Sample android chat app source code eclipse

Note: Because of the size or complexity of this view, the author presented it as a .zip file to reduce download time. Once downloaded, you'll need a program like Winzip to unpack it. Virus Note: All files are scanned once a day for SourceCodester.com for viruses, but new viruses come out every day, so no prevention program can catch 100% of them. FOR YOUR OWN SAFETY, PLEASE: 1. Re-scanning downloaded files by personal verification of the virus before using it. 2. NEVER ever run collected files (.exe's, .ocx's, .dll's, etc.) --only run the source code. Presented by Nicks12on Wed, 05/01/2013 - 16:24 screenshot change screenshot of your own applications, presented by Pawan Soanawane (not verified) on Sat, 09/21/2013 - 17:50 It's a really great app and works very well. Thank you!!! Presented by Olugenius oluwole (unverified) on Sat, 09/28/2013 - 14:30 It's fine, I'll contact you directly: I hope this app will work for me Presented bynishant13 (not verified) on Thu, 01/16/2014 - 01:44s pl send me ths code. (Secure email) Presented byshobar (not verified) on Fri, 01/24/2014 - 12:06, how can we see the phone number submitted byryan ap (not verified) on W, 02/04/2014 - 12:37 pls send me this code. u make a great application, I think (email protected) Presented bypravinraj gavhane (not verified) on Fri, 02/14/2014 - 23:45 send me too to email protected, please submitted bychirag patoliya (not verified) on W, 02/18/2014 - 13:09 pls send me the logic for a chat application, how to work in Android on my email ID (email protected) Submittedprabal (not verified) on Thu, 02/20/2014 - 15:38 Please send me the source code to email protected . Thanks Presented ByHiram (not verified) on W, 02/25/2014 - 11:54 Please send the source code to an email protected thank you! Presented bysrijini (not verified) on Thu, 02/27/2014 - 16:09 hello, good job, could you send me the source code on my ID (email protected) thanks to presented Nisit Chaun (not verified) on Thu, 02/27/2014 - 23:27 hello nice to see mock chat code you can send me this code? (email protected) Presented by Manish Sindhal (unverified) on Fri, 03/07/2014 - 17:32 hello, Excellent and good work. Can you send me the source code on my id (email protected) pre-submittedLikeSun (not verified) on W, 03/25/2014 - 16:06 hello, Excellent and good job. Could you send me this code? E-mail protected pre-submitted bylepa (not verified) on Fri, 03/28/2014 - 14:27 good work, please send source code (email Presented bysenthil134on Fri, 03/28/2014 - 18:06 My email ID (email protected) Submitted 04/03/2014 - 17:47 Please, Send the source code (email protected) Submitted byVuthang (not verified) on Mon, 04/21/2014 - 03:51 Please send the source code to my email: Email protected Thank you very much! Presented byFrede (not verified) on Mon, 05/05/2014 - 03:51 hy I can give me a sorce email code: email is protected PresentedDexter (not verified) on Mon, 05/05/2014 - 10:15 What server?? Can you post? Presented bybahnuprakash (not verified) on Mon, 05/05/2014 - 13:06 Please send me the source code tyo (email protected) Submitted bybijin (not verified) on Wed, 05/07/2014 - 12:15 Please send the source code to my email: Email protected by Much thanks Thank you Much thanks! Presented bydommy (not verified) on Fri, 05/09/2014 - 21:06 Please send the source code to my email: Email protected Thank you very much! Presented bysankaran (not verified) on Thu, 05/15/2014 - 19:42 its very nice. I really need it, please me (email protected) Submitted byPalraj A (not verified) on Mon, 05/19/2014 - 14:16 Please send the source code to my email: Email protected Thank you very much! Presented by Aftab Saraz (not verified) on Thu, 05/22/2014 - 05:10 Hi, could you share your source code with me. I really need it. Thank you. Presented by Markus Bohr (unverified) on Fri, 05/23/2014 - 04:00 Presented ByRamya.D (not verified) on Mon, 05/26/2014 - 20:14 Hi Santos, Could You Share With Me The Source Code on Email Protected By Hafizhan (Unverified) on Wed, 05/28/2014 - 17:52 Please Boss send me the source code for chatRoom thanks here my mail ID (email protected) Submitted 05/29/2014 - 01:26 Presented byrajiv Gupta (not verified) on Thu, 05/29/2014 - 20:05 Please send me the source code of this massenger chat app to understand this project briefly. Presented byrajiv Kumar (not verified) on The Sun, 06/01/2014 - 19:05 hi sir you did a very good job. I searched for this application but didn't get it. would y please send me the source code for this chat massenger app using Google cloud messages, please send to email protected presented byrahulk7858on Fri, 06/06/2014 - 13:30 Please please send me the source code as soon as possible (email protected) Sent byugoo (not verified) to The Sun, 06/29/2014 - 00:57 pls send the source code email protected Presented bybasav (not verified) to Mon, 06/30/2014 - 19:07 pl send the source code to