A MOBILE-AWARE BUSINESS LOGIC CONTAINER

Robert Steele
University of Technology, Sydney
PO Box 123 Broadway NSW 2007 Australia

Elaine Lawrence
University of Technology, Sydney
PO Box 123 Broadway NSW 2007 Australia

ABSTRACT

Currently providing such features in mobile applications as the sending of lower-bandwidth images when a client device is in a low-bandwidth access area, the business logic to display location-based services to users when they are in the relevant vicinity and the determining of device location from the network must be manually programmed in the mobile application's business logic. However these features are common to many mobile applications and involve repeating similar code in different applications. In this paper we propose a mobile-aware business logic container that provides these features inbuilt in the container for mobile applications, freeing application developers from duplicating such code many times.

KEYWORDS

Mobile applications, business logic, e-business, container

1. INTRODUCTION

The complexity of Web applications is a driving force for the adoption of a container-based approach [11] for non-mobile computing Web applications. Containers provide an execution environment for code and importantly provide many aspects of required functionality without requiring duplicated coding effort by individual application developers.

Mobile applications similarly also can have a high level of complexity if not higher than Web applications. By mobile applications is meant (centralized) Web server-based applications that can be accessed by multiple mobile clients from multiple locations. Such mobile applications allow the accessing of the same application as the user roams through different locations. We do *not* mean applications that can migrate from one host to another for execution. Mobile applications have many aspects of Web applications but in addition have a number of further characteristics. These further characteristics include:

- Mobile clients will as a norm, be used from continually differing locations.
- As clients move, their type and bandwidth of connectivity will typically vary
- Services will in some cases be location-based
- There is a greater heterogeneity of client device platforms

These characteristics can add greater complexity to mobile applications. However, as they are common to many mobile applications it is possible to move some aspects of mobile applications out of the business logic and into a mobile-aware business logic container. In this paper we propose a mobile-aware business logic container and describe how it can provide application developers with built-in functionality for a number of the aspects of mobile applications listed above.

2. BACKGROUND

A well-known example of a container-based specification is Java 2 Enterprise Edition (J2EE) [16]. J2EE is a specification (not a product) that provides a component model, container requirements (containers are an execution environment for components) and a set of standard services e.g. database, transactions etc.

A container can be divided into four parts [10]:

- Component contract – application components must be developed according to a defined framework
- Container service API – allows access to various services e.g. JNDI, JMS
- Declarative services – a declarative service is a service performed by the container for us. A deployment descriptor describes the contract between the container and the component
- Other container services – e.g. resource pooling

In this paper we introduce declarative services and some component contracts specific to mobile applications. Deployment descriptors, for the specifying of declarative services, might typically be written in XML [2, 3]. XML provides an appropriate format for specifying declarative services in a container as it is totally platform-independent and as such supports the platform-independence of an overall container-based specification.

In addition, the ability to transform high-level representations into XML-based representations [5, 6] provides a powerful way to transform from a high-level model into a deployment descriptor that provides implementation-independent configuration for an application.

There are a number of advantages [13, 14] to implementing mobile services as XML Web services [12, 19]. These include providing interoperability and support for automated communication.

3. MOBILE-AWARE CONTAINER

We assume a mobile computing architecture where a mobile application is deployed at a central server hosting a mobile-aware business logic container. Connectivity between mobile client devices and this central mobile application can varyingly be via GPRS [8], WiFi [17], Bluetooth [1] or other wireless technologies. This mobile application may provide a large number of services, many of which may be location-based services. They are location-based in that their availability to a client can vary with the location of the mobile client and their behaviour is a function of the location of the client accessing them.

The mobile-aware business logic container hosts the mobile application. As with the current J2EE specification in respect to the management of persistence in a container environment, the choice can be made between container managed functionality or bean (or component) managed functionality. Similar options can exist with a mobile-aware container for mobile applications. In this case there are three options:

- Container managed mobile functionality
- Container managed mobile functionality with declarative specification
- Component or “bean” managed mobile functionality

In the first case, the container will automatically provide particular mobile application functionality with no input from the mobile application developer. In the second case a deployment descriptor must be provided that declaratively specifies aspects of the mobile application. In the third case the mobile application developer will provide the mobile business logic manually.

3.1 Container support for varying connectivity bandwidth

As mobile devices roam they will pass through areas with varying connectivity type and bandwidth. For example a device may pass from the vicinity of a WiFi access points into an area providing just GPRS connectivity. In these different areas, the available bandwidth affects the size of images or other multimedia files that should be sent to the mobile device. In current mobile applications, to achieve this dynamic selection of file versions, requires the writing of explicit business logic that contingent upon the available bandwidth sends an image of the appropriate size. However this developer effort will be repeated by many developers and many times by a developer of the same application. That is, some conditional sending of images or other multimedia files will need to be coded for each such image or file sent by the application.

