Master Thesis in
Computer Science

Developing WAP services with Allaire's ColdFusion

By

Tuong Huynh

The Royal Institute of Technology
Kungliga Tekniska Högskolan

Final Report

Examiner: Prof. Seif Haridi
Department of Teleinformatics
The Royal Institute of Technology

Supervisor: Thomas Sjöland
Department of Teleinformatics
The Royal Institute of Technology

Supervisor: Mark Tierney
Research and Development
room33 AB

Abstract

The Wireless Application Protocol is designed first and foremost for resource-constrained devices. The programming model is similar to that used for web applications, but physically and logically, WAP-enabled devices bear little resemblance to their distant desktop/notebook relatives. The differences between the WAP programming model and the web model on which it is based translates into two key differences between web and WAP applications. First, the user agents requesting documents are likely to be less homogeneous than those visiting a web page. These days there are a much greater difference between the interactions of a digital pager and a PDA than there is between Internet Explorer and Netscape Navigator. Second, the language used to program the interaction is different. The text document delivered by the content server will be a Wireless Markup Language (WML) document, not a Hypertext Markup Language (HTML).
Because WML is designed for resource-constrained devices, there are limits to what one can accomplish without language extensions.

The competing I-mode has gained tremendous success in the Japanese market. Unlike WAP, it uses compact HTML and the devices are often equipped with color screens. It was originally designed to attract a young audience, especially teenage girls, while WAP was mainly targeted to the business market.

Allaire's ColdFusion Application Server has gained tremendous success, especially within the area of e-commerce sites. The ColdFusion Markup Language (CFML) creates a complete environment for building page-based applications, its library of more than 200 functions and over 70 tags promise developers rapid application development. The most important issue when building sites that need to keep users separated, like online stores or online services where users can customize their preferences, is the ability to keep track of each users session and hold them separately. WML itself doesn't provides the require session handling, which makes the development of e-commerce services impossible or hard to implement for WAP-enable devices. CFML does provide the option for session handling and support WML pages.

Table of Contents

1 Introduction

1.1 Project description

1.1.1 Background

This project was carried out at room33 AB, where the goal was to develop an application for WAP enabled mobile phones and for the World Wide Web as well. The wireless application protocol and the scripting language ColdFusion were also the objects within this study.

1.1.2 Goals

room33 AB develops the main part of the portal room33.com with Lutris Enhydra. One of the key-initiatives was to investigate how well services developed on a different platform will fit into the existent portal.

Primary goals for this project

- Studies of WAP and ColdFusion
- Develop an online community service with CFML and WML
- Integration between Lutris Enhydra and ColdFusion

1.2 Report outline

This report is organized as follow, chapter 1-4 give a general theoretical background for the project area, chapter 4 covers the existent alternative technologies, chapter 5-9 describes the implementation and integration and chapter 10 summarizes the work.

Theoretical background (1-4)

Chapter 2 introduces the Wireless Application Protocol and describes the general architecture. The different elements of the Wireless Application Environment are described.

Chapter 3 is an introduction to Allaire's ColdFusion Application Server and the development environment.

Chapter 4 discusses the existing alternatives like I-mode and Active Server Pages.

Implementation (5-9)

Chapter 5 describes the service Café33, the online community service developed within the scope of this project.

Chapter 6 introduces the reader to the room33 AB:s core platform, Lutris Enhydra Application Server.

Chapter 7 deals with the integration of ColdFusion and Enhydra.

Chapter 8 described in brief the translation intelligence developed for Café33.

Chapter 9 covers the experiences learned while setting up the entire development environment. Installation of the ColdFusion server on both Microsoft Windows NT 4 Server and SUN Microsystems Solaris. Configurations of the CF server and Apache web server 1.3.x on those operation systems.

Chapter 10 summarizes the work and suggestions for future work.

2 Wireless Application Protocol (WAP)

2.1 Background

In 1995, Ericsson initiated a project to bring a general protocol for value-added-services (VAS) on mobile networks. The outcome of that was Intelligent Terminal Transfer Protocol (ITTP). During this time, Unwired Planet (now openwave.com) and Nokia, presented additional work on this area, Unwired Planet with their Handheld Device Markup Language (HDML) and Handheld Device Transport Protocol (HDTP). HDML is optimized for mobile devices with limited screen area and input facilities

In June 26 1997, Ericsson, Nokia, Motorola and Unwired Planet began to work for a common standard and their efforts resulted in the foundation of WAP Forum in December 1997.

The WAP Forum's main objectives are:

- Independent of wireless network standard
- Open to all
- Will be proposed to the appropriate standard body
- Applications scale across transport options
- Applications scale across device types
- Extensible over time to new networks and transports

WAP uses Internet standards such as XML, user datagrams protocol (UDP), and IP, but have been optimized for the unique constraints of the wireless environment:

- low bandwidth.
- high latency.
- less connection stability (calls may drop).
- less predictable availability (network congestion).

Mass-market handheld wireless devices present a constraining computing environment:

- less powerful CPUs.
- less ROM and RAM.
- limited power supplies.
- smaller displays.
- restricted input devices.

The existing Internet standards today (HTML, HTTP, TCP) are inefficient over mobile networks, requiring large amounts of mainly text-based data to be sent. Standard HTML content cannot be effectively displayed on the small screens of mobile phones and PDAs. WAP uses binary transmission for higher compression rate of data and is designed for long latency and low bandwidth. WAP sessions cope with intermittent coverage and can operate over a wide variety of wireless transports.

The protocol stack is designed to minimize the required bandwidth and maximize the number of wireless network types that can deliver WAP content. Multiple networks will be targeted,

global system for mobile communications (GSM) 900, 1800 and 1900 Mhz, digital European cordless communication (DECT), time-division multiple access (TDMA) and code division multiple access (CDMA). All network technologies and bearers will also be supported, e. g. short message service (SMS), circuit-switched cellular data (CSD) and general packet radio services (GPRS).

2.2 Wireless Application Protocol Architecture

The WAP protocol suite contains four protocols that are responsible for the communication between clients and WAP Gateways. As the protocols used on the Internet, these can be used in four different configurations:

- Connectionless mode – only WSP and WDP are involved, no acknowledgements on the sent datagrams.
- Connectionless mode with security- additional encryption by WTLS.
- Connection mode – additional use of WTP, now the datagram being sent must be acknowledged and may be retransmitted if lost, which gives reliable transmissions.
- Connection mode with security – additional encryption by WTLS.

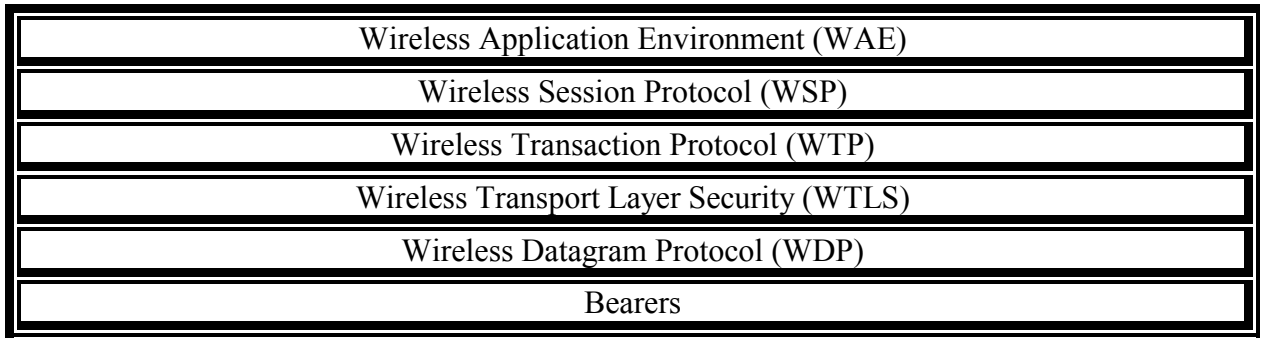


Diagram 2.2-1 WAP Protocol Stack

2.2.1 Wireless Application Environment (WAE)

The wireless application environment (WAE) provides a general-purpose environment for applications to be used on wireless devices. It connects the technologies used in the World Wide Web and mobile telephony to allow inoperability.

WAE contains of two logical layers, user agents and services/formats. Browsers, phonebooks and message editors are within the user agent layer while services and formats deal with common elements and formats accessible to user agents like WML, WMLScript and image formats.

2.2.2 Wireless Session Protocol (WSP)

The session protocol (WSP) layer provides a lightweight session layer to allow efficient exchange of data between WAE and the rest of the protocol stack. It provides negotiation capabilities, caching header and long-lived sessions. When running WSP in connection mode, it will set up a session between the client and the WAP Gateway. The session is assumed to be long-lived and can be resumed if suspended.

2.2.3 Wireless Transaction Protocol (WTP)

The wireless transaction protocol (WTP) runs on top of a datagram service such as User Datagram Protocol (UDP); part of the standard suite of TCP/IP protocols, to provide a simplified protocol suitable for low bandwidth mobile devices, reliable communication, controlling transmission and reception of message. It makes sure that messages are unique and retransmitted if they are lost. Messages are divided into three different classes.

- Unreliable send with no result message – no retransmission if the sent message is lost.
- Reliable send with no result message – retransmission if no acknowledgements are received.
- Reliable send with reliable result message – the sender acknowledges the acknowledgement.

WTP also supports Protocol Data Unit concatenation and delayed acknowledgement to help reduce the number of messages sent. This protocol therefore tries to optimize the user experience by providing the information that is needed when it is needed. It can be confusing to receive confirmation of delivery messages when you are expecting the information itself. By stringing several messages together, the end user may well be able to get a better feel more quickly for what information is being communicated.

2.2.4 Wireless Transport Layer Security (WTLS)

Wireless transport layer security (WTLS) is an optional security layer, which incorporates encryption facilities that provide the secure transport service for applications that must be secured during transactions between the client and the WAP gateway. WTLS can be used with both the connectionless and connection mode and is always placed on top of WDP.

2.2.5 Wireless Datagram Protocol (WDP)

The wireless datagram protocol (WDP) is the transport layer that sends and receives messages via any available bearer network. It is the base of the WAP protocol stack and hides the characteristics of different underlying bearers (GSM, SMS).

2.2.6 Wireless Markup Language (WML)

The wireless markup language (WML) is WAP's counter part to the hypertext markup language (HTML) on WWW. It is the page describing language used for authoring services and is designed to fit small handheld devices.

Like HTML, WML is also a tag-based language, but it has been designed for low-bandwidth wireless devices with limited input and output capabilities. WML documents navigate through a "card-and-deck" metaphor. A card is a single unit where information is shown to the user or the user can choose to input some data, and a deck is a related set of cards.
WML supports text and images, user inputs, navigation mechanism and variables.

2.2.7 WMLScript

WMLScripts can be used to enhance the functionality of a service, just as for example JavaScripts may be utilized in HTML. It makes it possible to add procedural logic and computational functions to WAP based services.

2.2.8 Wireless Telephony Application (WTA)

The wireless telephony application (WTA) framework defines a set of features to create telephony services.

2.3 WAP Gateway

The Internet is unprecedented in its impact on the world community of industries, institutions and individuals. Internet has affected the way we communicate and how we use our time. No media adoption curve has been faster than the Internet's. It took almost 40 years for 50 million people to use radio and 15 years for 50 million people to use TV and cellular communications in the USA. Internet users reached the 50-million mark in just 5 years.

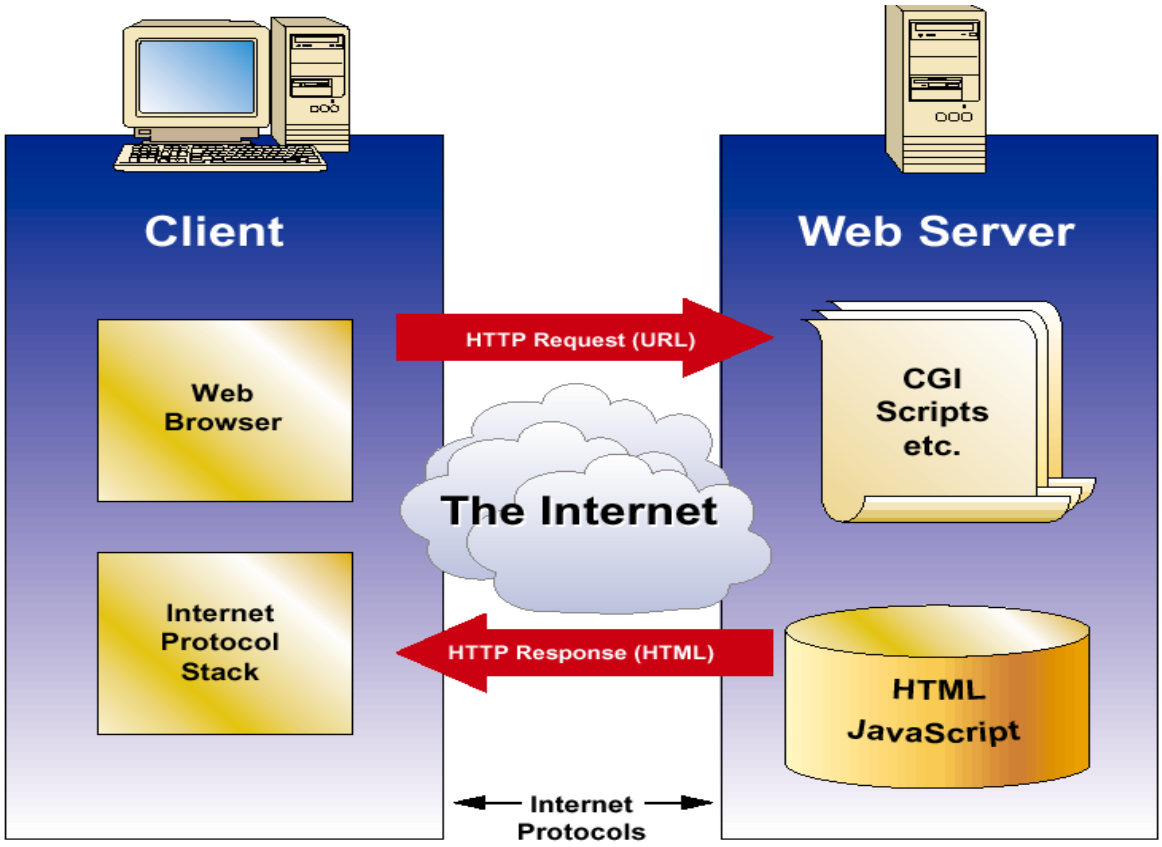


Figure 2.3-1 The Internet way.

All resources on the Internet's World Wide Web are named with Uniform Resource Locators (URLs). A client-side web browser sends HTTP requests for content to a web server by typing in the web address in the URL. The server then fetches the required contents and does the necessary processing, formatting and then sends the page back to the browser.

The WAP gateway can be seen as an access point to the Internet for WAP clients, typically mobile phones with display and software enhancements. Communication takes place between the WAP-compliant protocol set in the gateway and the TCP/IP protocol in the fixed network origin server.

