MODULE 2

Intents and Filters.

Intents

Intents are used as a message-passing mechanism that works both within your application, and between applications. Intents can be used to:

➤ Declare your intention that an Activity or Service be started to perform an action, usually with (or on) a particular piece of data

➤ Broadcast that an event (or action) has occurred

➤ Explicitly start a particular Service or Activity You can use Intents to support interaction among any of the application components installed on an Android device, no matter which application they're a part of. This turns your device from a platform containing a collection of independent components into a single interconnected system. One of the most common uses for Intents is to start new Activities, either explicitly (by specifying the class to load) or implicitly (by requesting that an action be performed on a piece of data).

In the latter case the action need not be performed by an Activity within the calling application. Intents can also be used to broadcast messages across the system. Any application can register Broadcast Receivers to listen for, and react to, these broadcast Intents. This lets you create event-driven applications based on internal, system, or third-party-application events.

Android broadcasts Intents to announce system events, like changes in Internet connection status or battery charge levels. The native Android applications, such as the phone dialer and SMS manager, simply register components that listen for specific broadcast Intents — such as ''incoming phone call'' or ''SMS message received'' — and react accordingly. Using Intents to propagate actions — even within the same application — is a fundamental Android design principle. It encourages the decoupling of components, to allow the seamless replacement of application elements. It also provides the basis of a simple model for extending an application's functionality.

Intents could be Implicit, for instance, calling intended actions, and explicit as well, such as opening another activity after some operations like onClick or anything else. Below are some applications of Intents:

1. Sending the User to Another App
2. Getting a Result from an Activity
3. Allowing Other Apps to Start Your Activity

Some Important Method of Intent and its Description

Methods	Description
Context.startActivity()	This is to launch a new activity or get an existing activity to be action.
Context.startService()	This is to start a new service or deliver instructions for an existing service.
Context.sendBroadcast()	This is to deliver the message to broadcast receivers.

Deep Linking

Deep Link is an URL that redirects the device to the API of that *Missing Application* and then service is run on the system to check if a version of that application exists on the device. For time being, let's assume that the application is not available on the device and no previous versions ever existed. This service then makes a call to the *Play Store* from the device and the application appears, just a matter of download.

Types of Android Intents

There are two types of intents in android:

1. Implicit and
2. Explicit.

MODULE 2

1. Implicit Intent

Implicit Intent doesn't specify the component. In such a case, intent provides information on available components provided by the system that is to be invoked. For example, you may write the following code to view the webpage.

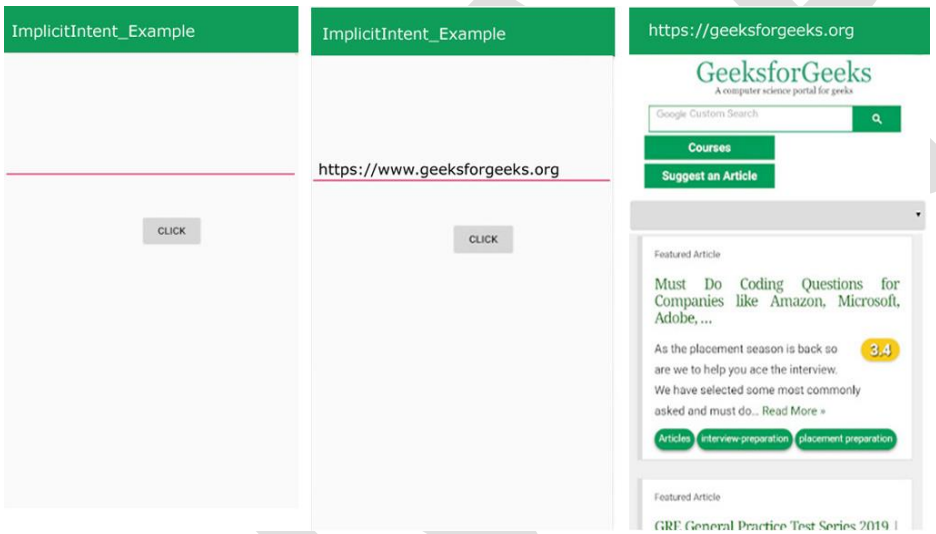
Intent intent=new Intent(Intent.ACTION_VIEW);

intent.setData(Uri.parse("https://www.google.com/"));

startActivity(intent);

Example:

In the below images, no component is specified, instead an action is performed i.e. a webpage is going to be opened. As you type the name of your desired webpage and click on 'CLICK' button. Your webpage is opened.



