThreeColumnPattern.display ();
}
}

```

Table 4-1

First, the portal application may request Fragment Manager to load required Fragments from Fragment Container. After loading Fragments, Fragment Manager returns the references of these Fragments to the portal application. Receiving fragment references, the portal application should hand the user request to the designated Fragment. Afterwards, the portal application calls the `arrange()` method to get UIs generated by all Fragments (UIs can be HTML code segments stored in String type, or even Java Applet), and arrange them in default, user-defined, or device-dedicated patterns. In this case, `arrange()` method exploits `ThreeColumnPattern` class which is provided by MVP framework as a default arrangement pattern. The instance of this class will first classify the Fragment contents into three columns. Thereafter, it will display these Fragment in the

designated order to form a whole page.

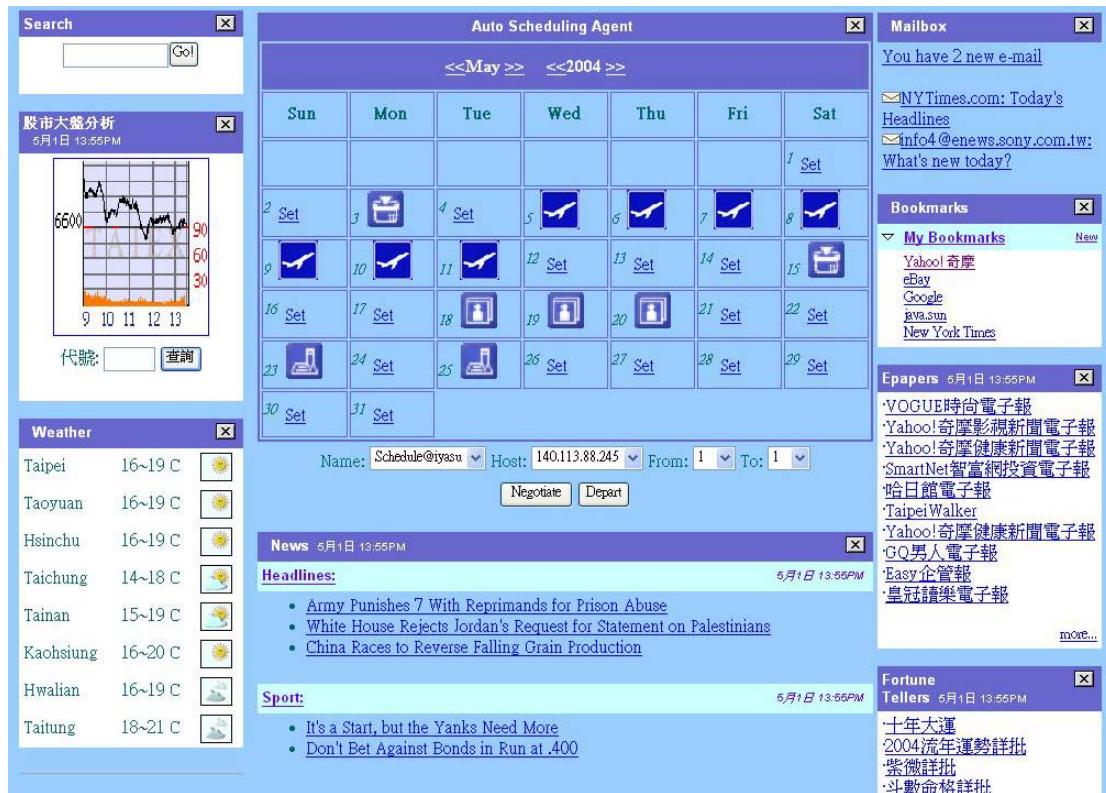


Figure 4-1: MVP Sample Portal View on a Desktop Browser



Figure 4-2: MVP Sample Portal View on Mobile Devices

Figure 4-1 and Figure 4-2 give a sample view of a MVP portal application. The portal appearance in Figure 4-1 shows a three-column pattern of display. Fragments appear in the page as HTML tables, and each of them has an appropriate region in the whole page. With the arrival and departure of Fragments or user preferences, the arrangement of the page can be flexibly changed. In Figure 4-2, the identical content of the portal is displayed on a mobile device. We can see that when the portal application detects that the requesting client is a mobile device, it can rearrange the portal content in a different pattern. Consequently, the mobile device users will see one Fragment at a time, and then click to choose the other fragment if they want the information displayed on other fragments. The left one of Figure 4-2 shows Fragment about the information of the stock market, while the right one shows the Fragment of auto scheduling agent.