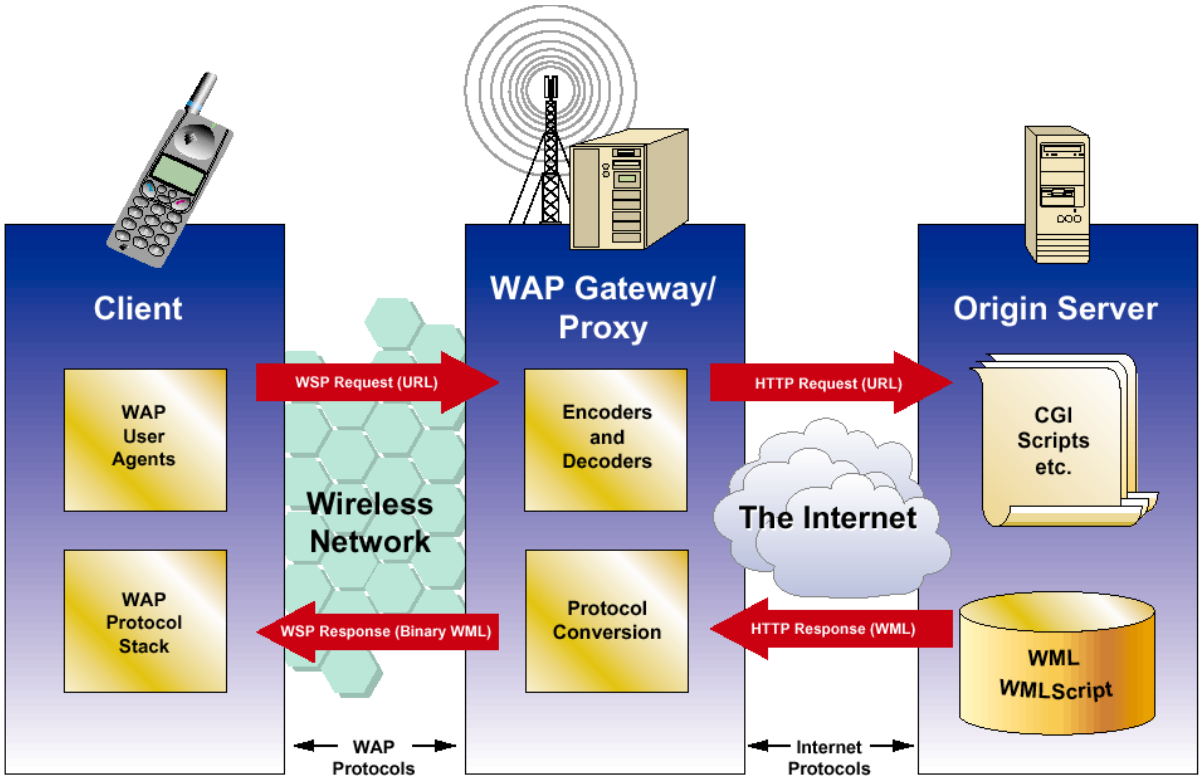


Figure 2.3-2 The WAP way.

The gateway simply acts as a data and protocol converter, providing a link between the mobile network and the Internet. This allows WAP-enabled mobile devices to request WAP services and information from World Wide Web servers. The mobile terminals make requests to Internet servers, which in turn, send WAP content to the WAP gateway. Then the gateway encodes WAP content into a compact binary form and forwards this content to the mobile device. The WAP-enabled mobile devices have their own WAP micro browser that displays interactive WAP contents to the user.

2.4 Data bearer

Although WAP is the key enabler, who marries the world of wireless telephony and the Internet, the speed and quality of the underlying data bearer will have an important impact on user perceptions. GSM networks generally provide a 9600 bits/s data bearer, but many operators do offer 14400 bits/s nowadays, which may be sufficient to provide a range of short message services, email and limited web browsing.

2.4.1 High Speed Circuit Switched Data (HSCSD)

It provides GSM users with a 57600 bits/s bearer (using four timeslots) but, in a world rapidly moving to IP networks, the fact that it is a circuit-switched technology limits its future-proofing. With HSCSD wireless users will get the speed they need for fast, robust connections

to data services. However, they will remain subject to time-related tariffs, rather than simply paying for data traffic.

2.4.2 General Packet Radio Services (GPRS)

GPRS is a packet-switched data bearer running at 57.6 kbits/s to 115.2 kbits/s, with up to 384 kbits/s envisaged, via channel aggregation. Although implementation of GPRS (2.5G) in mobile networks involves investments on additional hardware to current base stations, its two prime benefits are that it provides a highly functional bearer for current wireless data requirements and that it has a clear migration path to third generation (3G) mobile, via software upgrading.
GPRS existed prior to the World Wide Web and was originally specified for X.25. However, it is now seen as an ideal bearer for mobile IP and has the potential to effect the transformation of mobile operators to mobile Internet service providers. With GPRS, the user is always online (depends on the availability of free timeslots) and generally will pay only for data received/sent or perhaps some flat-rate system.

2.4.3 Enhanced Data-rates for GSM Evolution (EDGE)

EDGE is the final second generation bandwidth upgrade for GSM networks, prior to the advent of 3G mobile service, and uses a higher-level modulation scheme called octagonal phase shift keying (8PSK), EDGE will theoretically provide bandwidth up to 384 kbits/s.

2.4.4 Bluetooth

Bluetooth, while not a bearer technology per se, makes an elegant fit with the wireless lifestyle. The Bluetooth technology is the result of the joint achievements of nine leading company within the telecommunication and computer industries (3 Com, Ericsson, Intel, IBM, Lucent, Microsoft, Motorola, Nokia and Toshiba). Bluetooth enables users to interconnect a wide range of computing and telecommunications devices easily in a plug-and-play manner. Each Bluetooth enabled device contains a Bluetooth microchip that incorporates a radio transceiver and operates in a globally available frequency band for worldwide compatibility.
Examples of Bluetooth applications:

- Wireless headsets
- Wireless office peripherals
- Handheld devices

2.4.5 Other Solutions

The current race of 3G licenses has cost the European telecommunication operators some 10 billion USD. But according to professor [1] Arto Karila and Hannu Kari at the Helsingfors Institute of technology, will the hyped transfer speed of the future UMTS networks only are at

the modest 144 kbits/s and 10 kbits/s for GPRS. They instead advocate a combination of wireless LAN and the existing GSM networks upgraded with GPRS utilizing IP.
This solution will gain the maximal coverage and still offers the high bandwidth for most cases, i.e. in offices or in areas where the wireless LAN still covers. The wireless LAN is today capable of communications with 11 Mbits/s (half duplex) and 5 Mbits/s (full duplex), speeds that the UMTS networks will not offer in the intermediate future. Not if the operators want to give priority to reception coverage due to the relation between coverage area and communication speed. A research project has solved the main issue “passage problem”, which shortly is the problem of switching the communication between basestations when using IP over mobile networks. The research team has developed a prototype basestation called Mart, contains only standard components and costs less than 50 USD while a basestation for UMTS costs 1000 times more.

3 Allaire ColdFusion Application Server 4.5 Enterprise

3.1 Introduction

The first version of ColdFusion application server was released in 1995 and was the first web application server on Windows NT. The current version is available for several platforms (see comparison chart).
It offers developers to create open, scalable applications quickly for deployment on intranets and the Internet.

3.2 Technical Overview

- The **ColdFusion Server** is the deployment platform for delivering ColdFusion applications. It is a multithreaded service architecture application platform that can be scaled with multiple processors on HP-UX, Intel Win32 or SPARC Solaris for support of load balancing and fail over.
- The **ColdFusion Studio** is an integrated development environment for code writing and debugging. It also provides the developers some visual tools and wizards for HTML and database design. A development team has the possibilities of working and managing projects remotely.
- The **ColdFusion Administrator** is a configuration and remote server administration tool. It allows the server manager to monitor, tune, configure and maintain ColdFusion Servers, applications, and clusters.
- The **ColdFusion Markup Language** is a tag-based and server scripting language for building ColdFusion applications. It has a syntax similar to HTML and XML and provides a large range of common programming constructs, function library and expression syntax.

3.2.1 ColdFusion Server

ColdFusion Server runs as a multithreaded process with advanced thread pooling, database connection caching and just-in-time compilation. It's native support for server clustering ensures sites with high demand on availability to continuously stay alive. Copies of applications can run on several servers simultaneously in a clustered environment. If one of the servers is currently heavily loaded or out of service, future requests will be passed to the other servers that are less loaded or still up.
Native support for Open Database Connectivity (ODBC), OLE-DB, email through POP & SMTP, directories through LDAP, file servers through native file system support and FTP, distributed objects through COM, CORBA and EJB. Integration with SAP R/3 through Backsoft's b-Talk EAI server technology. Exchange of complex data between servers and with other programming environments is supported using Web Distributed Data Exchange (WDDX) and XML.
Server sandbox-security offers secures deployment – restricts the access that applications have to directories, components, databases or other resources on the server.

3.2.2 ColdFusion Studio

The ColdFusion Studio is an advanced editor for HTML, CFML and XML with color-coding and automatic tag completion.

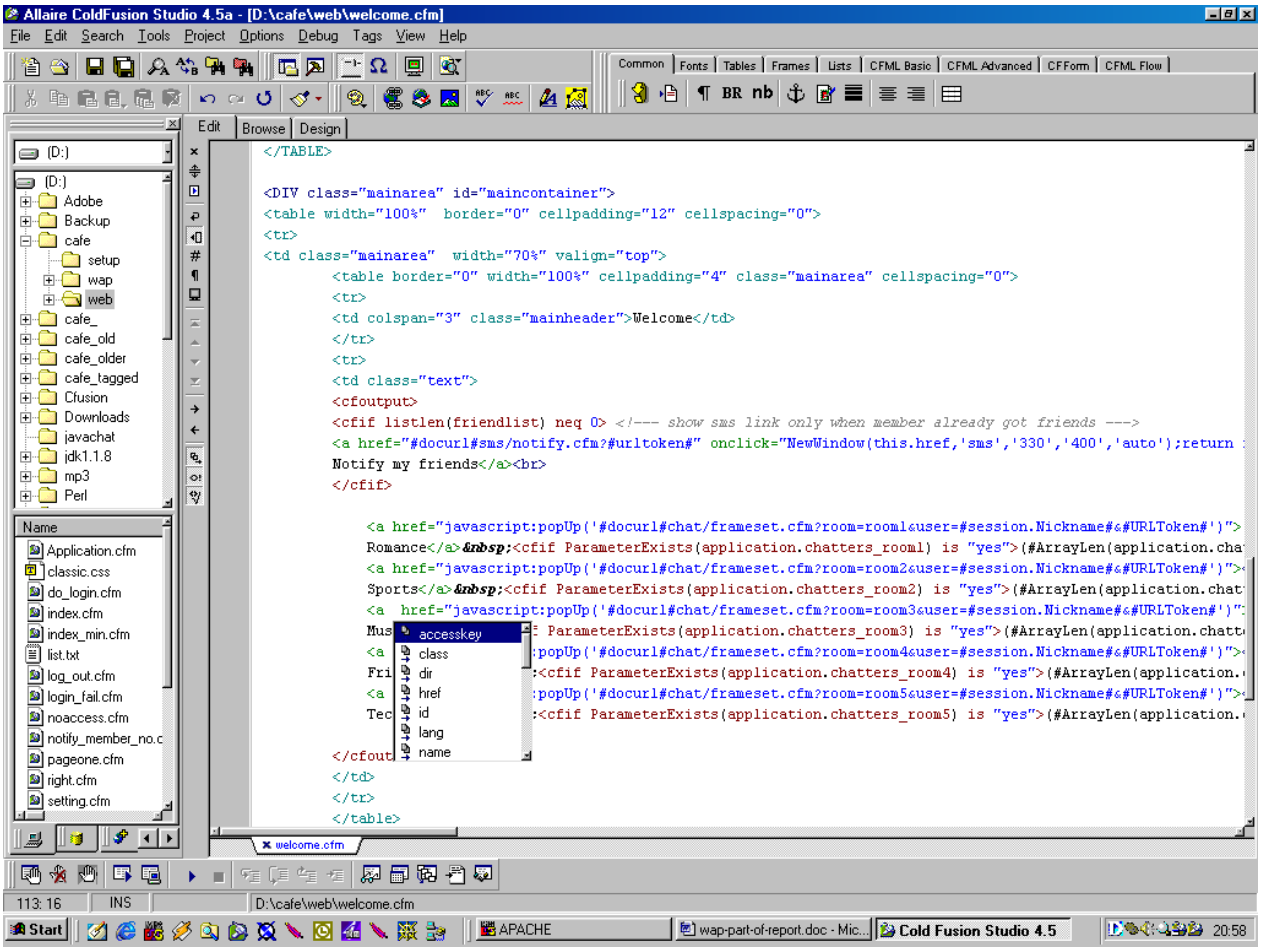


figure 3.2.2-1 Screendump of ColdFusion Studio

It also provides a line-by-line step through debugging tool for easy bug-hunting processes. Complex SQL statements is easy with the visual query builder, several other wizards will help and enable quick web application developments.

3.2.3 ColdFusion Administrator

The ColdFusion Administrator is used to manage and configure ColdFusion Application Server; it is a web-based management tool that allows ColdFusion administrators to use the web browser to manage ColdFusion, regardless of platform.
When the responsible administrator has been authenticated and logged in, he will be presented to a set of tools that will help him to control and maintain the application server's behavior.

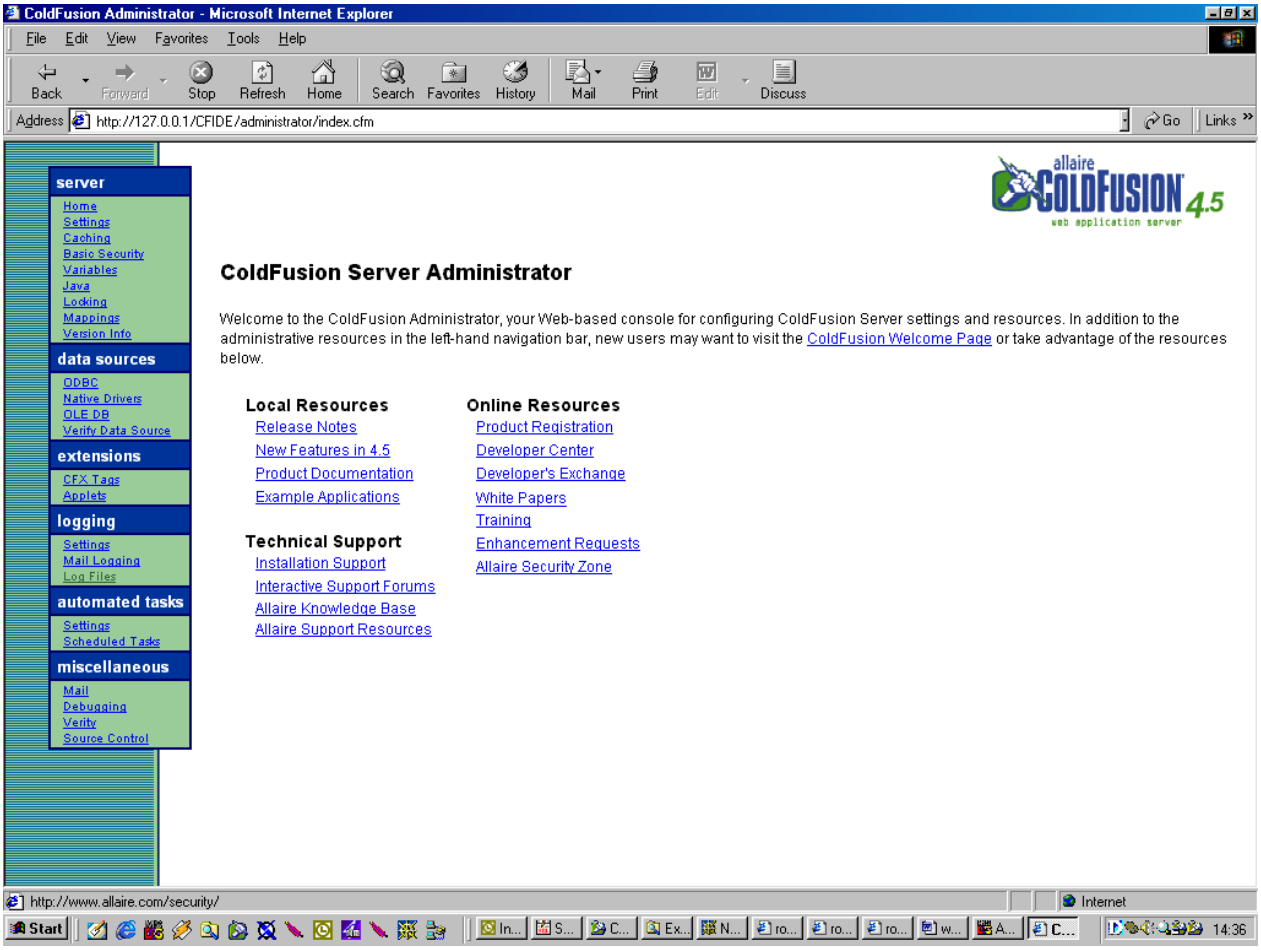


Figure 3.2.3-1 Screendump of the ColdFusion Administrator

Some of the key configurations are:

- Server settings – set limit of simultaneous requests, enable/disable the use of application- and session variables and their life-times, class path for Java applets, server mapping, request timeouts, cache size, security settings etc.
- Data sources settings – select ODBC/OLE-DB/native drivers, create/verify data sources.
- Logging settings – set directories for log files, administrator e-mail, and view log files.
- Automated tasks settings – scheduler refresh interval, schedule tasks for automatic executions.
- Debugger settings – enable/disable stack tracing, enable/disable display of CGI, URL, form and cookie variables, processing time, SQL queries information, restrict debug output information to a specific IP address only.

