

UI Plugins

Feature & implementation overview

Vojtech Szöcs

Sr. Software Engineer, Red Hat

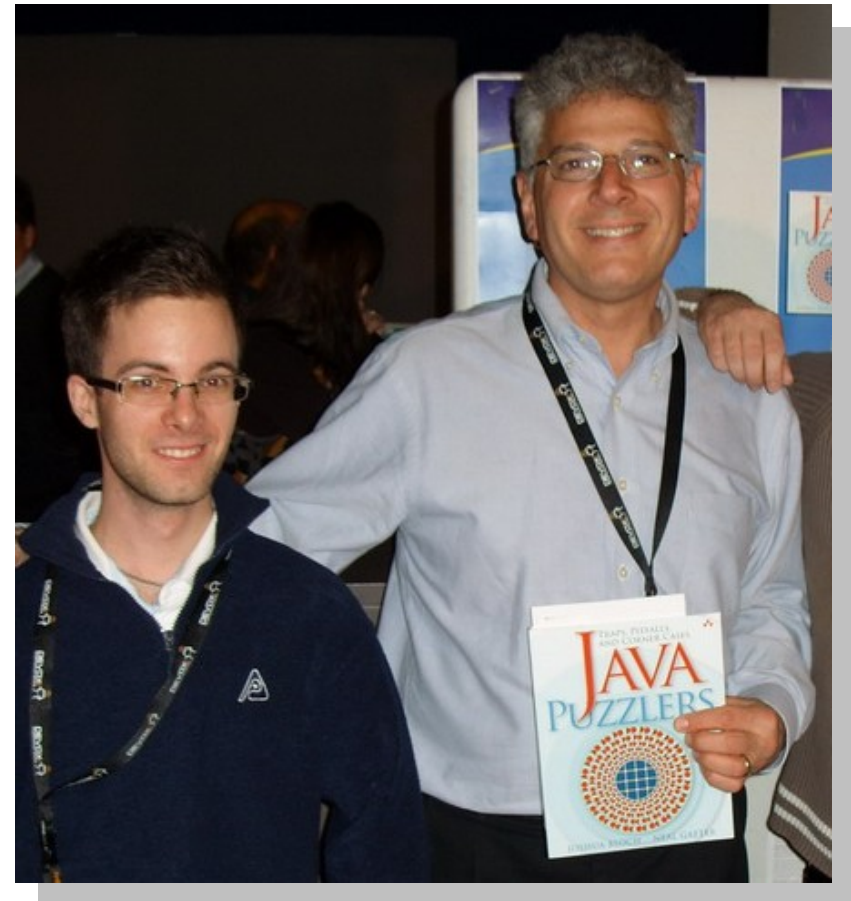
October 9, 2012



About the speaker



- oVirt / RHEV UI maintainer
- Member of UX team, maintaining Frontend application architecture and infrastructure
- Lead developer of UI Plugins feature



Topics covered in this session



- Feature overview
- Current state of plugin infrastructure
- Tutorial on writing simple plugin
- Tips for implementing new plugin features

Feature overview



- Extend or customize oVirt Engine functionality through WebAdmin user interface
- Plugins can be packaged and distributed for use with oVirt Engine
- Planned oVirt feature, currently in proof-of-concept development stage
- Updates and announcements on engine-devel list

