Developing Web Applications in a Mobile Computing Environment

JASON RAPP and JIANG B. LIU
Computer Science & Information Systems Department
Bradley University
Peoria, IL 61625, U.S.A.

Abstract: There are critical distributed computing processing issues in mobile computing systems due to the characteristics of mobile devices. In this research, we have developed web applications using J2EE technologies to address these mobile computing issues. The rapid development of wireless digital communication technology has created capabilities that software systems can be developed for distributed client/server computing. The falling cost of both communication and mobile computing devices (tablet computers, pocket PC, etc.) is making wireless computing affordable to both business and consumers users. Mobile computing is not just a `scaled-down' version of the established and well-studied field of distributed computing. The nature of wireless communication media and the mobility of computers combine to create fundamentally new problems in networking, operating systems, and information systems. Furthermore, many of the applications envisioned for mobile computing place novel demands on software systems. We have developed mobile distributed computing applications to study these challenge issues.

Key-words: Mobile Computing, Web Application Design

1 Introduction

When developing web applications in a mobile computing environment there are many different decisions have to be made [1-3]. These decisions are made based on the specific web application in how it will be used, where it will be used, and who will be using it; these along with many other variables go into deciding the underlying framework and design of a web application. In order to show the various decisions that have to be made as well as the benefits and detriments that these decisions can yield, we have developed several web applications with different underlying designs. Modern web applications are built upon smaller components that accomplish different tasks for the application as a whole. This creates a situation in which a developer must not only develop these multiple components but must also ensure that all the pieces will interact with each other. This communication allows the entire application to function. In order to alleviate a developer from this tedious task the Java 2 Enterprise Edition (J2EE) provides a variety of components that can aid in the design of a web application [4]. J2EE components are made to aid web applications that are built using a multitiered design model. Having standard multitiered architecture allows the J2EE components to be reusable for any conceived web application. A generic but common web application in a mobile system would have a multitiered structure similar to the figure below.

The web applications created has the common task of allowing a user to manipulate a database. From a remote location a mobile user will be able to obtain the client component. Through this client component the mobile user will be presented with an interface in order to connect to database found at different locations.

After establishing a connection the mobile user will be able to add, delete, and update data found on the database via the interface. The first design uses a two-tiered system, while the second and third designs employ the three-tiered paradigm.

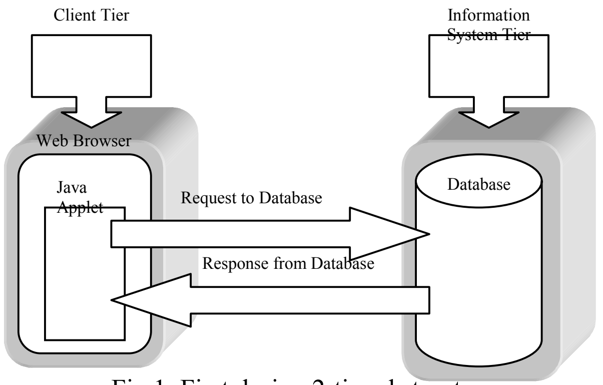


Fig 1. First design 2-tiered structure

In the first design no web components were used and thus no middle web tier is presented. In this construct a mobile user directs a web browser to a specific URL address. From this URL an applet is downloaded directly to the user's mobile device. This applet will not only give the user an interface to interact with but will also make direct connections to the database. In other words the client tier will connect directly to the information system tier. From this connection database manipulation requests are sent directly from the client component to the database.

The second design adds the middle web tier by running a J2EE compliant server which contains web components that will direct data flow. The client tier will involve an applet again just as the first design. However, this applet will be much “smaller” as it will not directly deal with the database manipulation requests. Instead the applet will simply send the request to the web tier components. In this design web components called servlets are used in the web tier. The web components will then relay these requests to the database located at the information system tier. The database will then send information back to the web component regarding the manipulations success or failure. The web component then sends this information back to the client tier’s applet.