3.2.4 ColdFusion Markup Language (CFML)

The ColdFusion Markup Language is a tag-based server-side scripting language that is used for writing ColdFusion applications. It offers the usual set of programming components such as variable manipulation, conditional statements, exception handling and dynamically resize of data types as arrays and structures.

Built-in support for the wireless markup language WML for developing killer applications for WAP enabled devices, and XML for large-scale data exchanging applications.

CFML has over 200 built-in functions for

- Array manipulation
- List manipulation
- Mathematical evaluation
- String manipulation/conversion
- Query database
- Date and time formatting

And over 70 tags for

- Forms
- Database manipulation
- Data output
- Exception handling
- File management
- Flow-control
- Java servlet and Java objects
- Variable manipulation
- Extensibility

All CFML tags begin with the prefix CF, for example:

<CFOUTPUT>#someVariable</CFOUTPUT>

to display the content in the variable someVariable.

Web pages written in CFML usually have the file-extension cfm to distinguish it from regular web pages or pages written in other scripting languages. The ColdFusion server processes CFML pages at run-time each time they are requested by a browser.

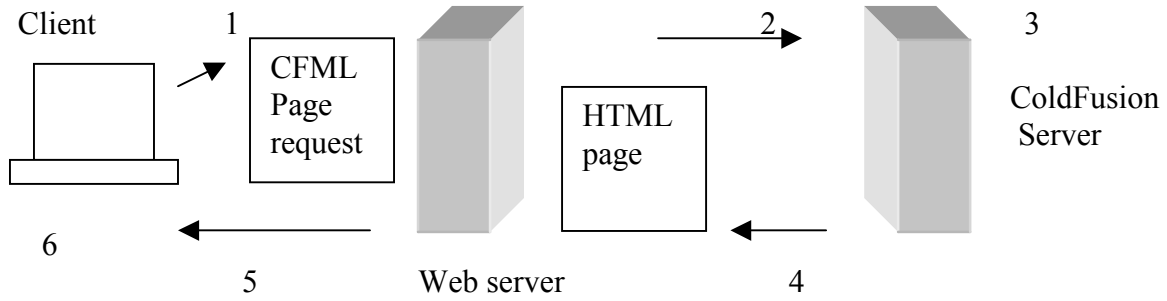


Figure 3.2.4-1 CFML page retrieval.

1. Client requests a CFML page.
2. The web server passes files to ColdFusion server if a page request contains a ColdFusion file extension (.cfm).

3. ColdFusion server read the page and processes all CFML tags.
4. ColdFusion server then returns only pure HTML to the web server.
5. The web server passes the page back to the client's browser.

4 Alternative to WAP

4.1 i-mode

The Japanese i-mode, developed by Nippon Telegraph and Telephone Docomo is based on packet data transmission technology. It has gained a tremendous success in its home market and keys to its huge success in Japan are that subscribers are always online and charged only for how much information they retrieved, not how long time they are online and the large number of services available.

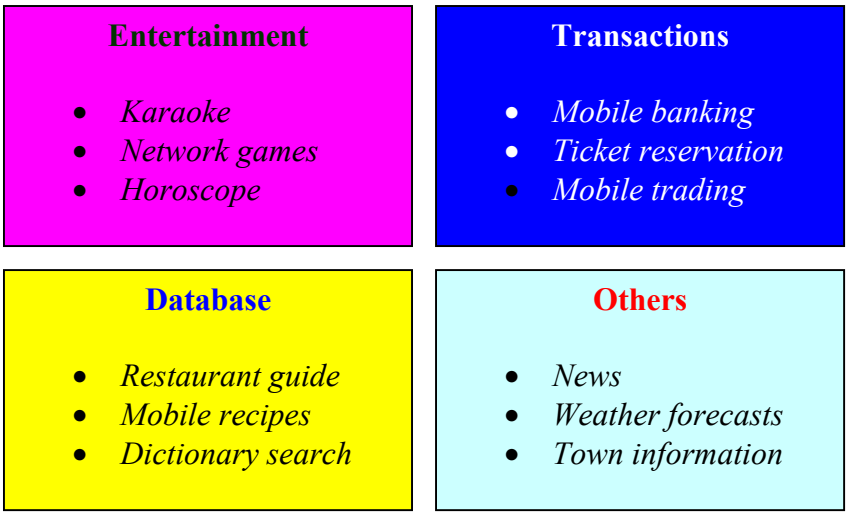


Figure 4.1-1 i-mode service line-up.

The i-mode relies on Code Division Multiple Access (CDMA) as data bearer. The multiple access technology uses the available frequencies more efficiently, allowing multiple users to share radio communications channels to simultaneously conduct communications. There are three ways to separate radio channels, Frequency Division Multiple Access (FDMA) that divides by frequency, Time Division Multiple Access (TDMA) that divides by time, and Code Division Multiple Access (CDMA) that divides by spread codes using the spectrum spread. Figure 4.1-2 shows the differences between the three systems of multiple access technology.

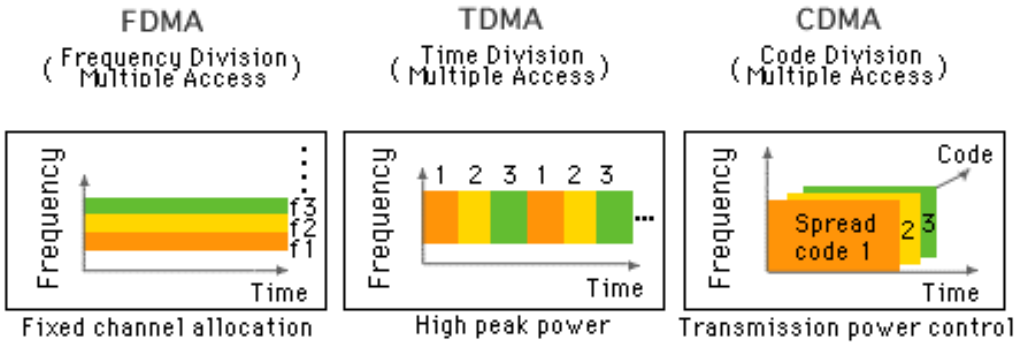


Figure 4.1-2 Multiple access technology

There are currently over 9000 i-mode enable web sites on the Net, approximately 4000 of those are own by business, while the remaining sites are personal pages.

The user's cellular phone connects to the Docomo i-mode center using packet transmission link at 9600 bps. The center provides the phone access to the sites where the services resides either via the Internet or a dedicated line if security is an issue.

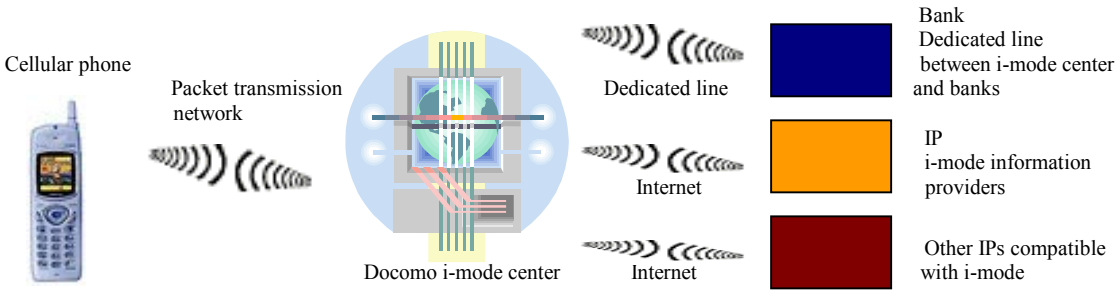


Figure 4.1-3 Network structure

The i-mode enabled sites are written in Compact-HTML (a subset of HTML 1.0) for description, but some Web features are excluded from its vocabulary:

- Tables
- JPEG images (uses GIF instead)
- Image maps
- Background color
- Multiple character fonts and styles
- Java and other scripting languages yet (although NTT Docomo and Sun Microsystems announced an alliance in March 2000 to incorporate Sun's Java, Jini, and Java Card technologies into future i-mode cellular phones.)

The number of I-mode customers exceeded 18 millions as of January 21, 2001.

5 room33.com

The following is a quote from room33's portal presentation page:

" room33 is pioneering the world of mobile Internet services. With the launch of our first offering in early 1999, we became one of the first companies to introduce commercially available mobile Internet services to users around the globe. Today, room33 offers a full suite of leading-edge services to consumers as well as a wide range of business solutions to telecom operators, ISPs and other enterprises that want to add value to their mobile offerings.

Our vision is to become the key to the mobile Internet. We make it easy for people to use and take advantage of the wireless world. Our products and services reflect our thorough understanding of the differing needs of mobile users. We support them as they communicate in diverse situations, use various mobile technologies and work within the interface constraints of today's wireless devices.

Delivering these services is simplified by our advanced, device-independent technology platform. Its future-proof design supports all next-generation mobile technologies, such as GPRS and UMTS.

A multicultural company represented by more than 16 different nationalities, room33's core principle is respect for the individual. Our different backgrounds and competencies shape our open and creative culture, the foundation of our success.
room33 was founded May 1998 in Stockholm, the heart of Sweden's Mobile Valley, by a core team of mobile-industry executives. Today, we employ more than 85 people and are realising our global vision with offices in Stockholm, London, Paris, Madrid and New York.

room33 is an active participant in wireless industry associations including the WAP Forum, the Mobile Applications Initiative, Bluetooth SIG, the World Wide Web Consortium and the Wireless Data Forum. "

5.1 Services

Currently, room33.com offers a bundle of services to their users without charge. The range of services spans from pure business oriented communication services to information-based services. The aimed target group of users is youngster and students between the ages of 17 to 25 years, since this group is quick to adopt new technologies and familiar to the Internet concept.

5.1.1 My room33

- My Links – increase the accessibility of users favorite bookmarks
- My WAP page – users' personal WAP pages
- My room33 - is the start page

5.1.2 Messages & Calendar

- Email – send and receive emails through WAP
- SMS
- Fax
- ICQ – send instant messages to ICQ friends
- Appointment – check your associates public agenda and book meetings
- Calendar
- Contacts – online "telephone and address book"
- Synchronisation – keep users' address books and calendars in sync with Calendar and Contacts

5.1.3 News Central

- Headlines – news flash
- Weather – forecasts for the major cities in the world

5.1.4 Fun Zone

- Ring Tones – send ring tones to mobile phones from the extensive library
- Icons – send logos to mobile phones
- Jokes
- Bar33 – drink dictionary
- ***Café33 – online communities***

5.1.5 Finder

- Mobile Directory – the largest collection of links to WAP pages in the world, down-loadable without charges
- City Guide – guide to bars, shops and restaurants
- Travel Directions – driving directions

5.2 Café33 - brief orientation

Café33 is intended to be the place where members of the room33 portal can exchange their thoughts, share interests and participate in discussion forums. This should be available for the users independently what devices they are using. No matter if they use a conventional web browser or a mobile phone with WAP capabilities to access the service, they will always get the full functionalities of Café33.
These functions was implemented in version 1.0, released in June 2000:

- On signup, the user have to choose at least two interests from the user supplied list of interests, they can put in new interests as well
- Discussion forums, participate in current forums or start new one (either public or private)
- Request/accept new friends
- Profile, add new interest(s), remove current interest(s), send an invitation to non-Café33 member

5.2.1 Welcome page

When the user hit the Café33 link in the service navigation bar, he will be redirected to the Café33 service and the welcome page.

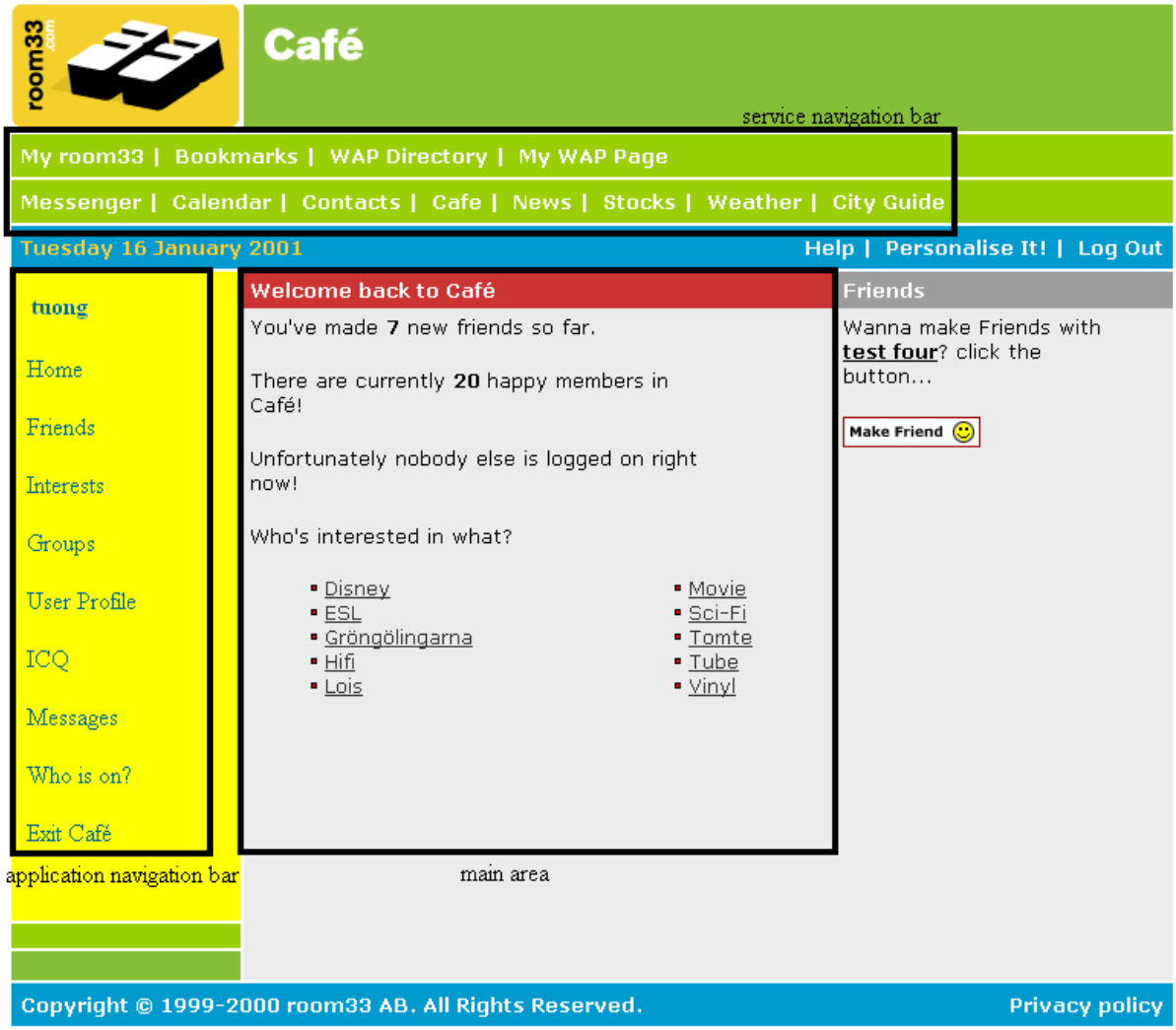


Figure 5.2.1-1 Café33 welcome page.

