Slack

Slack Notification　Bean Shell

1.

Slack　Joget Workflow

2.

　Slack

3.

Slack NotificationUser Notification

4.

/Slack

5./ API

slack-webhook　Slack　org.joget.apps.app.lib.UserNotificationAuditTrail

6.

Joget

Macbook ProJoget5.0.1

```
- Home
  - joget
    - plugins
    - jw-community
      -5.0.1
```

“”“jw-community”Joget Workflow

“plugins”maven

```
cd joget/plugins/
~/joget/jw-community/5.0.1/wflow-plugin-archetype/create-plugin.sh org.joget slack_notification 5.0.1
```

shellmaven

```
Define value for property 'version':  1.0-SNAPSHOT: : 5.0.0
[INFO] Using property: package = org.joget
Confirm properties configuration:
groupId: org.joget
artifactId: slack_notification
version: 5.0.0
package: org.joget
Y: : y
```

“BUILD SUCCESS”“plugins”“slack_notification”

IDEmaven NetBeans

7. !

a.

“org.joget”“SlackNotification” **org.joget.apps.app.lib.UserNotificationAuditTrail** **org.joget.plugin.base.DefaultAuditTrailPlugin** **org.joget.plugin.base.PluginWebSupport** Web Service .

b.

AppPluginUtil.getMessagei18nMESSAGE_PATH

Implementation of all basic abstract methods

```
package org.joget;

import org.joget.apps.app.lib.UserNotificationAuditTrail;
import org.joget.apps.app.service.AppPluginUtil;
import org.joget.apps.app.service.AppUtil;

public class SlackNotification extends UserNotificationAuditTrail {
    private final static String MESSAGE_PATH = "message/SlackNotification";

    @Override
    public String getName() {
        return "Slack Notification";
    }
    @Override
    public String getVersion() {
        return "5.0.0";
    }

    @Override
    public String getClassName() {
        return getClass().getName();
    }

    @Override
    public String getLabel() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.SlackNotification.pluginLabel", getClassName(),
MESSAGE_PATH);
    }
    @Override
    public String getDescription() {
        //support i18n
        return AppPluginUtil.getMessage("org.joget.SlackNotification.pluginDesc", getClassName(), MESSAGE_PATH);
    }

    @Override
    public String getPropertyOptions() {
        return AppUtil.readPluginResource(getClass().getName(), "/properties/slackNotification.json", null,
true, MESSAGE_PATH);
    }
}
```

UIgetPropertyOptions “/properties/slackNotification.json”“slack_notification / src / main”“resources / properties”“properties”“slackNotification.json”

