DOWNLOAD PDF ANDROID TUTORIALS FOR BEGINNERS IN ECLIPSE

Chapter 1 : Android Tutorial PPT for Beginners- JavaTpoint | JavaTpoint .com - theinnatdunvilla.com

Launch Eclipse from the Start Menu and click on File > New > Other and from the drop-down menu, click on Android Test Project, then at the bottom of the dialog click Next. Enter any name you want and in the Test Target panel, set An Existing Android Project and browse to the Rock Paper Scissors app you made.

Enjoy this article as well as all of our content, including E-Guides, news, tips and more. Step 2 of 2: You forgot to provide an Email Address. This email address is already registered. You have exceeded the maximum character limit. Please provide a Corporate E-mail Address. Please check the box if you want to proceed. I agree to my information being processed by TechTarget and its Partners to contact me via phone, email, or other means regarding information relevant to my professional interests. I may unsubscribe at any time. As easy as it might seem, it requires some hard work on your part before you can create an app that runs without errors. If you are reading this tutorial, it means you are probably new to the Android app development field. Set up the Android development environment Setting up Android development environment takes some time at first. Then, there are four tools that you will need and they are available on the Internet for free: Note for Windows Users: If you installed the JDK in C: After installing the Android SDK, you will get a window like this: Click on Install 7 packages to continue with the installation. You will get a dialogue box like this: It will take some time to install, so in the meanwhile you could do some other task to kill the time. How long will it take? Well, it depends on the speed of your Internet connection. Once it is done, you can close the SDK manager. After successful installation, it should display a window like this: This will display the following dialogue box. Just click on the Add button as shown in the picture and add https: When you press OK, Eclipse will start to search for the required plug-in and finally it will list the found plug-ins. Use the screenshot below to enter the correct values. You have successfully created Android Application Development environment. Before we write the code, you need to know how to take input from the user. The most efficient way of taking input from the user is to use the Scanner class, which is found in the java. Calling the Java code in Eclipse We will save you the trouble of writing the java code for a simple Rock Paper Scissors app and use can use the code below, but you are free to use your own code if you prefer. Use nested ifs if personPlay. Click on Finish to save it and it should appear in the Package Explorer window. Then we are supposed to add a package which will contain all our package files. Click on New Java Package icon to do this, as shown in the screenshot below. Name your project and then click Finish. Now we need to add a Java Class, which is as easy as adding a Java Package. After giving it a name, make sure that the following options are checked: After you create a new class, it will show up in the Work Space where you can write or copy the code. You have just finished writing your first Java Application in Eclipse. After selecting the export button, select Create new keystore and it will take you to the location where you want to save it, so give it a name and save it. Fill in all the required fields that are self-explanatory and save it. You have successfully exported the apk file to your computer and you can test the app it on your android device. Luckily, there is an integrated testing framework in Android Framework, which you can use to test all the aspects of your application. SDK tools can also help set up and test applications. SDK will help you test different aspects of your app no matter if you are planning on running your tests within an emulator or any Android device. I will recommend using ADT for the testing process, as it is comparatively easier than the other tools. Using ADT, you can easily create a test project and link it to the application under test. Click the Finish button to complete the Wizard and if it is disabled, look for error messages at the top to fix any problems. If you are looking for a step-by-step guide to Android testing, you will find this Activity Testing Tutorial very useful. This was last published in January Related Resources.

Chapter 2 : Getting started with Android development - Tutorial

This video is the first in a web series that I am creating to help you learn Android programming so you can create apps and eventually how to learn to write games for android phones.