The application navigation bar will also be exchanged to Café33's own one. The main area will tell the user how many members are logged on, the total number of members and display a list of interests. The Café33 engine will propose a friend for the user based on the interests shared between the two of them.

5.2.2 Friends page

The main purpose of Café33 is to bring people together and make new friends. In the "Friends" section, the user can manage and monitor his friends making process.

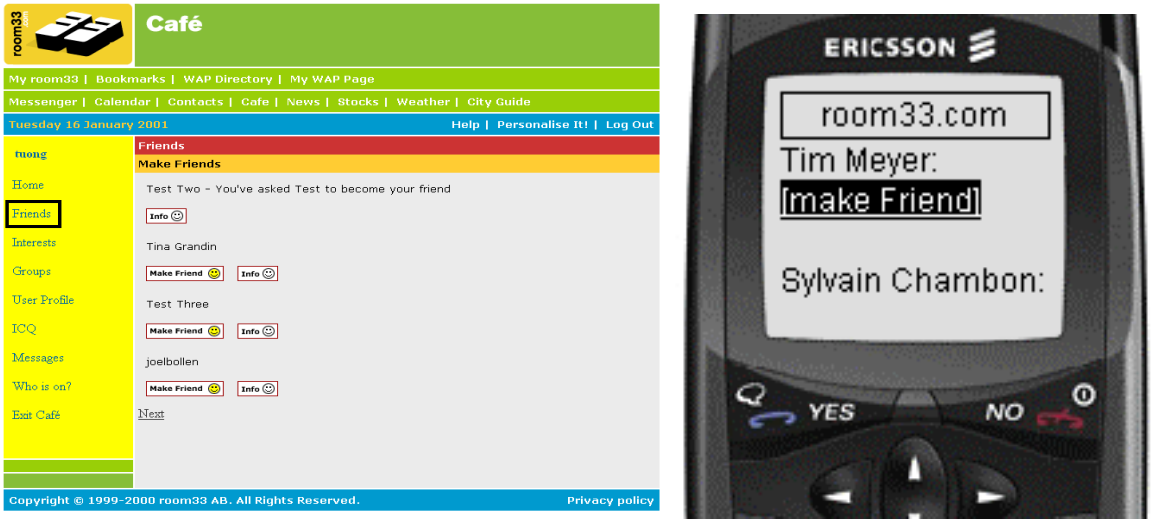


Figure 5.2.2-1 Café33 Make Friends page.

In the "Make Friends" section, user can browse through the current member list, get information about them and send a friendship request by hitting on the "info" or "Make Friend" button respectively.
The members are presented in a manner of "first in, first out" i.e. the newest member will be presented last.

In the "My Friends" section, the user can have an overview of his friends.

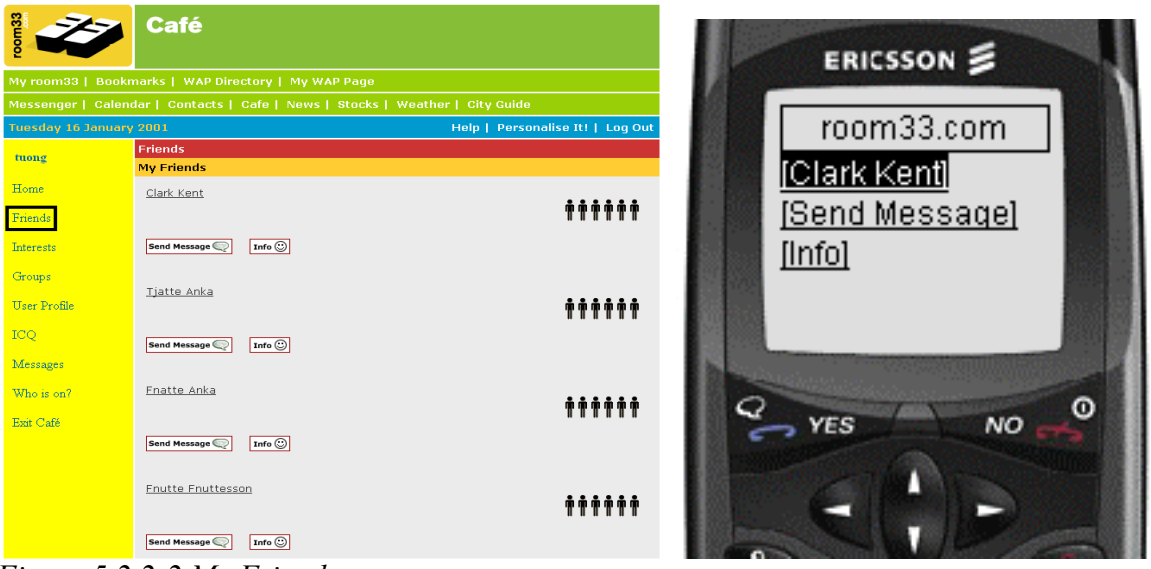


Figure 5.2.2-2 My Friends page.

The friends names are presented as links, if clicked, it will show that friend's friends.
By clicking on the "Send Message" button, users can send instant messages to their friends, while the "Info" button will provide information about that friend.

5.2.3 Groups

Café33 will also be used as a forum, where members can freely exchange their thoughts.

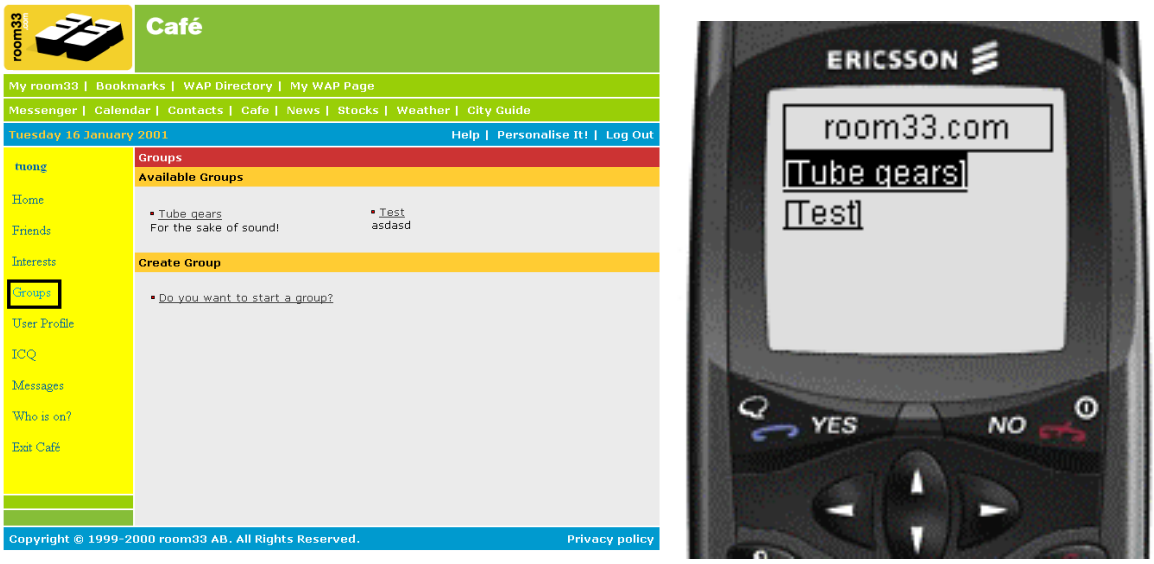


Figure 5.2.3-1 Groups main page.

The main area shows the current ongoing groups and the option to start a group as well. When the user clicks on the name of a group, it will take him to that group and display the postings, post a message and join that group.

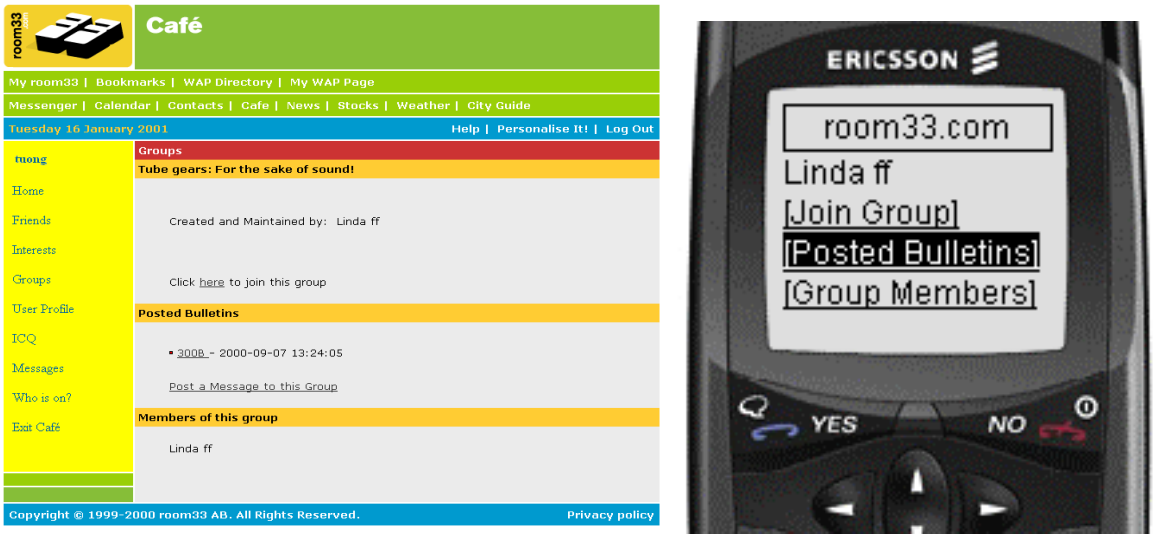


Figure 5.2.3-2 Entering a group page.

When entering a specific group, Café33 will display a list of posted messages with the subjects and date/time headlined, also the creator and the other members.

By clicking on the headline, the message body and the author will be displayed. The “info” button will provide further information about the author.

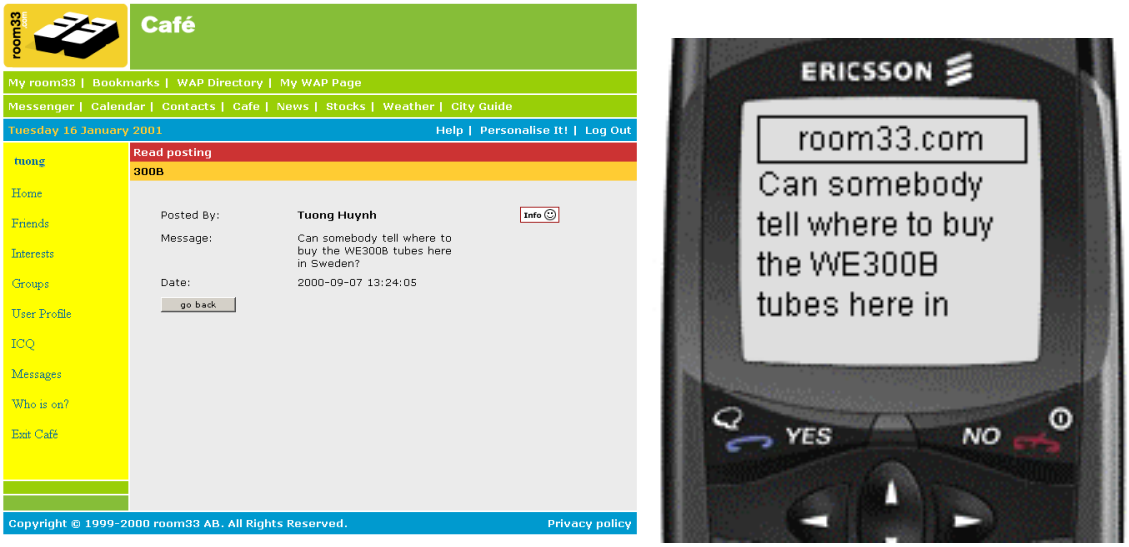


Figure 5.2.3-3 Read posting.

If the user is the creator of that group, an extra link to group managing will be available, by clicking on this link, user can remove members from the group, adding members to the group and remove the group itself as shown in figure 5.3.3-4.

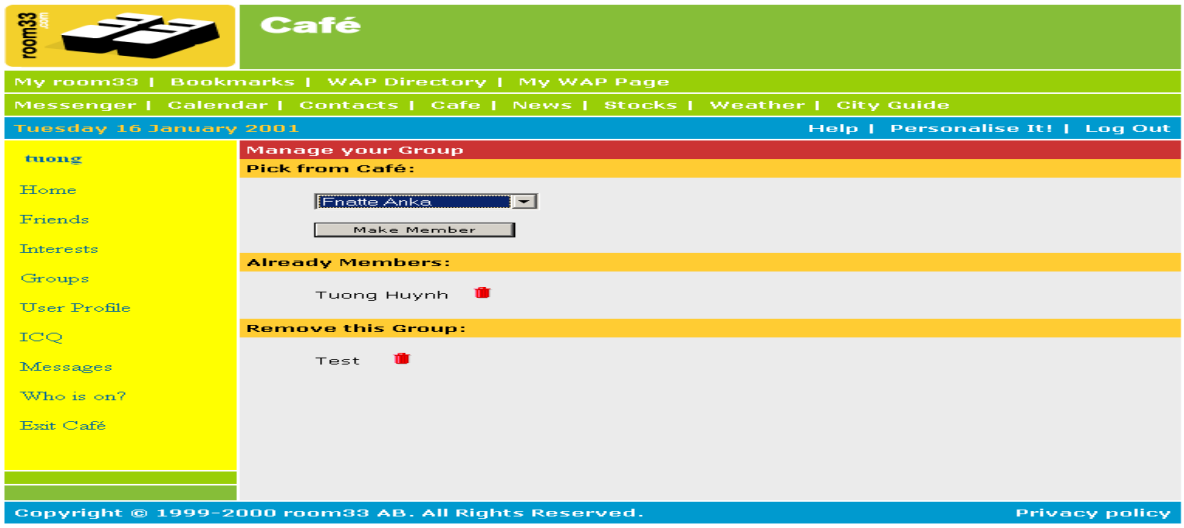


Figure 5.2.3-4 Group manage.