Technical background

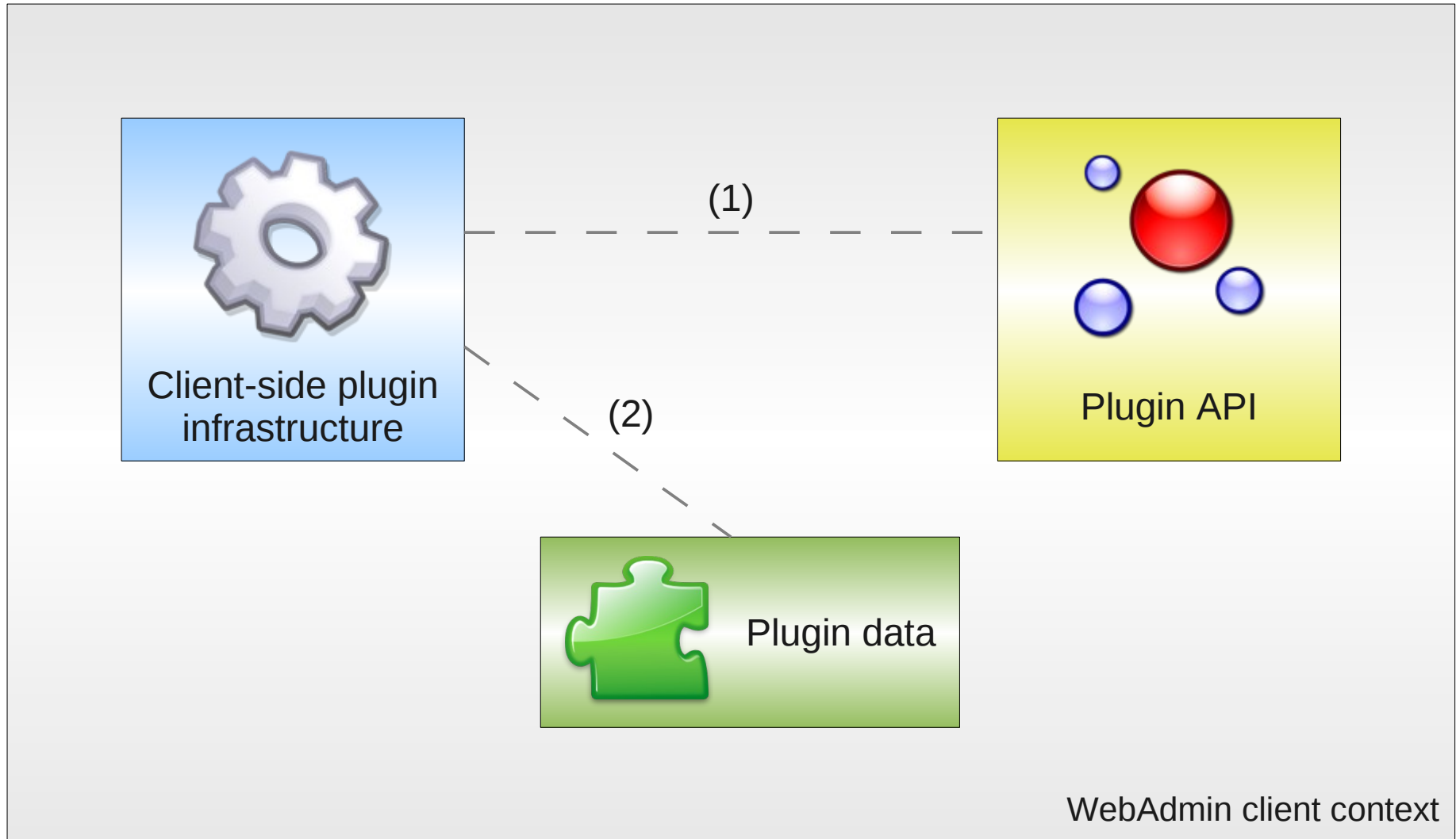


- Plugins integrate with WebAdmin on client using JavaScript programming language
- Simple and flexible plugin infrastructure
 - WebAdmin exposes plugin API
 - Plugins consume the API
 - WebAdmin calls (invokes) plugins at key events during the application runtime