A mobile-aware business logic container can remove this effort from the developer. The mobile container when it identifies an image or multimedia file embedded in a mobile application HTML/WML page can auto-generate lower-bandwidth versions of the image or file. This auto-generation of file "versions" is done just once for each image prior to any page requests – at application deployment time. When a mobile device requests an image or multimedia file the container will interpret this request "behind the scenes". It will detect the bandwidth currently available to the mobile device requesting the image and will redirect the HTTP request, invisibly to the developer, to the appropriate version of the multimedia file and hence serve the version of the file that is appropriate to the bandwidth available.

Also by drawing this mobile-specific behaviour into the container the presentation logic of the application remains disentangled from the code relating to varying connectivity and can be changed without requiring recoding of this varying connectivity behaviour. This container behaviour could be fully automated or could be specified by a deployment descriptor which would be represented in XML. It is possible to generate this deployment descriptor from a conceptual model.

In addition, by separating this behaviour out into a deployment descriptor it becomes possible to re-use presentation components developed for non-mobile applications in mobile applications without modification.

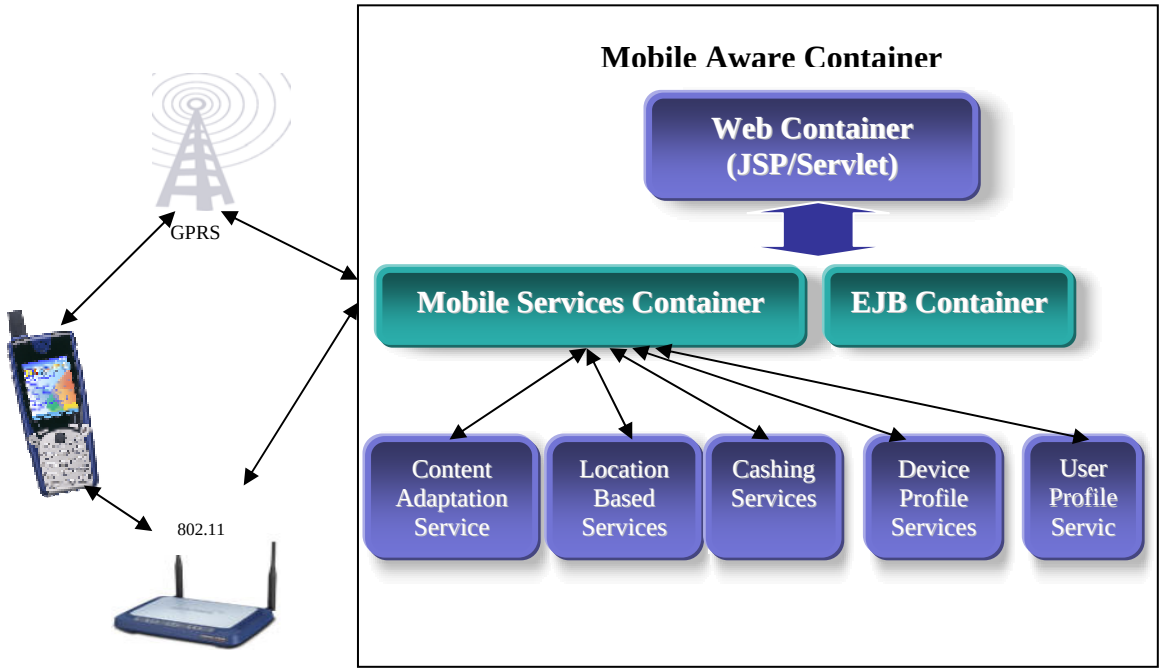

Figure 1. Mobile-aware container architecture

3.2 Container Support for Location-based Services

A second functionality that can be supported by a mobile-aware container is in relation to location-based services. By location-based services is meant services that only become available to a client when the client is in a specific physical area or location and/ or are a function of the client's physical location. Many of the services of interest in mobile computing systems are location-based services.

Consider the scenario where a centrally located mobile application, providing many location-based services, is being accessed from a roaming mobile device. Currently the developer of a mobile application would need to explicitly code the display of different services to mobile device users based on device location. When services are added or removed the business logic dealing with the selection and display of services would need to be altered and re-compiled; when the region of relevance of any service or services is changed the business logic dealing with the selection and display of services would also need to be altered and re-compiled. This is because in current systems such selection and display logic is commingled with the

other business logic of the application. This coding overhead in selection and displaying of services to devices within the relevant area of applicability will be present in all location-based mobile applications.

