7-2006

# A Framework for Developing Wireless Mobile Online Applications

Chong-wei Xu
*Kennesaw State University*, cxu@kennesaw.edu

# A Framework for Developing Wireless Mobile Online Applications

Chong-wei Xu
Computer Science and Information Systems
Kennesaw State University
cxu@kennesaw.edu

**Abstract**
Online applications based on the HTTP protocol are shifting from wired networks, known as Web applications and Web services, to wireless networks, known as wireless mobile online applications, due to the rapid growth of mobile devices, such as Personal Digital Assistants and cell phones. Among the enabling technologies, J2ME is the dominant and the most potential one for building up these wireless mobile online applications. This paper presents a framework that makes the modeling, implementation, and maintenance of wireless mobile online applications intuitive and easy, especially for students and beginners.

**Key words**
Mobile/Wireless Computing, Web Applications, Software Frameworks, Component-Based Software Engineering.

## 1. Introduction

Wireless communication eliminates the limitations of location and time. Mobile devices, such as, Personal Digital Assistants (PDAs) and cell phones, plus the wireless networks have provided a solid foundation for the ubiquitous computing. Today, the number of people accessing the Web through wireless devices has been larger than the number of people who access the Web through desktop computers. Just look at cell phones, in 2002, there were 400 M cell phones and its number grew to 1 billion by 2003 [Cell02]. In 2006, nearly 1 billion new cell phones will be sold [Mane05]. Cell phones plus wireless communication have been part of our daily life. Definitely, they are not only available for voice talks but also are the most suitable platform for accessing online services through HTTP protocol, such as browsing Internet, doing business through mobile e-business (m-business), accessing scientific data, and so on.

Wireless mobile online applications are similar with wired Web applications in terms of three-tier architecture, server side programming, and system-level supports. The difference between them mainly lays on the client side programming. Due to the limited resources of mobile devices, especially cell phones, some mobile devices do not have Web browser installed. Therefore, the communication between the client and server takes the form of end-to-end, that is, a specific client program should be implemented and deployed to an individual cell phone. There are different enabling technologies, such as i-mode, Wireless Application Protocol (WAP), XHTML, and J2ME (Java 2 Micro Edition) [Deit02]. Among them, the J2ME platform is getting popular due to the fact that the majority of mobile devices support J2ME and the majority of IT people (professionals and students) have been trained with Java programming.

Any software development goes through modeling, implementation, and maintenance processes. This paper proposes a framework for making the modeling, implementation, and maintenance of end-to-end mobile online applications based on J2ME intuitive and easy. Section 2 briefly introduces the architecture and APIs of J2ME. Section 3 describes the graphical language MVC diagram for modeling the mobile online applications. Section 4 presents the layered component structure for implementing the model. Then, section 5 further explores that

Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)
0-7695-2613-6/06 $20.00 © 2006 IEEE

the framework makes the maintenance much easier because of its support of the Open-Closed principle (OCP). Finally, the conclusion and future work are drawn in section 6.

## 2. The J2ME architecture and APIs

Java 2 introduced three editions that are J2EE, J2SE, and J2ME as shown in Figure 1. The J2ME supports programming in mobile devices, such as PDAs and cell phones. The architecture of J2ME consists of two parts: configuration and profile. Due to the fact that there are many different types of hardware in mobile devices, the Connected Device Configuration (CDC) and Connected, Limited Device Configuration (CLDC) can only provide common part of APIs and leave device specific needs (features) to so-called Mobile Information Device Profile (MIDP). In other words, the configuration defines the capability of J2ME while the profile adds additional functionalities for handling the new breed of mobile devices.

