Implementation Details

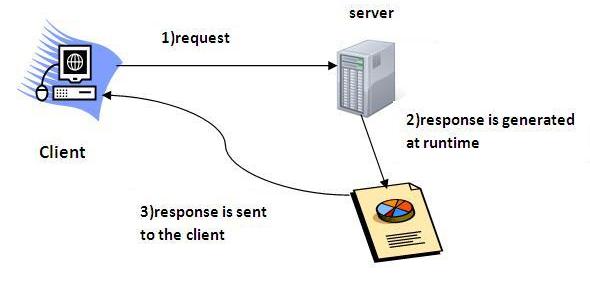
**ServerSide**

**Servlet:**

**What is a Servlet?**

Servlet can be described in many ways, depending on the context.

* Servlet is a technology i.e. used to create web application.
* Servlet is an API that provides many interfaces and classes including documentations.
* Servlet is an interface that must be implemented for creating any servlet.
* Servlet is a class that extend the capabilities of the servers and respond to the incoming request. It can respond to any type of requests.
* Servlet is a web component that is deployed on the server to create dynamic web page.



# How Servlet works?

When we run our project there are mane web pages like index, home, login, registration etc…But how would server knows which is the first file to run .When we run our project the first file is web.xml. In that we mention in welcome list File which is the first page to run.



From this it shows first page as [index.jsp. If](http://home.jsp.If) in home page we write one form, then it search the action tag.

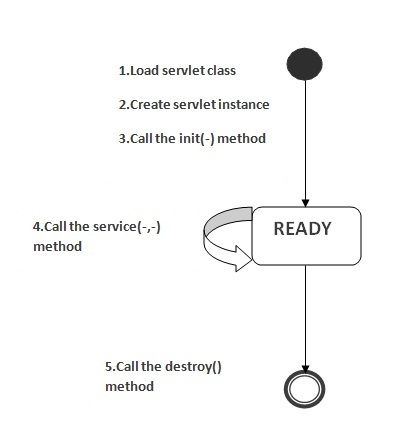
The public service method internally calls the protected service method

The protected service method calls the doGet or doPost method depending on the type of request.

# Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



**JSP**

JavaServer Pages (**JSP**) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types.

## What is JavaServer Pages?

JavaServer Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.

A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

## Why Use JSP?

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

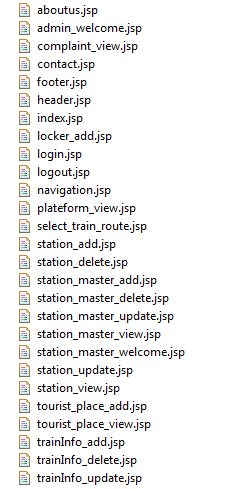
* Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
* JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
* JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.
* JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

**Advantages of JSP:**

Following is the list of other advantages of using JSP over other technologies:

* **Active Server Pages (ASP):** The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
* **Pure Servlets:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.
* **Server-Side Includes (SSI):** SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

**JSP Pages our project:**



**Apache Tomcat:**

**Apache Tomcat** is an open-source web server and [servlet](http://en.wikipedia.org/wiki/Java_Servlet) [container](http://en.wikipedia.org/wiki/Web_container) developed by the [Apache Software Foundation](http://en.wikipedia.org/wiki/Apache_Software_Foundation) (ASF). Tomcat implements several [Java EE](http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition) specifications including [Java Servlet](http://en.wikipedia.org/wiki/Java_Servlet), [JavaServer Pages](http://en.wikipedia.org/wiki/JavaServer_Pages) (JSP), [Java EL](http://en.wikipedia.org/wiki/Unified_Expression_Language), and [WebSocket](http://en.wikipedia.org/wiki/WebSocket), and provides a "pure [Java](http://en.wikipedia.org/wiki/Java_(programming_language))" [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) [web server](http://en.wikipedia.org/wiki/Web_server) environment for [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) code to run in.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the [Apache License](http://en.wikipedia.org/wiki/Apache_License) 2.0 license, and is [open-source software](http://en.wikipedia.org/wiki/Open-source_software).

### Catalina

Catalina is Tomcat's [servlet container](http://en.wikipedia.org/wiki/Web_container). Catalina implements [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems)' specifications for [servlet](http://en.wikipedia.org/wiki/Java_servlet) and [JavaServer Pages](http://en.wikipedia.org/wiki/JavaServer_Pages) (JSP). In Tomcat, a Realm element represents a "database" of usernames, passwords, and roles (similar to Unix groups) assigned to those users. Different implementations of Realm allow Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification.[[2]](http://en.wikipedia.org/wiki/Apache_Tomcat#cite_note-2)

### Coyote

Coyote is a Connector component for Tomcat that supports the HTTP 1.1 protocol as a web server. This allows Catalina, nominally a Java Servlet or JSP container, to also act as a plain web server that serves local files as HTTP documents.[[3]](http://en.wikipedia.org/wiki/Apache_Tomcat#cite_note-3)

Coyote listens for incoming connections to the server on a specific [TCP](http://en.wikipedia.org/wiki/Transmission_Control_Protocol) port and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client. Another Coyote Connector, Coyote JK, listens similarly but instead forwards its requests to another web server, such as Apache, using the [JK protocol](http://en.wikipedia.org/wiki/Apache_JServ_Protocol).[[4]](http://en.wikipedia.org/wiki/Apache_Tomcat#cite_note-4)This usually offers better performance.

### Jasper

Jasper is Tomcat's JSP Engine. Jasper parses [JSP files](http://en.wikipedia.org/wiki/JSP_files) to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.

