

How to Create a Dynamic Action Plug-in

We are going to develop a region type plug-in which provides a very simple Google Map integration as described in the white paper “Integrating Oracle Application Express with Google Maps” http://www.oracle.com/technology/products/database/application_express/pdf/Integrating_Application_Express_with_Google_Maps.pdf

1. Go to **Shared Components\User Interface\Plug-ins**
2. Click **Create >**
3. Enter **Simple Google Map** into the field **Name**
4. Enter **com.yourcompany.apex.simple_google_map** into the field **Internal Name**. Replace yourcompany with the name of your company.

Note: The value has to be unique within your application. If you want to contribute your plug-in to the APEX community, it should be unique world wide. That’s why you should use a reverse version of your companies domain name as prefix.

5. Set **Type** to **Region**
6. Leave everywhere else the default values and click **Create**
7. Go to section **Attributes**
8. Click **Add Attribute**
9. Enter the following values

Scope: **Application**

Attribute: **1**

Label: **API Key**

Type: **Text**

Required: **Yes**

Translatable: **No**

Display Length: **90**

Max Length: **120**

Default Value: **Get key at <http://code.google.com/apis/maps/signup.html>**

Note: A plug-in can have up to 10 attributes for each scope. These attributes are displayed in the APEX Builder, when a developer selects the plug-in in the “Create Region wizard” or on “Edit Region”. Attributes extend the existing properties of a region and allow a plug-in to prompt the developer in a declarative way for additional data the plug-in requires.

There are two scopes available. “Component” means that an attribute can be entered each time when the plug-in is used for a page item. “Application” means that you can just enter the values once for the whole application.

In our example we will have the same attributes for the scope Application and Component. This allows us to have application wide default values as long as they are not overwritten on Component level.

10. Click **Create and Create Another**

11. Enter the following values

Scope: **Component**
Attribute: **1**
Label: **Width**
Type: **Integer**
Required: **Yes**
Display Width: **4**
Maximum Width: **4**
Default Value: **600**

12. Click **Create and Create Another**

13. Enter the following values

Scope: **Component**
Attribute: **2**
Label: **Height**
Type: **Integer**
Required: **Yes**
Display Length: **4**
Max Length: **4**
Default Value: **400**

14. Click **Create and Create Another**

15. Enter the following values

Scope: **Component**
Attribute: **3**
Prompt: **Page Item containing Location**
Type: **Page Item**
Required: **Yes**
Default Value: **PXX_LOCATION**

16. Click **Create and Create Another**

17. Enter the following values

Scope: **Component**
Attribute: **4**
Prompt: **Page Item containing Tooltip**
Type: **Page Item**
Required: **No**

18. Click **Create**

19. Go to section **Source**

20. Paste the code which you can find in the file **simple_google_map_plugin.sql** into the **PL/SQL Code** field.

Note: See the code comments for additional information on how a render procedure has to look like.

21. Go to section **Callbacks**

22. Enter **render_simple_google_map** into the field **Render Procedure Name**

Note: Because of performance reasons you can also store the plug-in code in a PL/SQL package. In that case you would just use

package_name.render_simple_google_map instead and leave the PL/SQL code field blank. Especially during development of a plug-in this is a more convenient way than writing the PL/SQL code of the plug-in in the text area of the browser.

23. Click **Apply Changes**
24. Let's test our new plug-in!
25. Go to <http://code.google.com/apis/maps/signup.htm> and request an API Key
26. Go to the **definition of our plug-in**
27. In the section **Settings** enter your API key into the field **API Key**
28. Click **Apply Changes**
29. In the Sample Application **create a new blank page with page number 30**
30. Click **Create Region** and create a region with the following settings:

Identify the type of region to add to this page: **Plug-ins**

Plug-in: **Simple Google Map**

Title: **Map**

Width: **600**

Height: **400**

Page Item containing Location: **P30_LOCATION**

Page Item containing Tooltip: **P30_TOOLTIP**

31. Create two Text Field page items called **P30_LOCATION** and **P30_TOOLTIP**
32. Create a button "displayed among this region's items" called **P30_GO**
33. **Run** the page
34. Enter **40.759726,-73.980341** into the Location field and **New York** into the Tooltip field
35. Click **Go** button and it shows the address on the Google map

Note: As described in the white paper you can use the Google Geocode Service to translate a real address into the corresponding geo coordinates.

36. If you are done with testing, go again to **Shared Components\User Interface\Plug-ins\Simple Google Map**
37. Click **Export Plug-in** in the Task sidebar
38. Follow the wizard steps to export the plug-in
39. You are done! You can now share and use the plug-in in any application by installing it with the **Install button** in Shared Components\User Interface\Plug-ins