MVP framework provides a Java interface `Arrangement` to be implemented by other subclasses. The default arrangement `ThreeColumnPattern` also implements this interface. Therefore, if programmers want to design their own arrangement pattern, they can write their own arrangement class. The relationship among these classes is depicted in Figure 4-3. `Arrangement` interface mainly consists of two methods: `arrange()` and `display()`. The subclasses should implement these two methods for arrangement. For example, the `arrange()` method of `ThreeColumnPattern` classifies all `Fragment` instances into three groups, while the `display()` method displays all these `Fragments` in certain order to generate the portal page.

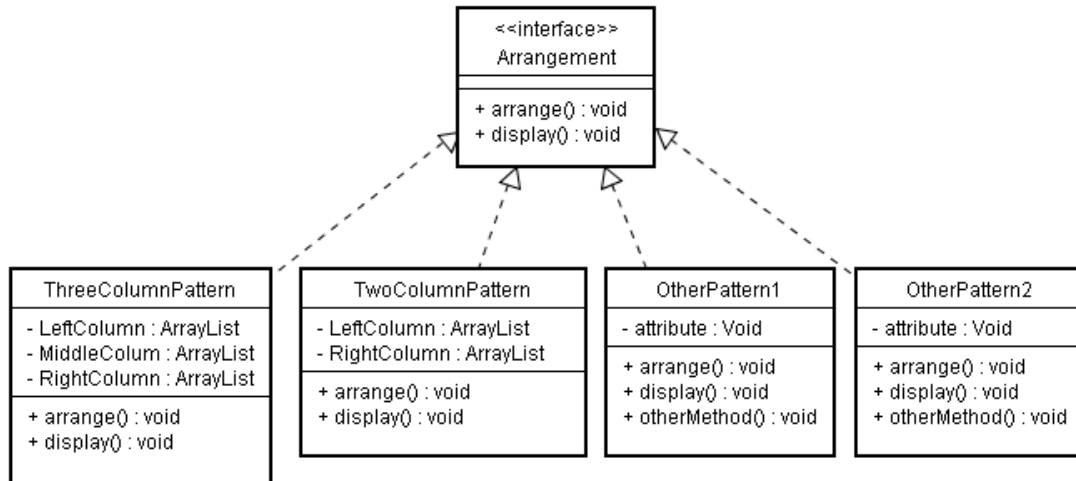


Figure 4-3: Class diagram for Arrangement

4.2 Development of Fragments

As the former section said, what portal application does is only load Fragment instances and aggregate content of them. The main interaction between a portal and users is handled by each of Fragment instances. In this section, we will explain how to develop Fragments on MVP framework.

It is also quite simple to develop Fragments on MVP framework. In MVP framework, there is an abstract class Fragment. It leaves four methods for programmers to implement: `action()`, `fragmentLogic()`, `init()`, and `render()`. As Table 4-2, Programmers should write a class that extends the abstract class Fragment, and then fill the four methods with the necessary code respectively. The functionalities of the methods are described as follows:

```

public class myFragment extends Fragment{
    public void action(.....){
        //request handler
    }
    private void fragmentLogic(){
        //Things that default thread will do
    }

    private void init(){
        //do initialization for this fragment
    }

    private void render(){
        //generate the user interface
    }
}

```

Table 4-2

1. action()

The user requests are going to be passed to `action()` method, so the intention of this method is to handle the incoming request from users. For example, the user request might contain parameters to fill in some form, to suspend a Fragment or ask it to leave. The programmer should write the corresponding code in this method to realize the functionalities that he allows users to invoke.