As of version 5, Tomcat uses Jasper 2, which is an implementation of the [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems)'s [JSP](http://en.wikipedia.org/wiki/JavaServer_Pages) 2.0 specification. From Jasper to Jasper 2, important features were added:

* JSP Tag library pooling - Each tag markup in JSP file is handled by a tag handler class. Tag handler class objects can be pooled and reused in the whole JSP servlet.
* Background JSP compilation - While recompiling modified JSP Java code, the older version is still available for server requests. The older JSP servlet is deleted once the new JSP servlet has finished being recompiled.
* Recompile JSP when included page changes - Pages can be inserted and included into a JSP at runtime. The JSP will not only be recompiled with JSP file changes but also with included page changes.
* JDT Java compiler - Jasper 2 can use the Eclipse JDT (Java Development Tools) Java compiler instead of [Ant](http://en.wikipedia.org/wiki/Apache_Ant) and [javac](http://en.wikipedia.org/wiki/Javac).

Three new components were added with the release of Tomcat 7:

### Cluster

This component has been added to manage large applications. It is used for [load balancing](http://en.wikipedia.org/wiki/Load_balancing_(computing)) that can be achieved through many techniques. Clustering support currently requires the JDK version 1.5 or later.

### High availability

A high-availability feature has been added to facilitate the scheduling of system upgrades (e.g. new releases, change requests) without affecting the live environment. This is done by dispatching live traffic requests to a temporary server on a different port while the main server is upgraded on the main port. It is very useful in handling user requests on high-traffic web applications.[]](http://en.wikipedia.org/wiki/Apache_Tomcat#cite_note-5)

### Web Application

It has also added user- as well as system-based web applications enhancement to add support for deployment across the variety of environments. It also tries to manage sessions as well as applications across the network.

Tomcat is building additional components. A number of additional components may be used with Apache Tomcat. These components may be built by users should they need them or they can be downloaded from one of the mirrors

**Database**

## What is Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.

So nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.

**A Relational DataBase Management System (RDBMS) is a software that**:

* Enables you to implement a database with tables, columns and indexes.
* Guarantees the Referential Integrity between rows of various tables.
* Updates the indexes automatically.
* Interprets an SQL query and combines information from various tables.

## MySQL Database:

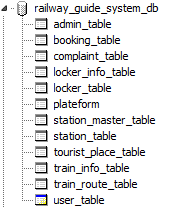
MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

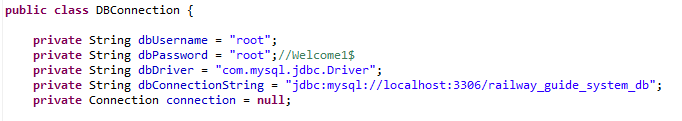
**How database works in our project?**

1 database in our project

Database name: railway\_guide\_system\_db



First connect database through connect method in our project



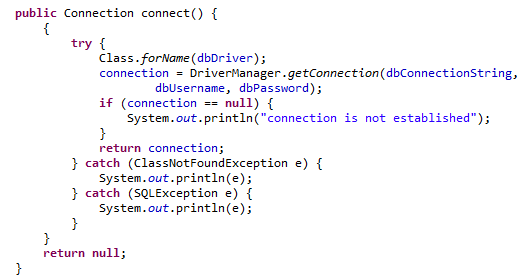
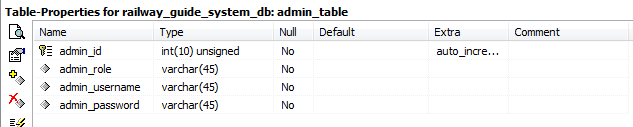
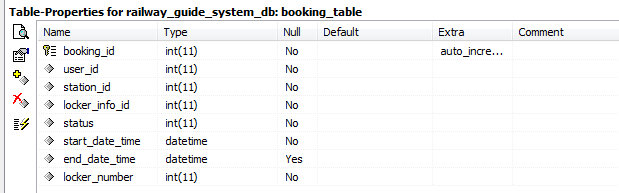
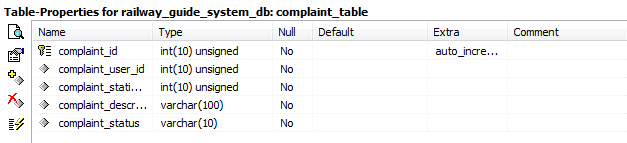
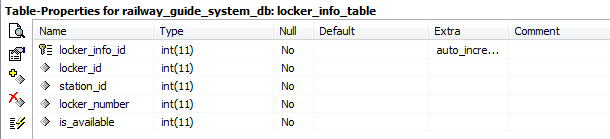


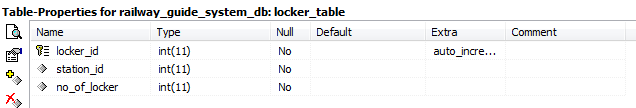
Table schema for railway\_guide\_system\_db database