2. Explicit Intent

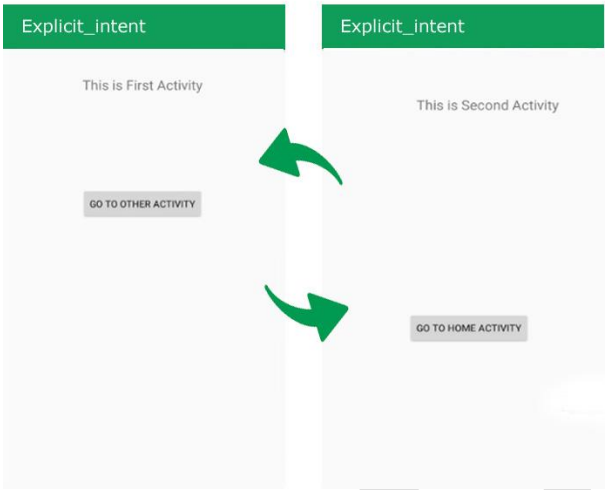
Explicit Intent specifies the component. In such a case, intent provides the external class to be invoked.

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);

startActivity(i);
```

Example:

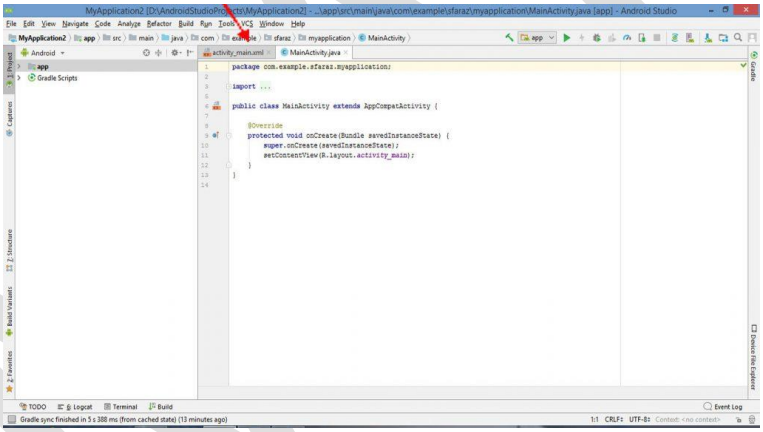
In the below example, there are two activities (FirstActivity, SecondActivity). When you click on the 'GO TO OTHER ACTIVITY' Button in the FirstActivity, then you move to the SecondActivity. When you click on the 'GO TO HOME ACTIVITY' button in the

SecondActivity, then you move to the first activity. This is getting done through Explicit Intent.



How to create an Android App to open a webpage using implicit intent

Step 1: Create XML file and Java File.



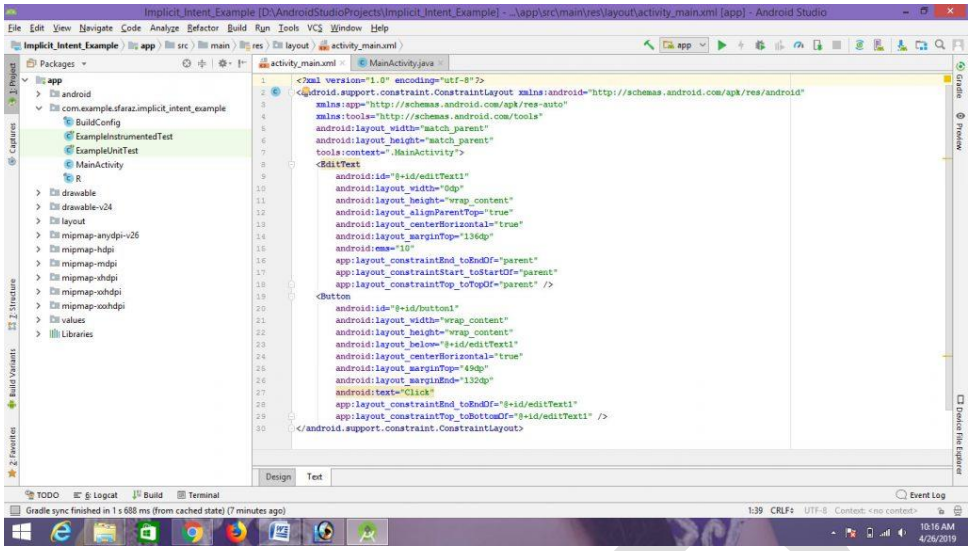
- **Step 2:** Open "activity_main.xml" file and add following widgets in a Constraint Layout.
  1. An **EditText** to input text.
  2. A **Button** to open webpage.

  Also, Assign **IDs** for each component as shown in the image and the code below. The **assigned ID** for a component helps in the identification of component and can be easily use in the Java files.

  Syntax:

  ```
  android:id="@+id/id_name"
  ```

  Here the given IDs: **editText1**, **button1**
  This will make the UI of the Application.



- **Step 3:** Now, after the UI, this step will create the Backend of the App. For this, Open “MainActivity.java” file and instantiate the component (Button) created in the XML file using findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.
  General Syntax:

  ComponentType object = (ComponentType)findViewById(R.id.IdOfTheComponent);

  Syntax for used components (EditText, Button):

  EditText editText = (EditText)findViewById(R.id.editText1);
  Button button = (Button)findViewById(R.id.button1);

- **Step 4:** This step involves setting up the operations to display the Toast Message. These operations are as follows:

  a). First Add the listener on button and this button will open webpage.

  button.setOnClickListener(new View.OnClickListener() {});

  b). Create String type variable to store the value of ‘EditText’. Value is accepted and converted to string.

  String url = editText1.getText().toString();

  c). Create an Intent object MainActivity.java class to open the webpage.

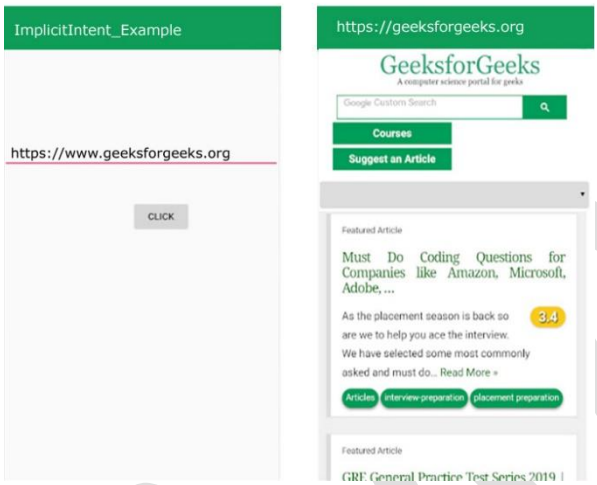
  Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));

d). The startActivity() method starts to call a webpage for opening specified by the intent.

startActivity(intent);

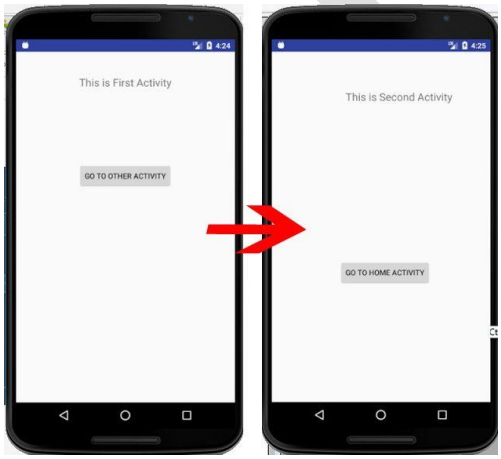
Therefore the code snippet for Implicit Intent:

```
String url = editText1.getText().toString();

Intent intent=new Intent(Intent.ACTION_VIEW, Uri.parse(url));