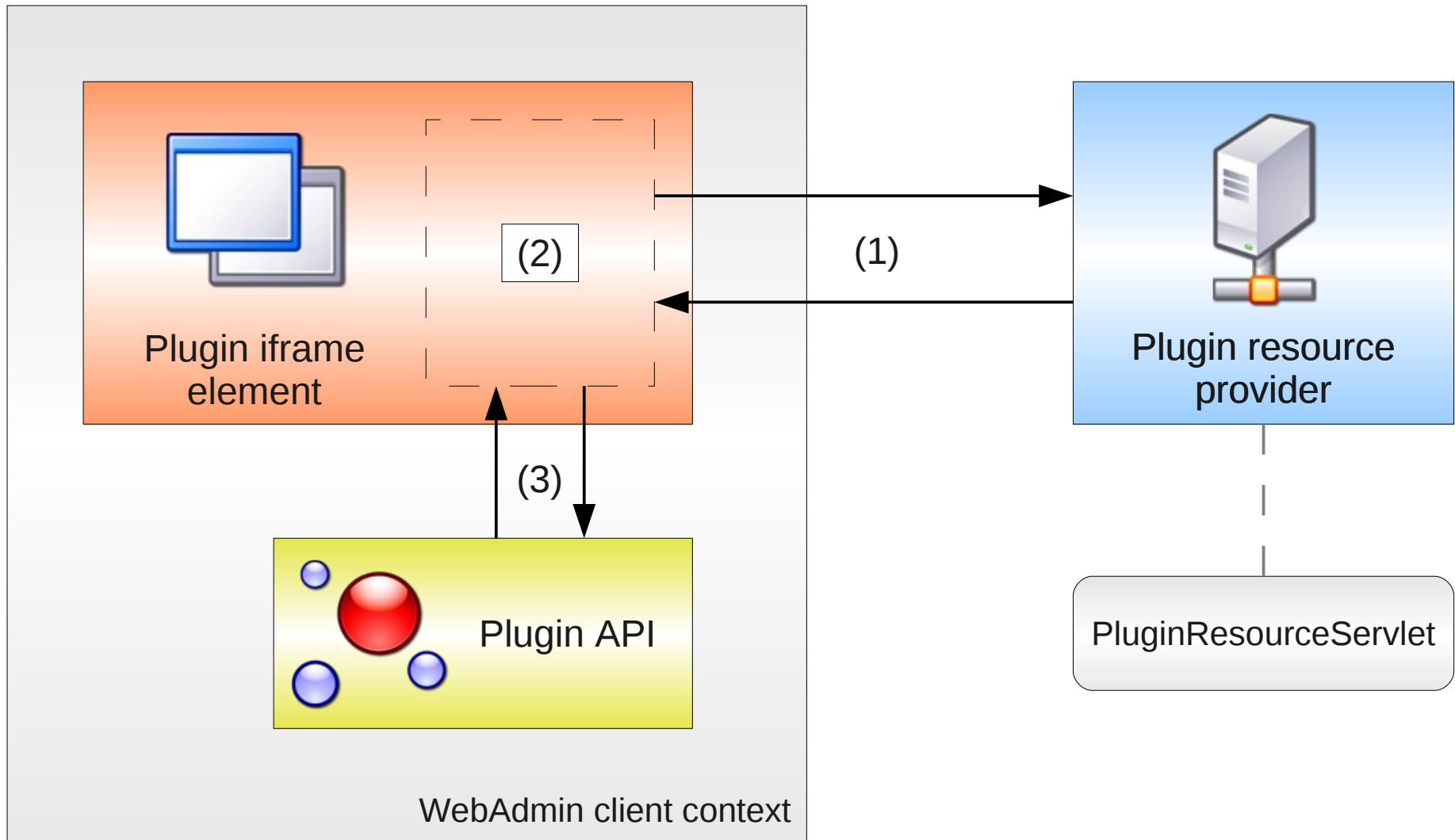
How it works, part 1



How it works, part 2



How it works, part 3



Plugin host page example, part 1



```
<!DOCTYPE html>
<html>
<head>
  <!-- Fetch additional resources if necessary -->
  <script type="text/javascript" src="jquery-min.js"></script>

  <!-- Actual plugin code -->
  <script>

  </script>

</head>
<body> <!-- HTML body is intentionally empty --> </body>
</html>
```

Plugin host page example, part 2



```
<!DOCTYPE html>
<html>
<head>
  <!-- Fetch additional resources if necessary -->
  <script type="text/javascript" src="jquery-min.js"></script>

  <!-- Actual plugin code -->
  <script>
    // Access plugin API from iframe context
    var api = parent.pluginApi('foo');

    // Register plugin event handler functions
    api.register({
      UiInit: function() {
        api.addMainTab('Foo Tab', 'foo-tab', 'http://foo.com/');
      }
    });

    // Tell plugin infrastructure that we are ready
    api.ready();
  </script>
</head>
<body> <!-- HTML body is intentionally empty --> </body>
</html>
```

Design implications, part 1



- JavaScript allows dynamic behavior in plugins

Design implications, part 2



- JavaScript allows dynamic behavior in plugins
- WebAdmin vs. plugin integration happens directly on the client

Design implications, part 3



- JavaScript allows dynamic behavior in plugins
- WebAdmin vs. plugin integration happens directly on the client
- Using iframe as plugin execution sandbox