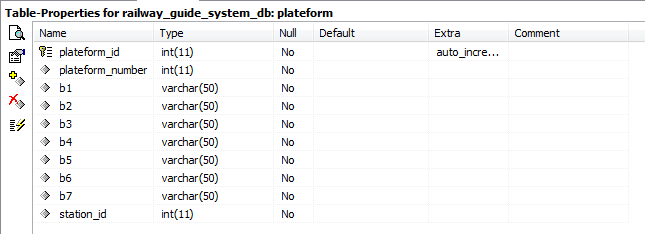


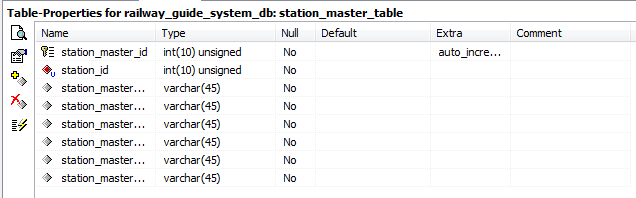


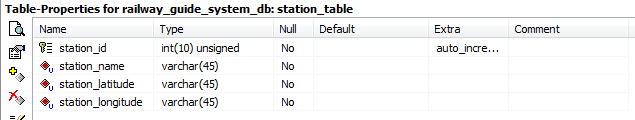


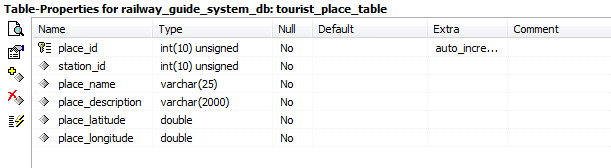


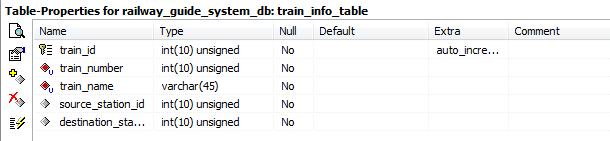


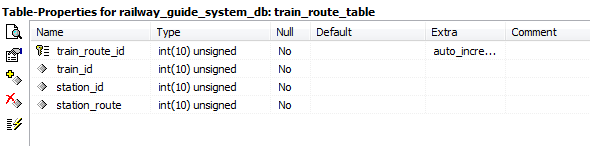


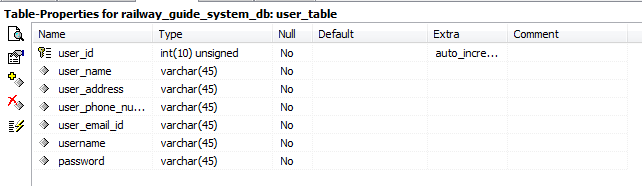




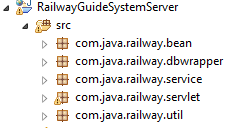








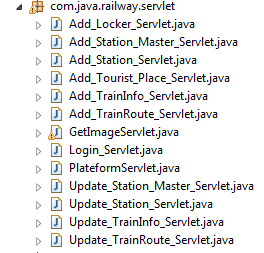
**Class wise description**

****

**Description:**

**Package name: com.java.railway.servlet**

In this package all servlet class is there. Our class extends with httpservlet class.

****

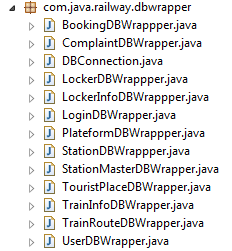
**Login Servlet.java**

This class is used to verify the admin

****

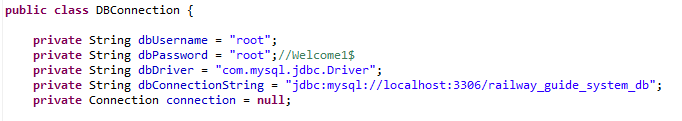
**Package name: com.java.railway.dbwrapper**

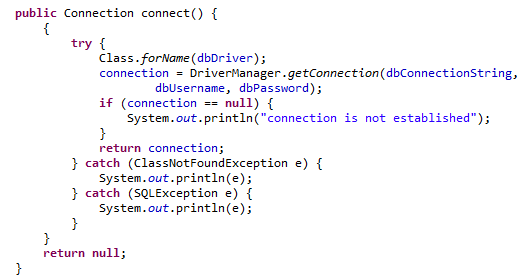
In this package all class is related to database operation. Means various class is used to add, update, view delete station info, train info etc

****

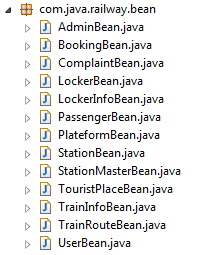
**DBConnection.java**

This is generic class and is used to make connection with Mysql database.





**Package name:com.java.railway.bean**

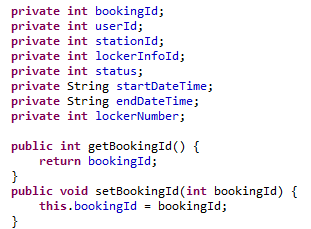


**Admin\_Bean.java**

This class is the place holder class and contains only getter setter methods to access values

**BookingBean.java**

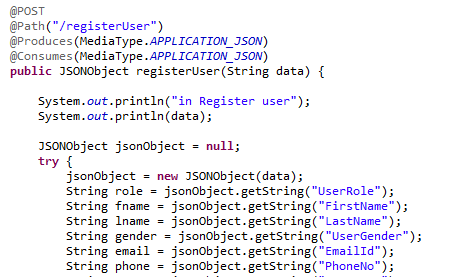
This class is the place holder class and contains only getter setter methods to access values

****

****

**Webservice.java**

This class is responsible for client server Communication. When client sent request to server then it connect from this class



**ClientSide**

**Android** is a software package and linux based operating system for mobile devices such as tablet computers and smartphones.

It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

There are many code names of android such as Lollipop, Kitkat, Jelly Bean, Ice cream Sandwich, Froyo, Ecliar, Donut etc .

**Activity Lifecycle**

Activities in the system are managed as an activity stack. When a new activity is started, it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

An activity has essentially four states:

* If an activity in the foreground of the screen (at the top of the stack), it is active or running.
* If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of our activity), it is paused. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.
* If an activity is completely obscured by another activity, it is stopped. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
* If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

The following diagram shows the important state paths of an Activity. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.