“@@ message.key @@”i18n userNotificationAuditTrail.json

```
[{
    title : '@@SlackNotification.config@@',
    properties : [
    {
        name : 'apiurl',
        label : '@@SlackNotification.url@@',
        type : 'textfield',
        required : 'true'
    },
    {
        label : '@@SlackNotification.from@@',
        type : 'header'
    },
    {
        name : 'username',
        label : '@@SlackNotification.fromUsername@@',
        type : 'textfield',
        value : '@@SlackNotification.fromUsername.value@@'
    },
```

```
{
    name : 'customIcon',
    label : '@@SlackNotification.customIcon@@',
    type : 'selectbox',
    value : 'joget',
    options : [{
        value : '',
        label : '@@SlackNotification.customIcon.none@@'
    },
    {
        value : 'joget',
        label : '@@SlackNotification.customIcon.joget@@'
    },
    {
        value : 'url',
        label : '@@SlackNotification.customIcon.url@@'
    },
    {
        value : 'emoji',
        label : '@@SlackNotification.customIcon.emoji@@'
    }]
},
{
    name : 'iconUrl',
    label : '@@SlackNotification.customIcon.url@@',
    type : 'textfield',
    required : 'true',
    control_field: 'customIcon',
    control_value: 'url',
    control_use_regex: 'false'
},
{
    name : 'iconEmoji',
    label : '@@SlackNotification.customIcon.emoji@@',
    type : 'textfield',
    required : 'true',
    control_field: 'customIcon',
    control_value: 'emoji',
    control_use_regex: 'false'
},
{
    label : '@@SlackNotification.to@@',
    type : 'header'
},
{
    name : 'usernameTransform',
    label : '@@SlackNotification.usernameTransform@@',
    description : '@@SlackNotification.usernameTransform.desc@@',
    type : 'textfield',
    value : '@@SlackNotification.usernameTransform.value@@',
    required : 'True'
},
{
    label : '@@SlackNotification.message@@',
    type : 'header'
},
{
    name : 'text',
    label : '@@SlackNotification.text@@',
    description : '@@SlackNotification.text.desc@@',
    type : 'codeeditor',
    required : 'True'
},
{
    name : 'unfurl_links',
    label : '@@SlackNotification.unfurl_links@@',
    description : '@@SlackNotification.unfurl_links.desc@@',
    type : 'checkbox',
    value : 'true',
    options : [{
        value : 'true',
```

```
            label : ''
        }]
    },
    {
        name : 'unfurl_media',
        label : '@@SlackNotification.unfurl_media@@',
        description : '@@SlackNotification.unfurl_media.desc@@',
        type : 'checkbox',
        value : 'true',
        options : [{
            value : 'true',
            label : ''
        }]
    }],
    buttons : [{
        name : 'sendTestMessage',
        label : '@@SlackNotification.sendTestMessage@@',
        ajax_url : '[CONTEXT_PATH]/web/json/app[APP_PATH]/plugin/org.joget.SlackNotification/service?
action=sendTestMessage',
        fields : ['url'],
        addition_fields : [
            {
                name : 'testChannel',
                label : '@@SlackNotification.sendTestMessage.testChannel@@',
                type : 'textfield'
            }
        ]
    }]
},
{
    title : '@@app.usernotificationaudittrail.notificationLink@@',
    properties : [
    {
        name : 'base',
        label : '@@app.usernotificationaudittrail.baseUrl@@',
        type : 'textfield',
        description : '@@app.usernotificationaudittrail.baseUrl.desc@@',
        required : 'True'
    },
    {
        name : 'url',
        label : '@@app.usernotificationaudittrail.url@@',
        type : 'textfield'
    },
    {
        name : 'urlName',
        label : '@@app.usernotificationaudittrail.urlName@@',
        type : 'textfield'
    },
    {
        name : 'parameterName',
        label : '@@app.usernotificationaudittrail.parameterName@@',
        description : '@@app.usernotificationaudittrail.parameterName.desc@@',
        type : 'textfield',
        value : 'activityId'
    },
    {
        name : 'passoverMethod',
        label : '@@app.usernotificationaudittrail.passoverMethod@@',
        type : 'selectbox',
        value : 'param',
        options : [{
            value : 'none',
            label : '@@app.usernotificationaudittrail.passoverMethod.none@@'
        },
        {
            value : 'append',
            label : '@@app.usernotificationaudittrail.passoverMethod.append@@'
        },
        {
            value : 'param',
```

```
            label : '@@app.usernotificationaudittrail.passoverMethod.param@@'
        }]
    }]
},
{
    title : '@@app.usernotificationaudittrail.advanced@@',
    properties : [{
        name : 'exclusion',
        label : '@@app.usernotificationaudittrail.activityExclusion@@',
        type : 'multiselect',
        size : '10',
        options_ajax : '[CONTEXT_PATH]/web/json/app[APP_PATH]/plugin/org.joget.apps.app.lib.
UserNotificationAuditTrail/service?action=getActivities'
    }]
}]
```