email protected presented bybasav (not verified) on Mon, 06/30/2014 - 19:07 pl send the source code to email protected presented bybasav (not verified) on Mon, 06/30/2014 - 19:07 pl send the source code to email Protected email submitted by Anonymous (not verified) on Wed, 07/02/2014 - 12:54 Please send me the source code to email protected . Thank you Submittedpromise (not verified) on Thu, 07/03/2014 - 01:48 Please send me the source code to an email protected submitted ByArsha Khan (not verified) on Thu, 07/03/2014 - 14:51 Please send me the source code to email protected presented presented (unverified) on Tue, 07/08/2014 - 15:29 saale teri maa ki faltu ka code deta hai... chutiya Presented byMaria K. Attit (not verified) on Thu, 07/10/2014 - 01:23 Sir, please send me the full source code of this project :(I just need it so badly :(Thank you, sir! Presented by Niladri Prasad Padi (unverified) on Thu, 07/10/2014 - 16:30 16:30 Sir, this is very useful for my android development carrier, could you send the source code of the application to this email ID protected.... In advance Thank Youssal presented by Farhat Ali (not verified) on Thu, 07/10/2014 - 18:32 Dear sir, it is very useful for my android development carrier, could you send the source code of the application to this email ID email protected . Presented byaziiz (unverified) on Fri, 07/11/2014 - 06:36 Please, send me the source code for an email protected by byvinay Kumar (not verified) on Sat, 07/12/2014 - 01:20 Dear Sir, could you send the source code of the application to this email id thank you in advance and all the best presentedabul Mustkeem (not verified) on Sat, 07/12/2014 - 23:0 send me the source code for an email protected byleeSY (not verified) on W, 07/15/2014 - 15:30 SubmittedPramod Wadkar (unverified) on W, 07/15/2014 - 17:05 Presented byprashant chaudhari (not verified) on Wed, 07/16/2014 - 15:53 Please send me a step-by-step process with a code for a simple chat app presented by Hereen TC (not verified) on Wed, 07/16/2014 - 19:31 My email ID is an email protected Start work with Google Android Development Tools Eclipse plugin Frank AblesonPubed February 266 , 2008 This tutorial introduces Android app development in the Eclipse environment, including building two example apps. First, it's a basic starter app, complete with all stages of construction and debugging. The second app looks at more complex Android features, including contact search and Google Maps search. To get most of this tutorial, mobile development experience is useful but not required. The skills ™ Java are necessary for Android apps, but are not a clear requirement for this tutorial. About this tutorialWhy do we care about Android? Android is an important platform for two reasons. First, the fact that Google presents it and the mind-share of Android got in such a small amount of time. Google flexes its muscles and tries to make a game for the crowded mobile market. His first salvo in this market, Android and Open Handset Alliance, is an impressive starting point. The second reason Android is important is because it's not just another mobile platform with a phone, menu and touch screen. As you'll find out in this tutorial, Android takes a different approach to apps. Android's architecture allows a highly customizable software environment thanks to its binding time of execution of requested actions and code to meet those requests. Whether it's market considerations or the technical aspects of Android, this platform is worth amazing. This tutorial is organized in the following sections:Android basics and necessary toolsThe Android developer Ensuring that and debugging the SaySomething Android appTocreate the content provider and Google Maps applicationSystem requirements This tutorial requires several technologies that work together. You need all of them for this tutorial. Eclipse PlatformEclipse is the platform on which the plug-in works. Get the latest version of the Eclipse Classic (V3.3.1 was used in this tutorial). Android Developer ToolsThe Android Developer Tools (Eclipse plug-in) can be installed by following the instructions found when installing Android SDK. Source codeSource snippets of code in this tutorial include: AndroidManifest.xml Snippet - This file is the handle of the deployment of applications for Android apps. IntentReceiver - This demonstrates the implementation of IntentReceiver, which is the class that handles the intentions advertised by the IntentFilter tag in the AndroidManifest.xml file. SaySomething.java - This implements the activities of Android, the main entry point to the sample application of this tutorial. Main.xml - This contains visuals, or resources, to use in Android activities. R.java - This file is automatically generated by Android Developer Tools and connects visual resources to java source code. AndroidManifest.xml full - It lists the full AndroidManfest.xml file, along with a