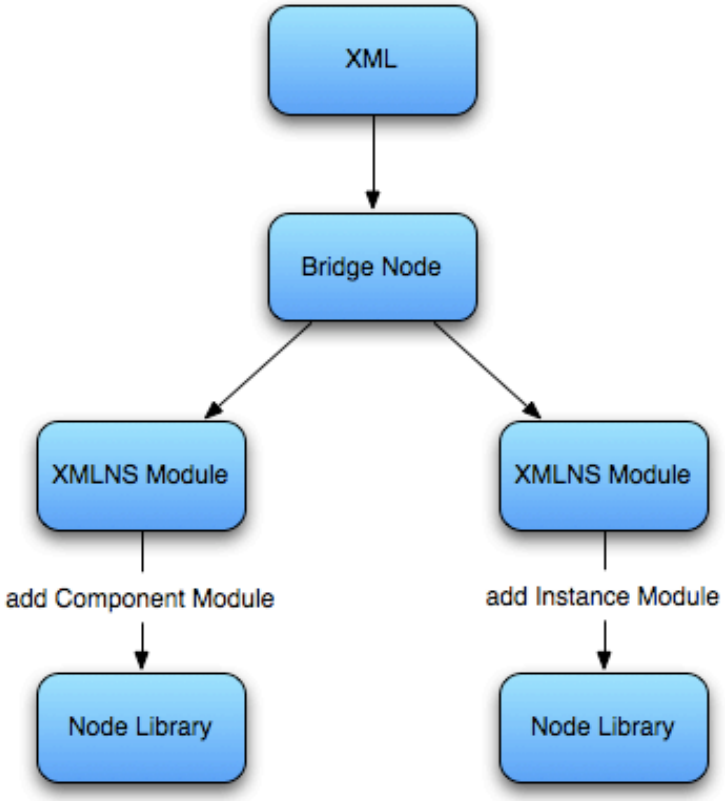
Bridge

About Bridge

Bridge a framework built in flash that enables the creation of Rich Internet Applications using XML. Bridge works by scanning through individual nodes for namespaces and sending those nodes to their rightful module. Here is an example:

```
<Bridge xmlns:view="com.ComponentModule" xmlns:instance="com.InstanceModule" />
```

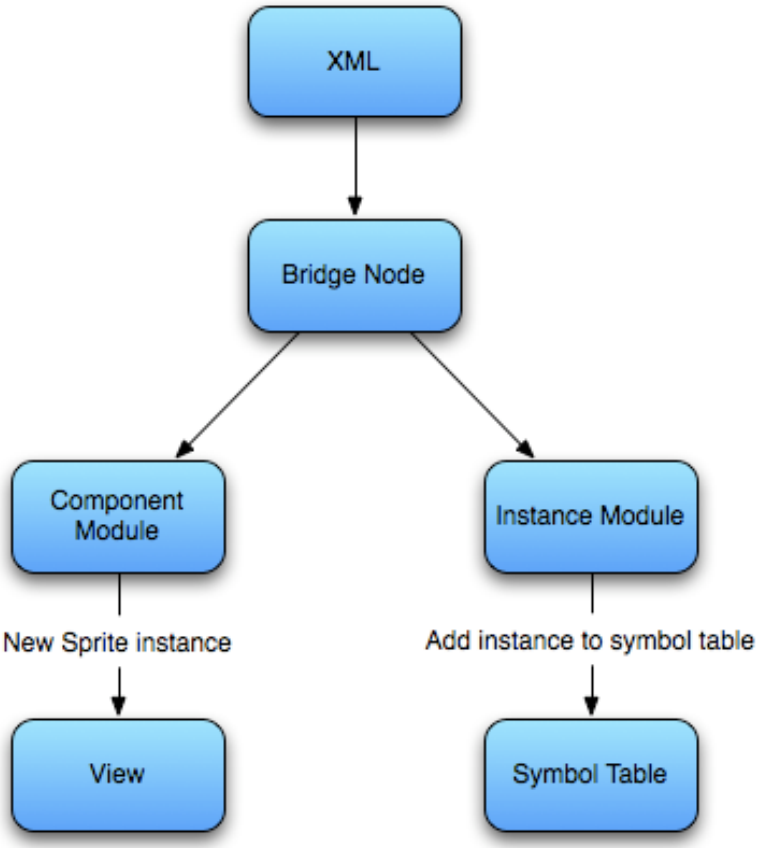


Here are the steps taken place when executing the node:

1. The node first scans through the attributes and finds the namespace “xmlns”.
2. The node is passed to the XMLNS module.
3. The XMLNS module registers a Component Module with a namespace “view”.
4. The XMLNS module registers an Instance Module with a namespace “instance”.