UserNotificationAuditTrailUserNotificationAuditTrailsendEmail

```
    private SlackApi api = null;

    @Override
    protected void sendEmail (final Map props, final AuditTrail auditTrail, final WorkflowManager
workflowManager, final List<String> users, final WorkflowActivity wfActivity) {
        new PluginThread(new Runnable() {
            public void run() {
                WorkflowUserManager workflowUserManager = (WorkflowUserManager) AppUtil.getApplicationContext().
getBean("workflowUserManager");
                String base = (String) props.get("base");
                String url = (String) props.get("url");
                String urlName = (String) props.get("urlName");
                String parameterName = (String) props.get("parameterName");
                String passoverMethod = (String) props.get("passoverMethod");
                String text = (String) props.get("text");
                String linkLabel = AppPluginUtil.getMessage("SlackNotification.viewAssignment", getClassName(),
MESSAGE_PATH);
                String activityInstanceId = wfActivity.getId();
                String link = getLink(base, url, passoverMethod, parameterName, activityInstanceId);
                if (!link.startsWith("http")) {
                    if (!link.startsWith("/")) {
                        link = "/" + link;
                    }
                    link = base + link;
                }

                SlackMessage message = createMessage();
                try {
                    for (String username : users) {
                        workflowUserManager.setCurrentThreadUser(username);
                        WorkflowAssignment wfAssignment = null;
                        int count = 0;
                        do {
                            wfAssignment = workflowManager.getAssignment(activityInstanceId);
                            if (wfAssignment == null) {
                                Thread.sleep(4000); //wait for assignment creation
                            }
                            count++;
                        } while (wfAssignment == null && count < 5); // try max 5 times
                        if (wfAssignment != null) {
                            String channel = getSlackUsername(username, wfAssignment);
                            if (channel != null && !channel.isEmpty()) {
                                message.setText(AppUtil.processHashVariable(text, wfAssignment, null, null));
                                message.setAttachments(new ArrayList<SlackAttachment>());
                                SlackAttachment attachment = new SlackAttachment();
                                attachment.setFallback(link);
                                if (urlName != null && !urlName.isEmpty()) {
                                    attachment.setTitle(AppUtil.processHashVariable(urlName, wfAssignment,
null, null));
                                } else {
                                    attachment.setTitle(linkLabel);
```

```
                                }
                                attachment.setTitleLink(link);
                                message.addAttachments(attachment);
                                try {
                                    LogUtil.info(SlackNotification.class.getName(), "Sending slack message to "
+ username);
                                    sendMessage(channel, message);
                                    LogUtil.info(SlackNotification.class.getName(), "Sending slack message
completed to " + username);
                                } catch (Exception ex) {
                                    LogUtil.error(UserNotificationAuditTrail.class.getName(), ex, "Error
sending slack message");
                                }
                            }
                        } else {
                            LogUtil.debug(UserNotificationAuditTrail.class.getName(), "Fail to retrieve
assignment for " + username);
                        }
                    }
                } catch (Exception e) {
                    LogUtil.error(UserNotificationAuditTrail.class.getName(), e, "Error executing plugin");
                }
            }
        }).start();
    }

    protected SlackApi getApi() {
        if (api == null) {
            api = new SlackApi(getPropertyString("apiurl"));
        }
        return api;
    }

    protected String getSlackUsername(String username, WorkflowAssignment assignment) {
        String syntax = getPropertyString("usernameTransform");
        syntax = syntax.replaceAll(StringUtil.escapeRegex("{username}"), StringUtil.escapeRegex(username));
        return AppUtil.processHashVariable(syntax, assignment, null, null);
    }

    protected void sendMessage(String channel, SlackMessage message) {
        if (message == null) {
            message = createMessage();
        }
        if (channel != null && !channel.isEmpty()) {
            message.setChannel(channel);
        }

        getApi().call(message);
    }

    protected SlackMessage createMessage() {
        SlackMessage message = new SlackMessage();

        String username = getPropertyString("username");
        if (!username.isEmpty()) {
            message.setUsername(username);
        }

        String customIcon = getPropertyString("customIcon");
        if (!customIcon.isEmpty()) {
            if ("joget".equals(customIcon)) {
                HttpServletRequest request = WorkflowUtil.getHttpServletRequest();
                if (request != null) {
                    String url = request.getScheme() + "://" + request.getServerName() + ":" + request.
getServerPort() + request.getContextPath() + "/images/v3/logo.png";
                    message.setIcon(url);
                }
            } else if ("url".equals(customIcon)) {
                message.setIcon(getPropertyString("iconUrl"));
            } else {
                message.setIcon(getPropertyString("iconEmoji"));
```

```
            }
        }

        message.setUnfurlLinks("true".equalsIgnoreCase(getPropertyString("unfurl_links")));
        message.setUnfurlMedia("true".equalsIgnoreCase(getPropertyString("unfurl_media")));

        return message;
    }
```

