Any good web designed knows how important the <select> element can be. In a touch screen environment, like Android, where users often don't have a full keyboard, like many Android phones, the ability to simply select an item from a pre-determined list is absolutely vital. It's easier for the user to select what they want - rather than have to type it, it protects your databases from junk data and typos by limiting choices to pre-defined, pre-screened responses. The Android OS has a very robust, user friendly and - relatively - programmer friendly way to handle this: the Android Spinner.

//

Spinners are simple to use, easy to program and just downright good-looking. However, there are several different ways to populate the list of items in an Android Spinner, and - as of the writing of this article - the Android Developer's Website (developer.android.com) only focuses on one of those methods. This article will break down both of the most popular ways to do populate an Android Spinner: with an array contained in your strings.xml file, and from a database query. Then it will also demonstrate how to manually insert items into a spinner array using code. For ease of navigation, I'm placing each of those on a separate page and including links to them so you can go to the page that best fits your needs as a developer.

- Populate Android Spinner from string array resource
- Populate Android Spinner from database query (using a cursor)
- Populate Android Spinner manually with Java code

Populating an Android Spinner from a string array resource is pretty well documented on developer.android.com. So this section will be pretty brief. Basically, if you have a preset list of items that you want to put into a spinner, and you have the luxury of knowing two things: one, that your list will be complete when you compile your .APK package; and two, that your list will be static; then you can fill your spinner using a string array contained as a resource in your Android .APK package.

To do this, all you need to do is put that list into your strings.xml file as a string-array, and then pass that array to your spinner using a simple array adapter. The syntax for a string-array looks like this:

```
<string-array name="array_name">  <item>Array Item One</item>  <item>Array Item
Two</item>  <item>Array Item Three</item>  </string-array>
```

As you can see, it's pretty straight-forward XML code. Just start a 'string-array' item, give it a unique name, and fill it with items. Next we need to create an actual spinner as a part of our layout file so that we have a spinner to populate. Technically, you can create a spinner in your Java code (an option that we'll visit later in this article) but for now, we're going to do it the easy way, in an XML layout file. Doing in your XML file is definitely the preferred method, since it helps to separate your code from your layout, it's easier and less prone to errors. Adding a spinner to a layout file is easy, in fact it will look almost exactly like most of your other view elements (like <TextView>, <EditText> or <Button> elements). It will contain the required layout attributes and, most importantly, it will have a unique ID that you can reference in your code (so that you can pass your array to it later). That will look like this:

```
  <Spinner    android:id="@+id/unique_id"    android:layout_width="wrap_content"
android:layout_height="wrap_content"  />
```

You could play around with the 'layout_width' and 'layout_height' depending on the layout of your activity, but for now I'll assume you want one just big enough to do the job.

//

Now we get to the fun part: populating the spinner from that string-array. This is pretty easy actually. First, we'll create an array adapter. Second, we'll create a reference to our spinner. And finally, we'll assign the array adapter to our spinner. The Java code for that goes like this:

```
  ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(    this,
R.array.array_name, android.R.layout.simple_spinner_item );
adapter.setDropDownViewResource( android.R.layout.simple_spinner_drop_down_item );
```

That does it for the array adapter. We just create one using the built-in 'createFromResource' method, which takes a context (usually 'this'), a reference to a resource (i.e. our string-array) and a reference to a layout view. Then we also specify a layout view for the drop down items when someone clicks on our spinner. Note that those last two references, we just used a built-in Android layout, which works very well for simple spinners, but this is where you have the opportunity to get creative, by creating your own layout views and supplying them here, you can create dynamic, colorful and more informative spinner items (but that's a topic for a separate article).

Next, we'll create a reference to our spinner and assing the array adapter to it:

```
  Spinner s = (Spinner) findViewById( R.id.unique_id );  s.setAdapter( adapter );
```

And that's it. To reference our spinner we use 'findViewById' which will be your go-to method to reference any view item that you have defined in your XML layout file, just be sure to cast the object as the correct type so that Java knows what methods it has available - that's why we put in that extra '(Spinner)' after the equals sign. Then we assigned our array adapter using the 'setAdapter' method of the spinner object.