In many cases it will be desirable however to be able to reconfigure the particular areas in which a location-based service is displayed to users, without having to change business logic code or recompile other parts of the application. Here we propose to achieve this by having the locations of relevance of services, specified declaratively in a deployment descriptor. This for example would have an XML syntax. This deployment descriptor, in the simplest case, would specify circular regions in which each service is relevant via specifying an "epicenter" and radius for each service (see Figure 2).

When a request comes from a mobile device, for a list of location-based services relevant to the device at its current location, the container searches through the deployment descriptor information checking for services for which the mobile device is located in their area (circular region) of relevance. Once the set of relevant location-based services is ascertained in this way the container automatically generates an interface to display them to the device user. In this way, the mobile application developer does not need to create code for selecting and displaying location-based services.

Via the use of a deployment descriptor it also becomes possible to easily reconfigure the mobile application without having to alter code and recompile – all that is required is an alteration to the deployment descriptor and the redeployment of the application. For example in Figure 2, the BigTopCircusGuide could give information about the BigTop Circus to mobile devices within 200 metres of where the circus is currently situated. If the BigTop Circus was to move just the deployment descriptor would need to be altered to reflect the new location and the application redeployed.

```
<?xml version="1.0"?>
<service-locations>
       <service>
              <name> RocksRestaurantGuide </name>
              <latitude> -33.8614 </latitude>
              <longitude> 151.2569 </longitude>
              <radius> 1000 </radius>
              <!-- radius in metres -->
       </service>
       <service>
              <name> BigTopCircusGuide </name>
              <latitude> -33.8603 </latitude>
              <longitude> 151.2566 </longitude>
              <radius> 200 </radius>
              <!-- radius in metres -->
       </service>
</service-locations>
```

Figure 2. Location-based service deployment descriptor snippet

It is possible to separate these display choices from the service functionality/ business logic itself and from other presentation logic. Once again this supports the re-use in mobile applications of business and presentation logic components used to build non-mobile applications.

It would also be possible to plug into the application server different modules that support different schemas for specifying the regions services are relevant in and how to calculate whether a given device location is within a region. This allows for extending the application server/ mobile-aware container to support more complex region specification techniques. Once again this could be carried out without requiring change to other business logic of the mobile applications.

3.3 Container Support for Location-parameterized Service Invocations

In Section 3.2 container support for selecting and displaying of location-based services relevant to particular geographic regions is discussed. While in some cases services may only need to be displayed in certain regions e.g. a service to search a shopping mall might typically be just displayed to devices in that mall, in other cases services will provide information relevant to many/most areas but will provide information based

on the inputting of the device location as an input parameter. That is their output will be a function (along with other inputs possibly) of the physical location of the mobile client device.

In this section we propose how such location-based services can also be supported by the mobile-aware container. This will require that mobile application developers create services/ classes that implement a standard interface. For a Java example of such a standard interface see Figure 3.

```
public interface Localized{

       void inputLocation( GeoLocation loc);
}

public class ExampleServiceLocalized implements Localized{

       GeoLocation location;

       void inputLocation( GeoLocation loc){
              this.location = loc;
       }

       // location-based service specific methods
}
```

Figure 3. Java example of a standard interface for a localized service

The application developer would construct a service/ class such as that shown in Figure 3. Within the code of the service specific methods the location variable can be accessed. At run time, if any of the services defined in this way are called by a mobile client the container will:

1. request the location of the mobile client from the network
2. call inputLocation to set the value of the location variable
3. call the service/ method requested by the mobile client

In this way the application developer will be spared from writing code to determine device location from the network before calling a location-based service and be able to obtain the most up-to-date information on the location of a mobile client device.

In addition obtaining the location of the client can be complex. The mechanism for determining location in a WiFi network is different from the techniques used in a GPRS network and specific to the network technology and even vendor product involved. From the application developer's perspective, they want to be able to write location-based service code without having to deal with the network technology-specific approaches to determining location. As such it is appropriate that the container provides as a standard service the hiding of such location determination code.