****

**There are three key loops you may be interested in monitoring within our activity:**

The entire lifetime of an activity happens between the first call to onCreate(Bundle) through to a single final call to onDestroy(). An activity will do all setup of "global" state in onCreate(), and release all remaining resources in onDestroy(). For example, if it has a thread running in the background to download data from the network, it may create that thread in onCreate() and then stop the thread in onDestroy().

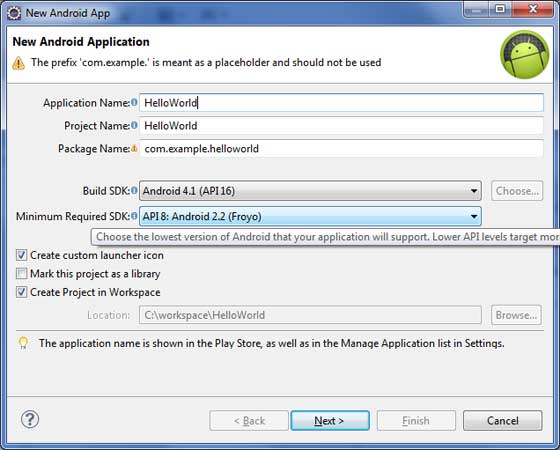
The visible lifetime of an activity happens between a call to onStart() until a corresponding call to onStop(). During this time the user can see the activity on-screen, though it may not be in the foreground and interacting with the user. Between these two methods you can maintain resources that are needed to show the activity to the user. For example, you can register a BroadcastReceiver in onStart() to monitor for changes that impact our UI, and unregister it in onStop() when the user no longer sees what you are displaying. The onStart() and onStop() methods can be called multiple times, as the activity becomes visible and hidden to the user.

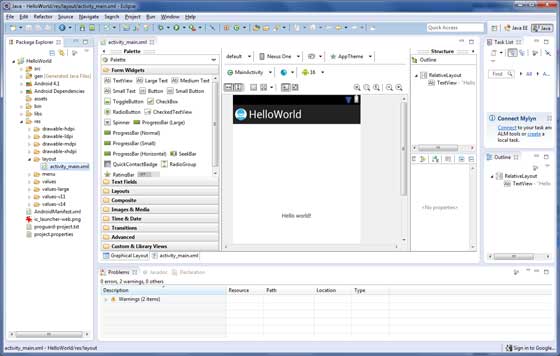
The foreground lifetime of an activity happens between a call to onResume() until a corresponding call to onPause(). During this time the activity is in front of all other activities and interacting with the user. An activity can frequently go between the resumed and paused states -- for example when the device goes to sleep, when an activity result is delivered, when a new intent is delivered -- so the code in these methods should be fairly lightweight.

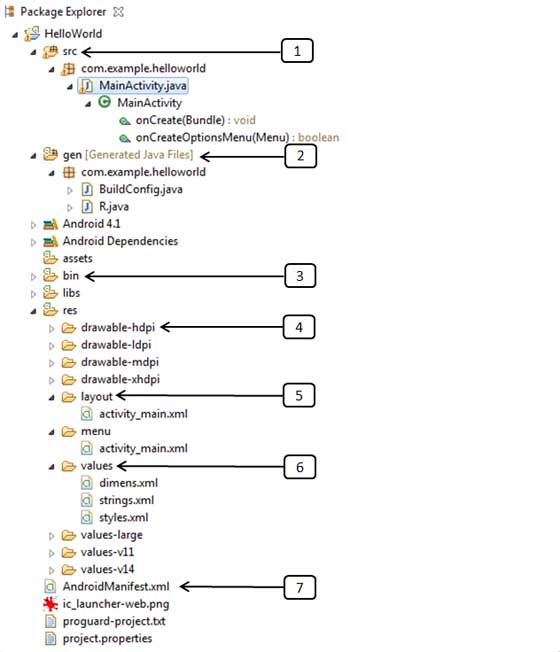
The entire lifecycle of an activity is defined by the following Activity methods. All of these are hooks that you can override to do appropriate work when the activity changes state. All activities will implement onCreate(Bundle) to do their initial setup; many will also implement onPause() to commit changes to data and otherwise prepare to stop interacting with the user. You should always call up to our superclass when implementing these methods.

**How to create program in android**

The first step is to create a simple Android Application using Eclipse IDE. Follow the option **File -> New -> Project** and finally select **Android New Application** wizard from the wizard list. Now name your application as **HelloWorld** using the wizard window as follows:



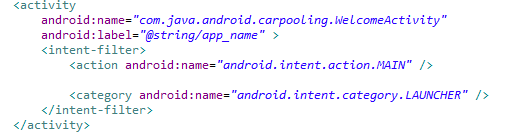




|  |  |
| --- | --- |
| **S.N.** | **Folder, File & Description** |
| 1 | **src** This contains the **.java** source files for your project. By default, it includes anMainActivity*.java* source file having an activity class that runs when your app is launched using the app icon. |
| 2 | **gen** This contains the **.R** file, a compiler-generated file that references all the resources found in your project. You should not modify this file. |
| 3 | **bin** This folder contains the Android package files **.apk** built by the ADT during the build process and everything else needed to run an Android application. |
| 4 | **res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens. |
| 5 | **res/layout** This is a directory for files that define your app's user interface. |
| 6 | **res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions. |
| 7 | **AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components. |

**Working of android project?**

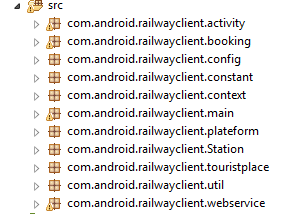
When we run our project first file running is manifest.xml. In manifest file we give class which is our first class from the list of classes.



