Bean Shell Programing - Invoke JSON Tool Plugin Programmatically

In Bean Shell plugin, we can call other Joget plugins programmatically to perform specific task without the need to rewrite specific functionality that existing plugins may have already addressed.

Example 1: JSON Tool

In this Bean Shell code, we will make use of JSON Tool.

Before we move on, let's inspect how the plugin is configured first.

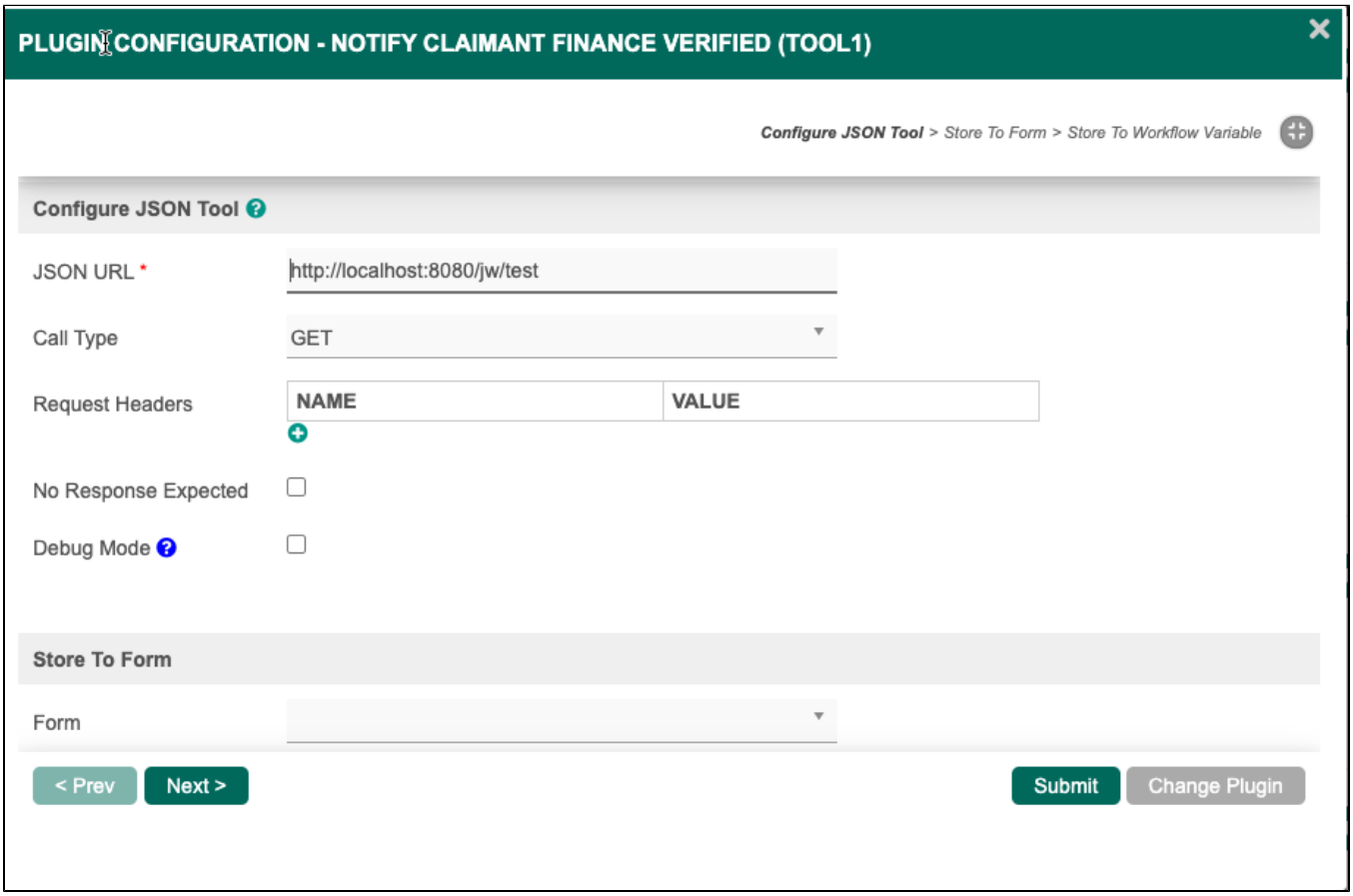

Figure 1: Configure JSON Tool

We will need to obtain the attributes of the configured plugin, so that we can then programmatically configure it with coding.