2. fragmentLogic()

As remarked above, a Fragment is always assigned to complete some tasks, so it often has a default thread to do the designated jobs. The `fragmentLogic()` method is designed for the default thread. Programmers should fill in this method with the code that will do the jobs this Fragment is born for. Besides, a Fragment certainly can be a common web component that cannot act as a mobile agent. In

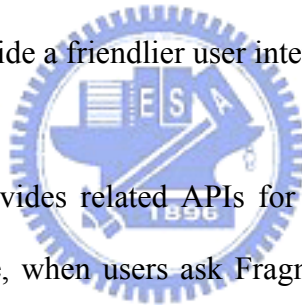
this case, the `fragmentLogic()` method can be leave to be blank.

3. `init()`

This method does the initialization that need only to be done once. For instance, the programmer can set the name, traveling route, or other configuration in this method.

4. `render()`

Programmers should write the code to generate the appearance of this Fragment. Basically, it is appropriate to show a Fragment with a HTML table for the ease to aggregate Fragments on a single page. It is fine to display a Fragment in whatever way as the programmers wish as long as there is corresponding design in the portal application. For example, programmers can also embed a Java applet into a HTML table to provide a friendlier user interface.



MVP framework also provides related APIs for Fragment developers to access Fragment Manager. Therefore, when users ask Fragments on a portal to perform as they suppose, Fragment Manager can facilitate Fragments to accomplish the tasks. For example, as mentioned in the description of `action()` method, if Fragments is assigned to leave or send message to other Fragment, the Fragments have to invoke the provided APIs for related services of Fragment Manager. These APIs are as follows:

1. `agentSpawn()`

As its name reveals, `agentSpawn()` method is used to create a new active Fragment instances. The code in this method takes the name and the path of the class of the Fragment to load the class and spawn a thread for the Fragment instance. Then the Fragment instance will be stored into Fragment Container. In this way, the Fragment instance can do the designated jobs asynchronously in the

background. Besides, this method also creates a new entry of `HFTable` for this Fragment instance. It results that Fragment Manager can keep track of the latest location of the Fragment instance, which is helpful to the communication among all Fragment instances.

2. `agentDestroy()`

`agentDestroy()` is the complement method of `agentSpawn()`. When this method is invoked, it will first delete the corresponding entry of `HFTable`, then stop the thread for this Fragment instance and remove it from the Fragment Container.

3. `agentDepart()`

When the Fragment instance is assigned to leave for next stop according to the itinerary, it has to call `agentDepart()`. This method will establish a connection to the destination in accordance with the configuration of Fragment's itinerary. Afterwards, `agentDepart()` method will use Java Serialization mechanism to serialize the Fragment instance and its states to the next stop. After arriving at the destination, the Fragment instance will be deserialized and recovered to the latest state. Therefore, the Fragment instance can continue to execute what it has to do.

4. `inform()`

When a Fragment instance has to communicate with other Fragment instances, it is necessary to invoke `inform()` method offered by Fragment Manager. After invoked, this method will set up a network connection to the host that the target Fragment instance is situated at. Then the message will be sent to the host, and the Fragment Manager on that host will deliver it to the target Fragment instance. The message is sent in String type. Therefore, the Fragment developer can make up unique message format such as XML message for their own Fragments.

Chapter 5

Related Work

MVP integrates two popular technologies in different domains to construct a whole new framework for building portals. In the following paragraphs, we will expound the related researches and development of these two domains: portals and mobile agents, and compare them with the design of MVP respectively.

5.1 Portal

Nowadays the number of Internet users is getting larger and larger, and each of them has his own desired services. For example, housewives might prefer shopping information while teenagers might prefer sports news and traveling information. As a result, some of the major portal vendors are gradually changing their authorizing strategies to satisfy the requirement of personalization of portals. The most well-known way to manage personalization issue is employing an idea of web components to make up a page. These kinds of web components are generally called Portlets.

In spite of the identical opinion and idea, portal vendors develop their respective Portlets. There is no open standard for these kinds of web components until Java Portlet Specification 1.0 was released from Java Community Process (JCP) on October 2003. But neither the standardized Java Portlet nor other kinds of vendor-specific Portlet design solves the problem thoroughly: they ignore the users of mobile devices.