Then we enter in the class which is mention in manifest. And by using setContentView(R.layout.welcome)we set layout to our class

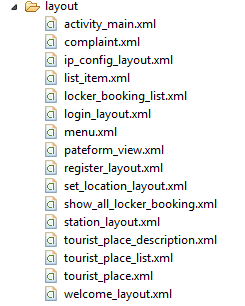
.

**Class wise Description Client Side**

****

**Layout**

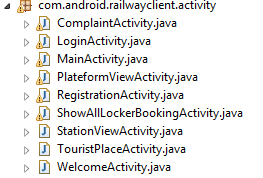
In layout all user interface pages are given

****

**Description:**

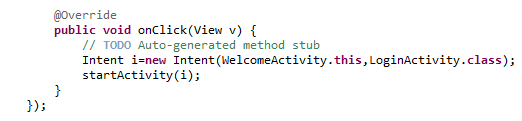
Among the many activity which is the first one is given in manifest file as shown below

****

****

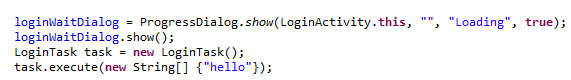
**WelcomeActivty.java:**

This class is first time called. In this layout we gave one buttons for start. And option for entering ip address.



**LoginActivity.java**

In this class we validate the user from server side. First we took data from login page then we use Login Task to communicate with server.

****

**AsyncTask:**

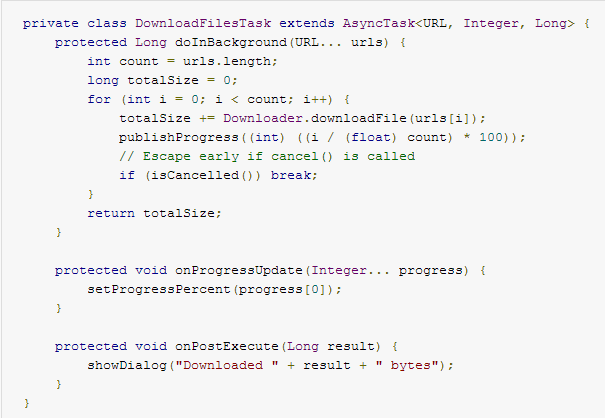
AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the java.util.concurrent pacakge such as Executor, ThreadPoolExecutor and FutureTask.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.

AsyncTask must be subclassed to be used. The subclass will override at least one method ([doInBackground(Params...)](http://developer.android.com/reference/android/os/AsyncTask.html#doInBackground(Params...))), and most often will override a second one ([onPostExecute(Result)](http://developer.android.com/reference/android/os/AsyncTask.html#onPostExecute(Result)).)

Here is an example of subclassing:



**When an asynchronous task is executed, the task goes through 4 steps:**

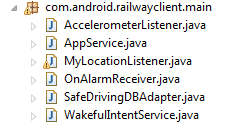
**onPreExecute():** Invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.

**doInBackground(Params...):** Invoked on the background thread immediately after onPreExecute() finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step. This step can also use publishProgress(Progress...) to publish one or more units of progress. These values are published on the UI thread, in the onProgressUpdate(Progress...) step.

**onProgressUpdate(Progress...):** Invoked on the UI thread after a call to publishProgress(Progress...). The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.

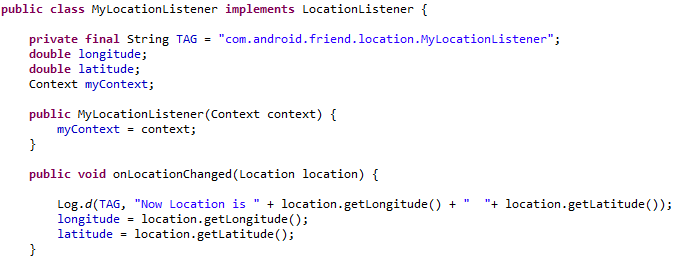
**onPostExecute(Result):** Invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

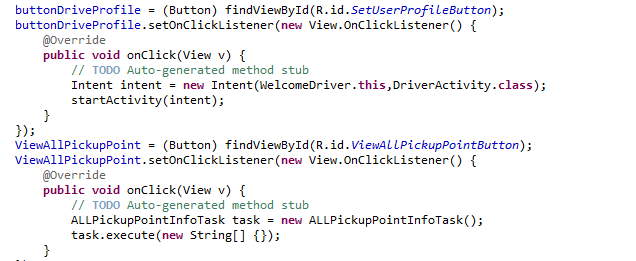
**Package name:com.android.railwayclient.main**

****

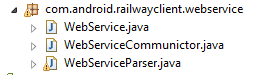
**MyLocationListner.java**

This class is used to get users latitude and location. For getting lat long we extend our class by location listener



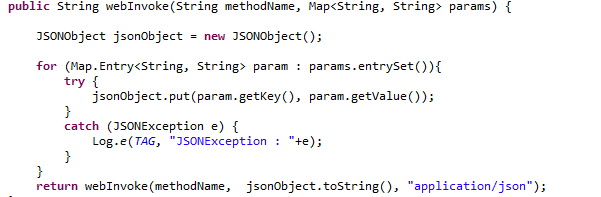
****

**Package name:com.android.railway.webservice**

****

**Webservice.java**

This class is responsible for client server Communication. When client sent request to server then it connect from this class



Web Service:

**Representational State Transfer** (**REST**) is a [software architecture style](http://en.wikipedia.org/wiki/Software_architecture_styles_and_patterns) consisting of guidelines and best practices for creating [scalable](http://en.wikipedia.org/wiki/Scalability) [web services](http://en.wikipedia.org/wiki/Web_service). REST is a coordinated set of constraints applied to the design of components in a distributed [hypermedia](http://en.wikipedia.org/wiki/Hypermedia) system that can lead to a more performant and maintainable [architecture](http://en.wikipedia.org/wiki/Software_architecture).

