IceGrid Templates

IceGrid templates simplify the task of creating the descriptors for an application. A template is a parameterized descriptor that you can instantiate as often as necessary, and they are descriptors in their own right. Templates are components of an IceGrid application and therefore they are stored in the registry's database. As such, their use is not restricted to XML files; templates can also be created and instantiated interactively using the graphical administration tool.

You can define templates for server and service descriptors. The focus of this section is server templates; we discuss service descriptors and templates in the context of IceBox integration.

On this page:

Server Templates

You may recall from a previous example that the XML description of our sample application defined two nearly identical servers:

XML

```
<icegrid>
    <application name="Ripper">
        <node name="Node1">
            <server id="EncoderServer1" exe="/opt/ripper/bin/server" activation="on-demand">
                <adapter name="EncoderAdapter" endpoints="tcp"/>
            </server>
        </node>
        <node name="Node2">
            <server id="EncoderServer2" exe="/opt/ripper/bin/server" activation="on-demand">
                <adapter name="EncoderAdapter" endpoints="tcp"/>
            </server>
        </node>
    </application>
</icegrid>
```

This example is an excellent candidate for a server template. Equivalent definitions that incorporate a template are shown below:

XML

```
<icegrid>
    <application name="Ripper">
        <server-template id="EncoderServerTemplate">
            <parameter name="index"/>
            <server id="EncoderServer${index}" exe="/opt/ripper/bin/server" activation="on-demand">
                <adapter name="EncoderAdapter" endpoints="tcp"/>
            </server>
        </server-template>
        <node name="Node1">
            <server-instance template="EncoderServerTemplate" index="1"/>
        </node>
        <node name="Node2">
            <server-instance template="EncoderServerTemplate" index="2"/>
        </node>
    </application>
</icegrid>
```

We have defined a server template named `EncoderServerTemplate`. Nested within the `server-template` element is a server descriptor that defines an encoder server. The only difference between this `server` element and our previous example is that it is now parameterized: the template parameter `index` is used to form unique identifiers for the server and its adapter. The symbol `${index}` is replaced with the value of the `index` parameter wherever it occurs.

The template is instantiated by a `server-instance` element, which may be used anywhere that a `server` element is used. The server instance descriptor identifies the template to be instantiated, and supplies a value for the `index` parameter.

Although we have not significantly reduced the length of our XML file, we have made it more readable. And more importantly, deploying this server on additional nodes has become much easier.

Template Parameters

Parameters enable you to customize each instance of a template as necessary. The example above defined the `index` parameter with a different value for each instance to ensure that identifiers are unique. A parameter may also declare a default value that is used in the template if no value is specified for it. In our sample application the `index` parameter is considered mandatory and therefore should not have a default value, but we can illustrate this feature in another way. For example, suppose that the path name of the server's executable may change on each node. We can supply a default value for this attribute and override it when necessary:

XML

```
<icegrid>
    <application name="Ripper">
        <server-template id="EncoderServerTemplate">
            <parameter name="index"/>
            <parameter name="exepath" default="/opt/ripper/bin/server"/>
            <server id="EncoderServer${index}" exe="${exepath}" activation="on-demand">
                <adapter name="EncoderAdapter" endpoints="tcp"/>
            </server>
        </server-template>
        <node name="Node1">
            <server-instance template="EncoderServerTemplate" index="1"/>
        </node>
        <node name="Node2">
            <server-instance template="EncoderServerTemplate" index="2"
                exepath="/opt/ripper-test/bin/server"/>
        </node>
    </application>
</icegrid>
```

As you can see, the instance on `Node1` uses the default value for the new parameter `exepath`, but the instance on `Node2` defines a different location for the server's executable.

Understanding the semantics of descriptor variables and parameters will help you add flexibility to your own IceGrid applications.

Adding Properties to a Server Instance

As we saw in the preceding section, template parameters allow you to customize each instance of a server template, and template parameters with default values allow you to define commonly used configuration options. However, you might want to have additional configuration properties for a given instance without having to add a parameter. For example, to debug a server instance on a specific node, you might want to start the server with the `Ice.Trace.Network` property set; it would be inconvenient to have to add a parameter to the template just to set that property.

To cater for such scenarios, it is possible to specify additional properties for a server instance without modifying the template. You can define such properties in the `server-instance` element, for example:

XML

```
<icegrid>
    <application>
        ...
        <node name="Node2">
            <server-instance template="EncoderServerTemplate" index="2">
                <properties>
                    <property name="Ice.Trace.Network" value="2"/>
                </properties>
            </server-instance>
        </node>
    </application>
</icegrid>
```

This sets the `Ice.Trace.Network` property for a specific server.

Default Templates

The IceGrid registry can be configured to supply any number of default template descriptors for use in your applications. The configuration property `IceGrid.Registry.DefaultTemplates` specifies the path name of an XML file containing template definitions. One such template file is provided in the Ice distribution as `config/templates.xml`, which contains helpful templates for deploying Ice services such as IcePatch2 and Glacier2.

The template file must use the structure shown below:

XML

```
<icegrid>
    <application name="DefaultTemplates">
        <server-template id="EncoderServerTemplate">
            ...
        </server-template>
    </application>
</icegrid>
```

The name you give to the application is not important, and you may only define server and service templates within it. After configuring the registry to use this file, your default templates become available to every application that imports them.

The descriptor for each application indicates whether the default templates should be imported. (By default they are not imported.) If the templates are imported, they are essentially copied into the application descriptor and treated no differently than templates defined by the application itself. As a result, changes to the file containing default templates have no effect on existing application descriptors. In XML, the attribute `import-default-templates` determines whether the default templates are imported, as shown in the following example:

XML

```
<icegrid>
    <application name="Ripper" import-default-templates="true">
        ...
    </application>
</icegrid>
```

Using Templates with `icegridadmin`

The IceGrid administration tools allow you to inspect templates and instantiate new servers dynamically. First, let us ask `icegridadmin` to describe the server template we created earlier:

```
$ icegridadmin --Ice.Config=/opt/ripper/config
>>> server template describe Ripper EncoderServerTemplate
```

This command generates the following output:

```
server template `EncoderServerTemplate'
{
    parameters = `index exepath'
    server `EncoderServer${index}'
    {
        exe = `${exepath}'
        activation = `on-demand'
        properties
        {
            EncoderAdapter.Endpoints = `tcp'
        }
        adapter `EncoderAdapter'
        {
            id = `EncoderAdapter${index}'
            replica group id =
            endpoints = `tcp'
            register process = `false'
            server lifetime = `true'
        }
    }
}
```

Notice that the server ID is a parameterized value; it cannot be evaluated until the template is instantiated with values for its parameters.

Next, we can use `icegridadmin` to create an instance of the encoder server template on a new node:

```
>>> server template instantiate Ripper Node3 EncoderServerTemplate index=3
```

The command requires that we identify the application, node and template, as well as supply any parameters needed by the template. The new server instance is permanently added to the registry's database, but if we intend to keep this configuration it is a good idea to update the XML description of our application to reflect these changes and avoid potential synchronization issues.

See Also

- Server Descriptor Element
- Server-Template Descriptor Element
- Server-Instance Descriptor Element
- icegridadmin Command Line Tool