Traditional portals always use various techniques which cannot be applied to mobile devices to fulfill many fancy styles and appearances. The users can only see the simplified versions or segmented pages of these portals on their mobile devices. When it comes to the simplified versions, users often hang back for fear of the bad look-and-feel. In addition, they need to make themselves be accustomed to another user interface different from the original version. As for researches about segmenting web pages [10] [11] [12], the result is always poor because there is no common styles and strategies for page arrangement. On the other hand, though the portlet-based pages are suitable to rearranging for mobile devices, the portlet-based systems or frameworks other than MVP do not think over this issue (see Figure 1-2 in chapter 1). MVP framework considers a page a set of Fragments, and it can rearrange the page according to the Fragment-based structure. Therefore the display for mobile devices is easy to be generated. Users will see that one Fragment is shown at a time, and they can choose whatever Fragment they want. Users can also configure that which Fragments are needed to be downloaded to mobile devices in advance.

Additionally, when using traditional portals, users will be bound to certain portals which offer certain services. They have to browse portal A for some services, and then get to portal B for others. MVP framework can aggregate all kinds of Fragments based on MVP from any portals throughout Internet by means of migrating the wanted Fragments to a single portal. In this way, users don't have to be on the run browsing numbers of portals any longer.

Portlet-based portals also leave two problems:

1. Portlets that follows Java Portlet Specification may adopt Web Services for Remote Portlets (WSRP [13]) to communicate with remote portlets. But in this way, portals still cannot acquire remote portlets to execute them locally. Thus, if the remote portals or their network connections failed, the way WSRP provide to communicate with remote portlets will also fail.
2. Portlet-based portals generate pages consisting of several portlets. If one or several of the portlets fail to execute or consume too much time, the performance of generating a page will get worse because it has to wait for all portlets which forms the page.

Compared with traditional portlet-based portals, MVP framework can rearrange pages for mobile devices and gather fragments from all MVP-based portals. Moreover, all Fragments in a page can be designed as smart as possible, and then they can negotiate with each other automatically for performance issues.

5.2 Mobile Agent

MVP also proposes a good way to manage and use mobile agents. Other researches about user interfaces for managing mobile agents can be roughly divided into two parts:

1. Developing standalone and dedicated applications to manage agents [14]. However, such kind of applications need to be installed in some specific procedures, and are usually bound to certain platform. Thus, it is inflexible to use these dedicated applications.

2. Adopting web-based user interfaces to manage mobile agents for users' conveniences [15]. For example, a fragment of HTML code or a Java Applet is mapped to a mobile agent to control it [16]. Users can use any kinds of client devices that have a browser to manage and control mobile agents. But it is still not flexible enough for users to access mobile agents when they want to control agents that are from other hosts. They cannot share their agents with others and make use of others' agents.

Using MVP framework, users can easily employ others' agents, as known as Fragments, and manage them in an efficient way. Mobile agents that travel around the network are seen as ships and then anchored to a portal based on MVP framework. Agents will be automatically arranged to be displayed on the portal pages, and the users who use desktop computers or mobile devices can access the agents through the portal.



Chapter 6

Conclusion and Future Work

MVP framework introduces several new ideas for both web technology and mobile agents. As for web technology, MVP framework is able to provide more versatile services that are not limited by HTTP response time. Portal users can use intact HTTP protocol to assign Fragments to fulfill kinds of asynchronous tasks autonomously and intellectually. MVP framework can also gather all necessary services for users to facilitate management and personalization. And in the aspect of mobile agents, MVP framework provides a much easier way to use and manage them. Users can centralize all agents they need with common web browsers into a single portal, and manipulate these agents simply in a very convenient way.

There are still many things to do with MVP framework in the future. They are:

1. Provide offline usage of MVP for mobile device users:

Mobile device users always have unreliable network connection for the sake of mobility. Under some circumstances, for example, before entering an environment without wireless connection, they know that the connection will be broken. If MVP framework offers such a service so that the users can in advance download the necessary Fragments with the computational logics to their mobile devices. Then they can do some negotiation or configuration with the Fragments even they are surrounded by a disconnected environment. After the connection is resumed, all things that have been done before will be automatically synchronized with the portal.

