


Lab 8.1 – Local Storage

Overview:

In this lab you will save data to local storage. You will save a serialized version of an new Event object as an XML file in local Android storage.

Step	Description
1.	<p>In this lab it is necessary to create files from event objects. The lab provides code for the methods necessary to perform the object serialization. Check "EventService.java" for the following methods:</p> <pre>writeXML serializeTag</pre> <p>If your code contains these methods you can skip to step 2. Otherwise you can get the code from the next step (step 1 a).</p> <p>The lab setup files contain code for serializing an event object as an XML string. That string will then be used for writing a file.</p>
a.	<p>Open the file in "C:/AndroidCourse/LabSetup/Lab8.1/xml.txt. This contains the code for two methods needed to serialize an event object. The methods are "writeXML" and "serializeTag". Copy these methods into the file "src/garagze.com.service/EventService.java". At the methods to the end of the class.</p> <p>Notice that this code uses "android.util.Xml". Be sure to read the javadoc for this class.</p>
2.	<p>Create a method in EventService to write a string out to a file on internal storage.</p>
a.	<p>Create a new method in "src/garagze.com.service/EventService.java". The method should return void and take a file name and an output string as parameters.</p> <p>The code should use the standard Java FileOutputStream technique for creating the file. Following is some sample code that you can use.</p> <pre>public static void writeFile(String fileName, String outputString) { FileOutputStream fos = null; try { fos = GaragzeApplication.getAppContext(). openFileOutput(fileName, Context.MODE_PRIVATE); fos.write(outputString.getBytes()); Log.v("EventService", "File written: " + fileName); } catch (FileNotFoundException e) { Log.e("EventService", "File not found", e); e.printStackTrace(); } catch (IOException e) { Log.e("EventService", "IO problem", e); e.printStackTrace(); } finally { try { fos.close(); } catch (IOException e) { Log.e("EventService", "IO problem", e); e.printStackTrace(); } } }</pre>

	<pre> } } } </pre> <p>However, there is one major different between the Android code and typical Java File IO code. The Android code returns a <code>FileOutputStream</code> object by calling the “<code>openFileOutput</code>” method on the Android app context. The problem for this method is to get the context.</p> <p>Note: This method will have a compiler error which you won’t be able to fix until the next step.</p>
b.	<p>To complete the method to write the file, you must get a reference to the context object for the application.</p> <p>If we were working within an activity, we could access the context object by simply running the following method:</p> <pre>Context context = getApplication();</pre> <p>But “<code>EventService</code>” does not extend activity so we must find another way. Fortunately, Android provides one for us. When Android first starts an application it can create an <code>Application</code> type object which we can use to hold the application context.</p> <p>You will now create an application class.</p> <p>Create a new file called “<code>com.garagze.GaragzeApplication.java</code>” and enter the following code:</p> <pre> public class GaragzeApplication extends Application { private static Context context; public void onCreate(){ GaragzeApplication.context = getApplicationContext(); } public static Context getAppContext() { return context; } } </pre> <p>Notice the static method “<code>getAppContext</code>” used to return the application object. It can be called from anywhere within the application. Once you have completed this class, the compiler error in “<code>EventService</code>” should go away.</p>
c.	<p>Modify the Android manifest to start up the custom application object whenever the application is started. Add the following attribute to the “<code><application></code>” tag in “<code>AndroidManifest.xml</code>”.</p> <pre>android:name="com.garagze.GaragzeApplication"</pre>
d.	<p>Whenever a new Event is created, you should also write an XML file for the event. You’ve created all the necessary method to do this now you just have to add some code to the “<code>addEvent</code>” method.</p> <p>The event object be initialized properly. It will need an id and a date. You can use the following code to assign these properties.</p> <p>In the “<code>addEvent</code>” method of “<code>EventService</code>” add the following code to write the event to an XML file:</p> <pre> event.setId(java.util.UUID.randomUUID().toString()); event.setDate(new java.util.Date()); </pre>

	<pre>writeFile(event.getId()+".txt", writeXml(event));</pre> <p>Run the application and add a new event. You should see the new event in the list view but how do you verify that it has been written to the Android file system?</p>
3.	<p>Verify that the file has been written. Review the logging but also check the file system for the file.</p>
a.	<p>Use DDMS to view the file. Start the DDMS perspective by selecting “Window > Open Perspective > Other > DDMS”</p> <p>In the DDMS perspective, select the “File Explorer” view.</p> <p>Drill down into “data > data > com.garagze > files”</p> <p>Select an XML file and click the icon  in the upper right corner of the view to “Pull a file from the device”</p> <p>Save the file to your local machine and open it to verify that it contains the data for the event that you entered.</p>
b.	<p>You may also see the file by using the Android command line shell. The instructor will demonstrate this technique in class.</p> <p>If you have time, you can learn more about this at the following URL:</p> <p>http://developer.android.com/guide/developing/tools/adb.html</p>