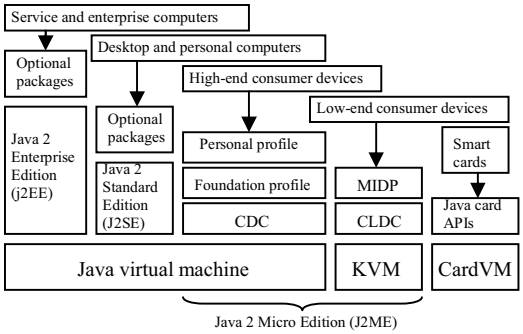


Figure 1. Three editions of Java language

The APIs provided by J2ME forms a class hierarchy as shown in Figure 2 [Well04]. They are categorized as high-level and low-level User Interfaces. The high-level APIs takes the Screen class as its root, which has four sub-classes, namely List, TextBox, Form, and Alert. Among them, the subclass Form is a container that may contain any objects instantiated from all subclass of the Item class, such as DateField, TextField, ImageItem, etc. These high-level APIs support a nice screen display with multiple functionalities. However, they are not flexible enough for creating more customized look-and-feel. The low-level APIs comes in handy. The major class in low-level APIs is the Canvas, which shares the same superclass Displayable with the Screen class. That is, the Canvas class has the similar behavior as the Screen class and can be used as a container or sub-container. Specifically, the Canvas class is designed for graphics with a built-in paint(Graphics g) method. With the Canvas class, all classes related with graphics, such as color, font, geometric shapes, images, and so on can be displayed as GUIs.

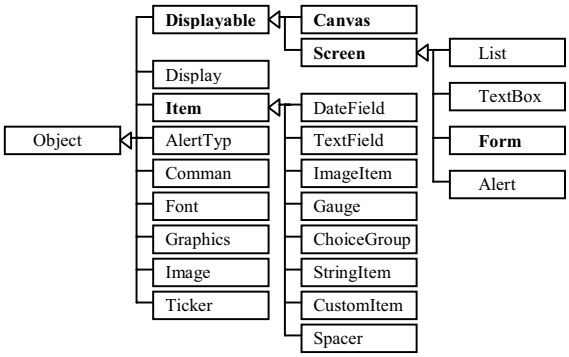


Figure 2. The class hierarchy of User Interfaces.

A wireless mobile online application is based on the three-tier architecture as shown in Figure 3. The HTTP is the communication protocol to bridge the client and the server. On the server side, the servlets accept the user's requests, dispatch the requests to other components for validation, manipulation, confirmation, and finally generate the corresponding responses. The JavaBeans are components that access the database for query, update, or insert data. The software components on the client side are based on the class MIDlet that is the superclass for supporting Application Manager of J2ME programs. The MIDlet implements the codes that will be deployed onto cell phones.

Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)
0-7695-2613-6/06 $20.00 © 2006 IEEE

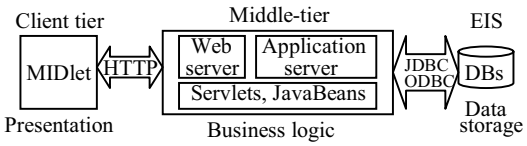


Figure 3. The three-tier architecture

## 3. Modeling with the MVC diagram

Based on the MVC (Model-View-Controller) model [HoCo05], [HDFW03], we have proposed a graphical language named MVC diagram for modeling, implementing, and maintaining Web applications [Xuch03]. The MVC diagram has been successfully applied to model wired Web applications [XuWH04], [Xuch05]. The MVC diagram can also be applied for modeling wireless mobile online applications. Figure 4 shows a MVC diagram for a J2ME application, named mathTest (Mathematics test). As the figure shows, a MVC diagram uses five symbols. A circle represents a screen display, such as "Welcome", "Login", "AQuestion", and so on. A pair of parallel lines refers to a database table, such as "login" table, "questions" table, etc. Arrows with labels exist between circles are known as event arrows, such as the arrow with the label "URL", "Submit", and so on. An arrow points to a circle from a pair of parallel lines indicates a query on the database table; points to a pair of parallel lines from a circle implies an update or insert operation on the database table. A small solid circle is the starting or the ending point.