startActivity(intent);
```



2. **Explicit Intent:** Using explicit intent any other component can be specified. In other words, the targeted component is specified by explicit intent. So only the specified target component will be invoked.

For Example:

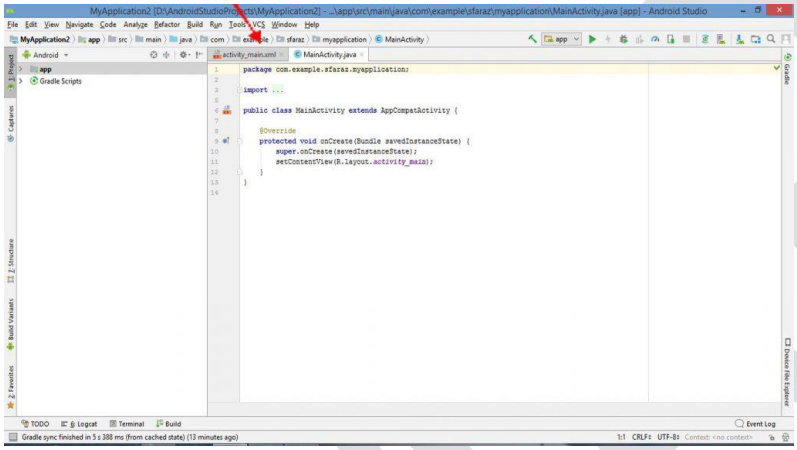


Explicit Intent Example

In the above example, There are two activities (FirstActivity, SecondActivity). When you click on 'GO TO OTHER ACTIVITY' button in the first activity, then you move to second activity. When you click on 'GO TO HOME ACTIVITY' button in the second activity, then you move to the first activity. This is getting done through Explicit Intent.

How to create an Android App to move to next activity using Explicit Intent(with Example)

- **Step 1:** Create XML file and Java File. Please refer the pre-requisites to learn more about this step.

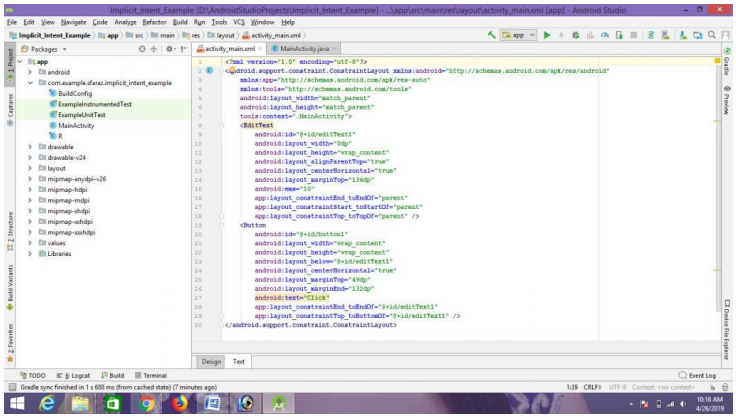


- **Step 2:** Open "activity_main.xml" file and add following widgets in a Constraint Layout.
  1. A **Button** for moving to second activity.
  2. A **TextView** for writing some text.

  Also, Assign **IDs** for each component (Button, TextView) as shown in the image and the code below. The **assigned ID** for a component helps in the identification of component and can be easily use in the Java files.
  Syntax:

  ```
  android:id="@+id/id_name"
  ```

  Here the given IDs: **Button01**, **TextView01**.
  This will make the UI of the Application.



- **Step 3:** Now, after the UI, this step will create the Backend of the App. For this, Open “MainActivity.java” file and instantiate the component (Button, TextView) created in the XML file using findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

  General Syntax:

  ComponentType object = (ComponentType)findViewById(R.id.IdOfTheComponent);

  Syntax for used components (Button, TextView):

  1. ***Button button = (Button)findViewById(R.id.Button01);***

  2. ***TextView textView = (TextView)findViewById(R.id.TextView01);***

- **Step 4:** This step involves setting up the operations to create explicit intent. These operations are as follows:

  a. First Add the listener on button and using this button you will move to other activity.

  button1.setOnClickListener(new View.OnClickListener() {});

  b. Now, Create an Intent.
  Intent i = new Intent(getApplicationContext(), SecondActivityName.class);

  Parameters: This is having two parameters:

- **getApplicationContext():** Returns the context for the entire application.
- **OtherActivityName.class:** You should use your activity name here.

  Therefore the code to make an explicit intent is:

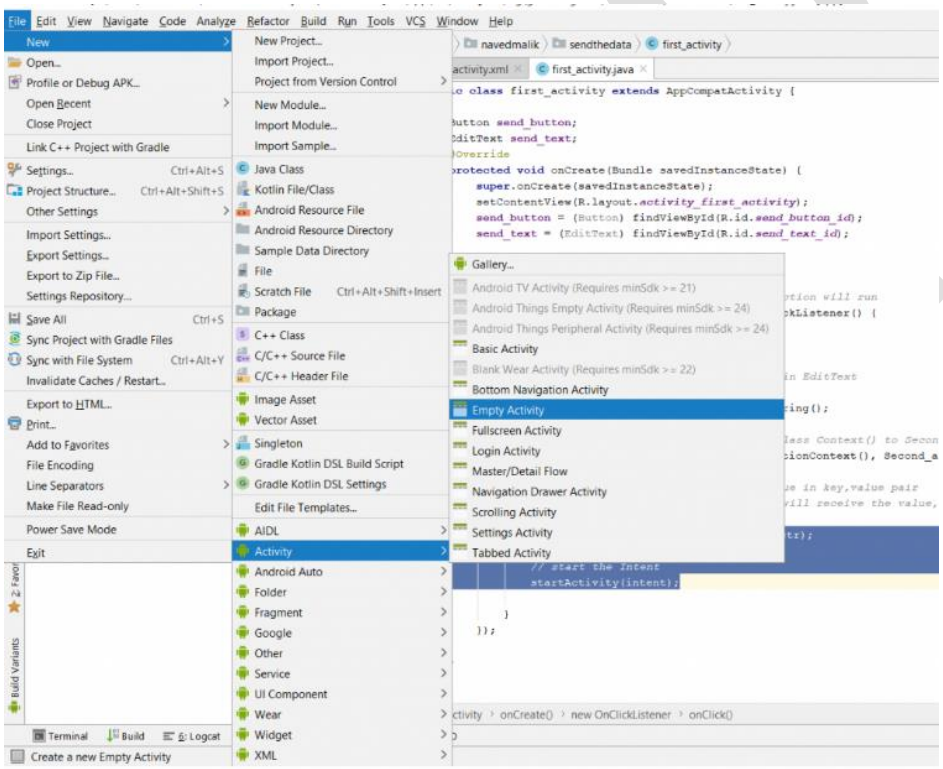
```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
```

c. Now Start the Targeted Activity.
startActivity(i): This starts target activity.
The code snippet for explicit intent:

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);

startActivity(i);
```