5.2.4 Personal profile

User has the possibilities to add and remove interests in their personal profile.

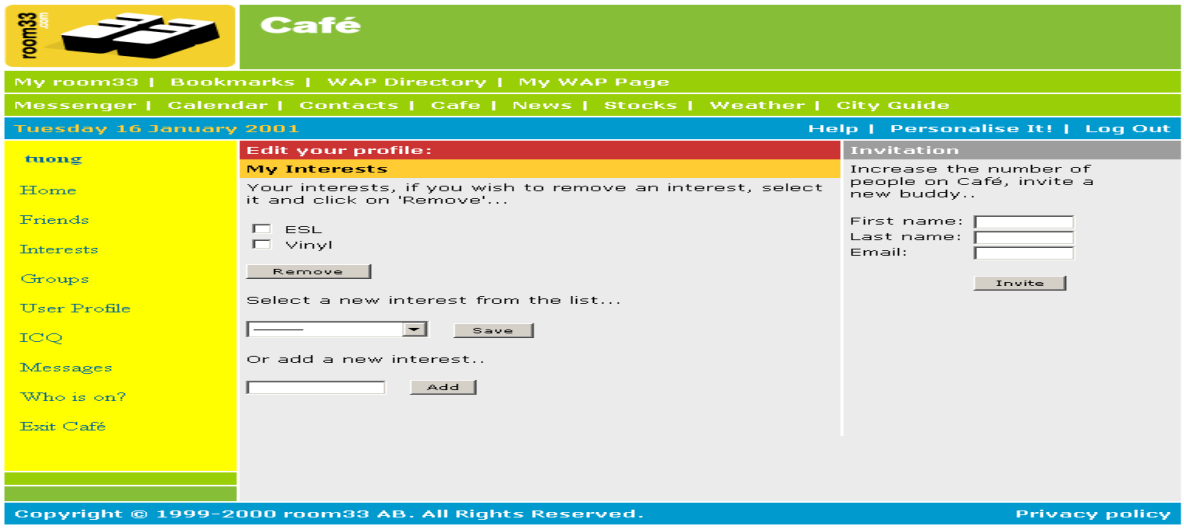


Figure 5.2.4-1 Personal profile.

User can either pick the interests contributed by other members in the Café33 database or they can add new into the database. There are no upper limit of how many interests user can have. To increase the population of Café33 members, user can simply send an invitation email to their friends with a pre-written message stating why they should join Café33 and from who the invitation is from.

5.2.5 Messages

Café33 members can send instant messages their friends.

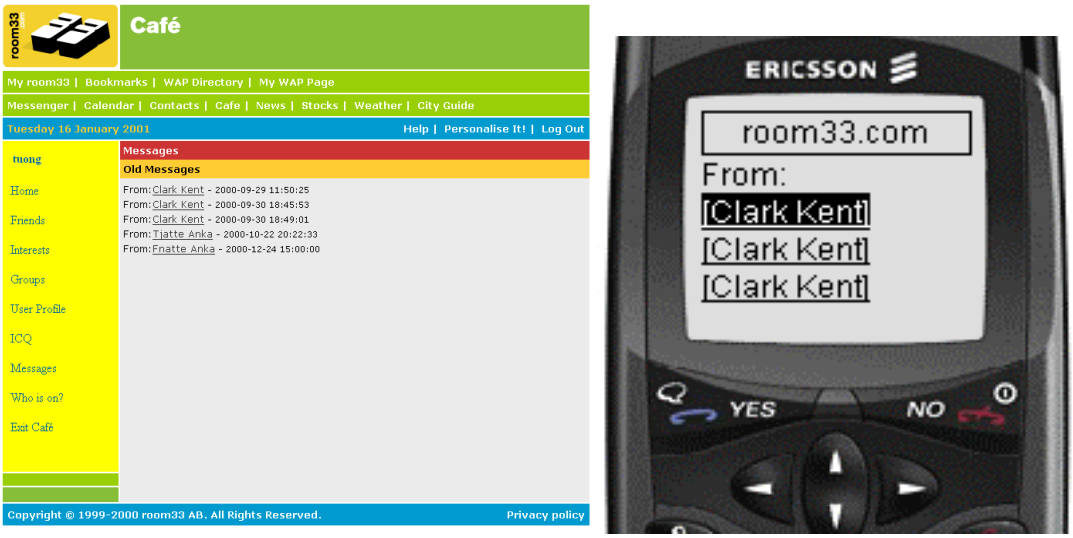


Figure 5.2.5-1 Inbox for messages.

The messages in the inbox are displayed as a list with the senders' name and a timestamp. When a new message arrives, Café33 will alert the user by firing a popup window.

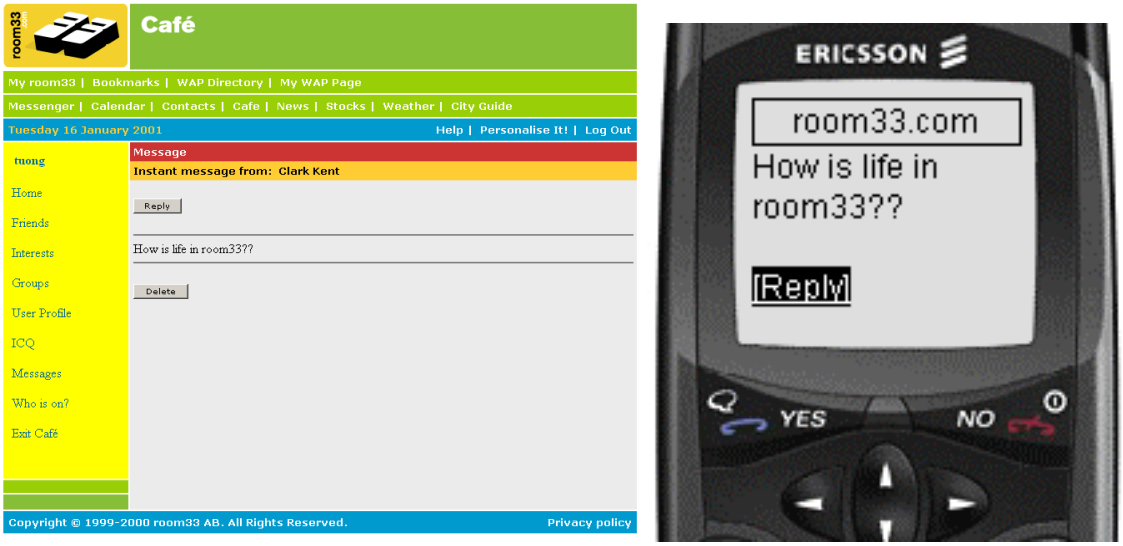


Figure 5.2.5-2 Read message.

After reading the message, user can either reply or delete it.

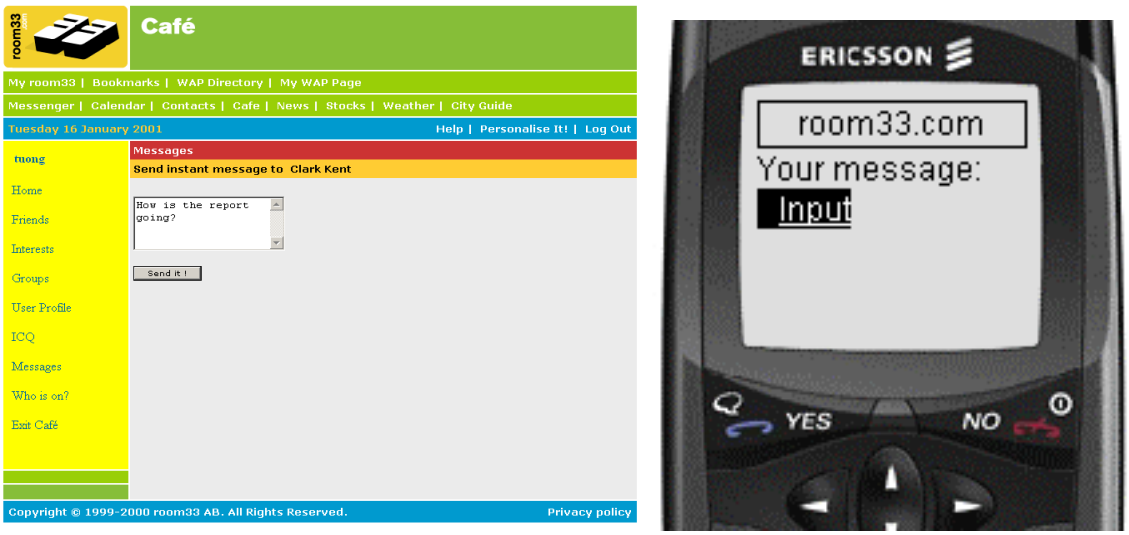


Figure 5.2.5-3 Sending/reply message.

The reply function is the same as in Friends/My Friends/Send message, but when replying a message, the receiver's id will automatically be pre-filled.

5.3 Café33 - technical overview and implementation

The Café33 application is written in ColdFusion Markup Language. The Web version relies on HTML elements for presentation and WML for the wireless version.

This service runs on Allaire's ColdFusion 4.5 Enterprise web application server.

The ColdFusion application server has several proven advantages:

- Multi platform support (Solaris, HP-UX, Linux, Windows NT)
- All codes are processed at runtime on the server side, the clients will only "see" plain HTML or WML code
- Rapid application development, powerful built-in tags and functions, reuse code by creating custom tags and template inclusions
- Database connectivity

When user goes from the portal application in Enhydra to Café33, the Redirect class in Enhydra will insert a session key associated with the user's useroid and a timestamp into the sessionholder table (and the page the redirection was initiated into the referrer column). It will next hook this key in the URL to Café33 and set off the redirection.
The redirection goes to the start.cfm, the first thing this file will do is make sure that the translation dictionary is loaded, then query the sessionholder table with the session key to obtain the user's useroid and set the result as a session variable:

```
<cfset session.key=#url.sk#>
<cfquery name="useroid" datasource="#psource#" dbtype="#pdbtype#" username="#pusername#" password="#ppassword#">
  SELECT useroid FROM sessionholder WHERE sk='#session.key#'
</cfquery>

<cfset session.PortalUserOid=#useroid.useroid#>
<cfset PortalUserOid=session.PortalUserOid>
```

Once we got the useroid we can then use it to get the user's userid:

```
<cfquery name="PortalUser" datasource="#psource#" dbtype="#pdbtype#" username="#pusername#" password="#ppassword#">
  SELECT userid, nickname, language FROM sysuser WHERE oid=#PortalUserOid#
</cfquery>
```

Now let's check if that user is a member of Café33 (i.e. if he has an entry in the database):

```
<cfquery name="Chk" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  SELECT Userid FROM FatUser WHERE Userid = '#PortalUser.userid#'
</cfquery>

<cfif #Chk.recordcount# gt 0>
  <cflocation url="http://room33.com/cafe/web/do_login.cfm?sk=#session.key#">
<cfelse>
  <cflocation url="http://room33.com/cafe/web/signup/do_signup.cfm?sk=#session.key#">
</cfif>
```

If the query yields 0 in result, meaning that he is not in the member database, he will be thrown to the signup page. Otherwise, the system will log him in and send him to the welcome page. The welcome page is in fact also the home page for Café33, where the user-supplied list of interests is displayed. The Café33 navigation in the left of the screen is always accessible as it is included into all the pages (represented as rectangles with thick border in the page flow[Appendix A & B], the rectangles with regular border are "sub menus"). The dashed boxes are options that will only appear if certain condition comes true, like if there are new friends to confirm.

The business logic for the HTML and WML version (except for some functions that are only available in HTML version) is quite similar in. Future code snippets will mostly be taken from the WML templates.
The code for the welcome page for the WML version:

```
<cfcontent type="text/vnd.wap.wml"><?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<cfinclude template="include/config.cfm">
<cfinclude template="include/session.cfm">
<cfinclude template="include/friends.cfm">



<wml>
<card id="welcome" title="room33.com">
<cfinclude template="include/navigation.cfm">
<do type="prev"><prev/></do>
<p><cfoutput>
#session.language.wmlWelcome1#<br/>
<a href="#docurl#new_friend.cfm?cfid=#client.cfid#&amp;cftoken=#client.cftoken#">[#session.language.wmlWelcome2#]</a><br/>
<a href="#docurl#whoson/index.cfm?cfid=#client.cfid#&amp;cftoken=#client.cftoken#">[#session.language.wmlWelcome3#]</a><br/>
<a href="#docurl#message/index.cfm?cfid=#client.cfid#&amp;cftoken=#client.cftoken#">[#session.language.wmlWelcome4#]</a><br/>
<a href="#docurl#friends/index.cfm?cfid=#client.cfid#&amp;cftoken=#client.cftoken#">[#session.language.wmlWelcome5#]</a><br/>
<a href="#docurl#interests/index.cfm?cfid=#client.cfid#&amp;cftoken=#client.cftoken#">[#session.language.wmlWelcome6#]</a><br/>
<a href="#docurl#groups/index.cfm?cfid=#client.cfid#&amp;cftoken=#client.cftoken#">[#session.language.wmlWelcome7#]</a><br/>
<a href="#docurl#log_out.cfm?cfid=#client.cfid#&amp;cftoken=#client.cftoken#">[#session.language.wmlWelcome8#]</a>
</cfoutput></p>
</card>
</wml>
```