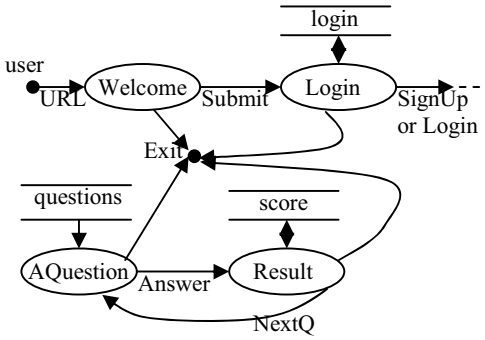


Figure 4. The MVC diagram of the wireless online application mathTest.

The semantics of the diagram can be briefly described as follows. Besides displaying welcome information to the user, the "Welcome" screen also provides a choice as either sign-up for new users or login for registered users. When the user makes a choice and clicks the "Submit" command the choice will be sent along the arrow "Submit" to the "Login" screen. Depending on the user's choice, the "Login" screen either allows a first-time user to sign-up his/her username and password or allows a registered user to login by providing his/her username and password. For the first-time user, the sign-up information will be accepted and sent to the server side. The data arrow points toward the database table "login" above the "Login" screen represents a JavaBeans component that inserts the username and password into the table "login". If the user expressed to login, his/her username and password will be sent to the server side to be verified. That is, the arrow points to the circle "Login" from the database table "login" serves a query that accesses the database table "login" and brings the information from the database to the corresponding JavaBeans component for the verification of the user's login information. The verification will return a true or a false. The true means the user is authorized to access the application, the application enters the next circle marked as "AQuestion". This screen displays a question with a set of multiple choices. The question comes from the database table "questions". When the user makes a selection and issues the command "Answer", the MIDlet sends the user's selection to the server side. The components in the server validate the answer, calculate the score, update the table "score", and send the related information to be displayed on the screen "Result" for the user. The user can use the "NextQ" (next question) command to continue the test and loop back to the screen "AQuestion". The remaining part of the diagram expresses that the user can exit from the test from any screen by clicking the command "Exit".

The MVC diagram models the entire application. It decomposes the application into the Model (database tables represented by the pairs of parallel lines; JavaBeans that access the database tables represented by the data arrows), the View (screen displays represented by circles), and the Controller (user actions represented by the event arrows with labels). The diagram clearly indicates all required components including all database tables, commands, screens, and their dynamic behaviors, as well as the enabling technologies, such as servlets, JavaBeans, etc. By using the MVC diagram, we can easily model a complicated mobile online application, visualize the interrelations among all components, and count all resources needed for the application. All these together definitely benefit the project management and teamwork a lot.

## 4. Implementation with reusable components

The implementation of a mobile online application consists of the server side and the client side. The server side is based on servlets and JavaBeans. The client side is based on a MIDlet. Considering the limited resources of mobile devices, implementing a real thin-client is the strategy for making a successful mobile online application, that is, moving all computations to the server side while leaving the display and commands only on the client side. Figure 5 depicts a programming template for the implementation. On the client side, the user issues a request by issuing a command, which sends a command and related data to the server side. Then the client side creates a new screen to display the response returned by the server side. In the new screen, the user can issue a new request by clicking a new command. Correspondingly, the server receives the request, processes the request, generates and sends a response to the client side. A different request needs a different processing. Consequently, the most comprehensive portion of the server side is the request-process, including validation, manipulation, and confirmation, as well as database accessing.

From the comparison of the programming template in Figure 5 with the MVC diagram in Figure 4, we can see that issuing a request corresponds to an event arrow (the Controller), processing a request corresponds to a data arrow (the Model), and displaying a response corresponds to a circle (the View). A request-response pair contains these three parts and forms a unit. Each unit is implemented by reusing component libraries in the layered component structure and each unit can be plug-and-play into the system as shown in Figure 6.

