Chapter 8: Design!
Using a DatePicker on a Tablet
Objectives

In this chapter, you learn to:

• Create an Android project on a tablet
• Understand tablet specifications
• Follow design principles for the Android tablet
• Add a second Android Virtual Device
• Add a custom launcher and tablet theme
• Understand the Calendar class
• Use date, time, and clock controls
• Determine the system date
Objectives (continued)

• Display a DatePicker control
• Launch a dialog box containing a DatePicker control
• Code an onDateSetListener method to await user interaction
• Determine the date entered on a calendar control
• Test an application on a tablet emulator
The Tablet Environment

- Android Apps not limited to phones
- Tablets are used to
  - Connect to the Internet
  - Play games
  - Use social media
  - Check e-mail
  - And more…

Figure 8-1 Marlin Adventures Android Tablet app
The Tablet Environment (continued)

Figure 8-2 Date Picker Calendar Control in a Dialog Box

Figure 8-3 TextView Control displays reservation
• Steps to complete the App:
  1. Add an Android Virtual Device specifically designed for tablets.
  2. Add the images used in this project.
  3. Change the theme and icon for the tablet display.
  4. Create a custom XML file with a Table layout.
  5. Add and initialize the TextView controls and the Button control.
  6. Initialize a DatePickerDialog with the present date and listen for the user to select a date.
  7. Return the selected date.
  8. Display the selected reservation date in the TextView control.
Designing a Tablet Application

- **Native applications** are programs locally installed on a specific platform (phone or tablet)
- An **emulated application** is converted in real time to run on a variety of platforms (various screens)
- Android tablet apps create an optimal user experience
- Different devices have different screen sizes
  - Common size is between 7 and 10.1 inches, 1280 x 800 pixel resolution and 16:9 screen ratio
  - iPads are 9.7 inches, 1024 x 768 and 4:3
Designing a Tablet Application (continued)

Figure 8-4 Android tablet displays Gallery controls
• **Design Tips for Tablets**
  – Keep screens uncluttered and use large controls
  – Make the design simple and make the app intuitive
  – Leave out “cool” interactions that distract the user
  – Use a flexible dimension – use dp and sp – not px or pt
  – Provide higher resolution – make your app look great on any screen size
  – Create a unique experience for phone and tablet designs
  – Use larger fonts than with a phone app. Consider printing out your user interface design to see how it looks.
Designing a Tablet Application

- Adding an Android Virtual Device for the Tablet
  - Uses Android Honeycomb 3.0 operating system
  - Recently updated (version 3.2 in Spring 2012)
  - Must add Android Virtual Devices (AVD) to Eclipse

Figure 8-5 Create new Android Virtual Device (AVD) dialog box
Designing a Tablet Application (continued)

• Creating a Tablet App

Figure 8-7 Application information for the new Android tablet project

Figure 8-8 Tablet emulator
Setting the Launcher Icon of a Tablet App

- Launcher icons are typically 72 X 72 pixels
- Preferred size for tablets is 96 X 96 pixels
  - Microsoft Paint offers a Resize button to make the adjustment automatically
- Custom Launcher is placed in the drawable-hdpi folder
  - Inside the application code, click in the line `android:icon="drawable/ic_launcher"`. Change the filename portion from “ic_launcher” to your icon
Designing a Tablet Application (continued)

• **Setting a Custom Theme of a Tablet**

![Custom theme displayed on tablet emulator](image)

*Figure 8-11 Custom theme displayed on tablet emulator*
• Setting a Custom Theme of a Tablet
  – Linear layout and tablet layouts used to create a simple, clean interface
  – User interface called **TableLayout** consists of four rows and four columns
    • Use the **padding property** to offset the content of one control from another
    • Use the **typeface** property to use font families
Designing a Tablet Application (continued)

Figure 8-14 TableLayout XML code for first two TableRows
Designing a Tablet Application (continued)

Figure 8-15 TableLayout XML code for last two TableRows
Designing a Tablet Application (continued)

Figure 8-16 mail.xml Table layout
Designing a Tablet Application (continued)

- Date, Time, and Clocks

Figure 8-17 TimePicker, DatePicker, CalendarView, Chronometer, AnalogClock and DigitalClock widgets
Designing a Tablet Application (continued)

• **Determining the Current Time**
  – Current date is the default in the DatePicker control
  – Uses a **static variable** – the value does not change through the execution of the app
  – currentYear, currentMonth, and currentDay hold the integer value of the system year, month, and day, respectively
Initializing the Button and TextView Controls

Figure 8-19 Class variables for Button and TextView controls
Initializing the Button and TextView Controls (continued)

Figure 8-20 Instance of the Button and TextView controls
ShowDialog Method

- When the user selects a button, the show dialog control appears

Figure 8-21 Inserting the Button onClick stub

```java
public void onClick(View v) {
    // TODO Auto-generated method stub
    showDialog(DATE_DIALOG_ID);
}
```

Figure 8-22 The onClick method launches a dialog box
Using the Calendar Class

- The Android System date is accessed by using the `Calendar class` which reads the updated time zone information from the wireless provider.
- The `getInstance` method returns the `YEAR`, `MONTH`, `DAY_OF_MONTH`, and the `DAY_OF_YEAR`
Using the Calendar Class (continued)

Figure 8-24 Current system date assigned to be displayed in the DatePicker dialog box

```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    btDate = (Button) findViewById(R.id.btnDate);
    reservation = (TextView) findViewById(R.id.txtReservation);
    btDate.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            showDialog(DATE_DIALOG_ID);
        }
    });

    final Calendar c = Calendar.getInstance();
    currentYear = c.get(Calendar.YEAR);
    currentMonth = c.get(Calendar.MONTH);
    currentDay = c.get(Calendar.DAY_OF_MONTH);
```
Adding the `onCreateDialog` Method

- The `onCreateDialog` method creates a dialog box based on the argument passed by the `showDialog` method call.