REST has gained widespread acceptance across the Webas a simpler alternative to [SOAP](http://en.wikipedia.org/wiki/SOAP) and [WSDL](http://en.wikipedia.org/wiki/WSDL)-based Web services. RESTful systems typically, but not always, communicate over the [Hypertext Transfer Protocol](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) with the same [HTTP verbs](http://en.wikipedia.org/wiki/HTTP_verbs) (GET, POST, PUT, DELETE, etc.) used by web browsers to retrieve [web pages](http://en.wikipedia.org/wiki/Web_page) and send data to remote servers.

The REST architectural style was developed by [W3C](http://en.wikipedia.org/wiki/World_Wide_Web_Consortium) Technical Architecture Group (TAG) in parallel with [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) 1.1, based on the existing design of HTTP 1.0.The [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web) represents the largest implementation of a system conforming to the REST architectural style.

**Architectural Constraint**

The architectural properties of REST are realized by applying specific interaction constraints to components, connectors, and data elements. The formal REST constraints are:

### Client–server

A uniform interface separates clients from servers. This [separation of concerns](http://en.wikipedia.org/wiki/Separation_of_concerns) means that, for example, clients are not concerned with data storage, which remains internal to each server, so that the [portability](http://en.wikipedia.org/wiki/Software_portability) of client code is improved. Servers are not concerned with the user interface or user state, so that servers can be simpler and more [scalable](http://en.wikipedia.org/wiki/Scalability). Servers and clients may also be replaced and developed independently, as long as the interface between them is not altered.

### Stateless

The client–server communication is further constrained by no client context being stored on the server between requests. Each request from any client contains all the information necessary to service the request, and session state is held in the client. The session state can be transferred by the server to another service such as a database to maintain a persistent state for a period and allow authentication. The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be *in transition*. The representation of each application state contains links that may be used the next time the client chooses to initiate a new state-transition.

### Cacheable

As on the World Wide Web, clients can cache responses. Responses must therefore, implicitly or explicitly, define themselves as cacheable, or not, to prevent clients from reusing stale or inappropriate data in response to further requests. Well-managed caching partially or completely eliminates some client–server interactions, further improving scalability and performance.

* 1. **Google Map Integration:**

### Google provides this facility using Google play services library which you have to download externally. After downloading, you have to integrate it with your project.In the end one have to integrate your application with Google via Google console

### GOOGLE MAP - ACTIVITY FILE

Google provides GoogleMap and MapFragmentapi to integrate map in your android application. In order to use GoogleMap , you have to create an object of GoogleMap and get the reference of map from the xml layout file.Its syntax is given below:

GoogleMapgoogleMap;

googleMap=((MapFragment)getFragmentManager().findFragmentById(R.id.map)).getMap()

### GOOGLE MAP - LAYOUT FILE

Now you have to add the map fragment into xml layout file. Its syntax is given below:

<fragment

android:id="@+id/map"

android:name="com.google.android.gms.maps.MapFragment"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

### GOOGLE MAP - ANDROIDMANIFEST FILE

The next thing you need to do is to add some permissions along with the Google Map API key in the AndroidManifest.XML file. Its syntax is given below:

<!--Permissions-->

<uses-permissionandroid:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<uses-permissionandroid:name="android.permission.INTERNET"/>

<uses-permissionandroid:name="com.google.android.providers.gsf.permission.READ\_GSERVICES"/>

<uses-permissionandroid:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<!--Google MAP API key-->

<meta-data

android:name="com.google.android.maps.v2.API\_KEY"

android:value="AIzaSyDKymeBXNeiFWY5jRUejv6zItpmr2MVyQ0"/>

## Customizing Google Map

You can easily customize google map from its default view , and change it according to your demand.

### ADDING MARKER

You can place a maker with some text over it displaying your location on the map. It can be done by via**addMarker()** method. Its syntax is given below:

finalLatLngTutorialsPoint=newLatLng(21,57);

Marker TP =googleMap.addMarker(newMarkerOptions().position(TutorialsPoint).title("TutorialsPoint"));

### CHANING MAP TYPE

You can also change the type of the MAP. There are four different types of map and each give different view of the map. These types are Normal,Hybrid,Satellite and terrain. You can use them as below

googleMap.setMapType(GoogleMap.MAP\_TYPE\_NORMAL);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_HYBRID);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_SATELLITE);

googleMap.setMapType(GoogleMap.MAP\_TYPE\_TERRAIN);

### ENABLE/DISABLE ZOOM

You can also enable or disable the zoom gestures in the map by calling the**setZoomControlsEnabled(boolean)** method. Its syntax is given below:

googleMap.getUiSettings().setZoomGesturesEnabled(true);

Integrating google maps in your application basically consists of these 4 steps.

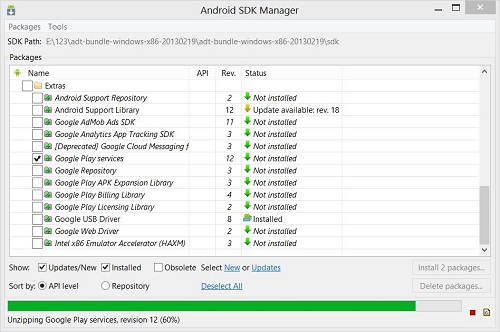
1. Download and configure. Google Play Services SDK
2. Obtain API key from google console
3. Specify Android Manifest settings

## Download and configure. Google Play Services SDK

### INSTALL GOOGLE SERVICES SDK

Open your SDK manager in the eclipse by clicking the Window and then selecting the Android SDK manager.