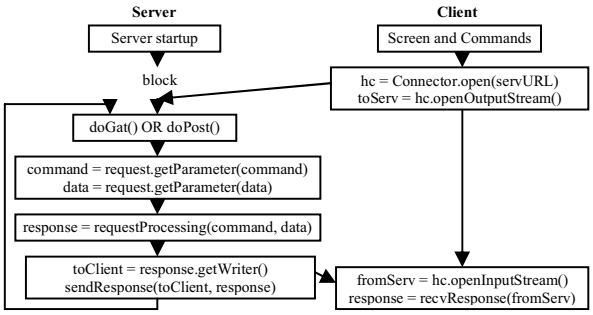


Figure 5. The programming template of wireless online applications

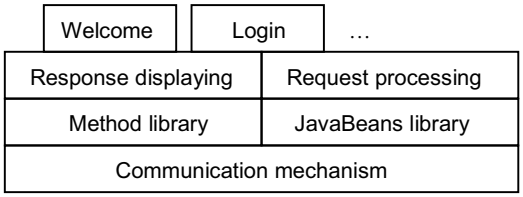


Figure 6. The layered structure of components.

In the layered component structure (Figure 6), the communication mechanism layer on the bottom refers to the protocol HttpConnection shown in Figure 7. CLDC network connections are based on the Generic Connection Framework (GCF). GCF defines that a certain type of devices must support a certain type of communication protocols. Due to the fact

that cell phones only have limited resources, GCF requires that all cell phones must support HTTP protocol and widely support Socket protocol. Thus, the HttpConnection interface is the most popular connection interface for implementing mobile online applications. It defines a variety of methods for network connections and data operations. GCF also commands that any connection must use Connector's open() method, for example for creating a HTTP connection using Connector.open("http://..."); for creating a socket connection using Connector.open("socket://…"); and so on. The second layer and the third layer in the structure are divided into two parts. The left part refers to the client side that displays a response with the method library. The right part refers to the server side that involves request processing using the JavaBeans library. Based on the component-based software engineering, both the method and the JavaBeans library contains reusable components. For example, on the client side, the method library includes high-level methods, such as getDataFromServer(), getImageFromServer(), getDataLineFromServer(), and low-level methods, such as getData(), postData(). These static methods are defined in a class named Util (Util.java). On the server side, when the doGet() or doPost() method receives a request, the request processing implements the business logic defined by the application for processing the request command and the data. The JavaBeans library contains classes including DBConnection, queryDBBean, updateDBBean for accessing database tables in order to support the servlets for processing the requests. These JavaBeans are formally designed and flexible enough to accept different SQL statements for handling connection, query, and update operations. Both the method library and the JavaBeans library can be reused for implementing other wireless online applications.

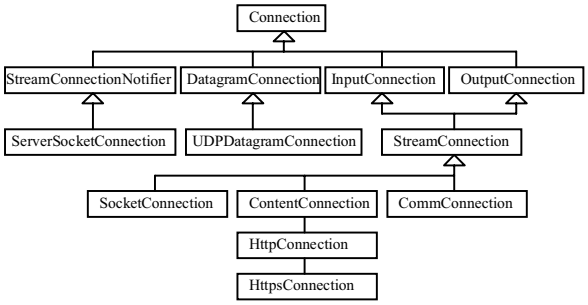


Figure 7. Connection hierarchy.

## 5. Maintenance with the OCP principle

The OCP (Open-Closed Principle) is the first principle in developing large software systems. The principle says, "Software entities should be open for extension, but closed for modification" [Meye88]. That is, a software system should be able to incorporate extensions for satisfying new requirements in order to increase software's adaptability and flexibility but without modifying its existing source code. The OCP principle is also very important for software maintainability and reusability [Mart00], [CoMK99]. The MVC diagram can easily accommodate extensions and modifications because any extension can be modeled as a request-response pair and inserted into the MVC diagram as a independent unit that consists of an event arrow, a circle, a data arrow and a database table. Its implementation is supported by the component structure as a plug-and-play unit.

