Chapter3: DESCRIPTION OF TOOL

3.1 Android Studio

Android Studio is the official IDE for android application development. It works based on IntelliJ IDEA, You can download the latest version of android studio from Android Studio 2.2 Download, If you are new to installing Android Studio on windows, you will find a file, which is named as android-studio-bundle-143.3101438-windows.exe.So just download and run on windows machine according to android studio wizard guideline.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.1 released in March 2018.

The following features are provided in the current stable version:-

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ and PyCharm e.g. Python and Kotlin and Android Studio 3.0 supports "Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features.

Everything you need to build on Android

Android Studio is Android's official IDE. It is purpose built for Android to accelerate your development and help you build the highest-quality apps for every Android device.

It offers tools custom-tailored for Android developers, including rich code editing, debugging, testing, and profiling tools.

Code and Iterate Faster Than Ever

Based on Intellij IDEA, Android Studio provides the fastest possible turnaround on your coding and running workflow.

Instant Run

Android Studio's Instant Run feature pushes code and resource changes to your running app. It intelligently understands the changes and often delivers them without restarting your app or rebuilding your APK, so you can see the effects immediately.

Intelligent code editor

The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis. As you type, Android Studio provides suggestions in a dropdown list. Simply press Tab to insert the code.

Fast and feature-rich emulator

The Android Emulator installs and starts your apps faster than a real device and allows you to prototype and test your app on various Android device configurations: phones, tablets, Android Wear, and Android TV devices. You can also simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.

Configure Builds Without Limits

Android Studio's project structure and Gradle-based builds provide the flexibility you need to generate APKs for all device types.

Robust and flexible build system

Android Studio offers build automation, dependency management, and customizable build configurations. You can configure your project to include local and hosted libraries, and define build variants that include different code and resources, and apply different code shrinking and app signing configurations.

Designed for teams

Android Studio integrates with version control tools, such as GitHub and Subversion, so you can keep your team in sync with project and build changes. The open source Gradle build system allows you to tailor the build to your environment and run on a continuous integration server such as Jenkins.

Optimized for all Android devices

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug.

Code with Confidence

At every step, Android Studio helps ensure that you're creating the best code possible.

Code templates and sample apps

Android Studio includes project and code templates that make it easy to add well-established patterns such as a navigation drawer and view pager. You can start with a code template or even right-click an API in the editor and select Find Sample Code to search for examples. Moreover, you can import fully functional apps from GitHub, right from the Create Project screen.

Lintelligence

Android Studio provides a robust static analysis framework and includes over 280 different lint checks across the entirety of your app. Additionally, it provides several quick fixes that help you address issues in various categories, such as performance, security, and correctness, with a single click.

Testing tools & frameworks

Android Studio provides extensive tools to help you test your Android apps with JUnit 4 and functional UI test frameworks. With Espresso Test Recorder, you can generate UI test code by recording your interactions with the app on a device or emulator. You can run your tests on a device, an emulator, a continuous integration environment, or in Firebase Test Lab.

Create Rich and Connected Apps

Android Studio knows not all code is written in Java and not all code runs on the user's device.

C++ and NDK support

Android Studio fully supports editing C/C++ project files so you can quickly build JNI components in your app. The IDE provides syntax highlighting and refactoring for C/C++, and an LLDB-based debugger that allows you to simultaneously debug your Java and C/C++ code. The build tools can also execute your CMake and ndk-build scripts without any modification and then add the shared objects to your APK.

Firebase and Cloud integration

The Firebase Assistant helps you connect your app to Firebase and add services such as Analytics, Authentication, Notifications and more with step- by-step procedures right inside Android Studio. Built-in tools for Google Cloud Platform also help you integrate your Android app with services such as Google Cloud Endpoints and project modules specially-designed for Google App Engine.

Eliminate Tiresome Tasks

Android Studio provides GUI tools that simplify the less interesting parts of app development.

Layout Editor

When working with XML layout files, Android Studio provides a drag-and-drop visual editor that makes it easier than ever to create a new layout. The Layout Editor was built in unison with the ConstraintLayout API, so you can quickly build a layout that adapts to different screen sizes by dragging views into place and then adding layout constraints with just a few clicks.