Next up, populating a spinner from a database query.

Populating an Android Spinner from a database is pretty similar to doing it from an array resource like we did on the previous page. The XML file is the same and the Java code to reference the spinner and set the adapter is the same as well (for more details on that, view the previous page of this article on populating android spinners from a string array). The primary difference is that instead of using an Array Adapter to grab a string array and pass that to our spinner, instead, we're going to use a Simple Cursor Adapter to connect a database query to our spinner. To do that, you need to query your database, which is sorta outside the scope of this tutorial, so I won't go into how to create a database, build a table, add records or anything like that. However, I will mention that to use a Simple Cursor Adapter, you need to query your database in a way that returns a cursor, something like this:

```
  Public Cursor fetchAllTitles(){  return mDb.query(    table_name, new String[]{KEY_ROWID,
KEY_TITLE}, null, null, null, null, null );  }
```

Note that you could also use something like "mDb.rawQuery(sql,null)" if you wanted to write your own SQL statement instead of using the query builder that comes with Android.

Once you have a method build to get that cursor, you need to call that method, and then build your Simple Cursor Adapter by passing it your cursor and a list of the fields from that cursor that you want to use. That should end up looking something like this:

```
  private void fillSpinner(){    Cursor c = mDbHelper.fetchAllTitles();  startManagingCursor(c);    //
create an array to specify which fields we want to display  String[] from = new
String[]{field_name};  // create an array of the display item we want to bind our data to  int[] to =
new int[]{android.R.id.text1};  // create simple cursor adapter  SimpleCursorAdapter adapter =
new SimpleCursorAdapter(this, android.R.layout.simple_spinner_item, c, from, to );
adapter.setDropDownViewResource( android.R.layout.simple_spinner_dropdown_item );  // get
reference to our spinner  Spinner s = (Spinner) findViewById( R.id.unique_id );
s.setAdapter(adapter);  }      //
```

There are a few assumptions built in here. First, we're assuming that the "fetchAllTitles" method that we created earlier is part of a public class, and that you have referenced that class as a member of your current Java activity (using the name "mDbHelper"). Second, we are assuming that you have given your Spinner a unique id in the XML layout file - this code is actually assuming that the id you gave the spinner was "unique_id", but you can change that to whatever id you want to, so long as the one in your Java matches the one in your XML. Also, just like in the string array example, the view items that we referenced are a standard part of the Android Package, but you can use custom ones if you would like, which is a great way to spice up your spinners with colors, graphics, multiple data fields, etc.

That just about covers it for populating an Android Spinner from a database, in the next few days, I'll be adding a section on how to populate an Android Spinner manually on the next page.

The final way to populate an Android Spinner - at least the final way we'll cover in this tutorial - is to manually populate the spinner. Doing this is relatively easy. All you need to do is create an empty Array Adapter, and then call the add() method on that object and it will insert your items. To create an empty Array Adapter, all you need to do is this:

```
  ArrayAdapter <CharSequence> adapter =    new ArrayAdapter <CharSequence> (this,
android.R.layout.simple_spinner_item );
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

As you can see, we just created an Array Adapter without passing it an array. It's important to include that <CharSequence> part so that they Array Adapter knows that type of data to expect. We also give the Array Adapter some standard Android Spinner layout items so that it will know how to display the content we are about to feed it. Now all we need to do is add some stuff to our Array Adapter and then we can pass that to our spinner.

```
  adapter.add("some string data");
```

Then just pass the Array Adapter to our Spinner:

```
  Spinner s = (Spinner) findViewById(R.id.unique_id);  s.setAdapter(adapter);      //
```

And that's it. Now you should know how to populate an Android Spinner using each of the three most popular methods. I hope you've found this tutorial helpful, if you have, consider donating using the PayPal button to the left. And if you have any questions or comments, feel free to email me (info@dcpagesapps.com) or submit a comment below.

Thanks,

dcPages