Design implications, part 4



- JavaScript allows dynamic behavior in plugins
- WebAdmin vs. plugin integration happens directly on the client
- Using iframe as plugin execution sandbox
- WebAdmin vs. plugin communication is subject to Same-Origin Policy

`http://my.domain/doc.html`

`https://other.domain/doc.html`

Current state of plugin infrastructure



- Core infrastructure pretty much complete (rev.5)
- Focus on specific plugin API features (rev.6)
 - Custom content shown in main / sub tabs
 - Data grid context-sensitive buttons / menu items

Infrastructure explained, part 1



- Plugin descriptor (JSON)

```
/usr/share/engine/ui-plugins/foo.json
```

```
{
```

```
// A name that uniquely identifies the plugin (required)
```

```
"name": "foo",
```

```
// URL of plugin host page that invokes the plugin code (required)
```

```
"url": "/webadmin/webadmin/plugin/foo/start.html",
```

```
// Default configuration object associated with the plugin (optional)
```

```
"config": { "band": "ZZ Top", "classic": true, "score": 10 },
```

```
// Path to plugin static resources (optional)
```

```
// Used when serving plugin files through PluginResourceServlet
```

```
// This path is relative to /usr/share/ovirt-engine/ui-plugins
```

```
"resourcePath": "foo-files"
```

```
}
```


Infrastructure explained, part 2



- Plugin configuration (JSON)

```
/etc/engine/ui-plugins/foo-config.json
```

```
{  
  
  // Custom configuration object associated with the plugin (optional)  
  // This overrides the default plugin descriptor configuration, if any  
  "config": { "band": "AC/DC" },  
  
  // Whether the plugin should be loaded on WebAdmin startup (optional)  
  // Default value is 'true'  
  "enabled": true,  
  
  // Relative order in which the plugin will be loaded (optional)  
  // Default value is Integer.MAX_VALUE (lowest order)  
  "order": 0  
}
```

Infrastructure explained, part 3



- Merged plugin configuration (JSON)

```
{ "band": "ZZ Top", "classic": true, "score": 10 }
```

+

```
{ "band": "AC/DC" }
```

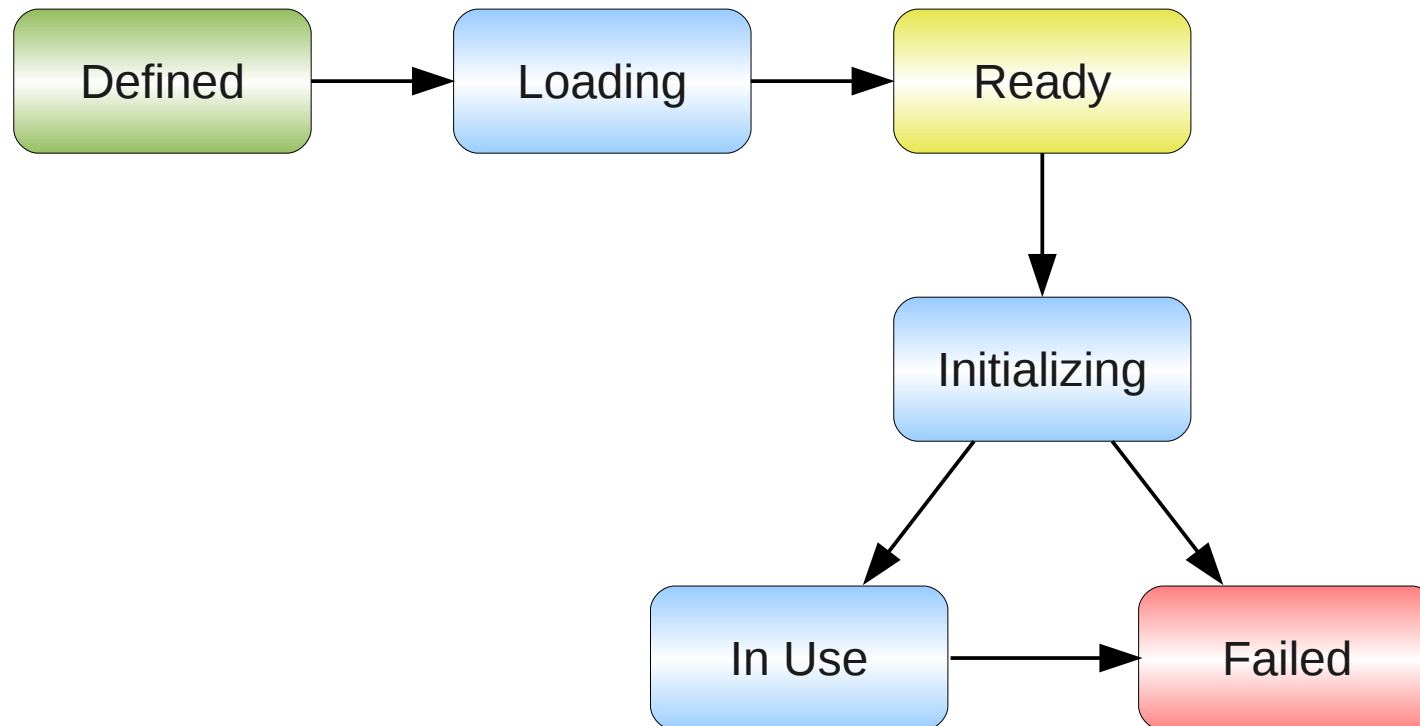
=

```
{ "band": "AC/DC", "classic": true, "score": 10 }
```