For example, suppose the mathTest application would add a new screen that displays the category of the mathematical questions for different levels of tests, it is simply to add a circle "Category", a database table "categories" with the data arrow, and an event arrow with the label "Que" into the extended MVC diagram as Figure 8 shows. Then apply the reusable components to implement it.

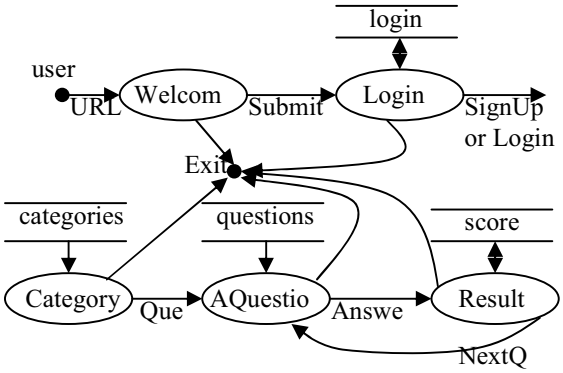


Figure 8. An extended MVC diagram.

The added unit is a complete encapsulated unit that won't interfere with the existing codes. The only thing needs to be done is to change the destination of the links "SignUp or Login" and "Que".

## 6. Conclusion and future work

The MVC diagram models a wireless mobile online application. The layered structure of components shown in Figure 6 is derived from the MVC diagram and supports the implementation of the model. They together provide a graphical modeling language for the specification of an application, guide the entire modeling, implementation, and maintenance processes, and support reusable components. This framework decomposes a complex online application into modules. Each module is a plug-and-play unit. The components in the libraries (the method library and the JavaBeans library) can be directly called and used. The MVC diagram facilitates the OCP principle, which makes the software with a strong adaptability and a high degree of flexibility. All these make a special sense to students and beginners.

We will further apply the framework to more practices, improve its structure, and enrich the libraries. Several more complicated applications, for example, an online bookstore, an online ticket office, an online IT terminology, an online conference submission systems are being developed for the goal.

## References

[Cell02] http://www.cellularphonenews.com/ebook/overview.html

[CoMK99] Peter Coad, Mark Mayfield, and Jon Kern, "Java Design – Building Better Apps and Applets", Prentice-Hall, 1999.

[Deit02] Deitel and Deitel, "Wireless Internet and Mobile Business—How to Program", Prentice-Hall, 2002.

[HDFW03] T. Husted, C. Dumoulin, G. Franciscus, and D. Winterfeldt, "Struts in Action", Manning, 2003.

[HoCo05] C.A. Horstmann and G. Cornell, "Core Java 2 Volume I – Fundamentals", 7/e, Sun Microsystems Press, 2005.

[Mane05] Tech guru dials into gaming's social side, By Kevin Maney, USA TODAY, Posted 12/11/2005 9:51 PM.

[Mart00] Robert Martin, "Design Principles and Design Patterns", www.objectmentor.com, 2000.

[Meye88] Bertrand Meyer, "Object Oriented software Construction", Prentice-Hall, 1988.

[Well04] Martine J. Wells, "J2ME Game Programming", Thomson Course Technology, 2004.

[Xuch03] C.-W. Xu, "A MVC Diagram – A Visualization Tool for the MVC Model", Proceedings of the first ACIS International Conference on Software Engineering Research and Applications (SERA'03), Crowne Plaza Union Square Hotel, San Francisco, June 25-27, 2003, pages 82-86.

[XuWH04] C.-W. Xu, C. Wang, and B. Huang, "WCBS: An Case study of MVC Diagrams", Proceedings of the ACIS 5th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2004), People's Palace Hotel, Beijing, China, June 30-July 2, 2004, pages 287-291.

[Xuch05] C.-W. Xu, "Applying the MVC Diagram for Building Web Applications in Struts", Proceedings of the 2005 International Conference on Internet Computing (ICOMP'05), Las Vegas, June 27-30, 2005.