description of each of the important items. MobileServiceCallContacts.java - This contains the code needed to display contacts, as well as respond to user input to follow google Maps search. Introduction to AndroidBefore diving right into all the inserts and exits of the Eclipse plugin and developing Android apps, let's take a look at the Android architecture and some key terms that will be useful in the tutorial and beyond as you start building Android apps for yourself. The terminology of AndroidAndroid application development in the Eclipse environment requires knowledge of the Eclipse environment and the Android platform. Understanding the terms below is useful in developing Android apps with the Eclipse plugin. Open Handset AllianceThis is an organization led by Google Inc., made up of numerous public and private organizations. Android Flagship Product of the Open Phones Alliance. It's an open source operating environment for mobile devices. EmulatorA is a software tool representative of another system - It is often an environment that works on a person computer (IBM®, Mac, Linux®) that simulates a different environment such as mobile computing devices. Linux Open Source operating system core at the center of many computing platforms, including servers, desktops, networking devices, and mobile computing devices. Android works on top of the Linux kernel. Dalvik Virtual MachineAlvik VM is an operating environment found in the Android stack, which application code while running. Dalvik VM is similar to a compatible Java VM, but they are not compatible. Android basics and necessary toolsAndroid is an open source operating system designed for mobile platforms. At the time of writing this article, this software is the only platform without public hardware devices. The Android platform is best described as a stack because it is a collection of components, including: Linux kernels based on the java programming system environmentTool chain, including compiler resources, push-up, and emulator Dalvik VM to launch applicationsThis that we briefly introduced the architecture of the Android platform, let's look at some important platform characteristics from a market perspective. Why is Android important? Computer technology has lavished attention on Android since its announcement and the initial release of SDK. Android is important as a platform for two disparate but compelling reasons, among many others. Android is a market engine. The space for mobile apps is crowded and it's hard to get a foundation for a beginner. Google has the resources and mind-share to make a splash in any market it puts in its sights. Google's entry into the mobile space has been in the works for several years. Android was a separate and separate company acquired by Google to give it a current on the mobile presence. Everything that Google does gets attention, and advertising is good for introducing new platforms. Score one for Android.The second reason Android is important is because of its app model. Android apps are not monolithic, menu-loaded apps that require a big tap and press button to work. Of course, there are menus and buttons to be used, but Android has an innovative design element in its architecture known as intention. Intent We intend is a design that allows the application to issue a request, which is sort of like a sign wanted by help. It may look like this: Wanted: an app that will help me find a contact or Wanted: an app that will help me display this image or Wanted: The app to perform this geographic search. Similarly, applications can register as capable and interested in meeting various requests or intentions. To follow the paradigm of classified advertising, they may look like this: Available: The app is ready and ready to submit contact records in a clear, concise manner, or Available: The app is ready and ready to perform a geographic search. These are examples of IntentFilters that are discussed further. IntentFilterApplications announce their availability for these types of operations through a design known as IntentFilter. is either registered during the run or listed in the AndroidManifest.xml file. The next snippet comes from app that responds to incoming SMS (text) messages: Listing 1. Android app in response to the zlt.receiver class. MySMSMailBox's intent-filter:action android:value android.provider.Telephony.SMS_RECEIVED/action-gt; Android Apps - A quick poll Give a minute to explore four main types of Android apps: Activity, Services, Receivers and ContentProvider. We'll also look at views to display user interface elements. ActivityThe activity is the most visible and visible form of the Android app. The action represents the user interface of the app along with a class known as the View. The view class is implemented as a variety of user interface elements, such as text boxes, tags, buttons, and other user interfaces typical of computing platforms, mobile or other. The