Introduction This tutorial is a starting point for developing Android apps. This tutorial explains Android beginners how to create an Android Project in Eclipse, work with resources, and create some first code. You can check what version of Java is installed, by going in the command line and typing java -version. First of all, start Eclipse. Here you can set up the project. First of all, we need to give the project a name, so type "Hello World" in the name box. Next you have to select the Android version you want to use. Here we choose version 2. Also, the project requires an application name. Notice that this name is also used in code, so the name should have no whitespaces. Usually, you use the project name and delete all whitespaces e. Next, you have to define the package of the project. We will use "com. In this example, we simply use "HelloWorldApp". Before we can finally start our first project, we need to create a configuration. This configuration specifies under which circumstances the app will be started. Also, you can choose different emulators to test the app with different versions of Android or in different screen sizes. Now click the "Android Application" tab at the side and then the New button above the tabs. Call the new configuration "HelloWorldConfig" and select our project over the Browse button. Now move on to the target tab. Here you can select the network speed and which emulator will be used. Click the automatic control to enable the buttons at the side and then click on the manager-button. Here, click the new button to the right to create a new emulator. In the following screen, you can enter a name for the emulator I have used "DefaultAndroidEmulator" and specify the details like Android version, SD card size, and much more. You can control every little detail of the emulator over the hardware section. Once you are done with that, click "Create AVD" and close the manager window. Now we have successfully created the run configurations. Click "Apply" and close the configurations. At least run your first Android project. It may take the emulator some time to get started, so be patient! You just created your first App! But before we can actually jump into the Java code, we need to understand the structure of an Android Application. Go to your Package Explorer and enlarge the "Hello World" project. You will see five folders and two files. This file contains all the information about your project, like the icon, the name of the author. In the upcoming tab, you can specify the package name and the version of your project. At the bottom, you will find additional tabs. I think most of the settings you will find are pretty much self-explanatory. Note the in front of some attributes. This shows that the following string is a reference to a resource. You can find the resources in the "res" folder of your project. If you enlarge it, you will notice that it has some subfolders. To be specific, the res folder can have seven types of subfolders: Here you can store all kinds of simple resources like strings, colors, numbers, dimensions, arrays, etc. By default, you will find the strings. The first is the message you see when you run your project, and the second is the name of your app. You can add new values if you want to use them later on in code or in the Manifest or Layout files. You can also create specific resources using quantifiers. If you do not add a quantifier, the resources are default. The default resources are used if no specific resources for the current system are found. If the project is started, all resources will be compiled as efficiently as possible and added to the package. Also, a reference will be created called R which allows you to access the resources in code. Since this is only a tutorial, I will not focus on all the types of resources here. You can find more information on resources and quantifiers here. At last, it is time to start coding! Go to the "src" folder. In the folder, you will find the package folder, open the HelloWorld. You will see the default code for an Android Activity: Of the base class super. Then we override the default onCreate function, which is called when the project is created. In there, we load our own layout from the resources and also call the onCreate function of the base class. You find it in the layout folder under resources. When you open it, it should look like this: As you you might already have figured out, there are different types of layouts: All controls children are placed in the upper left corner. The positions of the children are specified in relation to the other children. The child elements are placed with a grid. The child elements are positioned based on absolute coordinates in pixel. Once you have chosen a layout type, you can add child elements. In the code given, there is already a TextView, which is used to display text on the screen. The current content is a reference to a resource defined in the values. As you will see, it uses the whole width of the screen, but is only as long as it needs to, to display the content. We might start with some small changes. Next we create some code for the controls. Go to the helloworld. First of all, as before, we load the layout. In this OnKeyListener, we create the method onKey, which is called when a key is pressed, when the control is active. In the method, we perform two checks: If both checks are passed, we add the text of the EditText control to the TextView, and finally the text of the EditText control is deleted. Run and test the application. Great, you created your first real Android app. Android App Design As with every platform, Android has its own design challenges. Always keep in mind that you are developing for a mobile platform with limited memory, disk space, and processing power. Therefore, Android automatically kills processes each app runs in its own process to keep the system responsive. Processes are sorted after importance. The most important is the currently active process, followed by visible and stated service processes. The bottommost types of processes in hierarchy are background and empty processes. Also, you can use whatever hardware is built into the Android phone. But notice that not all phones might have this hardware, and so not all might be able to run your app. Hopefully, you understood the basics of Android development. Since this is my first article on CodeProject, I would really appreciate feedback! History 18 August, - Some minor changes.

