CHAPTER 1

Introduction to Mobile Application Development
The obvious reason for you to buy this book is that you want to start building mobile applications with Oracle tooling. Before you get started, you must understand some of the history, principles, and technologies used in modern-day mobile development. This chapter will give you an overview of the mobile landscape, including design, device, platform, and technology, before introducing Oracle’s Mobile Application Framework (MAF). But before this, let’s start with looking back to when there was no mobility at all.

A Short History

On April 3, 1973, Motorola employee Martin Cooper stood in midtown Manhattan and placed a call to New Jersey. Using a prototype of what would become the Motorola DynaTAC 8000x, the world’s first commercial cell phone, Cooper stood near a 900-MHz base station on Sixth Avenue in New York City and placed a call to the headquarters of Bell Labs in New Jersey. Back then, no one was yet thinking of watching movies on their telephones, or playing games on them, or writing letters on them that would fly through space and end up on someone else’s phone. But Cooper already understood that the big trick would lie in making the phones smaller and lighter. And this is exactly what happened (Figure 1-1).

![Figure 1-1. Size of smartphones from 1983 to 2012 (Source: http://qz.com/42150/a-history-of-mobile-devices-told-through-screen-sizes/)](http://qz.com/42150/a-history-of-mobile-devices-told-through-screen-sizes/)
Chapter 1: Introduction to Mobile Application Development

When I started working in IT in the mid-nineties, there was nothing mobile except for the end users themselves. A lot has changed since then. Over the last two decades, there has been a massive shift where the work we traditionally did on desktop PCs moved to laptops, and later our mobile phones became “smart,” fully featured computers in their own right as they moved way beyond just a simple phone to a fully working, mobile office.

In the beginning, the very first mobile devices didn’t support any kind of apps. My first encounter with mobile development was in 2004 when I created an HTML UI for a hand-held barcode scanner, calling out to a web service to retrieve product and stock information. The first real apps I created were apps for BlackBerry and Windows mobile devices. These apps were built with the Oracle Mobile Client Framework, which was retired in 2010. That’s when Oracle started to work on the ADF Mobile framework, which was released in 2012. Based on what was learned from ADF Mobile, in mid-2014 Oracle launched a completely new framework. This framework is known as the Oracle Mobile Application Framework, in short, Oracle MAF.

NOTE

“MAF” is the abbreviation of Mobile Application Framework and is pronounced M-A-F.

Mobile Design Principles

When you design for mobile, you need to understand that designing for mobile is significantly different from designing desktop applications. There is a great set of design principles available, and in this section you will learn about these design principles. At the core of these principles is the fact that it is all about the end user. Mobile workers have different needs and priorities than their colleagues who remain in one location throughout the day.

Some characteristics of the activities of mobile users include completing tasks in short spurts, moving from place to place, and being frequently distracted by changes in their physical environment.

For example, let’s take a look at a person completing an online form for a rental car reservation while waiting to get on an airplane. When the gate opens, this person must get in line. Then he gets a call from his office to confirm the hotel reservation that he submitted earlier. By the time he enters the plane and finds his seat, he has probably forgotten the place in the rental car form completion process. Therefore, tasks need to be simple, easily recoverable, and fast; anything more than a few minutes to complete a task will not be workable in a mobile application.

Kids use mobile devices to play games, but in the business world, at their core mobile devices are about connecting people and the systems people use to do their job. Collaboration and communication are central to the mobile experience. Integrating Short Message Service (SMS), Multimedia Message Service (MMS), instant messaging (IM), e-mail, and phone calls into the application can make task completion more efficient. For example, when regional sales managers are reviewing results by store location, they may scroll to a specific store and call the store manager by tapping the phone number on the screen. Mobile application design provides unique opportunities to leverage built-in communication functionality.

When transitioning to mobile design, an application designer’s first thoughts often focus on constraints, such as the small screen size. However, mobile applications are not just limited versions of desktop applications; they are compact apps that support constant updates, decision making, and data entry. With lower costs, portability, and better computing power, mobile devices have become a platform for providing analytics and unique functionality. Sales professionals know
where to go next by viewing mapped routes of upcoming sales appointments, managers stay updated with real-time performance metrics, and retail merchandisers stay current by taking pictures of competitor product marketing materials.