Infrastructure explained, part 4



- Plugin lifecycle (client-side)



Tutorial: writing simple plugin



- (1) Write plugin descriptor
- (2) Write plugin host page
- (3) Show plugin in action



Tip: new plugin API function, part 1



- Add new function in `PluginManager.exposePluginApi`

```
// GWT JSNI method that creates and exposes global plugin API object
private native void exposePluginApi() /*-{

    ...

    // Add new function to pluginApi prototype
    pluginApi.fn = pluginApi.prototype = {

        myFunction: function() {
            // Don't forget to validate plugin action
            if (canDoPluginAction(this.pluginName)) {
                // Delegate to Java method through GWT JSNI
                uiFunctions.@org...PluginUiFunctions::myMethod();
            }
        }
    };

    ...

} -*/;
```

Tip: new plugin API function, part 2



- Add new method in target Java class

```
package org.ovirt.engine.ui.webadmin.plugin;

// Implements UI related functionality exposed to plugins
public class PluginUiFunctions {

    ...

    public void myMethod() {
        // Plugin API function implementation
    }

    ...
}
```

Tip: new application event, part 1



- Create new event using @GenEvent

```
package org.ovirt.engine.ui.webadmin.plugin.event;
```

```
@GenEvent
```

```
public class HostActivated {
```

```
    List<VDS> hosts;
```

```
}
```

Tip: new application event, part 2



- Use EventBus to fire event within WebAdmin code

```
public class MainTabHostView ... {  
  
    void initTable() {  
  
        getTable().addActionButton(  
            new WebAdminButtonDefinition<VDS>(constants.activateHost()) {  
                @Override  
                protected UICommand resolveCommand() {  
                    return getMainModel().getActivateCommand();  
                }  
  
                @Override  
                public void onClick(List<VDS> selectedItems) {  
                    // Execute default (Activate) command logic  
                    super.onClick(selectedItems);  
  
                    // Fire event through the EventBus  
                    HostActivatedEvent.fire(eventBus, selectedItems);  
                }  
            });  
    }  
}
```


Tip: new application event, part 3



- Handle event in PluginEventHandler constructor

```
@Inject
public PluginEventHandler(EventBus eventBus, final PluginManager manager) {

    // Add HostActivatedEvent handler
    eventBus.addHandler(HostActivatedEvent.getType(),
        new HostActivatedHandler() {

            @Override
            public void onHostActivated(HostActivatedEvent event) {
                // Optional context object, could carry
                // data about hosts that were activated
                JavaScriptObject contextObject = null;

                // Use PluginManager to invoke specific event handler
                // function on all plugins which are currently in use
                manager.invokePlugins("HostActivated", contextObject);
            }
        });
}
```

Future plans



- Implement more plugin features (API, events)
- Update oVirt wiki page
<http://www.ovirt.org/wiki/Features/UIPlugins>
- Transition out of proof-of-concept stage



Thank you!

<http://www.ovirt.org/>

vszocs@redhat.com

vszocs at #ovirt (irc.oftc.net)