Chapter 3 : Android SDK tutorial for beginners - what you need to know

Succeeding Eclipse as the main IDE, Android Studio has come along way since its introduction in Here is an introduction tutorial for beginners.

This is the official IDE Integrated Development Environment for the Android platform, developed by Google and used to make the majority of the apps that you probably use on a daily basis. Prior to its release, Android development was handled predominantly through Eclipse IDE, which is a more generic Java IDE that also supports numerous other programming languages. Android Studio makes life significantly easier compared with non-specialist software, but is still has a little way to go before it can claim to be a completely intuitive and smooth experience. For complete beginners, there is an awful lot to learn here and much of the information available â€" even through official channels â€" is either out of date or too dense to make head or tails of. So just what is Android Studio? The programming language you will be using is Java and this will be installed separately on your machine. Think of this as an extension to the Java code that allows it to run smoothly on Android devices and take advantage of the native hardware. Java is needed to write the programs, the Android SDK is needed to make those programs run on Android and Android Studio has the job of putting it all together for you. At the same time, Android Studio also enables you to run your code, either through an emulator or through a piece of hardware connected to your machine. Google has done a lot of work to make Android Studio as powerful and helpful as possible. Setting up Setting up Android Studio is fairly straightforward and is easier than ever thanks to nearly everything being bundled into one installer. Remember, Android Studio is only really your window into Java! Follow the simple instructions during installation and it should also set you up with an Android platform that you will be able to develop with as well. Be sure to tick the checkbox to tell the installer that you want the Android SDK as well and make a note of where Android Studio itself and the SDK are being installed. These are the defaults that it selected for my installation: Pick a directory for the SDK that has no spaces in it. In some cases, this will be the entire app or in others, your app might transition from one screen to the next. This will include a menu in the top right corner, as well as a FAB button â€" Floating Action Button â€" which is a design choice that Google is trying to encourage. Pick the option that best suits the app you have in mind to build and this will impact on the kind of files you are presented with when you first start things up. What are all these files? To me, programming meant typing in a single script and then running that script. Android Development is rather different though and involves lots of different files and resources that need to be structured in a specific way. Android Studio exposes that fact, making it hard to know where to start! By default, this is MainActivity. Java but you may have changed that when you first set up the project. However, the actual layout of your app is handled in another piece of code entirely. Just to make things a little more complicated though, you can actually use any XML file to define the layout of any Java script called a class. This is set right at the top of your Java code, with the line: This also means that you could theoretically use the same XML file to set layouts for two different Java classes. A new empty activity, I love the smell of possibility in the morning! Your Java files are housed under java and then the package name of your app. Double click on MainActivity. Java and it will come to the fore in the window on the right. When you are editing XML files, you might notice two tabs down the bottom. In the Text view, you can make changes to the XML code directly by adding and editing lines. Everything in the resources folder needs to be lower case, which is why underscore is used a lot to separate file names into readable titles in the absence of camel case. This contains more XML files that hold the values of variables â€" things like app names and color values. You can create additional Java classes, XML files or entire activities at any point in order to add more functionality to your app. This is handy if you want to edit an image for example. Meet Gradle Android Studio tries to keep things nice and simple for users by providing all of the necessary tools and features in one place. Things only get more complicated once you need to interact with some of these other elements. You should be able to leave Gradle to do its thing most of the time, but you will occasionally need to jump into the build. One is to run it on your physical device and the other is to create a virtual device emulator to test it on. Running it on your device is simple. This is faster than ever right now thanks to the Instant Run feature. Should something go wrong causing your app to crash or become unresponsive, then red text will appear and this will give you a description of the problem. It essentially saves you a ton of time versus blindly trying to guess what went wrong. Make sure to filter the types of messages you want to see here. You can also switch to the monitors tab and see useful information such as the CPU usage etc. The Android Device Monitor takes this monitoring a step further and lets you monitor everything at once, complete with handy UI. However, one of the biggest challenges for Android devs is fragmentation. This is essentially an emulator that you can use to mimic the look and performance of any other Android device, setting such things as screen size, power and Android version. To use the virtual device though, you first need to build one by downloading the required components and setting the specifications as you want them. For those wondering, you can treat this just like any other emulator and even access the Play Store to download your apps. The SDK Manager If you want to target a specific version of Android, or if you want to create a virtual device running a specific version, then you are going to need to download the necessary platform and SDK tools. Make sure to keep-up-to-date! Google has made this easy by building support right into the IDE itself. Likewise, you may find yourself needing to use GitHub, which lets you backup your apps online and handles version control for streamlined collaboration. While this might all sound like a headache, Google is taking huge strides to keep making these processes as simple and easy as possible. This tutorial would have been much more confusing a few years ago, even just the set-up stage! The best strategy is to get stuck in with a simple app project and to only learn the more advanced features as you need them. No Coding Experience Required. Whether you are an absolute beginner with zero coding knowledge or a veteran programmer, this course will guide you through the process of building beautiful, functional Android apps and bring you up to speed on the latest features of Android and Android Studio. The package includes over 6 hours of high quality videos and over 60 different lessons. Claim your discount now using exclusive promo code: This is your ticket to a lucrative future in Android App Development. What are you wating for?