If mobile applications are not well designed, users will not use them. The following design principles are fundamental to maximizing the adoption of mobile applications.

During the design stage, consider the tasks a mobile user will be performing, keeping in mind that hand-held usage is different from that of a laptop or desktop computer. How will your Mobile application help users get their jobs done? How will the users interact with the device? The more streamlined the application, the more they will use it. The next steps are to determine whether the application is required to work in a connected or disconnected mode; understand the device services integration requirements; determine the server-side data source and protocol. When designing server-side services, it is critical to provide optimizations for mobile access: if server-side web services are complex, requiring many round trips, for example, it would be difficult for the mobile application to consume them. This is not only due to the amount of data that needs to be passed, but also the amount of client-side logic that must be written to process the results. It is preferable to expose a set of server-side interfaces provided specifically for mobile that provide quick, efficient results.

The secret of writing good mobile applications lies in the understanding of functional and data partitioning, and how to make it easy for users to complete the task at hand. Wireframes offer a great help for designing mobile user interfaces that are easy to use and understand.

You also need to understand the business services that must be developed on the device, such as all Java modules and controls that need to be created. In addition, you should create wireframes for the views and flow in the application, which can help you to visualize the application functionality and assist in the development process. As a final design step, you should consider how to partition the application functionality into separate application features that represent a group of functionality and associated views. Then you can start designing the client user interface and task flows by creating wireframes.

As mentioned before, mobile development and design are different from desktop design and development. If you want to create a best-practice mobile app that satisfies your users, you should obey the most important mobile design principles. When you stick to these principles, you are likely to create a successful mobile app. These principles are explained next.

First, you must know your end users. Before designing, spend time getting to know the role of your end users and the specific needs that they will have while mobile. Understanding key characteristics of your users, their work environments, and the tasks that they perform will help ensure that your product has the right features and optimal user experience.

**TIP**

You can use Personas during the design phase of your mobile app. Personas are fictional characters created to represent the different user types. By thinking about the needs of a fictional Persona, designers may be better able to infer what a real person might need.

Next, you must define the essential mobile tasks. When assessing how to convert an existing desktop application into a mobile design, be aware that the key task for the desktop may be quite
different from that of the mobile. It is best to think through the mobile use case rather than relying
on the desktop workflows. Do not hesitate to eliminate tasks that are not essential to the mobile
requirements. Successful mobile applications are frequently simplified to primary tasks, such as
searching for coworkers or managing task lists.

Also, application design must be contextual. Mobile applications are used in trains, warehouses,
taxis, oil platforms, and more where there may be poor connectivity, loud distracting noises,
limited access to the touch screen though gloves, and other constraining factors. Designs must
work in the target work environment and maximize context awareness of the mobile device. A
GPS-enabled device that maps sales representatives’ locations helps them arrive at their next
appointments on time.

It is important to create a flattened navigation model. With limited time and concentration,
users do not want to, and in many cases are incapable of, traversing deep data structures to
complete a task. A flattened navigation model gives users quick access to the most important tasks
without having to fully concentrate on the task at hand. Once users begin their work, in terms of
navigation through the application, you should provide a clear understanding of where they are
and how they can return to their starting point if they mess things up. Instead of having a user log
in, access a section, find an object, and then perform a task, a flattened navigation model provides
quick access to the mobile task immediately after login.

Next, think and design for “two-minute” tasks. Typically, mobile devices are used for short tasks,
while laptop and desktop computers are used for more extended work. Mobile users will not
tolerate designs that require a lot of data entry or multistep flows because this takes too much
time and concentration. Mobile users need to get to work fast and complete their tasks quickly.
Tasks should be simple and quick to complete. If you can prefill data for them or hide
nonessential fields for the most part, this will speed up the data entry exercise.

Also, it can be useful to integrate analytics. Analytics and business intelligence are not limited
to the desktop. Mobile users need analytics that work for small screens. A color-coded graph of
sales data draws immediate attention to good, moderate, and bad situations. Compare this to a
large table of text data that takes minutes to read. The first step in determining which analytics
will be useful is to understand the mobile use case and how to integrate analytics that help
decision making. A needless insertion of analytics takes up valuable real estate and makes it
harder for mobile workers to do their jobs.