2. Lookup for Fragments according to their functionalities:

The present MVP framework can only lookup Fragments in accordance with the complete name of them. That is, the user who wants to lookup some Fragment must give the complete name such as “`schecule@iyasu@dcslab.cis.nctu.edu.tw`”, and then MVP framework will return the current location of the wanted Fragment. It may be more convenient for users if MVP framework can lookup fragments according to certain given attribute or purpose. Users or Fragments thus can find any other adequate and competent Fragments with which they want to collaborate.

3. Locate Fragments in a more effective and efficient way:

In the current design of MVP framework, Fragments have to register to their birthplaces. Whenever leaving some host for another, Fragments report their current location to the birthplaces for keeping track of themselves. Afterwards, queries are sent to the birthplace, and the latest location of any Fragment born in that place will be returned. This scenario might be inefficient if the Fragments frequently travel among several hosts. Improved scenarios may need to be proposed. For example, Fragments can be seen as mobile phone or mobile device users and the portals can thus be regarded as base stations or access points, and then MVP framework can adapt the location-aware scenarios of mobile telephony or wireless technology to locate Fragments.

4. Security issue:

As mentioned in chapter 3, Fragments facilitate communication among heterogeneous platforms regardless of proprietary protocols. Fragments can migrate from one platform to another and then execute certain tasks or invoke certain services on the target platform. Under such circumstances, there are

potential crises that malicious Fragments may emerge to do harm to those platforms who accept these Fragments. Therefore, the security issue will be very important after the MVP framework is getting more and more popular. Designs for security should be added into MVP framework in the future.

5. Although portals based on MVP framework have Fragment-based structures and is quite suitable for mobile device display, there are still many portals that are authored in traditional ways. If someone wants to apply MVP framework to the existing traditional web portals, the only way at the present time to achieve the goal is re-authoring the current portals to conform to the specification of MVP framework. It is really time-consuming and tiring work. A façade gateway needs to be developed to solve this problem. We can see the scenario in Figure 6-1. The façade gateway is a web portal based on MVP framework. Fragments of the facade gateway can be automatically mapped to services on the traditional web portals. If the Fragment-based appearance is needed, users can directly access the façade gateway whether using normal desktop computers or mobile devices. Moreover, Fragments can not only be mapped to services on traditional web portal but at the same time do other jobs such as improving and integrating the functionality of the correspondent services. In this way, the traditional portal can also make use of MVP framework.