app may contain one or more actions. They are usually on a one-to-one relationship with the screens found in the app. The app goes from one action to another, triggering a method known as startActivity () or startSubActivity. The first method is used when the application wants to simply switch to a new activity. The latter is used when you want a simultaneous call/response paradigm. In both cases, the intention is referred to as an argument to the method. The operating system is responsible for identifying the most qualified activities to meet this intention. Services and receivers As well as other multi-tasking computing environments, there are applications running in the background that perform different responsibilities. Android calls these types of apps services. The Android service is an app that doesn't have a user interface. The receiver is a component of the application that receives requests for processing intentions. Like the service, the receiver doesn't have a user interface element in normal practice. Receivers are usually registered in the AndroidManifest.xml file. The fragment shown in Listing 1 is an example of a receiver application. Note that the key to the receiver class is the Java class responsible for implementing the receiver. List 2 is an example of a receiver code. List 2. Codepackage com.msi.samplereceiver; import android.content.Context; Import android.content.Intent; import android.content.IntentReceiver; The public class is expanding IntentReceiver - the public void onReceiveIntent (Context arg0, Intent arg1) / Do something when this method is called. Managing data with ContentProvider is an Android mechanism for abstraction of data storage. Consider the specific type of data found on a mobile device: an address book or a contact database. The address book contains all the contacts and phone numbers of the person's person require when using a mobile phone. ContentProvider is an abstract access mechanism to a specific data store. In many ways, ContentProvider acts as a database server. It is up to the content to read and record in a particular data store through the relevant ContentProvider, rather than directly accessing the file or database. There may be both customers and implementations of ContentProvider.The next section introduces Android Views, a user interface mechanism for putting things on the Android device screen. ViewsThe Android activity uses views to display user interface elements. The views follow one of the following layout projects: LinearVerticalEach follows its predecessor, flowing underneath it in a single column. The subsequent LinearHorizontalEach element follows its predecessor, leaking to the right in the same row. The subsequent RelativeEach element is described in terms of bias from the previous element. A series of TableA rows and columns similar to HTML tables. Each cell can hold one view element. After selecting a specific layout (or combination of layouts), separate views are used to represent the user interface. View elements consist of familiar UI elements, including:ButtonImImageButtonEditTextTextView (similar to the label)CheckBoxRadio ButtonGallery and ImageSwitcher to display multiple imagesListGridDatePickerPickerSpinner (similar to combo box) AutoComplete (EditText with automatic text function) Listing 3 shows an example of a simple linear layout. List 3. A simple linear layout of the ?xml version? LinearLayout xmlns:android/ orientation'vertical android:layout_width'fill_parent android:layout_height'fill_parent'gt; z lt'textView android:layout_width'fill_parent android:layout_height'wrap_content android:wrap_content TextView android:layout_width'fill_parent android:layout_height'wrap_content android:text'Threa'1, second text view!'lt;lt;zlt;button android:layout_width'wrap_content android:layout_height'wrap_content android:text'Switch To Activity 2 id'id/switchto2 The next section goes through getting an Android SDK and tweaking it for use with Eclipse.Android software developer KitNow that we have a feel for the Android platform, let's Eclipse environment configured to develop Android so we can create our sample apps. This section runs through Android SDK and tweaking it for use with Eclipse.Obtaining and installing EclipseIf Eclipse is not installed, download it and install the latest stable release of Eclipse IDE from the Eclipse Foundation. Installation is a compressed folder. Remove content to a comfortable place on your computer. The installer does not create any icons or shortcuts on the windows®. For the purposes of this tutorial, the Eclipse folder will be located in the C:'software-eclipse. To start Eclipse, double tap on eclipse.exe find in the eclipse installation catalog. This will launch IDE. The software tells you the workspace and offers a default location, such as c:documents and customization of the user's name workspace. Choose this location or provide an alternative location