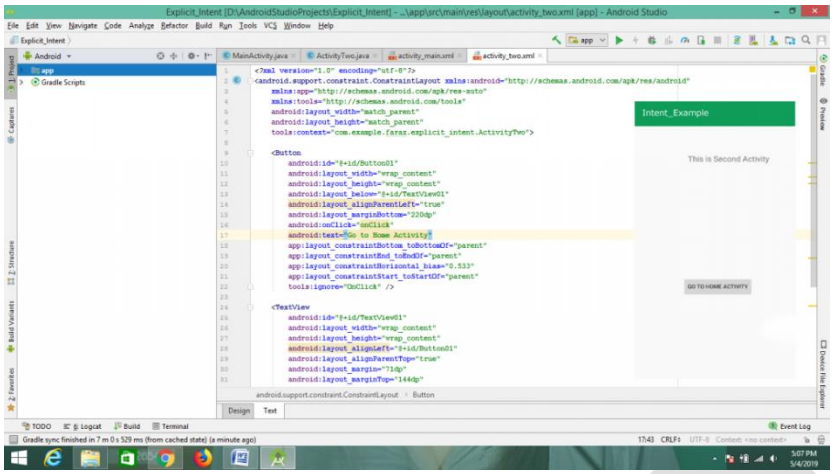
- **Step 5:** Now we have to create a second activity as a destination activity.
  The steps to create the second activity is as follows:

  android project > File > new > Activity > Empty Activity



- **Step 6:** Now open your second xml file.
  Add **Button** and **TextView** for moving back to home activity and to write some text on activity respectively. Assign ID to Button and Textview. Second Activity is

shown below:



- **Step 7:** Now, open up the your second activity java file and perform the following operation.
  a. First Add the listener on button and using this button you will move to home activity.

  button1.setOnClickListener(new View.OnClickListener() {});

  b. Now, Create an Intent.

  Intent i = new Intent(getApplicationContext(), OtherActivityName.class);

  Therefore the code to make an explicit intent is:

  ```
  Intent i = new Intent(getApplicationContext(),
  MainActivity.class);
  ```

  c. Now Start the Targeted Activity.