Chapter 4 : 12 Android Tutorials for Beginners

Android Hello World tutorial using Eclipse for beginners - Step by Step Android SDK implementation Android is a Linux-based operating system first introduced on Nov. 5, , was originally developed by Android Inc. and subsequently purchased by Google.

To make it easy for you and with no illusions that this list of Android tutorials is the best or complete, here are 12 Android tutorials to start with. Some of them start out for beginners and then delve into more advanced topics. If you encounter a hurdle, just spend more time with the tutorial, reading it a couple of times if necessary. If you are still not on friendly terms with it, there is no drama â€" just move forward and revisit it later. This is good because all the important content about the topic in one place and you just have to read it. This tutorial has more topics and information than the tutorial from Google, so if you are looking for an in-depth tutorial, this is one the. If you want to get the most from it, you will need quite a lot of time to read it from start to finish. It can be a great source if you need to consult a given topic in detail. Video Tutorials Series I find video tutorials less useful except when they teach design, animation, or any other visual topic but for many people they are the preferred way of learning. If you belong to this group, you will love this series of video tutorials. Similarly to the previous two tutorials, this series covers everything from absolute beginner level to advanced topics. The first two tutorials in this list are book-like but if you want something more authentic you could print them. Even better, a pdf tutorial, like this one , is a much better option. Similarly to the previous resource, this one might not be very up to date but it does cover the major principles of Android programming. This is one more general tutorial that covers Android development from beginner level to advanced. Game Development Series If you have some knowledge about Android but you want to delve into games development, this series of video tutorials is a great start. The series starts with the very basics of Android and Eclipse but my personal feeling is that if you are a total stranger to Android, the journey will be too hard. From what I saw, the series mentions general Android as well, not only game development. For some of these topics you can find information in the general tutorials as well but if you want more detail, this is for you. In this tutorial you will learn how to set up the action bar, how to add actions, how to split, hide, and overlay it, as well as how to add navigation. You will also learn about action bar interactivity, such as how to handle clicks on its items and to use action views. In such cases you need to know how to handle this data. This tutorial leads you step by step in the world of XML parsing. It also helps you create a parser that will look like the one shown in the next screenshot. Android for iOS Developers With the huge popularity of Android, even die-hard iOS developers are likely to consider switching or at least expanding to it. If you are an iOS developer, you are lucky because you are not new to mobile development as a whole. Of course, you could read the general Android tutorials I listed earlier but especially for you, here is a better tutorial. Unfortunately, some of the info in this tutorial might be outdated but with the rapid development of mobile programming technologies this is inevitable. This tutorial is great because it summarizes the differences between iOS development and Android development, thus making the change easier for you. The tutorial is a pretty detailed one â€" it starts with how to install Android Studio, how to create a new project, how to add functionality to it, how to run it, etc. Localizing Android Apps Android applications are popular all over the world. Your users speak different human languages, which means if you want to reach them, you need to think about localizing your Android apps. This tutorial explains it all. Getting Started with Android Library Projects At some point in your Android development career you will get tired of having to re-invent the wheel all the time and you will appreciate the advantages of reusable code. If you are already there, you will certainly want to know more about reusable code. In this case this tutorial will help you get started as quickly as possible. The first part warms you up with some basic concepts, while the other two delve into more detail about how and when to use Android Library Projects. So, if you have a spare minute, check the tutorials, learn something new and let us know your favorite tutorials. Meet the author Ada Ivanoff Ada is a fulltime freelancer and Web entrepreneur with more than a decade of IT experience. She enjoys design, writing and likes to keep pace with all the latest and greatest developments in tech. In addition to SitePoint, she also writes for Syntaxxx and some other design, development, and business sites.