It is essential that you simplify the search process. Search is a very important part of mobile
applications and must be quickly accessible. Because mobile data entry is more difficult, simplify
search entry requirements to a single field when possible and place the field above related data.
If a user is on an inventory screen within a hand-held application and initiates a search for an
item, all results should relate to inventory attributes. Do not require the user to enter text in
multiple fields.

Make use of the power of the device and embed collaboration. You can embed collaboration
into workflows, and include triggers to call a person, connect to a social network, and text using
SMS and IM. Social networking is used heavily within the work environment and demonstrates
the importance of keeping in touch with colleagues and affiliated professionals.

Do not show all information at once. Because screen real estate is small, you must consider
the type and quantity of data that you display when designing the application. Information must be
summarized with basic overviews and limited actions. Details and additional actions should
be available in drill-down pages.
Reduce free-text input and instead use select lists, single or multiple-choice components like check boxes and radio buttons, and similar. Even with all the beauty in mobile, the virtual keyboard hasn’t yet found its way into everyone’s hearts.

Finally, it is important to leverage the mobile platform. Mobile applications can be built to run in the browser or as native applications installed on the device. Enterprise applications should leverage mobile capabilities that enable a user to tap a phone number to call or text, touch an address to map its location, and rotate the device for an alternative view. Native enterprise applications enable more integration than those applications run in the browser and provide the ability to transfer enterprise data to local built-in applications, such as calendars and contacts, so that users can view important business information without signing in. Understanding each platform and maximizing the appropriate mobile actions will ensure a productive and natural mobile experience.

Mobile Platforms

Mobile application development is targeted for one or multiple mobile platforms. These platforms are the operating system that runs on the mobile device. There are many mobile platforms. Some of them are relatively new and/or unknown such as Bada and Tizen, while others are already gradually disappearing, such as BlackBerry and Symbian (Figure 1-2).

From the perspective of this book, there are two platforms that you need to know, Apple’s iOS and Google’s Android.

![Smartphone operating system market share](http://www.statista.com/chart/1899/smartphone-market-share/)

**FIGURE 1-2.** Global smartphone operating system market share (Source: [http://www.statista.com/chart/1899/smartphone-market-share](http://www.statista.com/chart/1899/smartphone-market-share))
What Is iOS?
The iOS operating system is owned, developed, and distributed by Apple. iOS was released in 2007 originally for the iPhone and iPod Touch, and has been extended to support other Apple devices such as the iPad and Apple TV. iOS can only be used on Apple-produced hardware and is not licensed for installation on other hardware. iOS is derived from OS X and is also seen as Apple's mobile version of the OS X operating system used on Apple computers.

The user interface of iOS is based on the concept of direct manipulation, using touch gestures rather than a mouse. Interface control elements consist of sliders, switches, and buttons. Interaction with the OS includes gestures such as swipe, tap, pinch, and reverse pinch, all of which have specific definitions within the context of the iOS operating system and its multitouch interface. Internal accelerometers are used by some applications to respond to shaking the device (one common result is the undo command) or rotating it in three dimensions (one common result is switching from portrait to landscape mode).

What Is Android?
Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially, the operating system was developed by Android, Inc., which Google later bought in 2005. The first Android-powered phone was sold in October 2008.

The opposite of iOS Android is open source, and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers, and enthusiast developers. Additionally, Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language.

These factors have allowed Android to become the world's most widely used smartphone platform. Android's open nature has further encouraged a large community of developers and enthusiasts to use the open source code as a foundation for community-driven projects.

What About Windows?
There's a broad set of Windows mobile solutions, including Windows 8 Pro Tablet, Windows RT, and Windows Phone. Windows Phone is a series of proprietary mobile operating systems developed by Microsoft, and is the successor to its Windows Mobile platform. With Windows Phone, Microsoft created a new user interface.

Additionally, the software is integrated with third-party services and Microsoft services, and sets minimum requirements for the hardware on which it runs. Because of the multiple Windows-related mobile platforms and fragmented market share, Oracle hasn't yet supported any Windows platforms. It looks as if Oracle is waiting to see which Windows platform (if any) gets market share before implementing a solution. This is probably the main reason why Windows Phone is currently not supported by Oracle MAF.

