

How to Create a Dynamic Action Plug-in

We are going to develop a dynamic action plug-in which highlights a DOM object (like a page item) for a few seconds to indicate to the user that a value has changed or the field is very important.

1. Go to **Shared Components\User Interface\Plug-ins**
2. Click **Create >**
3. Enter **Highlight** into the field **Name**
4. Enter **com.yourcompany.apex.highlight** 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 **Dynamic Action**
6. Leave everywhere else the default values and click **Create**
7. Open the file **highlight.js**
8. Replace **yourcompany** with your company name
9. Save the file.

Note: See the code comments for additional information on how such a plug-in javascript function has to look like.

It's highly recommended that you use a javascript/CSS compressor (eg.: <http://developer.yahoo.com/yui/compressor/>) to reduce the size of your javascript and CSS code when you are done with your development!

10. Go to section **Files**
11. Upload the **highlight.js** file

Note: A plug-in has its own file storage which can store images, css- and javascript files. This concept has the advantage that it allows a single click installation of the plug-in, without having to upload something to a web server, ... A user can still do that to optimize performance, but by default he is done after installing the plug-in.

12. Go to section **Attributes**
13. Click **Add Attribute**
14. Enter the following values

Scope: **Application**
Attribute: **1**
Label: **Color**
Type: **Text**
Required: **No**
Translatable: **No**

Display Length: **10**

Max Length: **10**

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 Dynamic Action wizard” or on “Edit Dynamic Action”. Attributes extend the existing properties of a dynamic action 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.

15. Click **Create and Create Another**

16. Enter the following values

Scope: **Application**

Attribute: **2**

Label: **Speed**

Type: **Select List**

Required: **No**

List Values: **Slow;slow,Normal;normal,Fast;fast**

17. Click **Create and Create Another**

18. Enter the following values

Scope: **Component**

Attribute: **1**

Label: **Color**

Type: **Text**

Required: **No**

Translatable: **No**

Display Length: **10**

Max Length: **10**

19. Click **Create and Create Another**

20. Enter the following values

Scope: **Component**

Attribute: **2**

Prompt: **Speed**

Type: **Select List**

Required: **No**

List Values: **Slow;slow,Normal;normal,Fast;fast**

21. Click **Create**

22. Go to section **Source**

23. Paste the code which you can find in the file **highlight_plugin.sql** into the **PL/SQL Code** field. Replace **yourcompany** with your company name.

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

24. Go to section **Callbacks**

25. Enter **render_highlight** 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_highlight` 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.

26. Click **Apply Changes**

27. Let's test our new plug-in!

28. In the Sample Application go to **page 6**

29. Create a **Dynamic Action** with the following values:

Name: **Highlight list price for computer**

Sequence: **10**

Element Type: **Item**

Element: **P6_CATEGORY**

Condition Type: **equal to**

Expression: **Computer**

30. Click **Create**

31. Go to section **Actions**

32. Click Add Action

33. Use the following values

Fire When Event Result is: **True**

Sequence: **10**

Action: **Highlight**

Fire On Page Load: **No**

Element Type: **Item**

Elements: **P6_LIST_PRICE**

34. Click **Apply Changes**

35. You can play with the color and the speed by clicking the **Edit Icon** of the just created Action. Enter `#ff0000` or red for the color.

36. **Run** the page

37. Change the Category to **Computer**

38. If you are done with testing, go again to **Shared Components\User Interface\Plug-ins\Highlight**

39. Click **Export Plug-in** in the Task sidebar

40. Follow the wizard steps to export the plug-in

41. 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