The codes/pages can roughly divide into two sections, presentation and business logic/ database logic. The files holding the logical are named as do_something.cfm.
For instance, the page prior to the welcome page is do_login.cfm and looks like:

```
<cfcontent type="text/vnd.wap.wml">
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<cfinclude template="include/config.cfm">

<cfset session.key=#sk#>
<cfset key=session.key>
<cfset signdate = dateformat(now(), "yyyy-mm-dd")>
<cfif ParameterExists(application.startedwml) is "No">
<cfinclude template="translation.cfm">
</cfif>
<cfquery name="useroid" datasource="#psource#" dbtype="#pdbtype#" username="#pusername#" password="#ppassword#">
select useroid from sessionholder where sk='#key#'
</cfquery>

<cfset session.PortalUserOid=#useroid.useroid#>
<cfset PortalUserOid=session.PortalUserOid>
```

```
<cfquery name="PortalUser" datasource="#psource#" dbtype="#pdbtype#" username="#pusername#" password="#ppassword#">
select lastname,firstname,userid,password,email,language from sysuser where oid=#PortalUserOid#
</cfquery>

<!--- Delete the session-key from the database --->
<!---cfquery datasource="#psource#" dbtype="#pdbtype#" username="#pusername#" password="#ppassword#">
  DELETE FROM sessionholder WHERE sk='#session.key#'
</cfquery--->

<cfset session.UserLanguage=#PortalUser.language#>
<cfset UserLanguage=session.UserLanguage>

<!--- Check which language to use --->

<cfif UserLanguage is "Danish"><cfset session.language=application.Danish>
<cfelseif UserLanguage is "Dutch"><cfset session.language=application.Dutch>
<cfelseif UserLanguage is "English"><cfset session.language=application.English>
<cfelseif UserLanguage is "Finnish"><cfset session.language=application.Finnish>
<cfelseif UserLanguage is "Flemish"><cfset session.language=application.Flemish>
<cfelseif UserLanguage is "French"><cfset session.language=application.French>
<cfelseif UserLanguage is "German"><cfset session.language=application.German>
<cfelseif UserLanguage is "Hungarian"><cfset session.language=application.Hungarian>
<cfelseif UserLanguage is "Icelandic"><cfset session.language=application.Icelandic>
<cfelseif UserLanguage is "Italian"><cfset session.language=application.Italian>
<cfelseif UserLanguage is "Norwegian"><cfset session.language=application.Norwegian>
<cfelseif UserLanguage is "Polish"><cfset session.language=application.Polish>
<cfelseif UserLanguage is "Portuguese"><cfset session.language=application.Portuguese>
<cfelseif UserLanguage is "Romanian"><cfset session.language=application.Romanian>
<cfelseif UserLanguage is "Russian"><cfset session.language=application.Russian>
<cfelseif UserLanguage is "Spanish"><cfset session.language=application.Spanish>
<cfelseif UserLanguage is "Swedish"><cfset session.language=application.Swedish>
<cfelseif UserLanguage is "Turkish"><cfset session.language=application.Turkish>
<cfelse><cfset session.language=application.English>
</cfif>

<cfquery name="Chk" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  SELECT Userid FROM FatUser where Userid = '#PortalUser.userid#'
</cfquery>

<cfif #Chk.recordcount# gt 0>
  <!--- Check if the man is a signed up user --->
  <cfquery name="fatuser" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
    SELECT oid, userid, Password FROM FatUser WHERE userid = '#PortalUser.userid#' AND password = '#PortalUser.password#'
  </cfquery>
  <!--- If the user is in the database, redirect the man to where he belongs... --->
  <!--- Check if he is a member of Café 33, if he is send him to welcome, if he is not send him to join --->
  <cfif fatuser.oid neq "">
   <cfquery name="member" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
    SELECT id, Userid FROM Member  WHERE Userid = #fatuser.oid#
  </cfquery>
  <cfif member.id neq "">
    <cfset session.currentuser = member.id><cfset thisuser = session.currentuser>
    <!--- Check if the user has an entry in the Logon table --->
    <cfquery name="Logon" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
    SELECT id FROM Logon WHERE Memberid = #member.id#
  </cfquery>
<cfif trim(logon.id) neq "">
  <!--- Update the entry for this user in the Logon table --->
  <cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
    UPDATE Logon SET Memberid = #member.id#, Dateentered = #CreateODBCDateTime(Now())# WHERE id = #Logon.id#
  </cfquery>
<cfelse>
  <!--- Write an entry to the Logon table for this user--->
  <cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
    INSERT INTO Logon (Memberid, Dateentered) VALUES (#member.id#, #CreateODBCDateTime(Now())#)
  </cfquery>
</cfif>
<!--- Write an entry to the Logonstats table for statistical purposes --->
<cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
INSERT INTO Logonstats (Memberid, Dateentered) VALUES (#member.id#, #CreateODBCDateTime(Now())#)
</cfquery>
<cflocation url="#docurl#welcome.cfm" addtoken="Yes">
<cfelse> <cflocation url="#docurl#login_fail.cfm" addtoken="No">
</cfif>
<cfelse>
```

```
<cflocation url="#docurl#login_fail.cfm" addtoken="No">
</cfif>
<cfelse>
<!--- If the man is not a Café 33 user, then sign him up --->
<cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  INSERT INTO FatUser (Userid, Firstname, Lastname, Password)
  VALUES ('#PortalUser.userid#', '#PortalUser.firstname#', '#PortalUser.lastname#', '#PortalUser.password#')
</cfquery>

<cfquery name="user" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  SELECT oid from FatUser WHERE userid='#PortalUser.Userid#'
</cfquery>
<cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  INSERT INTO Member (Userid, Dateentered) VALUES (#user.oid#, '#SignDate#')
</cfquery>

<!--- Give him two Interest by default --->
<cfquery Name="Interest_one" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#"
password="#fatpassword#">
  SELECT Id FROM Interest WHERE Name='#defaultinterest1#'
</cfquery>
<cfquery Name="Interest_two" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#"
password="#fatpassword#">
  SELECT Id FROM Interest WHERE Name='#defaultinterest2#'
</cfquery>

<cfquery name="member" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  SELECT id, Userid FROM Member WHERE Userid = #user.oid#
</cfquery>

<cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  INSERT INTO Memberinterest (Memberid, Interestid, Dateentered) VALUES  (#member.id#,#Interest_one.id#,'#SignDate#')
</cfquery>
<cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  INSERT INTO Memberinterest (Memberid,Interestid,Dateentered) VALUES (#member.id#,#Interest_two.id#,'#SignDate#')
</cfquery>
<!--- Check if the man is a signed up user --->
<cfquery name="fatuser" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  SELECT oid, userid, Password FROM FatUser WHERE userid = '#PortalUser.userid#' AND password = '#PortalUser.password#'
</cfquery>

<!--- If the user is in the database, redirect the man to where he belongs... --->
<!--- Check if he is a member of Fat Central, if he is send him to welcome, if he is not send him to join --->
<cfif fatuser.oid neq "">
<cfquery name="member" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  SELECT id, Userid FROM Member WHERE Userid = #fatuser.oid#
</cfquery>

<cfif member.id neq "">
  <cfset session.currentuser = member.id>
  <cfset thisuser = session.currentuser>
  <!--- Check if the user has an entry in the Logon table --->
  <cfquery name="Logon" datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
   SELECT id FROM Logon  WHERE Memberid = #member.id#
  </cfquery>
<cfif trim(logon.id) neq "">
  <!--- Update the entry for this user in the Logon table --->
  <cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
   UPDATE Logon  SET Memberid = #member.id#, Dateentered = #CreateODBCDateTime(Now())# WHERE id = #Logon.id#
  </cfquery>
<cfelse>
  <!--- Write an entry to the Logon table for this user--->
  <cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
   INSERT INTO Logon (Memberid, Dateentered) VALUES (#member.id#, #CreateODBCDateTime(Now())#)
  </cfquery>
</cfif>

<!--- Write an entry to the Logonstats table for statistical purposes --->
<cfquery datasource="#fatdatasource#" dbtype="#fatdbtype#" username="#fatusername#" password="#fatpassword#">
  INSERT INTO Logonstats (Memberid, Dateentered) VALUES (#member.id#, #CreateODBCDateTime(Now())#)
</cfquery>
<cflocation url="#docurl#welcome.cfm" addtoken="Yes">
<cfelse>
  <cflocation url="#docurl#login_fail.cfm" addtoken="No">
</cfif>
```

```
<cfelse>
  <cflocation url="#docurl#login_fail.cfm" addtoken="No">
 </cfif>
</cfif>
```

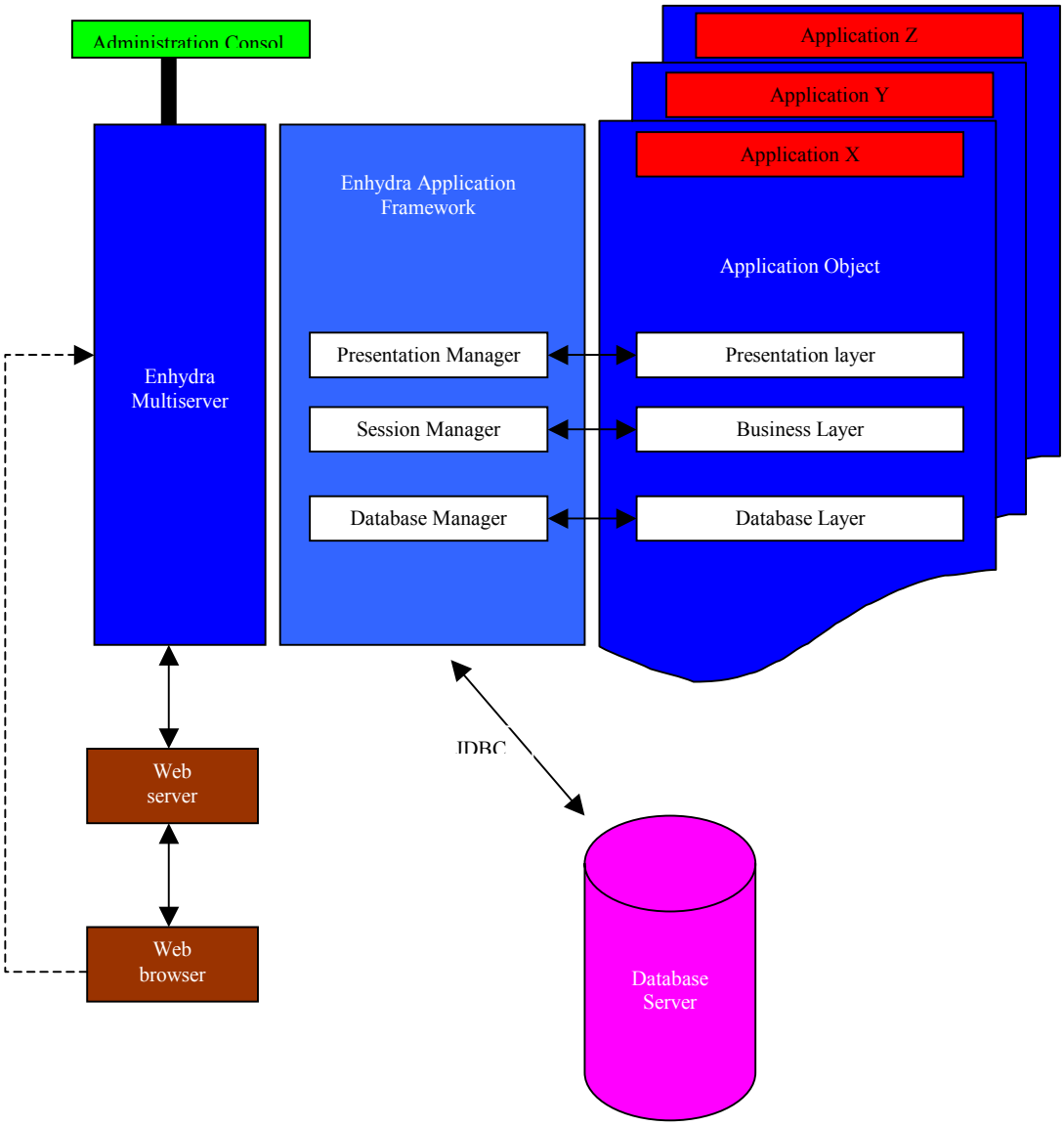
Often used-constants are set in the config.cfm file and then included into the templates when needed:

```
<!--- Directory and Webserver settings --->
<cfset rootdir = "d:\cafe\wap\">
<cfset docurl = "http://127.0.0.1/cafe/wap/">
<cfset portal = "http://room33.com/the/portal/">
<cfset AdminEmail = "tuong@hq.room33.com">

<!--- Database variables --->
<cfset fatdatasource="cafe">
<cfset fatdbtype="ODBC">
<cfset fatusername="dba">
<cfset fatpassword="pwd">

<cfset tsource="translation">
<cfset tdbtype="ODBC">
<cfset tusername="dba">
<cfset tpassword="pwd">

<cfset psource="portalSource">
<cfset pdbtype="ODBC">
<cfset pusername="dba">
<cfset ppassword="pwd">

<!--- Default interests --->
<cfset defaultinterest1="room33">
<cfset defaultinterest2="Café33">
```

6 Lutris Technologies Java Application Server Enhydra

Lutris Technologies Inc. initially created Enhydra in 1997. It is an application server for running robust and scalable multi-tier web applications, and a set of application development tools. Enhydra applications are written in Java that uses the Enhydra framework at runtime.

6.1 Architecture



Enhydra consists of the following parts:

- Multiserver, runs applications either by itself or with a web server
- Application framework, collection of Java classes providing the runtime infrastructure
- Tools for application developments

6.1.1 Application objects

The application object is the central hub of an Enhydra application. It stores application information such as:

- Name of the application
- Status of the application (running/stopped/dead)
- Name and location of the configuration file that initializes the application
- Logging
- References to the application: session manager, database manager and presentation manager

6.1.2 Presentation objects

Presentation objects generate dynamic content for pages in an Enhydra application. When a web browser requests files that end in .po, Enhydra passes the request on to the corresponding presentation object, instantiates and calls the presentation object.

6.1.3 Application layers

An Enhydra application should divide into three distinct parts/layers to keep it easy for maintenance and modular.

- Presentation layer contains presentation objects that handle how the application is presented to web browsers through HTML.
- Business layer contains business objects that hold the application's business logic, algorithms and specialized functions, but not data access or display functions.
- Data layer contains data objects that handle the communication with persistent data source.

6.2 Enhydra Multiserver

Each Enhydra application runs as a single servlet, which is a Java class that dynamically extends the functionality of a web server. The Enhydra Multiserver is the runtime component of Enhydra that provides services an application uses to interface with the web server as well as executes other runtime functions, i.e. a servlet runner.

7 Integrating ColdFusion with Enhydra

A unique ID, the session key in Enhydra and clientID in ColdFusion connects the applications in Enhydra and ColdFusion to a user. These ID:s must be passed between these two platforms when users hop from a application in Enhydra to ColdFusion and vice versa to keep track of the user's session data.

7.1 Implementation

The best way to enable the interchanging of session ID: s is to pass the relevant session key in Enhydra to ColdFusion via the URL. There are several reasons why we do not want to pass the user's userid nor his useroid in the URL. The main reason is the security issue; there is always a risk when sending confidential information in the URL.
When a user follows the link that will take him from the Enhydra framework to a ColdFusion application, the user's current session key and useroid are inserted into a database table. Among the useroid and session key, the referring page's URL and time are also inserted.

Sk	Useroid	Referrer	Time
Session key	Identify the user	Refers to the page from where the user was coming from	When the session-key was created

The sessionholder table

The Café33 start page then do a database query to the sessionholder table to get the useroid with the session key:

```
<cfset session.key=#url.sk#>

<cfquery name="UserOID" datasource="#psource#" dbtype="#pdbtype#" username="#pusername#" password="#ppassword#">
  SELECT useroid FROM sessionholder
  WHERE sk='#key#'
</cfquery>
```

Once the session key is used, e.g. when the ColdFusion application has retrieved all the necessary data from the sessionholder table and stored that as ColdFusion session variables, the entry is deleted.
With the useroid, we can now get all information we want to know about this user:

```
<cfquery name="PortalUser" datasource="#psource#" dbtype="#pdbtype#" username="#pusername#" password="#ppassword#">
  SELECT lastname,firstname,userid,password,email,language FROM sysuser
  WHERE oid=#UserOID.useroid#
</cfquery>
```

8 Translation

Since the room33 portal exists in twelve languages, it is desired that Café33 also go multilingual.

8.1 Implementation

The translation intelligence of Café33 is quite straightforward. The interpreter table for the portal is built as follow:

Page	Id	Language	Device	Text	Status	Version	Oid
	X	X		X			X

The X-marked columns are the one I need:

- Oid – unique counter for each entry in the table
- Id – tag identification, a unique variable for each language
- Text – the translated text
- Language – the language of the translated text

To make the Id unique (for each language), I named them in this manner:

1. for HTML pages : html + directory name + file name + running number
2. for WML pages : wml + directory name + file name + running number

Example: htmlProfileIndex23 for tag number 23 in file index.cfm in the profile directory.

