<service>

SYNTAX:

```
<service android:description="string resource"
         android:directBootAware=["true" | "false"]
         android:enabled=["true" | "false"]
         android:exported=["true" | "false"]
         android:icon="drawable resource"
         android:isolatedProcess=["true" | "false"]
         android:label="string resource"
         android:name="string"
         android:permission="string"
         android:process="string" >
    . . .
</service>
```

CONTAINED IN:

`<application>`

CAN CONTAIN:

`<intent-filter>`
`<meta-data>`

DESCRIPTION:

Declares a service (a `Service` subclass) as one of the application's components. Unlike activities, services lack a visual user interface. They're used to implement long-running background operations or a rich communications API that can be called by other applications.

All services must be represented by `<service>` elements in the manifest file. Any that are not declared there will not be seen by the system and will never be run.

ATTRIBUTES:

`android:description`

A string that describes the service to users. 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.

`android:directBootAware`

Whether or not the service is *direct-boot aware*; that is, whether or not it can run before the user unlocks the device.

> **Note:** During Direct Boot, a service in your application can only access the data that is stored in *device protected* storage.

The default value is `"false"`.

`android:enabled`

Whether or not the service 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 services. The

`<application>` and `<service>` attributes must both be "`true`" (as they both are by default) for the service to be enabled. If either is "`false`", the service is disabled; it cannot be instantiated.

`android:exported`

Whether or not components of other applications can invoke the service or interact with it — "`true`" if they can, and "`false`" if not. When the value is "`false`", only components of the same application or applications with the same user ID can start the service or bind to it.

The default value depends on whether the service contains intent filters. The absence of any filters means that it can be invoked only by specifying its exact class name. This implies that the service is intended only for application-internal use (since others would not know the class name). So in this case, the default value is "`false`". On the other hand, the presence of at least one filter implies that the service is intended for external use, so the default value is "`true`".

This attribute is not the only way to limit the exposure of a service to other applications. You can also use a permission to limit the external entities that can interact with the service (see the `permission` attribute).

`android:icon`

An icon representing the service. 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).

The service's icon — whether set here or by the `<application>` element — is also the default icon for all the service's intent filters (see the `<intent-filter>` element's `icon` attribute).

`android:isolatedProcess`

If set to true, this service will run under a special process that is isolated from the rest of the system and has no permissions of its own. The only communication with it is through the Service API (binding and starting).

`android:label`

A name for the service that can be displayed to users. 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 service's label — whether set here or by the `<application>` element — is also the default label for all the service's intent filters (see the `<intent-filter>` 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:name`

The name of the `Service` subclass that implements the service. This should be a fully qualified class name (such as, "`com.example.project.RoomService`"). However, as a shorthand, if the first character of the name is a period (for example, "`.RoomService`"), it is appended to the package name specified in the `<manifest>` element.

Once you publish your application, you should not change this name (unless you've set `android:exported="false"`).

There is no default. The name must be specified.

`android:permission`

The name of a permission that an entity must have in order to launch the service or bind to it. If a caller of `startService()`, `bindService()`, or `stopService()`, has not been granted this permission, the method will not work and the Intent object will not be delivered to the service.

If this attribute is not set, the permission set by the `<application>` element's `permission` attribute applies to the service. If neither attribute is set, the service is not protected by a permission.

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 where the service is to 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 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 service runs in that process. If the process name begins with a lowercase character, the service 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.

SEE ALSO:

`<application>`
`<activity>`

INTRODUCED IN:

API Level 1