What Technologies Do I Need to Know?
When you start developing mobile applications with the Oracle Mobile Application Framework, you might encounter a number of techniques that you have never used or even have never heard of. What technologies are important and which ones do you really need to know? The most important ones are discussed in the next sections.
HTML5

“HTML” stands for Hyper Text Markup Language, which is most of the code that makes up the web pages we view each day. “HTML5” refers to the fifth generation of the original language. Many features of HTML5 have been built with the consideration of being able to run on low-powered devices such as smartphones and tablets. It was specially designed to deliver rich content without the need for additional plugins.

The current version delivers everything from animation to graphics, music to movies, and can also be used to build sophisticated web applications. HTML5 is also cross-platform. It is designed to work whether you are using a PC, or a tablet, a smartphone, or a Smart TV.

CSS3

“CSS” is an acronym for Cascading Style Sheets, a web-based markup language used to describe the look and formatting of a web site to the browser, most commonly used in HTML or XHTML web pages. “CSS3” simply refers to the latest version of CSS, with additional capabilities compared to the first two generations. Because of its modular structure, CSS3 allows developers to build content-rich web pages with relatively lightweight code requirements. That means fancier visual effects, better user interfaces, and most importantly, cleaner pages that load faster than ever before.

JSON

JSON (JavaScript Object Notification) is a text-based open-standard data interchange format. JSON is easy for humans to read and write, and for software, it is easy to parse and generate. It is based on the JavaScript scripting language. Despite its relationship to JavaScript, it is language independent. There are many JSON parsers available in several programming languages, which makes JSON an ideal language for data exchange.

Apache Cordova and PhoneGap

PhoneGap is a mobile development framework that enables software programmers to build applications for mobile devices using JavaScript, HTML5, and CSS3, instead of device-specific languages such as Objective-C for the Apple iOS platform. The resulting applications are hybrid, meaning that they are neither truly native (because all layout rendering is done via web views instead of the platform’s native UI framework) nor purely web-based (because they are not just web apps, but are packaged as apps for distribution and have access to native device APIs). The software underlying PhoneGap is Apache Cordova. The software was previously called just “PhoneGap.” Apache Cordova is open source software.

NOTE

Throughout this book you may find that the terms Cordova and PhoneGap are used interchangeably. Both refer to exactly the same open source platform and library to enable you to create mobile applications built using HTML, JavaScript, and CSS. In 2011, the PhoneGap codebase moved to an open source Apache Software Foundation project under the name Cordova. Adobe still distributes the library under the PhoneGap name. Essentially, both the PhoneGap and Cordova projects are the same, and refer to the same free, open source library.
Chapter 1: Introduction to Mobile Application Development

A Challenge in Mobile Development

Essentially, developers can choose from three different application approaches, each with its own set of pros and cons. First, there are native applications. These refer to apps built for and installed on a specific platform, such as iOS or Android, using a platform-specific software development kit (SDK). For example, apps for Apple’s iPhone and iPad are designed to run specifically on iOS and are written in Xcode/Objective-C. Android has its own variation of Java, Windows uses C#, and so on. Native apps written for one platform cannot be deployed on another. Native apps offer fast performance and access to native-device services, but require additional resources to develop and maintain each platform, which can be expensive and time consuming.

Next there are mobile web applications. Unlike native apps, mobile web apps are not installed on the device; rather, they are accessed via a web browser. These are server-side applications that render HTML, typically adjusting the design depending on the type of device making the request. There are no program coding constraints for writing server-side apps—they can be written in any programming language. These apps work across platforms, but are limited to what you can do through a browser and require Internet connectivity. Oracle supports mobile web applications via MAF Faces (for tablets) and MAF Mobile browser (Trinidad) for smartphone and feature phones.