Navigate to the extras tab and select the Google play services and click on install this package. It would be like this.

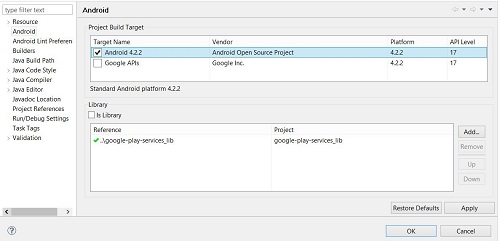


### IMPORT SDK TO ECLIPSE

After you download the SDK , click on file tab and select import option. Select existing android application code and press ok. Browse to your android folder and then sdk folder. In sdk folder expand extras folder. Expand google folder and select google play services.

### CONFIGURE YOUR PROJECT WITH SDK

After you import the SDK , you have to add it into your project. For this , right click on your eclipse project and select properties. Select android from left tab and then select add from right below panel and add the project. It would be like this



## Obtaining the API key

This part is furthur divided into two steps. First you have to get an SHA1 fingerprint key from your pc and then you have to get map API key from google console.

### GETTING CERTIFICATE FROM KEYTOOL

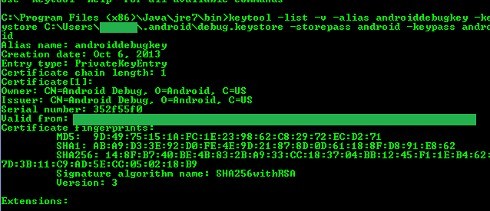
You need to get a certificate key because you have to provide it to google console in order to get your API key for map.

Open your command prompt and move to the path where your java jre has been placed. Now type this command.

keytool-list -v -aliasandroiddebugkey-keystore%%Your path%%-storepass android -keypass android

Replace the percentage part of the command with the path which you will copy from by selecting the window tab and selecting the preferences tab and then selectng the build option under android from left side.

Copy the default debug keystore path and replace it in the cmmand and hit enter. The following result would appear.



Copy the SHA1 key because you need it in the next step.

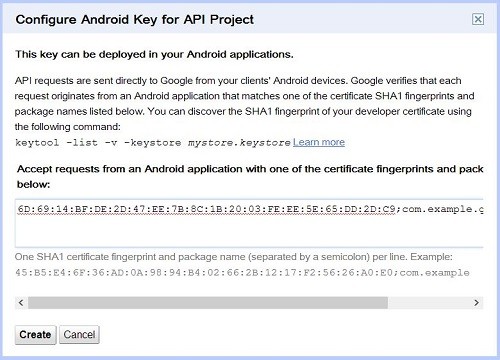
### GETTING KEY FROM GOOGLE CONSOLE

Open [Google Console](https://code.google.com/apis/console/) and sign in by clicking a new project.

Click on services from the left tab and then navigate to the Google Maps Android API v2. You have to turn them on like this



Now again go to the left tab and select API access. And click on create new android key. Now paste the key that you copied and put a semicolon and paste your project name and click create. It would be like this.



Now copy the API key that has been given to your by android , because you have to paste it into your manifest file.

## Specify Android Manifest settings

The final step is to add the API key to your application. Open your manifest file and place this code right before closing the application tag.

<meta-data

android:name="com.google.android.maps.v2.API\_KEY"

android:value="API\_KEY"/>

In the second line replace API\_KEY with your api key and you are done. You need to add some permissions in your manifest too which are given below in the manifest file.

## Adding Google Maps to your application.

Following is the content of the modifed main activity file**src/com.example.googlemaps/MainActivity.java**.

packagecom.example.googlemaps;

importcom.google.android.gms.maps.GoogleMap;

importcom.google.android.gms.maps.MapFragment;

importcom.google.android.gms.maps.model.LatLng;

importcom.google.android.gms.maps.model.Marker;

importcom.google.android.gms.maps.model.MarkerOptions;

importandroid.os.Bundle;

importandroid.app.Activity;

importandroid.widget.Toast;

publicclassMainActivityextendsActivity{

staticfinalLatLngTutorialsPoint=newLatLng(21,57);

privateGoogleMapgoogleMap;

@Override

protectedvoidonCreate(BundlesavedInstanceState){

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

try{

if(googleMap==null){

googleMap=((MapFragment)getFragmentManager().

findFragmentById(R.id.map)).getMap();

}

googleMap.setMapType(GoogleMap.MAP\_TYPE\_HYBRID);

Marker TP =googleMap.addMarker(newMarkerOptions().

position(TutorialsPoint).title("TutorialsPoint"));

}catch(Exception e){

e.printStackTrace();

}

}

}

Following is the modified content of the xml **res/layout/activity\_main.xml**.

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<fragment

android:id="@+id/map"

android:name="com.google.android.gms.maps.MapFragment"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

</RelativeLayout>

Following is the content of **AndroidManifest.xml** file.

<?xml version="1.0" encoding="utf-8"?>

<manifestxmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.googlemaps"

android:versionCode="1"

android:versionName="1.0">

<uses-permissionandroid:name="com.example.googlemaps.permission.MAPS\_RECEIVE"/>

<uses-sdk

android:minSdkVersion="12"

android:targetSdkVersion="17"/>

<permission

android:name="com.example.googlemaps.permission.MAPS\_RECEIVE"

android:protectionLevel="signature"/>

<uses-permissionandroid:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<uses-permissionandroid:name="android.permission.INTERNET"/>

<uses-permissionandroid:name="com.google.android.providers.

gsf.permission.