for the workspace. As soon as Eclipse is loaded, click Workbench - Go to the Work bench icon on the home screen. Now it's time to get an Android SDK. Receiving and installing Android SDK There are SDK installation versions available for Windows, Mac OS X (Intel®, and Linux (i386). Android SDK is a compressed folder. Download and extract the contents of this file in a convenient place on your computer. For the purposes of this tutorial, the SDK is installed on c:'software-google-android_m3-rc37a. Obviously, if you install this on Mac OS X and Linux, you should install an SDK where you usually put your development tools. Both Eclipse and Android SDK have been installed. It's time to install the Eclipse plug-in to take advantage of the Eclipse environment. Receiving and installing the Eclipse plugin The next steps demonstrate the installation of the Eclipse plugin, officially known as Android Developer Tools. Please note that alternative installation directions are available on the Android website. Install Android Developer Tools: Start Find and Install in Eclipse, found in the Help and Software Updates menu. Choose to search for new features to install the option. Choose a new remote site. Give this site a name, such as Android Developers Tools. Use the following URL in the conversation: . Please note https in the URL. It's a safe download. A full-size image is added to the list and verified by default. Click Finish. Search results show Android Developer Tools. Select the developer's tools and click Next.After reviewing and accepting the license agreement, click Next. Please note that the license agreement includes a special requirement to use the Google Maps API. Browse and take the installation location, then click Finish.The plugin is now loaded and installed. The plug-in is not signed (at the time of writing), so continue at your own comfort level by clicking on Set Everything and then restarting the Eclipse Eclipse plug-in restarts, it's time to connect the plug-in to install the SDK. Preferences in the Windows menu. Click on the Android item in the tree view to Left. In the right glass, specify the location of the SDK installation. The value used for this tutorial is that Mac OS X and Linux are used in c:'software-google-android-m3-rc37a (again, use the appropriate places). Three more sections can be configured once the SDK is positioned. They are mentioned here briefly: The Build section has options for recovering resources automatically. Leave it checked. The Build option can change the level of verbose. The default is normal. DDMS - Dalvik Debugging Monitoring Service is used to peer into a running VM. These settings include TCP/IP port numbers used to connect to a running VM with a disconnect document and different levels and registration options. The default settings should be in order. LogCat is a log file created on the basic Linux core. The font is selected in this dialogue. Adjust this as you wish. Congratulations! Eclipse is ready to create Android apps. Creating the SaySomething Android app This section creates a basic android app called SaySomething using Android Developer Tools. As soon as the app is created, we will debug and forget it. The new wizardThe first step is to create a new project. Choose the wizards for the Android project, as shown below. Figure 2. A new image of the wizardView project in full size Application Requirements: NameLocationPackage nameActivity - Think of it as the main form or screen of the app's titleSPrice look at the new project. Figure 3. A new full-size Android projectView image will create a default app ready to be created and launched. The components can be seen in Package Explorer, which we'll discuss later. Package ExplorerThe Package Explorer (located in Java's Eclipse perspective) displays all the components of the Android application sample (see Figure 4). Figure 4. The ExplorerView image package in full sizeItems note include: src folderIncludes package for sample app, namely com.msi.ibmtutorialR.javaThe Android Developer Tools create this file automatically and represents the constants needed to access the various resources of the Android app. Learn more about the relationship between class R and resources below. SaySomething.javaImplementation of the main class of application activity. Links libraryContains android.jar, which is an Android runtime class jar file found in the Android SDK.res folder will hold resources for the app, including:AndriodManifest.xmlDeployment descriptor sample app. Next, we'll take a closer look at the source code. The main activity of the Sampling app consists of one action, namely SaySomething. As described above, the SaySomething class is implemented in SaySomething.java.Listing 4. SaySomething.javapackage Import android.app.Activity; Import android.os.Bundle; SaySomething's public class expands the activity @Override when the action is first created. What to note about this