Chapter 5 : Android Development

Android is an operating system based on the Linux kernel. Android is developed in the Android Open Source Project (AOSP). This project is lead by Google. The Android operating system can be divided into the four areas as depicted in the following graphic. An Android application developer typically.

We will also take a look at the views to display the user-interface UI elements. Activity apps The activity is the most visible and prominent form of an Android application. An activity presents the UI to an application, along with the assistance of a class known as a view. The view class is implemented as various UI elements, such as text boxes, labels, buttons, and other UIs typical in computing platforms, mobile or otherwise. An application may contain one or more activities. An activity is typically defined on a one-to-one relationship with the screens found in an application. An application moves from one activity to another by calling a method known as startActivity or startSubActivity. In both cases, an intent is passed as an argument to the called method. Android calls these types of applications services. The service is an Android application that has no UI. The receiver is an application component that receives requests to process intents. Like the service, a receiver does not, in normal practice, have a UI element. Receivers are typically registered in the AndroidManifest. The following snippet which is the same as the one shown for IntentFilter is an example of a receiver application being defined. Note that the name attribute of the receiver is the Java class responsible for implementing the receiver. The address book contains all the contacts and phone numbers a person might require when using a mobile phone. The ContentProvider is a mechanism to abstract access to a particular data store. In many ways, the ContentProvider acts in the role of a database server. Operations to read and write content to a particular data store should be passed through an appropriate ContentProvider, rather than accessing a file or database directly. Android Views Android views are the UI mechanism for putting things on the screen of an Android device. The Android activity employs views to display UI elements. Some of the more popular layout designs include: LinearVertical â€” Each subsequent element follows its predecessor by flowing beneath it in a single column. LinearHorizontal â€” Each subsequent element follows its predecessor by flowing to the right in a single row. Each cell can hold one view element. After a particular layout or combination of layouts has been selected, individual views are used to define the UI. View elements consist of familiar UI elements, including: For example, this shows a simple LinearVertical view, with buttons and textviews defined: Obtaining and installing Android Studio This first step is very straight-forward. Open your web browser to https: Android Studio is available for Windows, Mac, and Linux. But, before we go any further, we need to have a discussion around Android versions. It is easy to be overwhelmed by the choices available. There are many versions, or revisions, of Android in the wild. If you are going to write an application for commercial release, you will need to know which devices you are targeting. Wikipedia has a nice summary of the Android versions, the names, the release dates as well as a graphic depicting the approximate number of each device in the marketplace. Newer devices run the latest code, and older devices can be somewhat constrained in their ability to upgrade to a newer version of Android. Sometimes the newer versions support capabilities that the older handsets are not capable of supporting think second camera, Near Field Communications, and so on. Supporting a new operating system requires resources on the part of the device manufacturer and the carrier. Market forces suggest that these market participants are more concerned with future sales than supporting prior sales and invest their human capital accordingly. As a software developer, what does it matter that there are new versions of the operating system? This breaks down into two broad categories when it comes to programming applications: Sometimes code needs to actually be removed from the API definitions. This is essentially a message to the developer that goes something like this: With iOS, even the programming language seems to change while we are not looking! While Android may be a bit more forgiving with its long-suffering of deprecated code, we do need to keep an eye on the documentation tag which says that a particular API is deprecated. Deprecation markers in the SDK are our roadmap to future opportunities for improvement and for potential headaches. Being a successful mobile application developer requires an ongoing investment over the lifetime of an application where time tends to pass like dog yearsâ€¦very quickly! This