![Diagram showing current system date in DatePicker if today were March 17, 2014]

Figure 8-25 Current system date in DatePicker if today were March 17, 2014
Adding the **OnCreateDialog** Method (continued)

- The **protected** keyword signifies that the method or variable can only be accessed by elements residing in its class

```java
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case DATE_DIALOG_ID:
        return new DatePickerDialog(this,
                                   reservationDate, currentYear, currentMonth,
                                   currentDay);
    }
    return null;
}
```
• Coding the `onDateSetListener` Method

```java
private DatePickerDialog.OnDateSetListener reservationDate = new DatePickerDialog.OnDateSetListener() {

    public void onDateSet(DatePicker view, int year, int month, int day) {
        reservation.setText("Your reservation is set for" + month + 1)+("-") + day + ("-") + year);
    }
};
```
Adding the **OnCreateDialog Method** (continued)

- The **onDateSet** event is fired after the user sets a date selection

```java
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case DATE_DIALOG_ID:
        return new DatePickerDialog(this, reservationDate, currentYear, 
                                   currentMonth, currentDay):
    return null;
    }

    private DatePickerDialog.OnDateSetListener reservationDate = 
    new DatePickerDialog.OnDateSetListener() {
        // Override
        public void onDateSet(DatePicker view, int year, int month,
                               int day) {
            // TODO auto-generated method stub
        }
    };
    }
```

Table 8-29 the onDateSet method reacts to the date selected by the user
```java
package net.androidbootcamp.merlinadventures;

import java.util.Calendar;

public class Main extends Activity {
    private int currentYear;
    private int currentMonth;
    private int currentDay;
    static final int DATE_DIALOG_ID = 0;
    private Button btDate;
    private TextView reservation;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btDate = findViewById(R.id.btDate);
        reservation = (TextView) findViewById(R.id.tvReservation);
        btDate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                showDialog(DATE_DIALOG_ID);
            }
        });

        final Calendar c = Calendar.getInstance();
        currentYear = c.get(Calendar.YEAR);
        currentMonth = c.get(Calendar.MONTH);
        currentDay = c.get(Calendar.DAY_OF_MONTH);
    }

    protected Dialog onCreateDialog(int id) {
        switch (id) {
        case DATE_DIALOG_ID:
            return new DatePickerDialog(this, reservationDate, currentYear,
                                      currentMonth, currentDay);
        }
        return null;
    }

    private int DatePickerDialog.OnDateSetListener reservationDate =
            new DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int month, int day) {
            // TODO Auto-generated method stub
            reservation.setText("Your reservation is set for "+
                                 (month + 1) + "-" + day + "-" + year);
        }
    };
```
Summary

• When designing apps for an Android tablet, keep your users’ objectives and the size of the device in mind.
• To use an Android emulator designed for tablets, you first add AVD configurations appropriate for tablets.
• You can combine the Linear layout and the Table layout to create a simple, clean layout that takes advantage of a tablet’s width. The TableLayout contains TableRow controls—one for each row in your table in main.xml. In each TableRow, you can insert a view control such as a Button or TextView.
• You can display a calendar tool called a DatePicker control in a dialog box so users can select a date from the control. The Time & Date category in the Palette contains many calendar controls, including TimePicker, DatePicker, CalendarView, Chronometer, AnalogClock, and DigitalClock.

• To display the current system date when the DatePicker control opens, you use the currentYear, currentMonth, and currentDay class variables which hold the integer value of the system year, month, and day.
Summary (continued)

• To create a DatePickerDialog instance, you must define a unique identifier to represent the dialog box that displays the DatePicker control.
• If you include a control, such as a Button, that users tap to display a calendar, use the setOnClickListener method to implement the Button.OnClickListener, which creates an instance of the OnClickListener and calls the onClick method.
Summary (continued)

- When the integer id for the DATE_DIALOG_ID constant is passed to the DatePicker dialog box in the onClick method, a new DatePicker Dialog is passed along with the values for year, month, and day.
- After an app calls a showDialog(DATE_DIALOG_ID) method in reaction to the user tapping a Button control, the showDialog method calls the OnCreateDialog callback method.
Summary (continued)

- When a dialog box containing a DatePicker appears, users can select a date and tap a Button control. Tapping the Button invokes an onDateSetListener in DatePickerDialog, which passes integers representing the year, month, and day from the DatePicker into onDateSet. The selected date can then be displayed in a TextView control using setText.