One quick way to learn the attributes needed is to make use of the browser's Developer Tools > Network to observe the submission made when we hit the submit button.

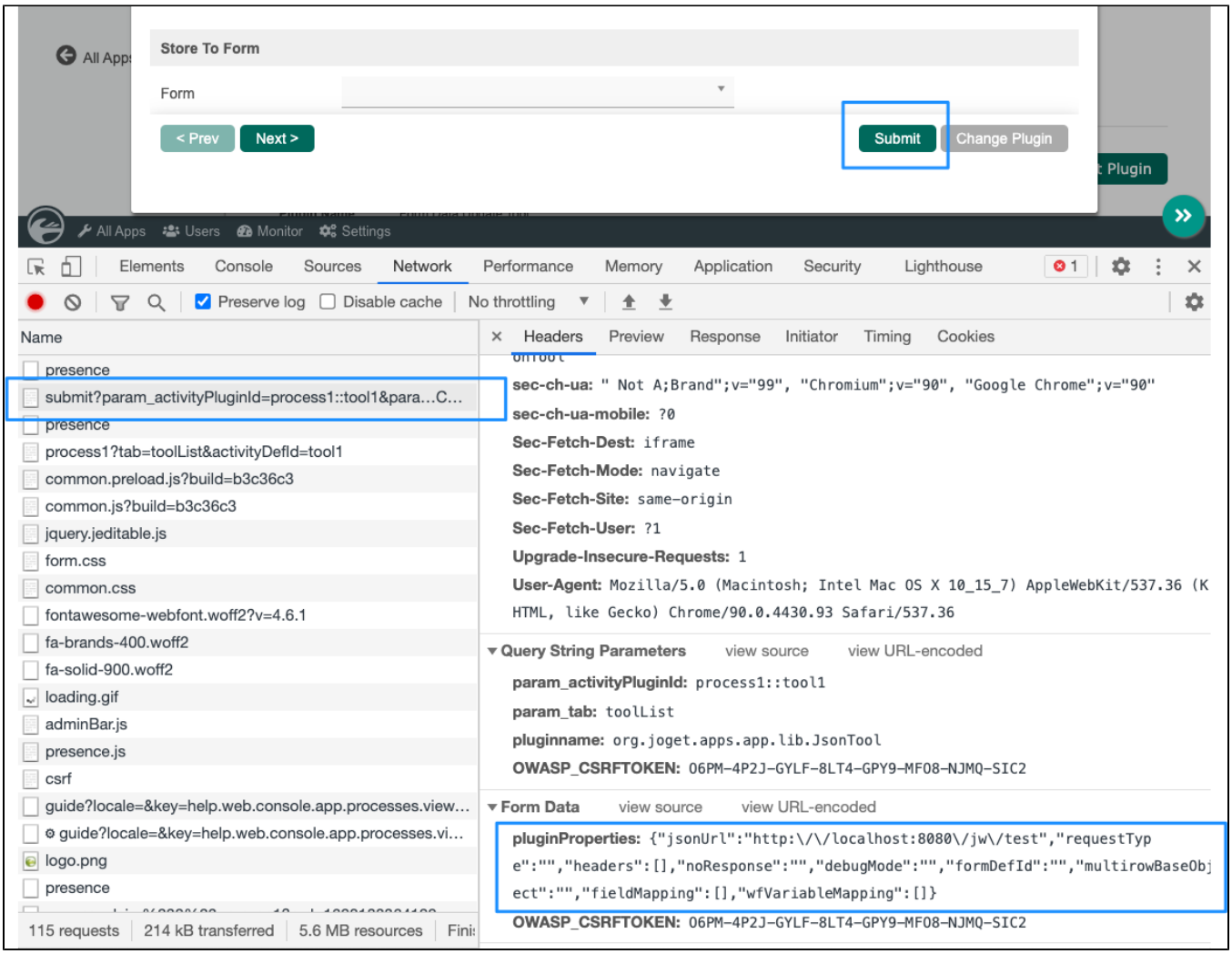

Figure 2: JSON Tool Attributes

Plugin Properties

```
{"jsonUrl":"http:\/\/localhost:8080\/jw\/test","requestType":"","headers":[],"noResponse":"","debugMode":"","
formDefId":"","multirowBaseObject":"","fieldMapping":[],"wfVariableMapping":[]}
```

We can also check out the plugin property options source code of the plugin too at https://github.com/jogetworkflow/jw-community/blob/7.0-SNAPSHOT/wflow-core/src/main/resources/properties/app/jsonTool.json or by inspecting the DOM elements in the JSON Tool configuration screen.