Finally, there are hybrid mobile applications. Hybrid apps combine technologies from native and mobile web apps to gain the benefits of each. These apps are installed on a device, like a pure native app, while the user interface (UI) is based on HTML5. This UI runs locally within the native container, which usually leverages the device’s browser engine. The advantage of using HTML5 is a consistent, cross-platform UI that works well on most devices. Combining this with the native container, which is installed on-device, provides mobile users with access to local device services, such as camera, GPS, and local device storage. Native apps may offer greater flexibility in integrating with device native services. However, since hybrid applications already provide device integrations that typical enterprise applications need, this is typically less of an issue. Oracle Mobile Application Framework is an HTML5 and Java hybrid framework that targets mobile app development to iOS and Android from one code base.

So what is the best approach?

While coding native applications, you will have to write the code in different languages, based on the platform. You will then have to compile the code for each platform and build binary packages that can run on the device. Upgrading the application to support the next version means you have to go back and redo the whole exercise of checking/fixing your code, rebuilding, and repackaging. Of course, there are advantages to using native applications. The performance of your application could be a very crucial factor. There are certain applications where you have to go native, especially when you expect real-time responses, as in games or command-and-control systems like drone remote controls. Also, with native apps, you can access core OS and device features, such as camera, accelerometer, contacts, and calendar. This is not easily done today with HTML5.

Access to enterprise applications with mobile devices is the standard nowadays rather than the exception. Such mobile applications increase efficiency because, unlike desktops, they can be used at any place at any time; but this also has a drawback. The speed with which mobile platforms are evolving creates big challenges for enterprises. Smartphones such as iPhone and tablets (iPad) are powerful platforms. However, when you need to develop for different mobile platforms, you usually need different tools and languages for developers. Developing mobile applications for each platform from the ground up typically means maintaining multiple code paths. Besides that, where do you find developers who are expert in all mobile platforms and their associated programming languages? And what if you find one? Can you afford to hire that developer? The Oracle Mobile Application Framework addresses these and more challenges.
Introducing Oracle Mobile Application Framework

Oracle Mobile Application Framework (MAF) enables rapid and declarative development of rich, on-device mobile applications. Oracle MAF-based applications are built using the Oracle MAF extension in Oracle JDeveloper or by using Eclipse and the Oracle Enterprise Pack for Eclipse (OEPE). Developers only need to write an application once, and then they can deploy the same application across multiple leading mobile platforms such as Android and Apple iOS. This means that MAF enables developers to build applications that are portable across devices and operating systems while still leveraging the device-specific capabilities and delivering an excellent user experience. Applications developed with Oracle MAF can be designed for phone and/or tablet form factors and can be packaged for either Apple iOS or Google Android. The applications will adapt to the form factor.

The Mobile Application Framework Runtime Architecture

In order to start mobile application development with Oracle MAF, you need to have some basic understanding of the MAF Runtime Architecture (Figure 1-3). Note that this section does not intend to go into detail about the various components. Its main purpose is to give you an
Chapter 1: Introduction to Mobile Application Development

overview of all components involved in the MAF Runtime Architecture. Further details will be covered throughout the book. Oracle MAF applications run in an application container that is compiled as a device-native application binary. It provides the runtime environment for a MAF application to run as an on-device native application right in the mobile device’s operating system (such as iOS or Android). The container not only hosts the client-side components of the MAF application, but it also provides navigation utilities such as a springboard and a navigation bar, both enabling access to the specific application features.

Inside the device-native container you find several components of the MAF Runtime Architecture. First, there is the web view. The web view uses the web engine of the mobile device to display web-based content. The web view is the primary mechanism to render and display the user interface of the Oracle MAF application. The user interface can be made up of several different technologies: Server HTML, Local HTML, and MAF AMX views.

Server HTML, for example, an ADF Faces or Java Server Faces application, represents a remote server-generated HTML-based web interface. It is delivered to the MAF application as a web page just as a regular HTML page is rendered in a browser. All of the HTML, business logic, and page flow are generated on a remote server. By using the Cordova JavaScript APIs, these server HTML pages can access device services such as the camera. In order to access these device features, the server HTML needs to be rendered inside the MAF application.

Local HTML, such as pages created using jQuery Mobile, refers to web pages that are directly embedded within and deployed as part of the MAF application. Local HTML files can access device-native features through the JavaScript APIs supported by Cordova.