menu is our primary means of interacting with the various SDKs available for the many versions of Android. With each release of the operating system, the API level increments. Notice that the revision increases slower than the API level. When you browse the Android documentation , be sure that you note which API level that a particular capability was introduced. Select one or two SDK levels to work with for now. This is the emulator. The emulator is helpful because it permits us to test applications on various device types, sizes, and versions. The screen shots in this tutorial are taken from the emulator. More on this later. Building your first Android application This tutorial steps you through creating a basic Android application, called SaySomething, using Android Studio and demonstrating it on the Android emulator. Fill out the Create Android Project dialog, and click Next. The requirements for the new project include: Application name Company domain this gets reversed into com. Project Location â€" where to store the files The Package name is suggested from the Application Name and Company domain By default, Android applications are written in Java. As mentioned earlier, Android is for more than just phones, though for our purposes we will simply select the Phone and Tablet form factor for this tutorial. This will create a default application ready to be built and run directly from Android Studio. On the Configure Activity dialog, name the files for the application. Review the code The following figure shows the components of our new project: There are two folders: The manifests folder in the app folder contains AndroidManifest. This file tells the Android device about how to interact with the application, which Activity to show when the application is started, which Intents the application services, which icons to display and much more. The java folder contains the source code for the application. In this case, the file which implements our activity plus a couple of class files for performing automated tests which we will ignore for this tutorial. The res folder contains the resources for the application, including icons, layout files, strings, menus, and so on. The Gradle Scripts folder contains all of the scriptable elements for building the application. The Gradle build system is a topic unto itself. For now, understand that these scripts govern the process of building the application â€" and importantly for more mature projects and teams, each of these steps is able to be run from the command line. This is important because this approach supports automation and fall into the category of Continuous Integration. Primary activity for the application Our sample application consists of a single activity, named MainActivity. MainActivity is a normal Java class, with a package and imports, as expected. MainActivity extends a base Android class named AppCompatActivity, which is located in the package named android. The onCreate method is the entry point to this activity, receiving an argument of type Bundle. The Bundle is a class which is essentially a wrapper around a map or hashmap. Elements required for construction are passed in this parameter. This is an identifier representing the main layout found in the resources of the application. Essentially all of the xml resource files under the res folder described earlier are accessed with the R class. Resources for the application Resources in Android are organized into a subdirectory of the project named res, as described previously. Resources fall into a number of categories. Three of these categories are: Drawables â€" This folder contains graphics files, such as icons and bitmaps Layouts â€" This folder contains XML files that represent the layouts and views of the application. These will be examined in detail below. Values â€" This folder contains a file named strings. This is the primary means for string localization for the application. This file is analogous to a css style sheet for those familiar with web programming techniques. This means it may be used in more than one activity, if desired. There is a single linear layout, which is oriented as a vertical layout, meaning all contained elements are in a single column. Within this encapsulating LinearLayout, there is a second LinearLayout organized as a horizontal layout. This inner layout contains a pair of button widgets. Below the horizontal LinearLayout there is a single TextView element, which can be likened to a label in other development environments. A TextView represents static text that is not editable. Below the TextView, there are two EditText widgets, configured to take numeric inputs.

Chapter 6 : Android Studio tutorial for beginners - Android Authority

Google provides the Android Development Tools (ADT) to develop Android applications with Eclipse. ADT is a set of components (plug-ins) which extend the Eclipse IDE with Android development capabilities.