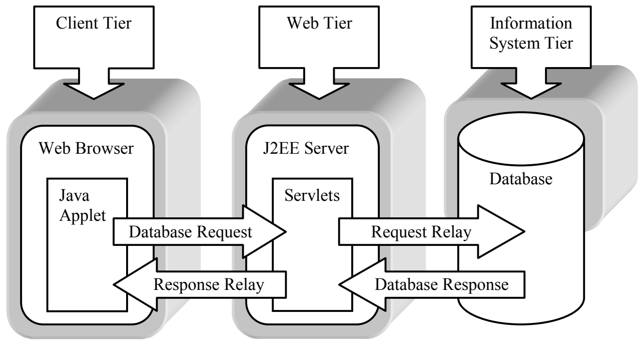


Fig 2. 2nd design three-tiered structure using servlets

In the third and final design the middle tier’s servlets are replaced with JavaServer Pages (JSP). These JSPs will also replace the applet used on the client side. When a user directs a web browser to a specific URL address they will enact the JSP on the web tier’s J2EE server.

This JSP will dynamically create HTML and send it to the user. This dynamic HTML will act as the interface for the user, thus acting as a client component. The interactions with the HTML will be sent back to the JSP on the web tier which will compute the user’s requests and send them to the database. As in the second design the database will then send messages back to the JSP. Once the JSP receives this message it is relayed back to the client tier via HTML generated by the JSP.

As these three designs show a web application in a mobile computing environment can be built using many different techniques and components. However, the same functionality can still be carried out using the different designs. Knowing that the same application may be built several ways it becomes the developers responsibility to use the correct design for the specific application. Of course lots of factors go into deciding what design is correct. The different components are better suited for certain factors than others. This is why the different components must be looked at individually while comparing the advantages of the component compared to the other components. Knowing which components to use in specific situations will allow a developer to create an efficient and robust mobile computing application.

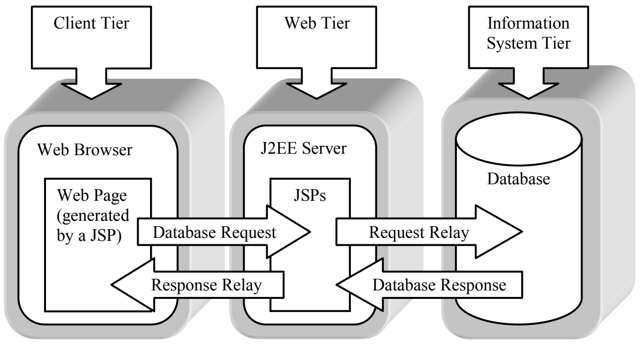


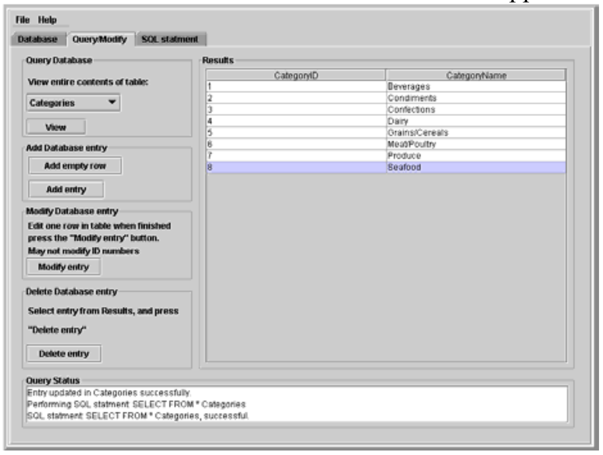
Fig 3. Third design three-tiered structure using JSPs.

2 Client Tier Design

The client component of a web application in a mobile computing environment could be considered the most import component. If the mobile user cannot interact or even understand their end of the application the backend design, no matter how well designed, will be obsolete. The end mobile user must be able to easily obtain the components he or she will interact with. Once the client’s components are obtained the user should not need to put a great deal of effort into running the application on his or her mobile device. When the user has the application up and running the developer must make sure that the application’s interface with the mobile user is clear and concise. The best way to achieve this is to develop a graphical user interface (GUI). The GUI must be easy for the user to use with little or no directions given to operate it. Of course the more advanced a GUI’s components are the more tools the developer will have to create a rich interface. The first two designs of the web applications used Java Applets to create a GUI, while the third design used JSPs.