Further, I thought it was a good idea to load the whole translation table into separate structures in the application scope; we will hence avoid unnecessarily future reloading of translation tags (unless the server is restarted).

```
<!--- Retrieve all the languages in the DB --->
<cfquery name="Lang" datasource="#tsource#" dbtype="#tdbtype#" username="#tusername#" password="#tpassword#">
  SELECT DISTINCT language FROM interpreter
</cfquery>

<!--- Create a query for each of the language --->
<cfloop query="Lang">
  <cfquery name="#language#" datasource="#tsource#" dbtype="#tdbtype#" username="#tusername#" password="#tpassword#">
    SELECT id,text FROM interpreter
    WHERE language='#language#'
    AND id LIKE 'html%'
  </cfquery>
</cfloop>

<cfset application.English=StructNew()>

<cfloop query="English">
  <cfset temp=StructInsert(application.English,"#id#","#text#")>
</cfloop>
…
```

We now have a complete set of lexicon for each language stored in a structure and could then output the text by calling the structure with the key for the corresponding value:

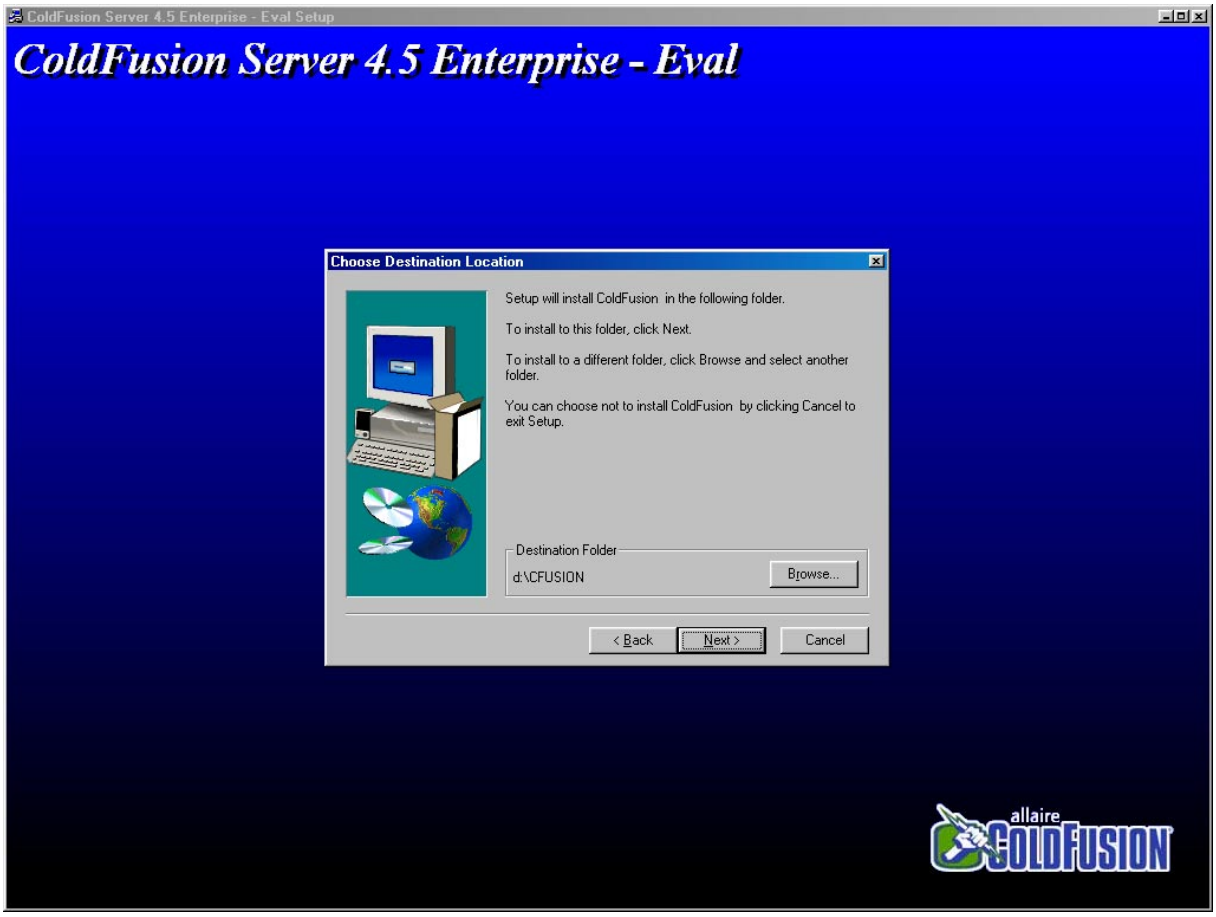
```
<cfoutput>#application.English.htmlProfileIndex23#</cfoutput>
```

This would for example output the text “Save” if we are using the English lexicon.

9 ColdFusion setup

9.1 Microsoft Windows NT 4 Server

The setup wizard is quite straightforward, as Windows user, you need only to click through the setup process.



During the setup, the wizard will ask for:

- where to put the files
- which web server to be configured for use with ColdFusion, if the web server was not detected by setup, choose “Other Server”
- the web server’ document directory for ColdFusion HTML components, examples and documentation
- components to be installed

When the setup is completed, it is necessary to reboot the system.

9.1.1 Configuring the Apache Web Server 1.3.x

Since the Apache web server was not detected by ColdFusion setup, it is necessary to configure it manually. To "tell" the web server how to handle ColdFusion templates, it needs to load a so called "dynamic load library", ApacheModuleColdfusion.dll on startup. This file can either be downloaded from the Allaire's support page or it can be found in the **cfusion/bin** directory. Simply put this file in the Apache's modules directory and add the following line to **httpd.conf**:

```
LoadModule coldfusion_module modules/ApacheModuleColdFusion.dll
```

Then restart the web server.

9.2 SUN Solaris

The setup on Solaris didn't differ much from the Windows installation, but the setup in Solaris environment was text based, there were the same options given.

9.2.1 Configuring the Apache Web Server 1.3.x

There is a similar module (*mod_coldfusion.so*) that Apache needs to load in order to handle ColdFusion templates. Apache need to be built and configure with the "mod_so" module enabled in order to dynamically load modules.
The share object file can either be downloaded from Allaire's support page or in the **coldfusion/webserver/apache/** directory. Place this file into the **apache/libexec** directory and edit the **httpd.conf** to include the following line:

```
LoadModule coldfusion_module libexec/mod_coldfusion.so
```

9.3 Distributed ColdFusion

ColdFusion 4.5 can be configured in a distributed manner where the ColdFusion engine is running on a separate computer from the web server. Running ColdFusion in this way might be called distributed or remote ColdFusion.
In addition to allowing the ColdFusion engine to be located on a separate machine from the web server, distributed ColdFusion provides the following unique capabilities:

- It allows the machine hosting the web server to potentially be of a different architecture from the machine hosting the ColdFusion engine.
- It allows more than one web server to be served by the same ColdFusion engine.

To provide some degree of security for the data being transferred between the web server and the ColdFusion engine, that conversation is encrypted using a standard, 56-bit DES encryption algorithm. Although it's possible for a ColdFusion engine to simultaneously service both local and remote requests, it is not possible for a single Web server to simultaneously dispatch both local and remote ColdFusion requests. When starting up, the ColdFusion Web server plug-in determines if it's to run in local or remote mode and remains in that mode until it's shutdown.

9.3.1 Setting it up

Make a standard installation of ColdFusion on all the machine involved, to ensure that both the computer running web server has loaded the ColdFusion server plug-ins correctly and the computer running the ColdFusion engine is set-up and operating correctly.
To run distributed ColdFusion, the ColdFusion web server plugin must be notified that it shall talk to a ColdFusion engine on another machine by simply making appropriate entries in an INI file. Then on the ColdFusion engine side, run an additional piece of software, known as

the **Network Listener Module**, that listens for incoming ColdFusion requests and forwards them to the ColdFusion engine running on that machine. The ColdFusion engine itself is a standard release version of the engine with no special modifications to accommodate remoting.
If the remote configuration operates successfully, the ColdFusion engine installed on the computer hosting the web server, should be disabled for security reasons, by renaming or deleting the following executable files in the cfusion/bin (for Windows) or coldfusion/bin (for Solaris) directory:

- cfserver
- cfrdsservice
- cfexec

This prevents any ColdFusion server-side process from running while generally preserving the ColdFusion configuration.

The web server side

In ColdFusion 4.5 all the Web server plug-ins are remote-capable so no special installation is required. All that need to do is letting the plug-in know that it should run in remote mode, by putting the following information in an INI file and putting that file in the root directory of the ColdFusion installation on the machine running the web server. That INI file must be named **cfremote.ini**. This INI file may be optionally set to be automatically deleted after being read at start-up to enhance security.

The ColdFusion Server side

The NLM (Network Listener Module) is a stand-alone program that acts as a network front-end for the standard ColdFusion Server. It runs on the same computer on which the ColdFusion Server is running. It listens for incoming requests via TCP/IP and forwards them on to the local ColdFusion Server. The ColdFusion Server then processes those requests, returning the results to the listener module that, in turn, returns them via the original TCP/IP connection. It is a silent, background process with no user interaction. On NT, it runs as an NT service. On UNIX, it runs as a daemon. For debugging or other special purposes, it may also be run as a command line program by specifying the appropriate command line option (-i) at start-up.

Installing the module on Windows NT

On NT, the module consists of a single executable file, cfdist.exe. Before it can runs as an NT service, the following installation step must be done.

To install the network listener module as a service:

Run the listener with the following special command line argument:

cfdist.exe –sINSTALL

If installation was successful, it should now appear on the Services list under the name ColdFusion NetListener. If it doesn't show up, look in the module's log file, distributed.log in

the log subdirectory of the ColdFusion installation, for information about why the install failed.

To uninstall the listener:

Invoke **cfdist.exe** with the -sREMOVE command line option. Notice of successful removal will be written to the listener log.

Installing the module on UNIX

On UNIX, the listener module consists of a single executable file, in this case named simply **cfdist**. It is not necessary to perform any special installation step on UNIX.

To start the listener as a daemon:

Type the executable's name (without the -i switch) and the process will start. Because it's running as a daemon, the command will return immediately having launched the process in the background.

To stop the daemon process:

Kill it by its process ID. Use the ps command to get the PID and then kill the process as demonstrated below.

ps -deaf | grep cfdist | grep -v grep

It returns the PID in a string something like:

ckintzin 980 1 0 15:48:12 ? 0:00 cfdist

The first number is the PID. Use it in the kill command to stop the process:

kill -INT 980

10 Summary and Conclusions

The benefit

Since the standard for providing the Internet to mobile devices was set to WAP by the major mobile phone manufacturers and that standard is also supported by most of the telecom operators in Europe, it falls natural to develop services for WAP.
As the processed CFML tags are transparent for the browsers (both HTML and WML), they provide the developer with a powerful way to create useful applications independently of what platform the user's browser maybe running on. The WML's lack of session management is a potential obstacle to create applications where the user information must maintain over more than one page, i.e. mobile commerce or usual portal functions where the user has to log in etc.

ColdFusion has both application and session managements. Each application shares the variables that are declared as an application variable within an application's scope name. The users' sessions are held separately by a unique url-token, i.e. variables declared as a session variable belong to that specific user only.

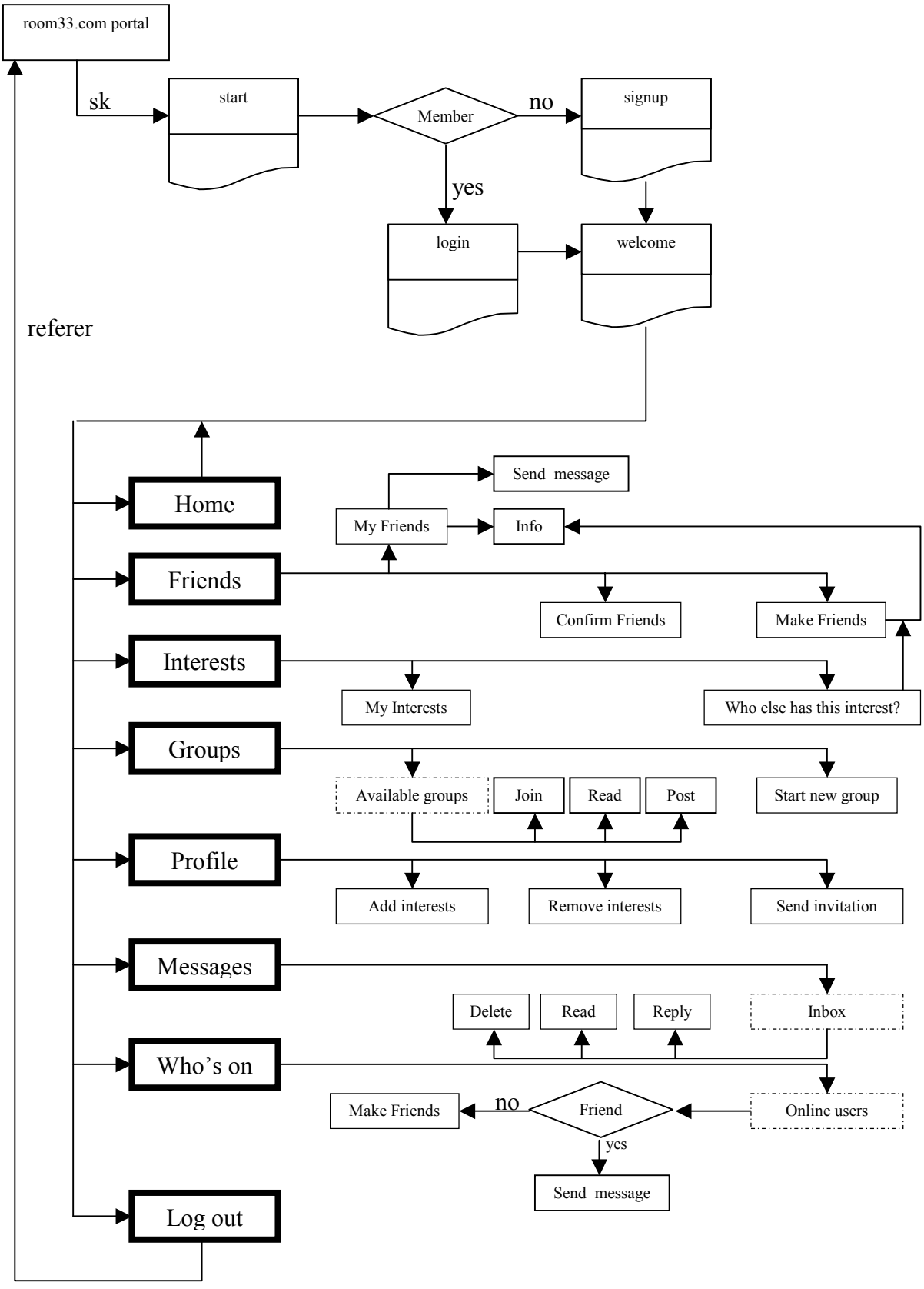
ColdFusion Studio

The CF Studio is the center-of-universe during application development. This editor retains the familiar two-paned interface of traditional editors. In the right pane is Allaire's Quickbar, which includes a series of tabs that change the toolbar to grouping of commands likely to be used in tandem, including those for fonts, tables, and forms. In the Resource Window, there are tabs for files, projects, site view, online help and tag inspector. The right side of the screen is the work area where I can switch among the code and preview mode. Working with the Studio is quite flexible, the program goes a long way toward assuring that my code is valid by anticipating the tags being typed in, displaying and letting me select the possible parameters on the fly, and closing tags automatically. It has further useful features that make the development process less painful like multifile search-and-replace, teamwork with projects, source control, debug etc.