MAF AMX views are based on the MAF AMX technology that delivers a JSF-like development experience to working with an HTML5-based user interface. MAF AMX views are defined using UI and code editors provided by JDeveloper and Oracle Enterprise Plugins for Eclipse (OEPE). These views are embedded into a MAF application and deployed to a mobile device. At runtime, the JavaScript engine in the web view renders MAF AMX view definitions as HTML5 components. Of the implementation approaches provided by MAF, application features developed using the MAF AMX components provide the most authentic device-native user experience through their extensive support of animation and gestures.

The controller is represented by a mobile version of Oracle ADF Controller that supports a subset of Oracle ADF task flow components available to a server-based Oracle ADF application. Both bounded and unbounded Oracle ADF task flows are supported, as well as a subset of events and scopes that are supported by the server-based ADF.

Java provides a Java runtime environment for MAF application. This Java Virtual Machine (JVM) is implemented in device-native code, and is embedded (or compiled) into each instance of the MAF application as part of the native application binary. The JVM is currently based on the JavaME Connected Device Configuration (CDC) specification. This JVM will be upgraded to be based on JavaSE8 in the near future. Inside the JVM you find three other components of the MAF Runtime Architecture.

First, there are managed beans. These are Java classes that can be created to extend the capabilities of MAF, such as providing additional business logic for processing data returned from the server. Managed beans are executed by the embedded Java support, and therefore must conform to the JavaME CDC specifications at the time this book is published.

The next component is the model. The model contains the binding layer that provides an interface between the business logic components and the user interface, as well as the execution logic to invoke REST or SOAP-based web services.
Finally, JDBC is part of the JVM. This enables a MAF application to connect to an on-device database. This is a data store that resides on the device. In MAF, this is implemented as encrypted SQLite database, which in Figure 1-8 is represented by “SQLite Local Database.” CRUD operations are supported to this local data store through the Java layer, using JDBC-based APIs.

The app config refers to services that allow application configurations to be downloaded and refreshed. For example, URL endpoints for a web service or remote URL connection. Application configuration services download the configuration information from a WebDav-based server-side service.

Credential management and access control refers to client-side services that provide security-related services for a MAF application, for example, a local credential store that securely caches user credentials to support an offline authentication, or access control services that display or hide application features based on user access privileges.

Apache Cordova is an open source code library that provides a common JavaScript API to access various mobile device services, such as the camera. Cordova provides a majority of the device services integration for a MAF application. Cordova JavaScript APIs are further abstracted as device data controls in the JDeveloper design time for MAF AMX-based views, allowing for integration of device services by simply dragging and dropping data controls to their MAF AMX views.

Server-Side Components of the Oracle Mobile Application Framework Runtime Architecture

On the server side, the configuration server refers to a WebDav-based server that hosts configuration files used by the application configuration services. The configuration server is delivered as a reference implementation. Any common WebDav services hosted on a common J2EE server can be used for this purpose.

Server-generated HTML refers to any framework for developing server-side applications that can be used for implementation of the remote URL MAF application feature. Also, a MAF application can use server-side SOAP and REST services to obtain data from server-side sources.

Finally, a MAF application can use APNS/GCM push services that allow for push notifications to the MAF application.

Developing with the Oracle Mobile Application Framework

When you start working with the Oracle Mobile Application Framework, you will typically go through several stages: design, develop, deploy. These stages are explained briefly in the next sections and in more detail throughout the rest of this book.

Design the Oracle Mobile Application Framework Application

During the design stage, consider the tasks a mobile user will be performing, keeping in mind that hand-held usage is different from that of a laptop or desktop computer. Work with the design principles explained previously in this chapter. How will your MAF application help users get
their jobs done? How will the users interact with the device? The more streamlined the application, the more they will use it. The next steps are to determine whether the application is required to work in a connected or disconnected mode, understand the device services integration requirements, and determine the server-side data source and protocol. When designing server-side services, it is critical to provide for optimization for the mobile access: if server-side web services are very complex, it would be difficult for the mobile application to consume them. This is not only due to the amount of data that needs to be passed, but also the amount of client-side logic that must be written to process the results. It is preferable to expose a set of server-side interfaces provided specifically for mobile. You also need to understand the client business services that must be developed, such as all Java modules and data controls that need to be created. In addition, you should create wireframes for the views and task flow in the application, which can help you to visualize the application functionality and assist in the development process. As a final design step, you should consider how to partition the application functionality into separate application features that represent a group of functionality and associated views. Then you can start designing the client user interface and task flows by creating wireframes.

