An application programming interface (API), in the context of Java, is a collection of prewritten packages, classes, and interfaces with their respective methods, fields and constructors. Similar to a user interface, which facilitates interaction between humans and computers, an API serves as a software program interface facilitating interaction.

In Java, most basic programming tasks are performed by the API’s classes and packages, which are helpful in minimizing the number of lines written within pieces of code.

Java Development Kit (JDK) is comprised of three basic components, as follows:

Java compiler

Java Virtual Machine (JVM)

Java Application Programming Interface (API)

The Java API, included with the JDK, describes the function of each of its components. In Java programming, many of these components are pre-created and commonly used. Thus, the programmer is able to apply prewritten code via the Java API. After referring to the available API classes and packages, the programmer easily invokes the necessary code classes and packages for implementation.

The JavaScript Object Notation (JSON) file format is a text-based, open standard format which is used to serialize and transmit structured data between a server and web application. The JSON format is easy for humans to read and write. It is also easy for machines to parse and generate. Although it is based on a subset of the JavaScript programming language, it is completely language independent. The JSON format is smaller, faster and easier to parse than XML. Because of these properties, the JSON format is the ideal data-interchange language.

DBMS AND RDBMS

* SQL databases are primarily called as Relational Databases (RDBMS); whereas NoSQL database are primarily called as non-relational or distributed database.
* SQL databases have predefined schema whereas NoSQL databases have dynamic schema for unstructured data.
* SQL database examples: MySql, Oracle, Sqlite, Postgres and MS-SQL. NoSQL database examples: MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb
* import java.io.File;
* import java.io.IOException;
* import jxl.Cell;
* import jxl.Sheet;
* import jxl.Workbook;
* import jxl.read.biff.BiffException;
* public class NewExcel
* {
* private String inputFile;
* String[][] data = null;
* public void setInputFile(String inputFile)
* {
* this.inputFile = inputFile;
* }
* public String[][] read() throws IOException
* {
* File inputWorkbook = new File(inputFile);
* Workbook w;
* try
* {
* w = Workbook.getWorkbook(inputWorkbook);
* // Get the first sheet
* Sheet sheet = w.getSheet(0);
* data = new String[sheet.getColumns()][sheet.getRows()];
* // Loop over first 10 column and lines
* // System.out.println(sheet.getColumns() + " " +sheet.getRows());
* for (int j = 0; j <sheet.getColumns(); j++)
* {
* for (int i = 0; i < sheet.getRows(); i++)
* {
* Cell cell = sheet.getCell(j, i);
* data[j][i] = cell.getContents();
* // System.out.println(cell.getContents());
* }
* }
* /\* for (int j = 0; j < data.length; j++)
* {
* for (int i = 0; i <data[j].length; i++)
* {
* System.out.println(data[j][i]);
* }
* } \*/
* }
* catch (BiffException e)
* {
* e.printStackTrace();
* }
* return data;
* }
* }

**WebDriver** is a web automation framework that allows you to execute your tests against different browsers, not just Firefox (unlike Selenium IDE).

driver.findElement(By.linkText("ToolsQA")).click();

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | WebElement element = driver.findElement(By.id("UserName"));  boolean status = element.isDisplayed();  ***driver.findElement(By.id(“Element ID”))***  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  public class FindElementCommands1 {  public static void main(String[] args) {  // Create a new instance of the FireFox driver  WebDriver driver = new FirefoxDriver();    // Launch the ToolsQA WebSite  driver.get("http://toolsqa.wpengine.com/Automation-practice-form/");    // Type Name in the FirstName text box  driver.findElement(By.name("firstname")).sendKeys("Lakshay");    //Type LastName in the LastName text box  driver.findElement(By.name("lastname")).sendKeys("Sharma");    // Click on the Submit button  driver.findElement(By.id("submit")).click();  WebElement radioBtn = driver.findElement(By.id("toolsqa"));  radioBtn.click();  Dropdown Menu::  Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));  oSelect.selectByValue("2014");  Maven allows a project to build using its project object model (POM) and a set of plugins that are shared by all projects using Maven, providing a uniform build system. Once you familiarize yourself with how one Maven project builds you automatically know how all Maven projects build saving you immense amounts of time when trying to navigate many projects.  **Maven** is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process like ANT. But it is too much advanced than ANT.  TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as:   * Annotations. * Run your tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...). * Test that your code is multithread safe. * Flexible test configuration. * Support for data-driven testing (with @DataProvider). * Support for parameters. * Powerful execution model (no more TestSuite). * Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...). * Embeds BeanShell for further flexibility. * Default JDK functions for runtime and logging (no dependencies). * Dependent methods for application server testing.   @Test  **public** **void** **registerAccount**()  {  System.**out**.println("First register your account");  }  @Test(priority=2)  **public** **void** **sendEmail**()  {  System.**out**.println("Send email after login");  }  @Test(priority=1)  **public** **void** **login**()  {  System.**out**.println("Login to the account after registration");  }  }  **Maven** does so by reading project files (pom.xml) of **dependencies**, figure out their **dependencies** and so on. We only need to define direct **dependency** in each project pom. **Maven** handles the rest automatically. With transitive **dependencies**, the graph of included libraries can quickly grow to a large extent. |

As today the Popularity of handheld devices are growing at an incredible pace. Database sizes for small applications need to store more data than many databases were meant to handle. As the amount of data that developers need to store grows, developers difficulty for scaling their databases is also growing. Either they should go for scale-up (i.e getting a bigger machine) or scale-out (partitioning data across more machines). We can scale-up upto a certain limit because large machines are expensive, and physical limit can be reached where a more powerful machine cannot be purchased at any cost. So we need to go for scale-up option for our databases.

MongoDB was designed from the beginning to scale out. Its document-oriented data model allows it to automatically split up data across multiple servers. It can balance data and load across a cluster, redistributing documents automatically. MongoDB tries to simplify database administration by making servers administrate themselves as much as possible

Getting started with MongoDB

MongoDB is a document-oriented database, not a relational one. Non relational means it does not store data in tables but in the form of JSON document. The primary reason for moving away from the relational model is to make scaling out easier. Here in MongoDB row is replaced with document and table is replaced with collection. It can be thought of as a group of documents.

Document is the basic unit of data for MongoDB, roughly equivalent to a row. It is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values maybe documents, arrays, and arrays of documents.

Hide   Copy Code

{

"name": "binit",

"age": "24",

"subject": ["mathematics", "science", "litrature"]

}