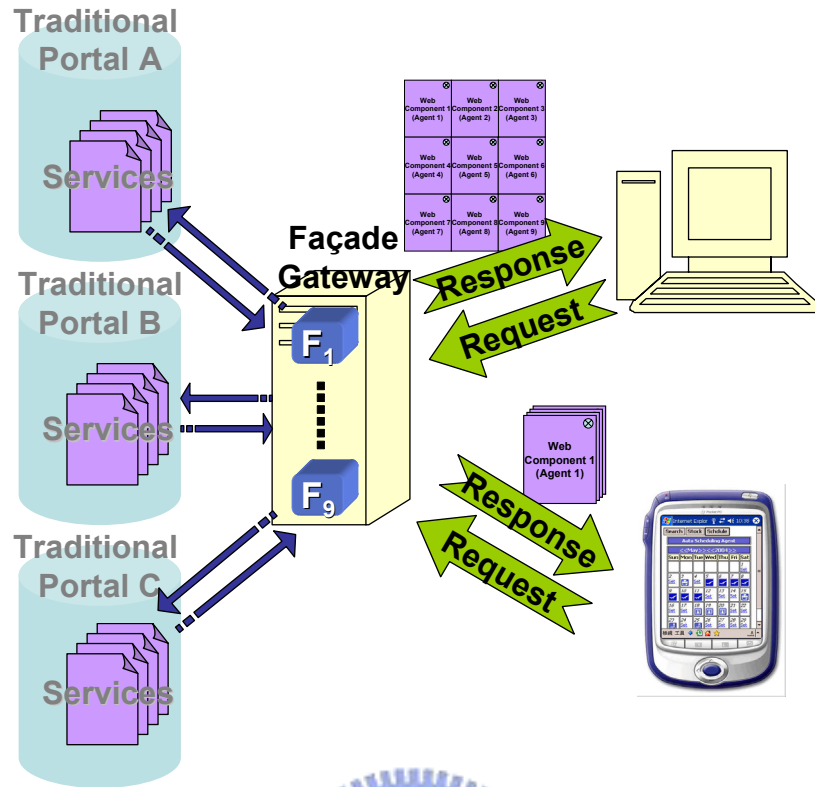


Figure 6-1: Façade gateway scenario



References

- [1] **PHP**, <http://www.php.net>
- [2] Mark Roth, Eduardo Pelegri-Llopart, “**JavaServer Pages™ Specification Version 2.0**”, Java Specification Requests – 152, November 24, 2003
- [3] Danny Coward, Yutaka Yoshida, “**Java™ Servlet Specification Version 2.4**”, Java Specification Requests – 154, November 24th, 2003
- [4] Sun Microsystems, **Java**, <http://java.sun.com>
- [5] Alejandro Abdelnur, Stefan Hepper, “**Java™ Portlet Specification Version 1.0**”, Java Specification Requests – 168, October 7, 2003
- [6] Stefan Hepper, Stephan Hesmer, “**Introducing the Portlet Specification**”, <http://www.javaworld.com/javaworld/jw-08-2003/jw-0801-portlet.html>, August 1, 2003, <http://www.javaworld.com/javaworld/jw-09-2003/jw-0905-portlet2.html>, September 5, 2003
- [7] David Wong, Noemi Paciorek, Dana Moore, “**Java-based mobile agents**”, Communications of the ACM, Volume 42 , Issue 3: 1999, pp. 92-102.
- [8] Danny B. Lange, Mitsuru Oshima, “**Seven Good Reasons for Mobile Agents**”, Communications of the ACM, Volume 42 , No. 3: 1999, pp. 88-89.
- [9] Danny B. Lange, Mitsuru Oshima, “**Programming and Deploying Java™ Mobile Agents with Aglets™**”, Addison-Wesley Pub Co, September 15, 1998
- [10] Yonghyun Hwang, Eunkyung Seo and Jihong Kim, “**WebAlchemist: A Structure-Aware Web Transcoding System for Mobile Devices**”, Proc. Mobile Search Workshop, Honolulu, Hawaii, May 2002
- [11] Jinlin Chen, Baoyao Zhou, Jin Shi, HongJiang Zhang, Qiu Fengwu, “**Function-based object model towards website adaptation**”, WWW 2001:

- [12] Joseph W. Sullivan, T. Bickmore, A. Girgensohn, “**Web Page Filtering and Re-Authoring for Mobile Users**” , The Computer Journal - Special Issue on Mobile Computing, Volume 42, Issue 6: 1999
- [13] Alan Kropp, Carsten Leue, Rich Thompson, “**Web Services for Remote Portlets Specification Version 1.0**”, August 2003
- [14] Alberto Silva, , M. Mira da Silva and Artur Romão, “**User Interfaces with Java Mobile Agents: The AgentSpace Case Study**”, Proceedings of the First International Symposium on Agent Systems and Applications (ASA '99), Palm Springs, California. IEEE Computer Society, October, 1999.
- [15] Anselm Lingnau, Oswald Drobnik, “**Agent-User Communications: Request, Results, Interaction**”, In Proceedings of Second International Workshop on Mobile Agents, MA'98, Stuttgart, Germany, 1998.
- [16] A. Rodrigues da Silva, M. Mira da Silva, Artur Romão, “**Web-based Agent Applications: User Interfaces and Mobile Agents**”, Proceedings of the IS&N'2000 Conference (Athens, Greece, February 23-25 2000), Lecture Notes in Computer Science, Vol. 1774, Springer 2000.