4. DISCUSSION

Moving functionality into the mobile-aware container has the following advantages:

- the complexity experienced by developers is decreased
- recurring code found in mobile applications is moved to the container thereby avoiding developer duplication of effort
- Web application business and presentation logic components can be re-used in mobile applications without modification
- components can be re-used by different mobile applications without recompiling and just re-configured via deployment descriptors

In relation to any of the three areas of use discussed in Section 3, the application developer can specify which of the three modes of development they wish to use; container managed mobile functionality; container managed mobile functionality with declarative specification; or component or "bean" managed mobile functionality. That is, in the third option, the developer can opt to write all aspects of the business logic themselves – this option may be preferred when greater development flexibility is the priority.

The technique for using a deployment descriptor to provide conditions for which services to display at which location can be generalized. Ontologies [20] could be used to generalize relationships of how the presence/ occurrence of a certain concept e.g. a user is at a train station, a user is running, implies the display of a certain set of services (annotated with some other concepts that are related in the ontology) to this user. By using a deployment descriptor these relationships can be very easily changed without having to change the other business logic code.

5. CONCLUSION

Mobile applications like, Web applications, can be extremely complex. A common contemporary architecture for Web applications is a container-based architecture that places common functionality and services in a pre-written execution environment for the application (a container) thereby decreasing the complexity faced by the application developer amongst other things. In this paper we propose a mobile-aware container that achieves a similar result for mobile application developers. In particular we look at its utility for three particular common aspects of mobile applications: varying bandwidth available to mobile clients, location-based services and the determination of client location from the network.

REFERENCES

[1] Bluetooth Specification. http://www.bluetooth.com/pdf/Bluetooth_11_Specifications_Book.pdf, February, 2001.

[2] Consortium, W. W. W. 2000. Extensible markup language (XML) 1.0. Avail. at http://www.w3.org/ TR/REC-xml

[3] Consortium, W. W. W. 2001. XML Schema Part 0: Primer. Available at http://www.w3.org/TR/ xmlschema-0/.

[4] Dillon, T., Tan, P. Object Oriented Conceptual Models. Prentice Hall. Inc. 1993.

[5] Feng, L., Chang, E., Dillon, T. A semantic network-based design methodology for XML documents. ACM Transactions on Information Systems (TOIS), Volume 20, No. 4, 2002, pp 390 – 421

[6] Feng, L., Chang, E., Dillon, T. Schemata Transformation of Object-Oriented Conceptual Models to XML. Intl. Journal of Computer Systems Science & Engineering, 1, 45-60, 2003.

[7] Fensel, D., Angele, J., Decker, S., Erdmann, M., Schnurr, H.-P., Staab, S., Studer, R., and Witt, A., On2broker: Semantic-based access to information sources at the WWW, in World Conference on the WWW and Internet (WebNet99). 1999: Honolulu, Hawaii.

[8] GPRS Platform. Available at http://www.gsmworld.com/technology/gprs/index.shtml.

[9] Hendler, J. Agents and the Semantic Web. IEEE Intelligent Systems. March/April 2001 (Vol. 16, 2).

[10] Perrone, P., Chaganti. V. Building Java™ Enterprise Systems with J2EE. Sams Publishing, 2000. Paperback edition - 1500 pages book & CD-ROM. ISBN 0-672-31795-8. (June 7, 2000 edition.

[11] Rouvoy, R., Merle, P. Abstraction of Transaction Demarcation in Component-Oriented Platforms. ACM/IFIP/USENIX Intl. Middleware Conference (Middleware'03), Rio de Janeiro, Brazil, 16-20 June 2003

[12] SOAP 1.1 Technical Report, http://www.w3.org/TR/SOAP/, W3C, 2000.

[13] Steele, R., A Web Services-based System for Ad-hoc Mobile Application Integration, IEEE Intl. Conf. on Information Technology: Coding and Computing '03, 2003.

[14] Steele, R., Ventsov, Y., Dillon, T. XML Schema-based Discovery and Invocation of Mobile Services. Accepted to IEEE EEE04, Taipei, March, 2004.

[15] Steele, R., Lubonski, M., Ventsov, Y., Lawrence, E. Accessing SMIL-based Dynamically Adaptable Multimedia Presentations from Mobile Devices. Accepted to IEEE ITCC04, Las Vegas, April, 2004.

[16] Sun Microsystems, Java 2 Platform Enterprise Edition (J2EE), http://java.sun.com/j2ee/.

[17] Wi-Fi Specification. Available at http://grouper.ieee.org/groups/802/11/.

[18] WAP Forum. Wireless Markup Language Version 2.0. Available at http://www1.wapforum.org/tech/documents/WAP-238-WML-20010911-a.pdf.

[19] Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl

[20] Wouters, C., Dillon, T., Rahayu, W., Chang, E.: A Practical Walkthrough of the Ontology Derivation Rules. DEXA 2002: 259-268