The first advantage applets have over client applications using HTML is that they are not limited by the form widgets included with HTML. With a richer GUI, an application can become much easier for the mobile user to control it, thus allowing the user to operate the application as a whole more efficiently. Another advantage gained from using Java Applets is that they are downloaded and interpreted on the client side which will cause less strain on the server that the applet was downloaded from, since the client will actually carry out the computations generated from the applet.

Using Java Applets however does bring some disadvantages to the application. The major disadvantage of using an applet in a mobile computing environment is the download time needed for a client to obtain the applet's files. An applet must have all of its files located on the client's mobile device before it can run. Depending on the applet there could be a great deal of files to download. This download can obviously be very time consuming for the mobile client. Also, the rest of the page will load in the user's browser before the applet can be downloaded and ran. This can lead to the mobile user thinking that the applet is not working or will not load for them when in fact the download is just taking longer than the rest of the page. This download time is worsened by the fact that the applet files are not always cached locally depending on the browser. This means that each time the applet is viewed in the mobile client's browser, no matter how frequently; the applet files must be downloaded again. When a web application is being designed for a mobile distribution there is no telling the speed at which the client can download the applet. Obviously, an applet becomes unacceptable if clients need extensive time to obtain the applet. Another down fall of Java Applets is the possible unpredictability of the mobile client using the applet. A developer cannot assume that all mobile users will already have Java software installed or have the access or know-how to install the software to run the applet



CategoryID	CategoryName
1	Beverages
2	Condiments
3	Confections
4	Dairy
5	Grains/Cereals
6	Meat/Poultry
7	Produce
8	Seafood

```
Entry updated in Categories successfully.
Performing SQL statment SELECT FROM * Categories
SQL statment SELECT FROM * Categories, successful.
```

Fig 4. Screenshot of applet.

The next design used the J2EE component JavaServer Pages (JSP) to construct the client component. JSPs provide presentation for the client tier yet also provide logic for the web tier, which will be explained more in depth when comparing the web components.

The first advantage seen with JSPs over applets is the reduction in overhead needed to get the component up and running for the client. Instead of sending multiple files for an applet only a single HTML page needs to be sent, thus the mobile client is not bogged down by possible download times to start interacting with the application. Another advantage lies in the dynamic nature of JSPs. JSPs contain snippets of Java code they are granted a degree of logic. This allows a JSP to decide what HTML to generate under specific conditions allowing the returned HTML to change depending on the interactions made by the user, which would not be possible under pure HTML. This ability to create dynamic pages is extremely useful for a web application as all possible interactions can be handled under one dynamic HTML page as opposed to many static HTML pages in which a new page is brought up depending on the mobile user's interactions.

Of course as far as the GUI is concerned JSPs fall short of applets due to HTML's much to be desired graphical widgets. Since JSPs do generate HTML the only GUI components available to them are those offered by HTML. Using more primitive graphics leads to a more primitive GUI that will not be as robust or not be able to maximize the physical mobile device screen space. Without being able to make as rich of a GUI user comprehension of the GUI will be decreased, as the GUI becomes harder to navigate and use. Any inefficiency in using the interface will adversely affect the efficiency of the mobile user to use the web application as a whole. JSPs also carry one small disadvantage dealing with speed, when the JSP is accessed for the first time. Before a JSP can generate HTML to send to the client it must first be compiled as Java code, which is not done until the first access of the JSP. Thus, if a mobile user is the first to access the JSP page then he or she must wait for compilation and HTML generation, where only HTML generation needs to be waited on in subsequent accesses of the JSP. However, this is easy remedied as long as the developer is sure to test all JSP pages before deploying them to the public and thus causing them all to be compiled before anybody else has access to them.