READ\_GSERVICES"/>

<uses-permissionandroid:name="android.permission.

WRITE\_EXTERNAL\_STORAGE"/>

<uses-permissionandroid:name="android.permission.

ACCESS\_COARSE\_LOCATION"/>

<uses-permissionandroid:name="android.permission.ACCESS\_FINE\_LOCATION"/>

<uses-feature

android:glEsVersion="0x00020000"

android:required="true"/>

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme">

<activity

android:name="com.example.googlemaps.MainActivity"

android:label="@string/app\_name">

<intent-filter>

<actionandroid:name="android.intent.action.MAIN"/>

<categoryandroid:name="android.intent.category.LAUNCHER"/>

</intent-filter>

</activity>

<meta-data

android:name="com.google.android.maps.v2.API\_KEY"

android:value="AIzaSyDKymeBXNeiFWY5jRUejv6zItpmr2MVyQ0"/>

</application>

</manifest>

Let's try to run your GoogleMaps application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Eclipse, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Before starting your application, Eclipse will display following window to select an option where you want to run your Android application.



Now what you need to do is to tap on the ballon to see the text.



Now you can customize the google map according to your choice with the functions given in the GoogleMap API.

# LocationManager

## Class Overview

This class provides access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location, or to fire an application-specified [Intent](http://developer.android.com/reference/android/content/Intent.html) when the device enters the proximity of a given geographical location.

You do not instantiate this class directly; instead, retrieve it through [Context.getSystemService(Context.LOCATION\_SERVICE)](http://developer.android.com/reference/android/content/Context.html#getSystemService(java.lang.String)).

Unless noted, all Location API methods require the [ACCESS\_COARSE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION) or [ACCESS\_FINE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION) permissions. If your application only has the coarse permission then it will not have access to the GPS or passive location providers. Other providers will still return location results, but the update rate will be throttled and the exact location will be obfuscated to a coarse level of accuracy.

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) GPS\_PROVIDER

Added in [API level 1](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Name of the GPS location provider.

This provider determines location using satellites. Depending on conditions, this provider may take a while to return a location fix. Requires the permission[ACCESS\_FINE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION).

The extras Bundle for the GPS location provider can contain the following key/value pairs:

* satellites - the number of satellites used to derive the fix

Constant Value: "gps"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) KEY\_LOCATION\_CHANGED

Added in [API level 3](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Key used for a Bundle extra holding a Location value when a location change is broadcast using a PendingIntent.

Constant Value: "location"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) KEY\_PROVIDER\_ENABLED

Added in [API level 3](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Key used for a Bundle extra holding an Boolean status value when a provider enabled/disabled event is broadcast using a PendingIntent.

Constant Value: "providerEnabled"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) KEY\_PROXIMITY\_ENTERING

Added in [API level 1](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Key used for the Bundle extra holding a boolean indicating whether a proximity alert is entering (true) or exiting (false)..

Constant Value: "entering"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) KEY\_STATUS\_CHANGED

Added in [API level 3](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Key used for a Bundle extra holding an Integer status value when a status change is broadcast using a PendingIntent.

Constant Value: "status"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) MODE\_CHANGED\_ACTION

Added in [API level 19](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Broadcast intent action when [LOCATION\_MODE](http://developer.android.com/reference/android/provider/Settings.Secure.html#LOCATION_MODE) changes. For use with the [LOCATION\_MODE](http://developer.android.com/reference/android/provider/Settings.Secure.html#LOCATION_MODE) API. If you're interacting with [isProviderEnabled(String)](http://developer.android.com/reference/android/location/LocationManager.html#isProviderEnabled(java.lang.String)), use[PROVIDERS\_CHANGED\_ACTION](http://developer.android.com/reference/android/location/LocationManager.html#PROVIDERS_CHANGED_ACTION) instead. In the future, there may be mode changes that do not result in [PROVIDERS\_CHANGED\_ACTION](http://developer.android.com/reference/android/location/LocationManager.html#PROVIDERS_CHANGED_ACTION) broadcasts.

Constant Value: "android.location.MODE\_CHANGED"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) NETWORK\_PROVIDER

Added in [API level 1](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Name of the network location provider.

This provider determines location based on availability of cell tower and WiFi access points. Results are retrieved by means of a network lookup.

Constant Value: "network"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) PASSIVE\_PROVIDER

Added in [API level 8](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

A special location provider for receiving locations without actually initiating a location fix.

This provider can be used to passively receive location updates when other applications or services request them without actually requesting the locations yourself. This provider will return locations generated by other providers. You can query the [getProvider()](http://developer.android.com/reference/android/location/Location.html#getProvider()) method to determine the origin of the location update. Requires the permission [ACCESS\_FINE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION), although if the GPS is not enabled this provider might only return coarse fixes.

Constant Value: "passive"

#### public static final [String](http://developer.android.com/reference/java/lang/String.html) PROVIDERS\_CHANGED\_ACTION

Added in [API level 9](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

Broadcast intent action when the configured location providers change. For use with [isProviderEnabled(String)](http://developer.android.com/reference/android/location/LocationManager.html#isProviderEnabled(java.lang.String)). If you're interacting with the [LOCATION\_MODE](http://developer.android.com/reference/android/provider/Settings.Secure.html#LOCATION_MODE)API, use [MODE\_CHANGED\_ACTION](http://developer.android.com/reference/android/location/LocationManager.html#MODE_CHANGED_ACTION) instead.

Constant Value: "android.location.PROVIDERS\_CHANGED"