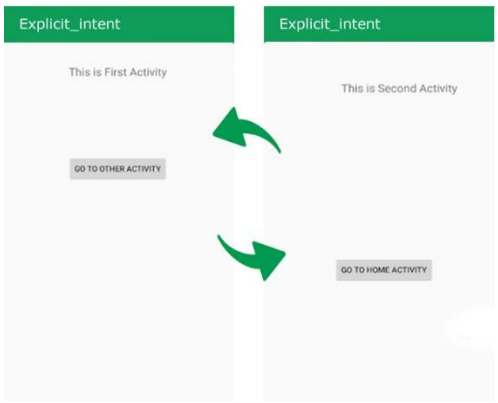
  startActivity(i):*This starts target activity.*

  The code to show the Toast message:

  Intent i = new Intent(getApplicationContext(), MainActivity.class);
  startActivity(i);

- **Step 8:** Now Run the app and operate as follows:
  When the app is opened, it displays a “GO TO OTHER ACTIVITY” button.
  Click the “GO TO OTHER ACTIVITY” button.
  Then you will move to second activity.

Also, you will get a “GO TO HOME ACTIVITY” button on the second activity, When you will click this, you will move to home activity.



Intent Filters

- Implicit intent uses the intent filter to serve the user request.
- The intent filter specifies the types of intents that an activity, service, or broadcast receiver can respond.

- Intent filters are declared in the Android manifest file.
- Intent filter must contain <action>

Example:

```
<intent-filter
  android:icon="drawable resource"
```

```
   android:label="string resource"
   android:priority="integer" >
    . . .
</intent-filter>
```

Most of the intent filter are describe by its

1. **<action>,**
2. **<category>** and
3. **<data>**.

1. <action>

Syntax:

```
<action android:name="string" />
```

Adds an action to an intent filter.

An <intent-filter> element must contain one or more <action> elements.

If there are no <action> elements in an intent filter, the filter doesn't accept any Intent objects.

Examples of common action:

- **ACTION_VIEW:** Use this action in intent with startActivity() when you have some information that activity can show to the user like showing an image in a gallery app or an address to view in a map app
- **ACTION_SEND:** You should use this in intent with startActivity() when you have some data that the user can share through another app, such as an email app or social sharing app.

2. <category>

Syntax:

```
<category android:name="string" />
```

Adds a category name to an intent filter. A string containing additional information about the kind of component that should handle the intent.

Example of common categories:

- **CATEGORY_BROWSABLE:** The target activity allows itself to be started by a web browser to display data referenced by a link.

3. <data>

Syntax:

```
<data android:scheme="string"
      android:host="string"
      android:port="string"
      android:path="string"
      android:pathPattern="string"
      android:pathPrefix="string"
      android:mimeType="string" />
```

Adds a data specification to an intent filter. The specification can be just a data type, just a URI, or both a data type and a URI.

Note: *Uniform Resource Identifier (URI) is a string of characters used to identify a resource. A URI identifies a resource either by location, or a name, or both.*

You have seen how an Intent has been used to call an another activity.

Android OS uses filters to pinpoint the set of Activities, Services, and Broadcast receivers that can handle the Intent with help of specified set of action, categories, data scheme associated with an Intent.

You will use **<intent-filter>** element in the manifest file to list down actions, categories and data types associated with any activity, service, or broadcast receiver.

Following is an example of a part of **AndroidManifest.xml** file to specify an activity **com.example.My Application.CustomActivity** which can be invoked by either of the two mentioned actions, one category, and one data −

```
<activity android:name=".CustomActivity"
  android:label="@string/app_name">

  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <action android:name="com.example.My Application.LAUNCH" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="http" />
  </intent-filter>

</activity>
```

Once this activity is defined along with above mentioned filters, other activities will be able to invoke this activity using either the **android.intent.action.VIEW**, or using the **com.example.My Application.LAUNCH** action provided their category is **android.intent.category.DEFAULT**.

The **<data>** element specifies the data type expected by the activity to be called and for above example our custom activity expects the data to start with the "http://"

There may be a situation that an intent can pass through the filters of more than one activity or service, the user may be asked which component to activate. An exception is raised if no target can be found.

There are following test Android checks before invoking an activity −

- A filter <intent-filter> may list more than one action as shown above but this list cannot be empty; a filter must contain at least one <action> element, otherwise it will block all intents. If more than one actions are mentioned then Android tries to match one of the mentioned actions before invoking the activity.
- A filter <intent-filter> may list zero, one or more than one categories. if there is no category mentioned then Android always pass this test but if more than one

categories are mentioned then for an intent to pass the category test, every category in the Intent object must match a category in the filter.

- Each <data> element can specify a URI and a data type (MIME media type). There are separate attributes like **scheme, host, port**, and **path** for each part of the URI. An Intent object that contains both a URI and a data type passes the data type part of the test only if its type matches a type listed in the filter.

****