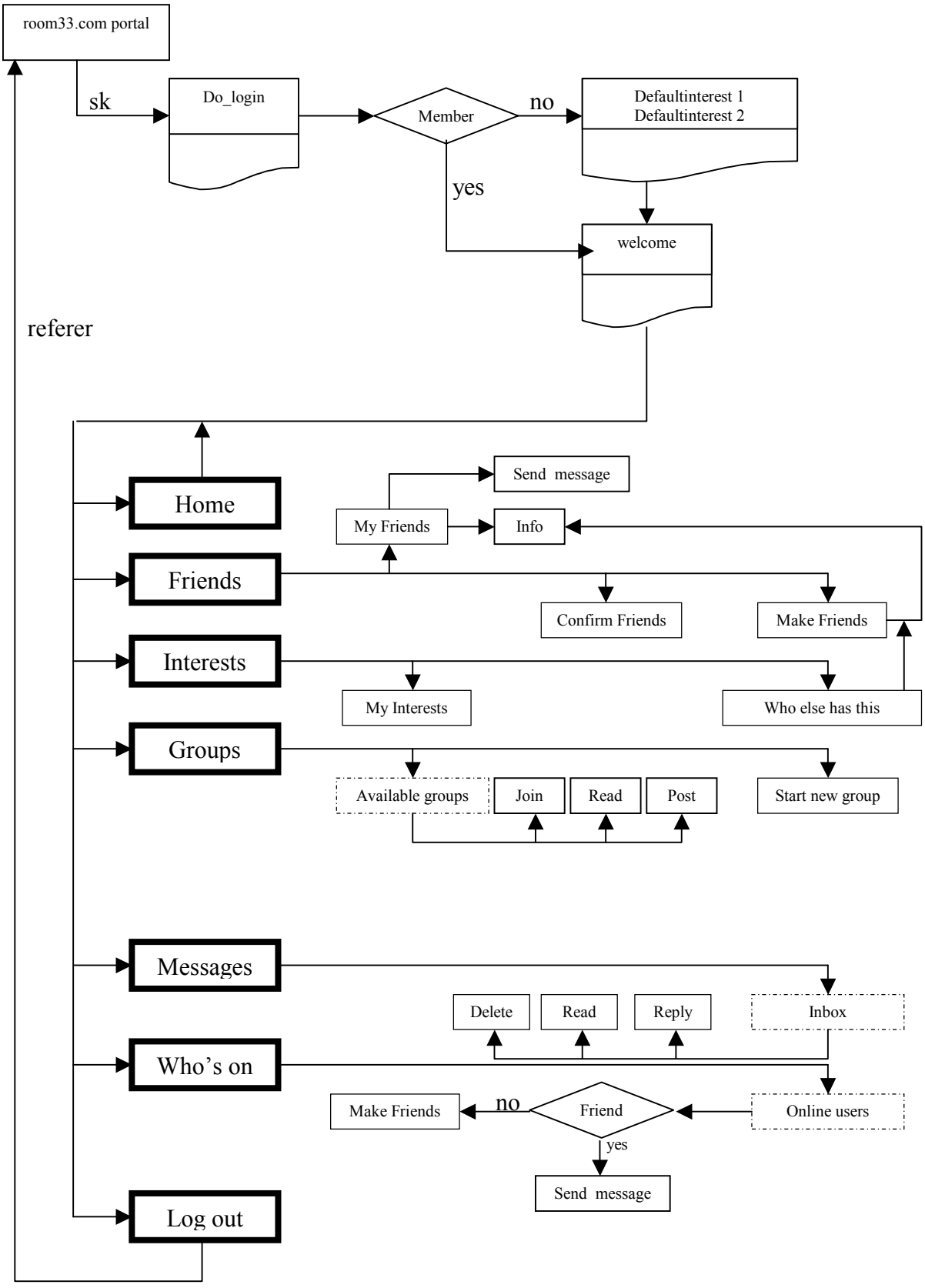
Future work

During the finalization of this report, large efforts have been put to improve the functionality and usability of Café33. Within the area of new functionality, next version of Café33 will be released with a full-blown chat, not only on the HTML version, but also on WML.
Cares also have been taken of considering usability; the main goal was to simplify the reach ness of the functions. Fewer clicks to send an instant message to friends and the introduction of best friends list, where the user can rename their best friends to whatever they want to. To avoid getting spammed by hostile members, the ignore-list will prevent users to receive unwanted messages from members they put into his/her ignore-list. Ability to send an SMS to one or all offline friends without the information about their cell phone number and invite them to come online will make it easier to get in touch with "virtual" friends (users can of course turn off the SMS alerting).

Appendix A Page flow (html version)



Appendix B Page flow (wml version)



Appendix C WML Reference

Decks and cards

Element	Syntax
<**wml**>	<wml xml:lang="***lang***" > ***content*** </wml>
<**card**>	<card id="***name***" title="***label***" newcontext="***boolean***" style="***style***" onenterforward="***url***" onenterbackward="***url***" ontimer="***url***" > ***content*** </card>
<**template**>	<template onenterforward="***url***" onenterbackward="***url***" ontimer="***url***" > ***content*** </template>
<**head**>	<head> ***content*** </head>
<**access**>	<access domain="***domain***" path="***path***" />
<**meta**>	<meta name="***name***"|http-equiv="***name***" content="***value***" forua="true | **false**" />

Timers

Element	Syntax
<**timer**>	<timer name="***variable***" value="***value***" />

Variables

Element	Syntax
<**setvar**>	<setvar name="***name***" value="***value***" />

Anchored links

Element	Syntax
<**anchor**>	<anchor title="***label***">***task text***</anchor>
<**a**>	 ***task*** ***text***

Events

Element	Syntax
<**do**>	<do type="***type***" label="***label***" name="***name***" optional="***boolean***" > ***task*** </do>
<**onevent**>	<onevent type="***type***" > ***task*** </onevent>

Tasks

Element	Syntax
<**go**>	<go href="***url***"

	sendreferer="***boolean***" method="***method***" accept-charset="***charset***" ***content*** </go>
<**prev**>	<prev> ***content*** </prev>
<**noop**>	<noop/>
<**refresh**>	<refresh> ***content*** </refresh>

Images

Element	Syntax
<**img**>	

User input

Element	Syntax
<**input**>	<input name="***variable***" title="***label***" type="***type***" value="***value***" default="***default***" format="***specifier***" emptyok="***boolean***" size="***n***" maxlength="***n***" tabindex="***n***" />
<**select**>	<select title="***label***" multiple="***boolean***" name="***variable***" default="***default***" iname="***index_var***" ivalue="***default***"

	tabindex="***n***" > ***content*** </select>
<option>	<option title="***label***" value="***value***" onpick="***url***" > ***content*** </option>
<optgroup>	<optgroup title="***label***" > ***content*** </optgroup>
<fieldset>	<fieldset title="***label***"> ***content*** </fieldset>

Layout and text formatting

Element	Syntax
****	 ***text***
<big>	<big> ***text*** </big>
** **	
****	 ***text***
****	 ***text***
<p>	<p align="***alignment***" mode="***wrapmode***" />
<small>	 ***text***
****	 ***text***
<table>	<table align="***alignment***" title=***"label"*** columns=***"n"***/>

<**td**>	<td>***content***</td>
<**tr**>	<tr> <td>***content***</td> </tr>
<**u**>	<u> ***text*** </u>

Special characters

Element	Display character
<	< (less than)
>	> (greater than)
'	' (apostrophe)
"	" (quote)
&	& (ampersand)
$$	$ (dollar sign)
	Non-breaking space
	Soft hyphen

Appendix D CFML Tag Reference

CFML Tag	Description
CFABORT	Stops processing of a ColdFusion page at the tag location.
CFAPPLET	Embeds Java applets in a CFFORM.
CFAPPLICATION	Defines application name, activates client variables.
CFASSOCIATE	Enables sub-tag data to be saved with the base tag.
CFAUTHENTICATE	Authenticates a user and sets the security context for an application.
CFBREAK	Breaks out of a CFML looping construct.
CFCACHE	Caches ColdFusion pages.
CFCOL	Defines table column header, width, alignment, and text.
CFCOLLECTION	Creates and administers Verity collections.
CFCONTENT	Defines the content type and, optionally, the filename of a file to be downloaded by the current page.
CFCOOKIE	Defines and sets cookie variables.
CFDIRECTORY	Performs typical directory-handling tasks from within your ColdFusion application.
CFERROR	Displays customized HTML error pages when errors occur.
CFEXECUTE	Executes any developer-specified process on the server machine.
CFEXIT	Aborts processing of currently executing CFML custom tag.
CFFILE	Performs typical file-handling tasks from within your ColdFusion application.
CFFORM	Builds an input form and performs client-side input validation.
CFFTP	Permits FTP file operations.
CFGRID	Used in CFFORM to create a grid control for tabular data.
CFGRIDCOLUMN	Used in CFFORM to define the columns used in a CFGRID.
CFGRIDROW	Used with CFGRID to define a grid row.
CFGRIDUPDATE	Performs updates directly to ODBC data source from edited grid data.

CFHEADER	Generates HTTP headers.
CFHTMLHEAD	Writes text, including HTML, to the HEAD section of a specified page.
CFHTTP	Used to perform GET and POST to upload files or post a form, cookie, query, or CGI variable directly to a specified server.
CFHTTPPARAM	Used with CFHTTP to specify parameters necessary for a CFHTTP POST operation.
CFIF CFELSEIF CFELSE	Used to create IF-THEN-ELSE constructs.
CFIMPERSONATE	Allows you to impersonate a user defined in a security context defined in Advanced Security.
CFINCLUDE	Embeds references to ColdFusion pages.
CFINDEX	Used to create Verity search indexes.
CFINPUT	Used in CFFORM to create input elements such as radio buttons, checkboxes, and text entry boxes.
CFINSERT	Inserts records in an ODBC data source.
CFLDAP	Provides access to LDAP directory servers.
CFLOCATION	Opens a ColdFusion page or HTML file.
CFLOCK	Ensures data integrity and synchronizes the execution of CFML code.
CFLOOP	Repeats a set of instructions based on a set of conditions.
CFMAIL	Assembles and posts an email message.
CFMAILPARAM	Attaches a file or adds a header to an email message.
CFMODULE	Invokes a custom tag for use in your ColdFusion application pages.
CFOBJECT	Creates and uses COM, CORBA, or JAVA objects.
CFOUTPUT	Displays output of database query or other operation.
CFPARAM	Defines a parameter and its initial default value.
CFPOP	Retrieves messages from a POP mail server.
CFPROCESSINGDIRECTIVE	Suppresses extraneous white space, and other output.
CFPROCPARAM	Specifies parameter information for a stored procedure.
CFPROCRESULT	Specifies a result set name that other ColdFusion tags use to access the result set from a stored procedure.
CFQUERY	Passes SQL to a database.
CFQUERYPARAM	Reads, writes, and deletes keys and values in the system registry.
CFREGISTRY	Reads, writes, and deletes keys and values in the system

	registry.
CFREPORT	Embeds a Crystal Reports report.
CFRETHROW	Rethrows the currently active exception.
CFSCHEDULE	Schedules page execution with option to produce static pages.
CFSCRIPT	Encloses a set of CFScript statements.
CFSEARCH	Executes searches against data indexed in Verity collections using CFINDEX.
CFSELECT	Used in CFFORM to create a drop-down list box form element.
CFSERVLET	Executes a Java servlet on a JRun engine.
CFSERVLETPARAM	Used to pass data to the Java servlet. CFSERVLETPARAM is a child tag of CFSERVLET.
CFSET	Defines a variable.
CFSETTING	Define and control a variety ColdFusion settings.
CFSILENT	Suppresses all output that is produced by the CFML within the tag's scope.
CFSLIDER	Used in CFFORM to create a slider control element.
CFSTOREDPROC	Specifies database connection information and identifies the stored procedure to be executed.
CFSWITCH CFCASE CFDEFAULTCASE	Evaluates a passed expression and passes control to the CFCASE tag that matches the expression result.
CFTABLE	Builds a table.
CFTEXTINPUT	Places a single-line text entry box in a CFFORM.
CFTHROW	Raises a developer-specified exception.
CFTRANSACTION	Groups CFQUERYs into a single transaction; performs rollback processing.
CFTREE	Used in CFFORM to create a tree control element.
CFTREEITEM	Used with CFTREE to populate a tree control element in a CFFORM.
CFTRY CFCATCH	Allow developers to catch and process exceptions in ColdFusion pages.
CFUPDATE	Updates rows in a database data source.
CFWDDX	Serializes and de-serializes CFML data structures to the XML-based WDDX format.

Appendix E Web server and Database Connectivity

This matrix lists the Web servers supported by the various versions of ColdFusion Server.	Express Windows	Express Linux	Professional Windows	Professional Linux	Enterprise Windows	Enterprise Solaris	Enterprise HP-UX	Enterprise Linux
Web Servers								
Microsoft Internet Information Server (IIS) v4.0	✓		✓		✓			
Netscape Enterprise Server (NSAPI) v3.5.1 and 3.6	✓		✓		✓	✓	✓	
Apache Web Server v1.3.6 and 1.3.9	✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Personal Web Server (PWS) v4.0	✓		✓					
O'Reilly WebSite (WSAPI)	✓		✓		✓			
Internet Server API (ISAPI)	✓		✓		✓			
Common Gateway Interface (CGI)	✓		✓		✓	✓		
Databases								
ODBC								
MERANT INFORMIX 7.x/9.x Driver				✓		✓	✓	✓
MERANT Sybase 11 Driver				✓		✓	✓	✓
MERANT dBase/FoxPro Driver						✓	✓	
MERANT IBM DB2/6000 Driver						✓	✓	
MERANT OpenIngres 1.x Driver						✓	✓	
MERANT OpenIngres 2.x Driver						✓	✓	
MERANT Oracle 7 Driver						✓	✓	
MERANT Oracle 8 Driver				†		✓	✓	†
MERANT Text Driver						✓	✓	
MERANT Microsoft SQL Server Driver				†		✓		†
MERANT MySQL Driver		†		†				†
Microsoft Access Driver	✓		✓		✓			
Microsoft SQL Server Driver			✓		✓			
Microsoft dBase Driver	✓		✓		✓			
Microsoft FoxPro Driver	✓		✓		✓			
Microsoft Visual FoxPro Driver	✓		✓		✓			
Microsoft Excel Driver	✓		✓		✓			
Microsoft Text Driver	✓		✓		✓			
FileMaker ODBC Driver	✓		✓		✓			
Native Drivers								
Sybase System 11					✓	✓	✓	✓
Oracle 7.3, 8.0, and 8i					✓	✓	✓	✓
Informix 7.3					✓	✓	✓	
DB2 Universal Database 5.2/6.1					✓	✓	✓	✓
OLE DB								
SQLOLEDB			✓		✓			
Microsoft.Jet.OLEDB			✓		✓			

Appendix F List of References

Programming Applications with the Wireless Application Protocol:
The Complete Developer's Guide
Mann, Steve, ISBN 0471327549, John Wiley & Sons, 1999

WAP White Paper
AU-System Radio AB, 1999

Developing Web Application With ColdFusion
Allaire Corporation, 1999

ColdFusion 4.5 White Paper
Allaire Corporation, 1999

Administering ColdFusion Server
Allaire Corporation, 1999

CFML Language Reference
Allaire Corporation, 1999

Using ColdFusion Studio
Allaire Corporation, 1999

The ColdFusion Web Application Construction Kit, third edition
Forta, Ben, ISBN 0-7897-1809-X, 1998

MSDN Online Web Workshop
http://msdn.microsoft.com/workshop

[1] Computer Sweden
Ogelid, Håkan, October 16, 2000, volume 18, issue 97