You will need these tools regardless of which version of Android you are targeting. These are what will actually create the APK â€” turning your Java program into an Android app that can be launched on a phone. These include a number of build tools, debugging tools, and image tools. The Build tools were once categorized under the same heading as the Platform tools but have since been decoupled so that they can be updated separately. As the name suggests, these are also needed to build your Android apps. This includes the zipalign tool for instance, which optimizes the app to use minimal memory when running prior to generating the final APK, and the apksigner which signs the APK surprise! The Platform tools are more specifically suited to the version of Android that you want to target. Generally, it is best to install the latest Platform tools, which will be installed by default. After first installation though, you need to keep your Platform-tools constantly updated. The tools should be backwards compatible, meaning that you will still be able to support older versions of Android. Anatomy of an app: It relies on Platform-tools in order to understand the Android version that is being used on said device and hence it is included in the Platform-tools package. You can use ADB to access shell tools such as logcat, to query your device ID or even to install apps. The Android emulator is what lets you test and monitor apps on a PC, without necessarily needing to have a device available. To use this, you also get an Android system image designed to run on PC hardware. I also recommend this resource on the build process that will help put the SDK into a little more context. They provide a kind of bridge between Android Studio and a physical device or emulator so that your app can be appropriately packaged and then tested as you develop. For the most part, you can leave the SDK alone: Android Studio will recommend necessary updates and it will call upon the required components when you hit Run or Build APK. That said, a few of the tools are also directly accessible, which will be used for things like updating the SDK, or directly monitoring and communicating with your Android device. If you are following along with an Android development tutorial, then you might sometimes get directed here in order to ensure that specific components are up-to-date. This lets you build your own emulators. This works with either an emulator or a connected device and will go a little deeper in monitoring the way your Android device and app are behaving. To do this, you will need to find your Android SDK installation folder and navigate to the platform-tools directory. On Windows, hold shift and right click anywhere in the folder to open a command line. On Mac, just open Terminal from Launchpad usually found in the Other folder. Now you can use a number of commands. You can find a list of the ADB commands here. Accessing the Documentation Looking for a specific Android development tutorial? There was a time when the Android SDK would also come packaged with a selection of useful sample projects. Today this is no longer the case, but you can find them instead by opening Android Studio and navigating to File â€” New â€” Import Sample. You may wish to use another IDE Integrated Development Environment , for instance if you want to streamline the process of making a 3D game in which case, you may wish to use Unity or Unreal , or if you are interested in cross platform mobile development in which case you might use Xamarin. You can also find the location of the Android SDK in Android Studio, in case you should ever need to move it, or just for your own reference. Just go to File â€” Project Structure. Be aware that this folder is hidden on Windows by default, so you might have a hard time finding it. This gives you access to certain libraries and can help to squeeze a little more performance out of a device â€” making it useful for game development, among other things. As mentioned, if it is just the SDK you are interested in, then you can download this on its own by visiting the downloads page and then choosing to include the sdkmanager. This will allow you to update the SDK through the command line. But for the vast majority of users, it makes a lot more sense to install the full suite and enjoy the graphical interface and other conveniences â€” even if you intend on using a different IDE for development. And this is the really good news: Android development is now easier than ever before thanks to the leaps and bounds that Google has made with Android Studio. There was a time when setting everything up was considerably more complex. There has never been a better time to start Android development!

Chapter 7 : Android Hello World tutorial using Eclipse for beginners - Step by Step Android SDK implemen

This tutorial explains Android beginners how to create an Android Project in Eclipse, work with resources, and create some first code. Setup Eclipse and the Android SDK If you don't already have a running environment to develop Android apps, follow the instructions at this link.

It primarily runs on cell phones, but Android devices do so much more than just act as a phone. The Android open-source software stack consists of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation. Google Android platform is fast becoming one of the most popular development platforms for your mobile devices. As more people are using their mobile devices for things normally done from a laptop or desktop means your customers or other internal employees such as sales people will soon start expecting you to offer them the capability to communicate with your data and business logic through mobile devices. The best part, you can download them for free: You can skip this step and go to Step 3. After that it will install the Android SDK. Then go and take care of Eclipse in Step 2. Step by Step to get Hello World! You can go to the Oracle website http: Here is a direct link to the JDK 1. You can go to http: In that folder you will find the eclipse application a big blue dot. We recommend you create a shortcut on the desktop to simplify the launching of eclipse. Notice that unlike Java, Eclipse does not have an installation process. Once you have unzipped the file you are done. After being launched Eclipse will ask you to specify the workspace to use. The workspace is a folder used by eclipse to keep all your work. Specify an already existing folder or accept the default provided by Eclipse or provide a new folder. You are all set for now. If you want you can read the manuals or click the curved arrowed you will eclipse IDE. Download the Android SDK.