3 Web Tier Design

Web components reside in a J2EE server on the host, when the server receives a request from a client, in this case an applet or web browser; it directs it to the appropriate web component. Once the web component handles the request it generates a response, usually an HTML page or pure data, and sends this to the J2EE server. The server then passes the response to the client that made the original request or passes on the data to another tier in the architecture.

The first design cut out web components completely. This was achieved by packaging the applet with the code that would connect and manipulate the database. With this architecture a mobile user's interaction with the applet results in a direct connection both to and from the database. With this design when a user downloads the applet they are not only downloading the graphical interface but downloading the business logic as well or the code that the applet uses in order to communicate with the database. This creates a "thick client" which will do all processing on the client's machine. The immediate advantage of this design is the lowered overhead for the server, after the applet has been downloaded. Once downloaded the applet has no more need for the server from which it came, since the applet can directly correspond with the database all from the user's mobile device. The server will have more resources free as the applet will not be continuously making requests to it, Without the middle tier the user will see faster response times as their requests will be directly sent to the database, instead of being sent to the server then redirected to the database, back to the server, and finally back again to the applet.

Although this design does have some performance advantages it defeats the purpose of J2EE components. The components need to be kept small and only achieve one task, and build an entire system when put together. This allows a logical separation between components as well as the application's programming logic. However, when the web tier is skipped the client tier not only supplies the presentation logic but the business logic as well. With no separation the web application becomes harder to maintain in the future since the one component becomes too large and complex. Also future upgrades in the system will become complicated. Since any changes to the business logic would require changes to the presentation logic as well since the business and presentation logic are connected to one component. Another disadvantage would be the time strain suffered by the mobile client when downloading the applet as mentioned previously. With the applet containing the business logic it becomes much larger in size than it would if it only contained the presentation logic. With this additional code and data the download time needed becomes a greater burden for the mobile client. Thus, the major disadvantage of Java Applets, their download time, is made worse.

The second design incorporated servlets in the web tier. The goal is to allow mobile clients and servers to communicate using the HTTP protocol, the same protocol used for regular web browsing. The first advantage of using servlets is that it makes it much easier to separate business logic from presentation logic by breaking it up into separate components. There is a very logical separation as the presentation is solely on the client side, while all the computations, or database access, is done at the server, and the two components: the servlet and the graphical applet are located on two different tiers. Separating the business logic from the presentation also provides the advantage of "thin clients." This means the client will only need to download a lightweight interface. Of course this is good for the mobile user as a smaller download is needed when compared to the first design.

There are not many disadvantages in using servlets in a three-tiered web application, as Java has become a very popular programming language to use on the server side of a mobile computing application. The only problem that could deter a developer from using servlets is the extra strain put on the server, as well as the extra overhead applied to the actual web application. Also, from a mobile user's perspective their requests must make three stops before they see the results, in this particular web application. First the user's request is sent to the server which then sends the request to the database which carries out the request then sends the result back to the server, which finally sends it back to the user.

JavaServer Pages (JSPs) were mentioned earlier from the perspective of the client; however, JSPs true power is shown from the web tier's standpoint. A JSP allows a developer to combine snippets of Java code that determines how the page constructs the dynamic HTML content. The JSP engine compiles the JSP into a regular Java servlet which is handed to the servlet container, which deals with this JSP turned servlet just as if it were originally a servlet. The container then executes the servlet when the J2EE server directs control to it. After the first time the JSP page has been called it is kept in the servlet container as a servlet, thus it only needs to be compiled by the JSP engine one time. Unique to JSPs when compared to other similar technologies, developers can create custom tags that can be used on the JSP. Custom tags are used in an HTML like syntax and execute Java code physically located on another file when executed. In other words Java code can be added to the JSP through a custom tag as opposed to adding snippets of actual Java code. In the final design a mobile user is presented with an HTML interface that was generated from a JSP. Every time a user wants to manipulate the database the request is sent to the JSP, now a servlet, on the server. Once the request is made it is handled in the exact same way as the pure servlet did from the second design.