APK Analyzer

You can use the APK Analyzer to easily inspect the contents of your APK. It reveals the size of each component so you can identify ways to reduce the overall APK size. It also allows you preview packaged assets, inspect the DEX files to troubleshoot multidex issues, and compare the differences between two APKs.

Vector Asset Studio

Android Studio makes it easy to create a new image asset for every density size. With Vector Asset Studio, you can select from Google-provided material design icons or import an SVG or PSD file. Vector Asset Studio can also generate bitmap files for each screen density to support older versions of Android that don't support the Android vector drawable format.

Translations Editor

The Translations Editor gives you a single view of all of your translated resources, making it easy to change or add translations, and to find missing translations without opening each version of the strings.xml file. It even provides a link to order translation services.

3.1.1 JDK

The Java Development Kit (JDK) is an implementation of either one of the Java Platform, Standard Edition, Java Platform, Enterprise Edition, or Java Platform, Micro Edition platforms[1] released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, macOS or Windows. The JDK includes a private JVM and a few other resources to finish the development of a Java Application. Since the introduction of the Java platform, it has been by far the most widely used Software Development Kit (SDK). On 17 November 2006, Sun announced that they would release it under the GNU General Public License (GPL), thus making it free software. This happened in large part on 8 May 2007, when Sun contributed the source code to the OpenJDK.

JDK CONTENTS

The JDK has as its primary components a collection of programming tools, including:

- appletviewer – this tool can be used to run and debug Java applets without a web browser
- apt – the annotation-processing tool
- extcheck – a utility that detects JAR file conflicts
- idlj – the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.
- jabswitch – the Java Access Bridge. Exposes assistive technologies on Microsoft Windows systems.
- java – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
- javac – the Java compiler, which converts source code into Java bytecode
- javadoc – the documentation generator, which automatically generates documentation from source code comments
- jar – the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
- javafxpackager – tool to package and sign JavaFX applications

- jarsigner – the jar signing and verification tool
- javah – the C header and stub generator, used to write native methods
- javap – the class file disassembler
- javaws – the Java Web Start launcher for JNLP applications
- JConsole – Java Monitoring and Management Console
- jdb – the debugger
- jhat – Java Heap Analysis Tool (experimental)
- jinfo – This utility gets configuration information from a running Java process or crash dump. (experimental)
- jmap Oracle jmap - Memory Map– This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump. (experimental)
- jmc – Java Mission Control
- jps – Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system. (experimental)
- jrunscript – Java command-line script shell.
- jstack – utility that prints Java stack traces of Java threads (experimental)
- jstat – Java Virtual Machine statistics monitoring tool (experimental)
- jstatd – jstat daemon (experimental)
- keytool – tool for manipulating the keystore
- pack200 – JAR compression tool
- policytool – the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources.
- VisualVM – visual tool integrating several command-line JDK tools and lightweigh performance and memory profiling capabilities
- wsimport – generates portable JAX-WS artifacts for invoking a web service.
- xjc – Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

Experimental tools may not be available in future versions of the JDK.

The JDK also comes with a complete Java Runtime Environment, usually called a private runtime, due to the fact that it is separated from the "regular" JRE and has extra contents.

It consists of a Java Virtual Machine and all of the class libraries present in the production environment, as well as additional libraries only useful to developers, such as the internationalization libraries and the IDL libraries.

Copies of the JDK also include a wide selection of example programs demonstrating the use of almost all portions of the Java API.

Ambiguity between JDK and SDK

The JDK forms an extended subset of a software development kit (SDK). It includes "tools for developing, debugging, and monitoring Java applications".[5] Oracle strongly suggests to now use the term JDK to refer to the Java SE Development Kit. The Java EE SDK is available with or without the JDK, by which they specifically mean the Java SE 7 JDK.