Develop the Oracle Mobile Application Framework Application

Before you can start developing, you must first set up your development environment. You need to download and install the MAF extension or the plugin, and then install the necessary components and complete the required setup for development and deployment. All of this will be discussed in Chapter 3.

When creating your MAF application using JDeveloper or OEPE, you use the MAF application creation wizard and dialog boxes rather than code from the outset. This is one of the key features of JDeveloper and Oracle’s development tooling, that wizards and smart editors supplement many of the repetitive tasks, and you only need to code by exception.

NOTE

Oracle supports two IDEs with MAF: Oracle JDeveloper and Eclipse through Oracle Enterprise Pack for Eclipse (OEPE). OEPE, like JDeveloper, is available as a free download from the Oracle Technology Network (OTN) web site. There will be three Eclipse plugins for MAF: one for design time, one for runtime deployment, and one for sample applications. These plugins are all added to Oracle Enterprise Plugin for Eclipse.

The artifacts that result from creating the application include descriptor files for the MAF application and for the application features, default images for icons and tabs for all supported platforms, and a set of data controls used for accessing the services of a mobile device (such as camera, GPS, or e-mail).

When implementing the application features, you perform a thorough evaluation of the business need to determine which application features should be included within the MAF application. Using the overview editors provided by MAF, your tasks for implementing an application feature include identifying its type (HTML, remote URL, or MAF AMX, or native UI), its display properties (display name, navigation bar, and springboard icon), and its display behavior as dictated by both the mobile device capabilities and the user role.
Deploy the Oracle Mobile Application Framework Application

During the application deployment stage, you start with creating a deployment profile that will support devices and simulators for its respective platform. Creating a deployment profile may include selecting the display icon used for the MAF application itself in various orientations (such as landscape or portrait) and setting the application’s signing options (such as debug or release). You then proceed to deploying your application to the mobile device or simulator.

**NOTE**

With MAF applications, it is required that you deploy to the device or simulator before attempting any testing and debugging; in other words, the application cannot be run until you deploy it. Once the application has been deployed, it can be tested, debugged, and optimized.

**Other Tasks During Application Development**

Enabling and configuring security for the application typically require configuring the login server, such as the Oracle Identity Connect server, or it can be any web page protected by the basic HTTP authentication mechanism. In addition, you may have to configure the access control server.

After ensuring that your application functions as expected at a basic level, you can implement the Java code to access the server-side data. For connected applications, these Java classes should invoke web services directly. If your application uses SOAP or REST XML-based data sources, you invoke web services through data controls, with the assistance of a set of helper classes that you can invoke from your code to invoke the data controls and return data. If your application uses JSON-based data sources, your code should directly invoke the JSON service and return data, after which you need to parse the JSON data from the server and populate the objects holding data collections accordingly. For disconnected applications, your code should populate the local SQLite database. Then, the code that backs the user interface can retrieve data from the SQLite database instead of directly invoking web services.

Ensure that after adding security to your application and enabling access to the server-side data, the application deployment runs as expected and the application is ready for the final testing and debugging.

**Deploy the Oracle Mobile Application Framework Application to Production**

Deploying the application to the production environment typically involves publishing to an enterprise server, the enterprise App Store, the Apple App Store, or an application marketplace, such as Google Play. After you publish the MAF application, end users can download it to their mobile devices and access it by clicking the designated icon. The application features bear the designated display icons and display as appropriate to the end user and the end user’s device.
Chapter 1: Introduction to Mobile Application Development

Summary
Mobile development has gone through many stages of evolution over the last couple of decades, starting with the first mobile call in 1973 all the way to modern smartphones. Modern mobile devices require modern development frameworks. Oracle’s Mobile Application Framework enables you to build hybrid on-device applications for both the Android and iOS platform.

In this chapter you were introduced to

- Technologies involved in mobile development
- Mobile design principles
- The Oracle Mobile Application Framework
- Steps involved in developing a MAF application