snippet source: SaySomething is a normal Java class, with package and import, as expected. SaySomething expands the basic Android class called Activity, which is in the android.app package. The onCreate method is the entry point to this activity, receiving a Bundle-type argument. The kit is a class that is essentially a wrapper around a card or hash card. The elements required for construction are transmitted by this parameter. This tutorial does not address this option. SetContentView (..) is responsible for creating the main user interface using the R.layout.main argument. This is the identifier that represents the main layout found in the app's resources. The next section looks at the resources to apply the sample. Resources for the Android Resource app are organized into a sub-direct project called res, as described earlier. Resources fall into three main categories: Drawables This folder contains graphics files such as icons and bitmapsLayoutsThis folder contains XML files that represent layouts and app views. They will be discussed in detail below. ValuesThis folder contains a file called strings.xml. This is the main line localization tool for the app. The next section dissects the main.xml file to view the main user interface of the resources.main.xml Application Example contains one action and a single view. The app contains a file called main.xml, which presents visual aspects of the main user interface of the action. Please note that there is no link to main.xml where the layout is used. This means that it can be used in more than one action if desired. List 5 contains the contents of the layout file. List 5. File layout?xml version?1.0 encoding'utf-8?'lt;lt;LinearLayout xmlns:android' orientation'vertical android:layout_width'fill_parent:android:layout_height'fill_parent'gt; TextView android:layout_width fill_parent android:layout_height'wrap_content android:text'Hello World, SaySomething'gt;lt;textView'gt; zlt;'LinearLayout'gt;'it's the most simplified layout. There is a single linear layout that is oriented as a vertical layout, meaning all the contained elements are in the same column. There is one TextView element that can be compared to a label in other development environments. TextView is a static text that is not edited. Note that each view element (the layout and TextView in this example) has attributes in the Android name space. Some attributes shared for all views - android: layout_width and and attributes, for example. Values available for these attributes: Fill ParentThis expands the view element to occupy the maximum space. This can also be seen as a signifying stretch. Wrap Content This value tells Android to draw items one by one without stretching. All resources are compiled during the build process. One of the products of this process is the R.java file, which represents resources for the remainder of the application. Next, the R.java file is discussed. R.javaThe R.java file is created automatically when assembled, so don't change it manually because all changes will be lost. List 6 contains an R.java file for the sample app. List 6. R.java file / AUTO-GENERATED FILE. DON'T CHEAT. This class is automatically generated by the aapt tool from the resource data found. It should not be manually altered. com.msi.ibmtutorial package Public final grade R - public static final class attr - public static final grade drawable - public static icon int'0x7f020000; - public static layout of the end class app_name - public static finale int main'0x7f030000; The R class contains anonymous subclasses, each containing identifiers for different previously described resources. Note that all of these classes are static. Note the item presented: R.layout.main. This identifier represents a layout defined by main.xml. Recall that this value is used in onCreate activity method as follows: setContentView (R.layout.main); This is the point at which specific actions (in this case, SayAnything) and a specific layout (core) are related to each other during execution. Building applicationsFiles are compiled every time they are saved by default. Figure 5. The paneView image error in full size We entered the bug into the source code, where we added extra space between setContent and View. When the file is saved, it is compiled and any errors appear in the problem bar at the bottom of the screen. Once the bug has been corrected, the application is properly created and errors are removed from the problem list. AndroidManifest.xmlThe AndroidManifest.xml file is the deployment handle for the Android app. The file lists any activities, services, content provider or recipient contained in the app, as well as relevant IntentFilters supported by the app. Here's the full AndroidManifest.xml file for sample app: Listing 5. File AndroidManifest.xml'lt;xml version?1.0 encoding'utf-8?'lt;lt;xmlns:android' package/com.msi.ibmtutorial'gt; zlt'@drawable/icon SaySomething