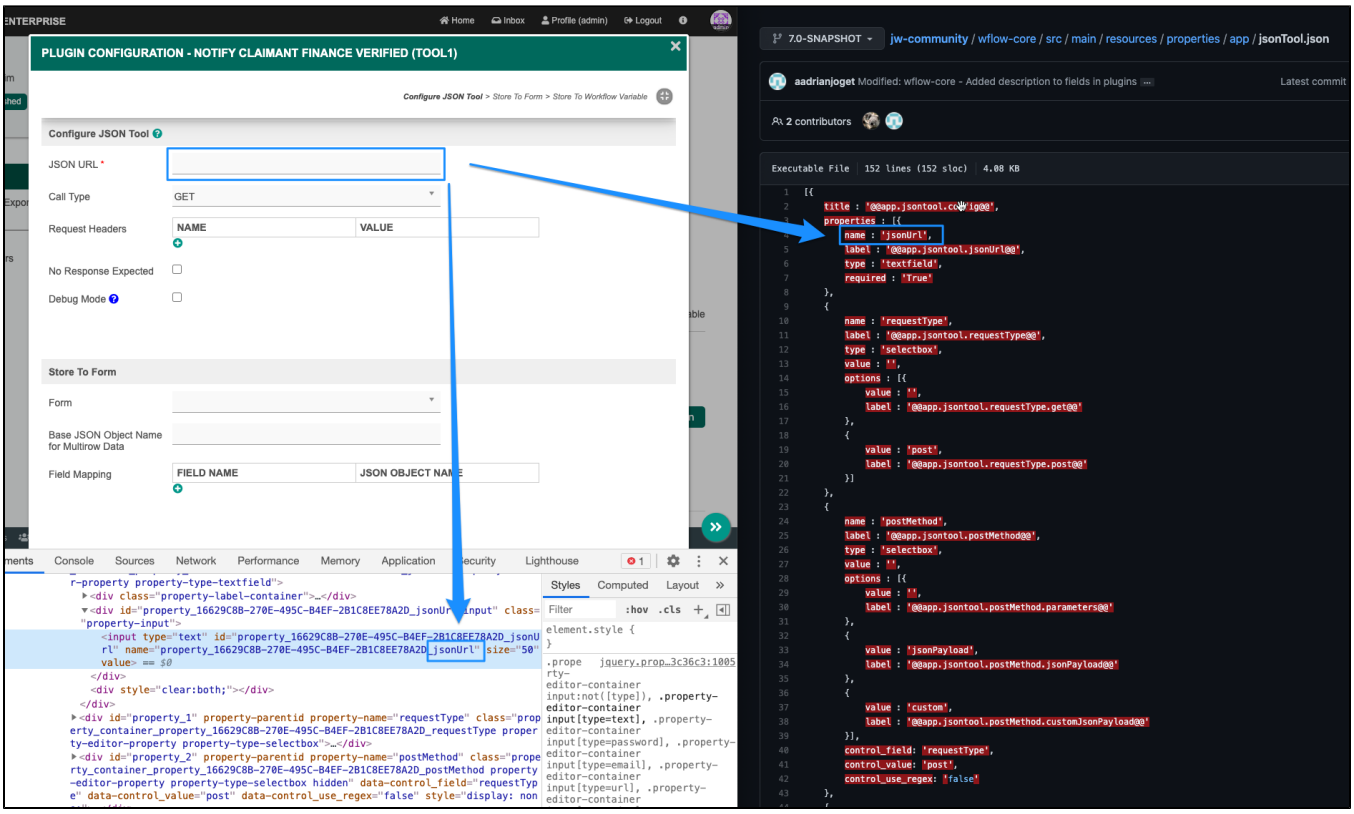

Figure 3: Obtaining JSON Tool Attrbutes

With this infomation on hand, we can now configure the plugin through coding. Configurations are made in the **propertiesMap** variable in the sample code below.

Sample code:

```
import java.util.Map;
import java.util.HashMap;
import javax.servlet.http.HttpServletRequest;
import org.joget.apps.app.service.AppUtil;
import org.joget.plugin.base.Plugin;
import org.joget.plugin.base.PluginManager;
import org.joget.workflow.util.WorkflowUtil;
import org.joget.apps.app.model.AppDefinition;
import org.joget.apps.app.service.AppPluginUtil;
import org.joget.plugin.property.model.PropertyEditable;
import org.joget.workflow.model.service.WorkflowManager;
import org.joget.workflow.model.WorkflowProcess;
import org.joget.apps.app.service.AppUtil;
import org.joget.workflow.model.WorkflowProcessLink;
import org.joget.workflow.model.WorkflowProcessResult;
import org.joget.workflow.model.WorkflowActivity;
import org.joget.workflow.model.WorkflowVariable;
import org.joget.commons.util.LogUtil;
import org.joget.apps.app.lib.JsonTool;

public void callPlugin(){
        try {
        PluginManager pluginManager = (PluginManager) AppUtil.getApplicationContext().getBean("pluginManager");

                //plugin to call
        String pluginName = "org.joget.apps.app.lib.JsonTool";
        Plugin plugin = pluginManager.getPlugin(pluginName);
        AppDefinition appDef = AppUtil.getCurrentAppDefinition();

                //JSON API to call to start a new process
        String jsonURL = "http://localhost:8080/jw/web/json/workflow/process/start/expenseclaim:latest:
process1";
```

```
                //prepare workflow variables for the process
        List params = new ArrayList();

        Map param = new HashMap();
        param.put("name", "var_approval");
        param.put("value", "New");
        params.add(param);

                param = new HashMap();
        param.put("name", "var_SelectApprover");
        param.put("value", "clark");
        params.add(param);

        Object[] paramsArray = params.toArray();

                //prepare propertiesMap mapped specifically for JSON tool
        Map propertiesMap = new HashMap();
        propertiesMap.put("jsonUrl", jsonURL);
        propertiesMap.put("requestType", "post");
        propertiesMap.put("debugMode", "true");
        propertiesMap.put("headers", new Object[]{});
        //propertiesMap.put("params", new Object[]{ params });
        propertiesMap.put("params", paramsArray );

                //obtain default properties set, if any
                propertiesMap = AppPluginUtil.getDefaultProperties(plugin, propertiesMap, appDef, null);

        //set properties into the JSON tool programmatically
        if (plugin instanceof PropertyEditable) {
            ((PropertyEditable) plugin).setProperties(propertiesMap);
            LogUtil.info("migrateProcess", "set properties");
        }

        //invoke the JSON plugin
                plugin.execute(propertiesMap);

        LogUtil.info("callPlugin", "execution finished");
    } catch (Exception ex) {
        LogUtil.error("callPlugin", ex, "Error");
    }
}

callPlugin();
```