<provider>

SYNTAX:

```
<provider android:authorities="list"
          android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:grantUriPermissions=["true" | "false"]
          android:icon="drawable resource"
          android:initOrder="integer"
          android:label="string resource"
          android:multiprocess=["true" | "false"]
          android:name="string"
          android:permission="string"
          android:process="string"
          android:readPermission="string"
          android:syncable=["true" | "false"]
          android:writePermission="string" >
    . . .
</provider>
```

CONTAINED IN:

<application>

CAN CONTAIN:

<meta-data>
<grant-uri-permission>

DESCRIPTION:

Declares a content provider — a subclass of ContentProvider — that supplies structured access to data managed by the application. All content providers that are part of the application must be represented by <provider> elements in the manifest file. The system cannot see, and therefore will not run, any that are not declared. (You need to declare only those content providers that you develop as part of your application, not those developed by others that your application uses.)

The Android system identifies content providers by the authority part of a content: URI. For example, suppose that the following URI is passed to ContentResolver.query():

```
content://com.example.project.healthcareprovider/nurses/rn
```

The content: scheme identifies the data as belonging to a content provider and the authority (com.example.project.healthcareprovider) identifies the particular provider. The authority therefore must be unique. Typically, as in this example, it's the fully qualified name of a ContentProvider subclass. The path part of a URI may be used by a content provider to identify particular data subsets, but those paths are not declared in the manifest.

For information on using and developing content providers, see a separate document, Content Providers.

ATTRIBUTES:

android:authorities

A list of one or more URI authorities that identify data under the purview of the content provider. Multiple authorities are listed by separating their names with a semicolon. To avoid conflicts, authority names should use a Java-style naming convention (such as com.example.provider.cartoonprovider). Typically, it's the name of the ContentProvider subclass.

There is no default. At least one authority must be specified.

android:enabled
Whether or not the content provider can be instantiated by the system — "true" if it can be, and "false" if not. The default value is "true".

The <application> element has its own enabled attribute that applies to all application components, including content providers. The <application> and <provider> attributes must both be "true" (as they both are by default) for the content provider to be enabled. If either is "false", the provider is disabled; it cannot be instantiated.

android:exported
Whether or not the content provider can be used by components of other applications — "true" if it can be, and "false" if not. If "false", the provider is available only to components of the same application or applications with the same user ID. The default value is "true".

You can export a content provider but still limit access to it with the permission attribute.

android:grantUriPermissions
Whether or not those who ordinarily would not have permission to access the content provider's data can be granted permission to do so, temporarily overcoming the restriction imposed by the readPermission, writePermission, and permission attributes — "true" if permission can be granted, and "false" if not. If "true", permission can be granted to any of the content provider's data. If "false", permission can be granted only to the data subsets listed in <grant-uri-permission> subelements, if any. The default value is "false".

Granting permission is a way of giving an application component one-time access to data protected by a permission. For example, when an e-mail message contains an attachment, the mail application may call upon the appropriate viewer to open it, even though the viewer doesn't have general permission to look at all the content provider's data.

In such cases, permission is granted by FLAG_GRANT_READ_URI_PERMISSION and FLAG_GRANT_WRITE_URI_PERMISSION flags in the Intent object that activates the component. For example, the mail application might put FLAG_GRANT_READ_URI_PERMISSION in the Intent passed to Context.startActivity(). The permission is specific to the URI in the Intent.

If you enable this feature, either by setting this attribute to "true" or by defining <grant-uri-permission> subelements, you must call Context.revokeUriPermission() when a covered URI is deleted from the provider.

See also the <grant-uri-permission> element.

android:icon
An icon representing the content provider. This attribute must be set as a reference to a drawable resource containing the image definition. If it is not set, the icon specified for the application as a whole is used instead (see the <application> element's icon attribute).

android:initOrder
The order in which the content provider should be instantiated, relative to other content providers hosted by the same process. When there are dependencies among content providers, setting this attribute for each of them ensures that they are created in the order required by those dependencies. The value is a simple integer, with higher numbers being initialized first.

android:label
A user-readable label for the content provided. If this attribute is not set, the label set for the application as a whole is used instead (see the <application> element's label attribute).

The label should be set as a reference to a string resource, so that it can be localized like other strings in the user interface. However, as a convenience while you're developing the application, it can also be set as a raw string.

android:multiprocess
Whether or not an instance of the content provider can be created in every client process — "true" if instances can run in multiple processes, and "false" if not. The default value is "false".

Normally, a content provider is instantiated in the process of the application that defined it. However, if this flag

is set to "true", the system can create an instance in every process where there's a client that wants to interact with it, thus avoiding the overhead of interprocess communication.

android:name
The name of the class that implements the content provider, a subclass of ContentProvider. This should be a fully qualified class name (such as, "com.example.project.TransportationProvider"). However, as a shorthand, if the first character of the name is a period, it is appended to the package name specified in the <manifest> element.

There is no default. The name must be specified.

android:permission
The name of a permission that clients must have to read or write the content provider's data. This attribute is a convenient way of setting a single permission for both reading and writing. However, the readPermission and writePermission attributes take precedence over this one. If the readPermission attribute is also set, it controls access for querying the content provider. And if the writePermission attribute is set, it controls access for modifying the provider's data.

For more information on permissions, see the Permissions section in the introduction and a separate document, Security and Permissions.

android:process
The name of the process in which the content provider should run. Normally, all components of an application run in the default process created for the application. It has the same name as the application package. The <application> element's process attribute can set a different default for all components. But each component can override the default with its own process attribute, allowing you to spread your application across multiple processes.

If the name assigned to this attribute begins with a colon (':'), a new process, private to the application, is created when it's needed and the activity runs in that process. If the process name begins with a lowercase character, the activity will run in a global process of that name, provided that it has permission to do so. This allows components in different applications to share a process, reducing resource usage.

android:readPermission
A permission that clients must have to query the content provider. See also the permission and writePermission attributes.

android:syncable
Whether or not the data under the content provider's control is to be synchronized with data on a server — "true" if it is to be synchronized, and "{@ code false}" if not.

android:writePermission
A permission that clients must have to make changes to the data controlled by the content provider. See also the permission and readPermission attributes.

INTRODUCED IN:
API Level 1

SEE ALSO:
Content Providers

← Back to The AndroidManifest.xml File ↑ Go to top