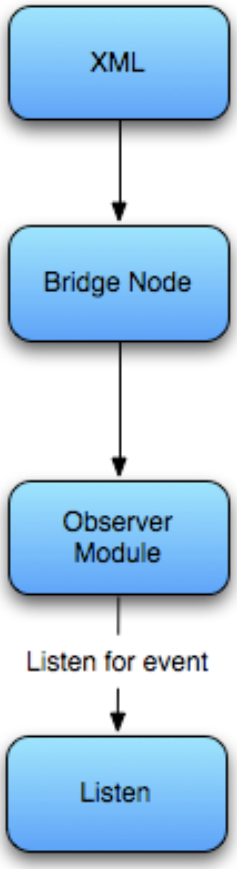
If you were to use those modules you could type the following node:

```
<view:Sprite instance:id="myComponent" />
```



In this example, “view” is found and the node is sent to the Component Module. The Component Module then creates a new instance of Sprite and then adds it to the View. The Instance Module then executes the “instance:id” property, and registers the new instance of Sprite to the Symbol Table with the name “myComponent”. This will allow the new Sprite instance to be used throughout the document. To access the new instance, we can use curly brackets. Here’s an example using the Observer Module:

```
<observer:listen to="{myComponent}" for="mouseUp">
                    do stuff
</observer:listen>
```



In this example we are listening for a mouse up event dispatched by our Sprite Instance. Attribute values wrapped in curly brackets are parsed by the Runtime Compiler and the result replaces the text. Without curly brackets, the attribute value is parsed as a regular string, and if the instance is not found in the symbol table, a “null” value will be returned. This also applies to boolean values. For example:

If I were to create a new instance of sprite with a visibility of “false”, I would have to use **visible=”{false}”**. If I were to omit the curly brackets, **visible=”false”** would return true, since a string converted to a boolean value literally translates as boolean true.

I can also omit some parts of attribute values from being parsed. For instance:

If I wanted to show where my new sprite was, I could type
<view:Label text=”the position of myComponent is x:{myComponent.x} y:{myComponent.y}” />

and it would output as “the position of myComponent is x:0 y:0”.

Component Module

Command: classDefinition

Description:
Registers a class to be used by the component module

Parameters:

Parameter	Description	Optional	Type
source	the class path	no	attribute
name	the name of the class	yes	attribute
constructor	takes the constructor parameters	yes	node
handle	directions on how to handle a particular class	yes	node

Example:

```
<view:classDefinition source="com.ei.utils.Tween" name="AwesomeTween">
        <handle childHandler="target" />
        <constructor>
                <param name="target"      />
                <param name="property"  />
                <param name="start"       />
                <param name="stop"        />
                <param name="duration"  />
        </constructor>
</view:classDefinition>
```

Command: [CLASS NAME]

Description:
Creates a new instances of a class registered

Parameters:

Parameter	Description
[PARAM]	the required contructor parameter, or property of the object

Example:

```
<view:Tween property=”alpha” start=”0” stop=”1” duration=”1000”>
        <view:Sprite />
</view:Tween>
```

Actionscript Module

Command:
Script

Description:
Compiles Actionscript code and parses it.

Use:

<[namespace]:script>

</[namespace]:script>

Parameters: None

Example:

```
<action:script>
        <![CDATA[
                trace(“Hello World!”);
        ]]>
</action:script>
```

Observer Module

Command: listen

Description:
Listens to a particular object for a event

Parameters:

Parameter	Description
to	the object to listen to
for	the event to listen for

Example:

```
<view:Sprite instance:id="testInstance" />
<observer:listen to="{testInstance}" for="mouseUp">
        <action:script>
                <![CDATA[
                        trace('pressed!;');
                ]]>
        </action:script>
</observer:listen>
```

Command: remove

Description:
Removes an event from a particular object

Parameters:

Parameter	Description
from	the object to remove the event from
event	the event name

Example:

```
<view:Sprite instance:id=”testInstance” />
<observer:listen to=”{testInstance}” for=”mouseUp”>
	<action:script>
		<![CDATA[
			trace(‘pressed!;’);
		]]>
	</action:script>
	<observer:remove event=”mouseUp” from=”{testInstance}” />
</observer:listen>
```

Command: [namespace]:[EVENT]

Description:
a quick way to add an event to an object

Note:
The parameter for the event must not contain curly brackets

Parameters:

Parameter	Description
[EVENT]	the code to execute after

Example:

<view:Sprite instance:id="testInstance" **observer:mouseUp="trace('hello world!'); "** /
>

Load Manager Module

Command:
media

Description:
loads a media file such as an image or swf

Example:

<view:Image source="{**#LoadManager:media('myImage.png')**}" />

Command:
text

Description:
loads in a text file

Example:

<view:TextField text="{**#LoadManager:text('termsOfUse.txt')**}" />

Command:
execute

Description:
pauses the script until all the content under the given namespace is loaded.

Example:

```
<LoadManager:execute  />

<view:TextField text="{#LoadManager:text('termsOfUse.txt')}" />
<view:Image source="{#LoadManager:media('myImage.png')}" />
```