android:label=@string/app_name> <intent-filter> <action android:value=android.intent.action.MAIN></action android:value=android.intent.action.MAIN ></intent-filter></activity></application></manifest> ></intent-filter></activity></application></manifest> What should be noted: The name of the package from the source file is presented here. This follows a similar pattern to the original Java file and import. Tag, in fact, import classes from this package. All not fully qualified classes in this file are in the package identified in the package attribute. The tag has an attribute that refers to a resource from the app's resources. Note the q symbol that precedes the drawable ID. This is a hint for a file to look into the app's resource folder for a resource called The Icon. The tag contains the following attributes and note values: The class represents the Java class by implementing this action: the label is the name of the application. Note that it comes from one of the line's resources. The string.xml file contains localized lines for the app. The intent-filter is a intentFilter available in the application example. This is the most common IntentFilter seen in Android apps. This filter essentially says that it implements the main action (or entry point) and is located in the OS launcher. In English this means that it can be launched as an app from the main list of apps on an Android device. The next section describes the launch of the app on android emulator from

Eclipse.Running appThere, now that the application has been successful, it's time to launch a sample of the app. Choose Open Run Dialog or shortcut on the toolbar in Eclipse. This opens up a dialogue in which startup configurations are created. Highlight the Android Application option and tap the new.Figure 6 icon to show the values used for the tutorial sample. Figure 6. You'll run the dialogView image at full sizeGive name configuration. The textbook sample uses the name Learning Configuration. Select the ibmtutorial project from the list of available projects (click View to see available projects). Choose start-up activity when you fall. Now select the emulator tab to indicate the emulator settings as you wish. By default, you can leave it alone. There are a few items to note as described in Figure 7.Figure 7. You run the dialogue, Emulator tabView image in full sizeThe multiple screen sizes and orientations to choose from, as well as the choice of network. Choosing a network is important when creating apps that use an Internet connection as a mobile device that has different network speeds. Choose the full speed of the network and no latency when prototyping the application. Once the basic functionality is present, it's a good idea to check in with less-than-ideal network conditions to see how the app reacts in suboptimal situations к сети. Выберите Run, чтобы увидеть</intent-filter> </activity> </application> </manifest> </manifest> use in action. Figure 8. EmulatorView image in full sizeTime that the app works on the emulator, it's time to see what happens behind the scenes. Dalvik (DDMS) debugging monitoring service will help with this. Debugging the appIn what happens to the running app, it's useful to connect to the running Dalvik VM. To turn this out from Eclipse, select the window for the other. This shows a dialog window where DDMS can be selected. This opens up a new perspective in Eclipse with a number of interesting windows. Here's a quick introduction to available resources in the DDMS perspective: LogCat is a running log of activities taking place in VM. Apps can make their own entries to this list with a simple line of code as follows: Log.i (tag, message); where tags and messages are Java lines. The Log class is part of the android.util.Log package. Figure 9 shows LogCat in action. Figure 9. LogCat in actionView image in full sizeAll convenient tool in DDMS is a file researcher that allows the file system to access the emulator. Figure 10 shows where an example of a textbook application is deployed on Emulator.Figure 10. An example of an app deployed in the EmulatorView image in full sizeUser applications is deployed in /data/app, while built-in Android apps are in the /system/app catalog. The list of running processes is also available in DDMS. Figure 11. The launch of the listView image in full Ful scale debugging the Android app goes beyond this tutorial. Creating a content provider and Google Maps exampleNow that you've seen is a complete example of the app, let's take a quick look at the more complex app. Content Provider and Google MapsIt's second app reviewed in this tutorial is built with the theme of a mobile service professional (perhaps a device repair technician) who should outline his way to the next service call. The app uses an built-in Android contact database as a record store. This tutorial will give you a feel for access to data from the content provider, as well as a look at the