OTHER JDKs

In addition to the most widely used JDK discussed in this article, there are other JDKs commonly available for a variety of platforms, some of which started from the Sun JDK source and some that did not. All adhere to the basic Java specifications, but often differ in explicitly unspecified areas, such as garbage collection, compilation strategies, and optimization techniques. They include:

In development or in maintenance mode:

- Azul Systems Zing, low latency JDK for Linux;
- Azul Systems / OpenJDK-based Zulu for Linux, Windows, Mac OS X, embedded and the cloud;
- OpenJDK / IcedTea;
- GNU's Classpath and GCJ (The GNU Compiler for Java);
- Aicas JamaicaVM
- IBM J9 JDK, for AIX, Linux, Windows, MVS, OS/400, Pocket PC, z/OS;

Not being maintained or discontinued:

- Apache Harmony
- Apple's Mac OS Runtime for Java JVM/JDK for Classic Mac OS
- Blackdown Java – Port of Sun's JDK for Linux
- Oracle Corporation's JRockit JDK, for Windows, Linux, and Solar

3.2 Database

SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.[6] SQLite has bindings to many programming languages

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite libraryis linked in and thus becomes an integral part of the application program. The library can also be called dynamically. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication. SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

Due to the server-less design, SQLite applications require less configuration than client-server databases. SQLite is called zero-conf because it does not require service management (such as startup scripts) or access control based on GRANT and passwords. Access control is handled by means of file system permissions given to the database file itself. Databases in client-server systems use file system permissions which give access to the database files only to the daemon process.

Another implication of the serverless design is that several processes may not be able to write to the database file. In server-based databases, several writers will all connect to the same daemon, which is able to handle its locks internally. SQLite on the other hand has to rely on file-system locks. It has less knowledge of the other processes that are accessing the database at the same time. Therefore, SQLite is not the preferred choice for write-intensive deployments. However, for simple queries with little concurrency, SQLite performance profits from avoiding the overhead of passing its data to another process.

SQLite uses PostgreSQL as a reference platform. “What would PostgreSQL do” is used to make sense of the SQL standard. One major deviation is that, with the exception of primary keys, SQLite does not enforce type checking; the type of a value is dynamic and not strictly constrained by the schema (although the schema will trigger a conversion when storing, if such a conversion is potentially reversible). SQLite strives to follow Postel's Rule

Features

SQLite implements most of the SQL-92 standard for SQL but it lacks some features. For example, it partially provides triggers, and it cannot write to views (however it provides INSTEAD OF triggers that provide this functionality). While it provides complex queries, it still has limited ALTER TABLE function, as it cannot modify or delete columns.

SQLite uses an unusual type system for an SQL-compatible DBMS; instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is dynamically typed. Moreover, it is weakly typed in some of the same ways that Perl is: one can insert a string into an integer column (although SQLite will try to convert the string to an integer first, if the column's preferred type is integer). This adds flexibility to columns, especially when bound to a dynamically typed scripting language. However, the technique is not portable to other SQL products. A common criticism is that SQLite's type system lacks the data integrity mechanism provided by statically typed columns in other products. The SQLite web site describes a "strict affinity" mode, but this feature has not yet been added. However, it can be implemented with constraints like `CHECK(typeof(x)='integer')`.

SQLite with full Unicode function is optional.

Several computer processes or threads may access the same database concurrently. Several read accesses can be satisfied in parallel. A write access can only be satisfied if no other accesses are currently being serviced. Otherwise, the write access fails with an error code (or can automatically be retried until a configurable timeout expires). This concurrent access situation would change when dealing with temporary tables. This restriction is relaxed in version 3.7 when write-ahead logging (WAL) is turned on enabling concurrent reads and writes.

SQLite version 3.7.4 first saw the addition of the FTS4 (full text search) module, which features enhancements over the older FTS3 module. FTS4 allows users to perform full text searches on documents similar to how search engines search webpages. Version 3.8.2 added support for creating tables without rowid, which may provide space and performance improvements. Common table expressions support was added to SQLite in version 3.8.3.

In 2015, with the json1 extension and new subtype interfaces, SQLite version 3.9 introduced JSON content managing.

Development

SQLite's code is hosted with Fossil, a distributed version control system that is itself built upon an SQLite database.

A standalone command-line program is provided in SQLite's distribution. It can be used to create a database, define tables, insert and change rows, run queries and manage an SQLite database file. It also serves as an example for writing applications that use the SQLite library.

SQLite uses automated regression testing prior to each release. Over 2 million tests are run as part of a release's verification. Starting with the August 10, 2009 release of SQLite 3.6.17, SQLite releases have 100% branch test coverage, one of the components of code coverage. The tests and test harnesses.