Chapter 8 : Android - A beginner's guide - CodeProject

Android i About the Tutorial Android is an open-source, Linux-based operating system for mobile devices such as smartphones and tablet computers.

A view in Android represents a widget, e. All views in Android extend the android. The main packages for views are: View allows to add data to each widget via the setTag Object and setTag int key, Object tag. To retrieve this information use the getTag and getTag int key methods. A layout manager is responsible for the layout of itself and its child views. The base class for these layout managers is the android. Layout managers can be nested to create complex layouts. The most relevant layout managers in Android are: Children can also define attributes which may be evaluated by their parent layout. Performance considerations with layouts Calculating the layout and drawing the views is a resource intensive operation. You should use the simplest layout possible to achieve good performance. For example, you should avoid nesting layout managers too deeply or avoid using complex layout managers in case a simple layout manager is sufficient. A fast and still powerful layout manager in ConstraintLayout. Layout files Android activities define their user interface with views widgets and fragments. You can also mix both approaches. Defining layouts via XML layout files is the preferred way. This separates the programming logic from the layout definition. It also allows the definition of different layouts for different devices. A layout resource file is referred to as layout. The following code is an example for a simple layout file. Views can define their size. This can be done in units of measurement or via pre-defined layout values. For example, as dp. The effect of these elements is demonstrated in the following graphics. ConstraintLayout ConstraintLayout is provided by an external library. It allows you to use a flat view hierarchy and has great performance. Also the design tools support constraint layout very well. New projects should prefer the usage of constraint layout. ConstraintLayout allows you to define constraints for views. By setting the width of TextView1 to 0dp the view expands to fulfill its horizontal constraints. Instead use 0dp to make the view fulfilling its constraints. There are several attributes in ConstraintLayout to define the size or position of a view. To size elements you can define an aspect ratio e. To define an aspect ratio one dimension has to be set to 0dp match constraints. In xml you can use app: To align elements which size change dynamically you can define a barrier. To position multiple elements at once you can define a chain. A chain groups multiple elements. FrameLayout FrameLayout is a layout manager which draws all child elements on top of each other. This allows to create nice visual effects. The following screenshot shows the Gmail application which uses FrameLayout to display several button on top of another layout. LinearLayout puts all its child elements into a single column or row depending on the android: Possible values for this attribute are horizontal and vertical. If horizontal is used, the child elements are layouted as indicated by the following picture. Vertical would result in a layout as depicted in the following picture. LinearLayout can be nested to achieve more complex layouts. LinearLayout supports assigning a weight to individual children via the android: This value specifies how much of the extra space in the layout is allocated to the corresponding view. RelativeLayout RelativeLayout allows positioning the widget relative to each other. This can be used for complex layouts. RelativeLayout is a complex layout manager and should only be used if such a complex layout is required, as it performs a resource intensive calculation to layout its children. A simple usage for RelativeLayout is if you want to center a single component. Just add one component to the RelativeLayout and set the android: GridLayout separates its drawing area into: You can specify how many columns the grid should have. For each view you can specify in which row and column it should be placed and how many columns and rows it should use. If not specified, GridLayout uses defaults, e. The following layout file defines a layout using GridLayout. ScrollView The ScrollView or the HorizontalScrollView class is useful to make views available, even if they do not fit onto the screen. A scroll view can contain one view, e. If the child view is too large, scroll view allows scrolling the content. The following code shows an example layout file which uses a ScrollView.

Chapter 9 : Step-by-step guide to Android development with Eclipse

DOWNLOAD PDF ANDROID TUTORIALS FOR BEGINNERS IN ECLIPSE

The series starts with the very basics of Android (and Eclipse) but my personal feeling is that if you are a total stranger to Android, the journey will be too hard.