Since JSPs are compiled into servlets and placed into the servlet container they share the same advantages of servlets. However, their unique design does create some new advantages. The biggest advantages for JSPs are brought by the custom tags, and how they help the developers. Custom tags immensely help the separation of presentation and business logic, which is always a goal when using

J2EE components. Since the JSP essentially combines the client and web tiers as HTML is generated while server side processing takes place a clear separation is needed. This separation is done when all the business logic is done in Java code within the custom tag, and all the presentation is accomplished in HTML on the actual JSP page. The ultimate goal would be to have a JSP page that contains only HTML and custom tags, and all the Java code execute by the JSP is done through the custom tags.

Again it is hard to find disadvantages in the JSP component from the web tier point of view, as the technology enhances Java servlet technology. They do however share the servlet disadvantage of adding extra computation and communication strain to the server. The other disadvantages of JSPs are not from the JSP component itself but the language in which they are written in, Java. All Java code is interpreted by the JVM on the machine it is running on while many other languages that have technology similar to JSPs compiles code into a language that the specific machines can understand. This means that Java runs on software while other languages run on hardware which is inherently faster. Thus JSPs may lose some performance when executed as opposed to other similar technologies written in other programming languages.

4 Web Application Security

With any program associated with the wireless network security must never be an afterthought. This is the case with any Java built mobile computing application where security is always a concern for both the mobile users and the developers. As a web application spans over the wireless network the mobile users will be suspicious of malicious programs from an unknown source, while the developer must reassure the mobile user that the application is safe to use. Security concern is a major aspect that must be considered when using applets. Applets have special security measures to protect the end user since the actual applet runs on the user's mobile device. Thus, the developer must take special steps before deploying the mobile computing application to the public that would not need to be done otherwise. When using Java Applets security can be accomplished in two different manners. The user can establish a security policy regarding applets for their local machine, or a developer may digitally sign their applet which essentially overrides the user's security policy.

Java security involving applets has evolved from two different methodologies. One idea is that all applets downloaded from a network are deemed untrustworthy and thus should not have any access to the local system's resources unless the machine owner explicitly grants certain access. The other school of thought is to have a developer digitally sign the applet. Since the digital signature uniquely identifies the developer the receiver of the applet may decide from the digital signature if they trust the developer or not.

The security policy must be configured specifically for the mobile computing applications. This is possible as the security policy is designed to be highly configurable to allow or disallow any type of action. The major advantage of configuring the security policy is that it allows the user to run the applet on their machine under their terms and conditions. Not only will it run but the mobile user can be assured that the applet will not step over the bounds that the user has set by the policy. Constructing the security policy personally the user does not need to completely trust the applet developer, since malicious code will be stopped through the policy. In the hands of an experienced user a security policy can be extremely customizable. This customization assures the mobile user that any applet downloaded from the wireless network may safely run on the mobile device based on his or her set policy.

Although the security policy system allows for a highly defined security system, it can be very difficult for novice users to setup the policy appropriately. It is very likely that the client will not have a good understanding of the security or may not know how to configure the policy correctly. Of course if the mobile user cannot adjust the policy correctly the web application will be not be able to run the way the developer intended as the client's mobile device will refuse to run the prohibited code. A mobile user must also know the specific privileges to grant the web application. Thus, the developer must find a way to inform the user exactly what privileges need to be granted for the applet to run. The user cannot simply trust the developer that the applet will not be malicious once the user grants the applet the specific privileges it needs. From the mobile users perspective configuring the security policy becomes very inefficient. The user now must go through a serious of tasks before the web application's applet will ever work, in commercial applications an applet this user-unfriendly applet would be unacceptable.