intention in action as we use the address data found in the contact database to perform a Google Maps search. For this tutorial to work properly on your Android emulator, be sure to have one or more contacts recorded and be sure to fill out a home address box. Figure 12 shows an emulator with multiple records in the contact application. Figure 12. The recording emulator in the full-sizeView contact app Is the first of two code fragments for the second app. Please note that the main activity class of this app expands ListActivity. This is because we will display the information in the list. List 6. The first snippet of the second MobileServiceCallContacts Expands ListActivity - Final Final MSCC tag Called when the action is first created. issue/@Override public void onCreate (Bundle icicle) - super.onCreate (icicle); setContentView (R.layout.main); Get a cursor with all the people of Cursor c and getContentResolver (People.CONTENT_URI, null, null, null); startManagingCursor (c); ListAdapter adapter - new SimpleCursorAdapter (it, android.R. layout.simple_list_item_1,c,c,new String,People.NAME, new int. R.id.text1); setListAdapter (adapter); } ... } Please note the use of a cursor class to request a contact database. This cursor set of results is related to the user interface through a class known as ListAdapter. Figure 13 shows the app in action because it represents the contacts available on the device. Please note that this display does not use the sorting order. Figure 13. Application in the full-size actionView image, as one of the contacts can be selected at the touch of a mouse, a central button on the emulator, or a key-press on the keyboard. Once this entry is selected, the code must perform a check to get the address of the contact you choose. This is where onListItemClick () redefined method comes into play. Implementation of this method has four important arguments. One of the most interesting here is the dbidentifier method. Because the cursor was tied to the user interface when this method is called, it actually gets an ID to the underlying source of data. The dbidentifier field can be used to request a contact database to obtain the desired information. It can also be used to simply run an application of contacts using intent, as shown in commented from the code in Listing 7.Listing 7. OnListItemClick () redefined method of @Override protected void on ListItemClick (ListView list, view view, int position, long dbidentifier) - super.onListItemClick (list, view, position, dbidentifier); Try // This commented code below will launch the Contacts app // and view contact Intention myIntent - new intention (android.content. // Intent.VIEW_ACTION,new ContentURI (content://contacts/people/ // dbidentifier)); startSubActivity (myIntent,position); Let's see the specifics of this ContentURI theContact entry - the new ContentURI (android.provider.Contacts.ContactMethods.CONTENT_URI.toURI(); !) IMPORTANT ! / To use this exemplary application, you need to have at least a ... one contact record on your Android emulator! / and don't forget to populate the Home Address box! / this is where the reservation is for the home address and for the person's entry / selected in GUI (id, dbidentifier) Courser c - managed by Cueri (TheContact, null) type 1 and person - db.7 if (!c.first)) - showAlert (MSCC, No available methods Return Line c.getString (c.getColumnIndex); Address : address.replace Address : address.replace Address: address.replace (((((((GeoIntent's intention is a new intention (android.intent.action.VIEW, new ContentURI (geo:0.0?q - address)); startActivity - Catch (Excluding ee) - Log.i (tag.ee.getMessage); Once you've received an address, you need a few simple lines to clean up your data to prepare it for a Google Maps query. GeoIntent is a new intention created to perform geo-search, which in the image of the Android emulator by default is satisfied with the call in Google Maps.All of the basic elements of the first application is still true for this application. There is one action that is launched from the main screen of the app. There is of course an AndroidManifest.xml file revealing our new app. Keep in mind that the full source code is available in the Download section. There is one last piece of information that is important for this second application example. There is an additional entry in the AndroidManifest.xml file that gives the app permission to read the contact database: zlt;uses-permission android.permission.READ_CONTACTS id. Without this explicit resolution, the Linux kernel will not allow the app to access the contact database. SummaryThis tutorial introduced the Android platform, Android Developer Tools, and key elements of Android's development in Eclipse. Android Developer Tools enable you to use The Rich Eclipse development environment to create and test android apps. Now you have to be ready to build your own Android apps. Resources you load