Another way to allow an applet, which needs local resources, to run on a client's machine is to digitally sign it. When a developer digitally signs an applet he or she is implicitly vouching for the code, and leaves it up to the end user to decide if the developer is trustworthy enough to run the applet. In order to sign the applet the developer creates a pair of keys one which is public and another that is private. The private key belongs solely to the developer and should not be shared, this key is used to encrypt or sign the applet code using the public-key encryption algorithm, while the public key is distributed to all those attempting to run the applet. The public key is used to verify the digital signature made by the developer. Now the user can tell exactly who created the applet and based on that can make a decision to either trust and run the applet or not allow the applet to run on his or her machine. The clear advantage of signing an applet is to completely alleviate the mobile user from configuring their system to run the one program. The only work for the user to do now is check the signer's credentials and base their trust, their decision to run the applet with full access, based on that. This alleviates the trouble of informing the mobile user of exactly how to run the applet, and allows the user to run it without needing to know how to construct their security policy. A trusted signed applet also benefits from the fact that it will run the same on any given mobile device. Under the security policy method a client's mobile device will allow the applet different privileges than another client mobile device if the security policies on both devices are different. However, if a signed applet is accepted the developer specifies which privilege the applet will need.

Unfortunately the mobile user bears most disadvantages of a signed applet. Once the applet is received and the signature is shown to the client they must verify the developer's credentials to be sure he or she can safely run the applet. Normally the mobile user will not trust an applet even if the developer digitally signs it. A mobile user cannot always trust a developer just because the developer told the user the applet is safe to run. This is why the user must look into the developer's credentials. Of course this could add a great deal of work and stress to the mobile user as he or she must now research the developer and decide if the applet is truly safe to run. If the developer instead, simply told the user exactly what security permissions the applet needed and the user adjusted the security policy accordingly the user would not need to trust the developer. The mobile user will also be weary since once an applet is declared trusted it can have full access to the systems resources. This is just another variable that would cause a user not to trust a developer, making another barrier for the web application.

5 Conclusion

Wireless technologies are transforming the distributed computing at organizations for their business applications with the lightweight, ultra-mobile technology and software system. These new development enables users to be fully productive from anywhere. In this research, we can clearly see a developer must make many decisions on what components to use when building mobile computing applications. There is clearly no silver bullet as to what architecture will provide the best results. Different situations and needs for the mobile computing application drive the decisions on what components to utilize. Some applications will solely be used in a wireless local area network while others will be used worldwide over the Internet. Some environments allow the developer to have access to the client mobile device to be certain each device will be able to run the application, and other times the developer will not be able overview each mobile device that will use the application. Under some conditions many mobile users will be accessing the application at the same time requiring a very powerful server. While in other situations only a simple computer is needed as user traffic is controlled so not as many users may use the program simultaneously. There are many different variables that play into picking the design for a mobile computing application. These design considerations will determine the performance of the mobile computing systems.

References:

[1] Jiang B. Liu, Dairui Chen, and Srihari Muthyala, "Web based Enterprise Computing Development Strategies," Proceedings of the 2004 International Conference on Internet Computing, Las Vegas, Nevada, June 2004, pp. 641-647.

[2] T.F. Abdelzaher, K.C. Shin and N. Bhatti, "Performance Guarantees for Web Server End-Systems: A Control-Therоretical Approach," IEEE Transactions on Parallel Distributed Systems, Vol. 13, NO. 1, 2002, pp 80-96.

[3] Karen Holtzblatt, "Designing for the Mobile Device: Experiences, Challenges, and Methods," Communications of the ACM, Vol. 48, NO. 7, 2005, pp 33-35.

[4] Jiang B. Liu, Web based Enterprise Computing Development using J2EE, Industrial Information Technology Handbook, Edited by Richard Zurawski, CRC Press, 2005, pp. 1-30.

[5] Jiang B. Liu and Pavan Manemela, "Internet2 End-to-End Performance Tuning for Distributed Computing Applications," Proceedings of IEEE International Conference on Industrial Technology, Hong Kong, December 2005, pp. 587-592.

[6] Srihari Muthyala and Jiang B. Liu, "Developing Internet Computing Applications using Web Services," WSEAS Transactions on Communications, Issue 1, Vol. 3, June 2004, pp. 116-121.

[7] Tim Hill and Jiang B. Liu, "Develop Web Application with XML and Java," Proceedings of the 18th International Conference on Computer and Their Applications, Honolulu, Hawaii, March 2003, pp. 434-437.