webServiceAPI

```
    public void webService(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        boolean isAdmin = WorkflowUtil.isCurrentUserInRole(WorkflowUserManager.ROLE_ADMIN);
        if (!isAdmin) {
            response.sendError(HttpServletResponse.SC_UNAUTHORIZED);
            return;
        }

        String action = request.getParameter("action");
        if ("sendTestMessage".equals(action)) {
            String message = "";
            try {
                AppDefinition appDef = AppUtil.getCurrentAppDefinition();

                String url = AppUtil.processHashVariable(request.getParameter("url"), null, null, null, appDef);
                String testChannel = AppUtil.processHashVariable(request.getParameter("testChannel"), null,
null, null, appDef);

                setProperty("apiurl", url);
                setProperty("text", AppPluginUtil.getMessage("SlackWebhookTool.testMessage", getClassName(),
MESSAGE_PATH));

                if (testChannel != null && !testChannel.isEmpty()) {
                    sendMessage(testChannel, null);
                } else {
                    sendMessage(null, null);
                }

                message = AppPluginUtil.getMessage("SlackWebhookTool.sendTestMessage.success", getClassName(),
MESSAGE_PATH);
            } catch (Exception e) {
                LogUtil.error(this.getClassName(), e, "Fail to send Test Message to Slack");
                message = AppPluginUtil.getMessage("SlackWebhookTool.sendTestMessage.fail", getClassName(),
MESSAGE_PATH) + "\n" + StringEscapeUtils.escapeJavaScript(e.getMessage());
            }
            try {
                JSONObject jsonObject = new JSONObject();
                jsonObject.accumulate("message", message);
                jsonObject.write(response.getWriter());
            } catch (Exception e) {
                //ignore
            }
        } else {
            response.setStatus(HttpServletResponse.SC_NO_CONTENT);
        }
    }
```

c.

POM“jsp-api”“slack-webhook”

```
        <!-- Change plugin specific dependencies here -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.0</version>
        </dependency>
        <dependency>
            <groupId>net.gpedro.integrations.slack</groupId>
            <artifactId>slack-webhook</artifactId>
            <version>1.1.1</version>
        </dependency>
        <!-- End change plugin specific dependencies here -->
```

d.

getLabelgetDescriptioni18ni18n

"slack_webhook / src / main" "resources / message" "SlackWebhookTool.properties"

```
org.joget.SlackNotification.pluginLabel=Slack Notification
org.joget.SlackNotification.pluginDesc=Send notification message to Slack user when an assignment is available.
SlackNotification.config=Configure Slack Notification
SlackNotification.url=Webhook URL
SlackNotification.from=From
SlackNotification.fromUsername=Username
SlackNotification.fromUsername.value=Joget Workflow
SlackNotification.customIcon=Custom Icon
SlackNotification.customIcon.none=None
SlackNotification.customIcon.joget=Joget Workflow Logo
SlackNotification.customIcon.url=Image URL
SlackNotification.customIcon.emoji=Emoji Code
SlackNotification.to=To
SlackNotification.usernameTransform=Transform username to Slack username
SlackNotification.usernameTransform.desc=Hash Variable can be used to transform username to Slack username. Eg.
@#form.slack.username[{username}]#
SlackNotification.usernameTransform.value=@{username}
SlackNotification.message=Message
SlackNotification.text=Text
SlackNotification.text.desc=Refer to <a href="https://api.slack.com/docs/formatting" target="_blank">Slack
Message Formatting</a>.
SlackNotification.unfurl_links=Unfurling Links
SlackNotification.unfurl_links.desc=Automatically find URLs in a message and create attachments based on the
content of those URLs
SlackNotification.unfurl_media=Unfurling Media
SlackNotification.unfurl_media.desc=Automatically find Media URLs in a message and create attachments based on
the media of those URLs
SlackNotification.sendTestMessage=Send Test Message
SlackNotification.sendTestMessage.testChannel=Test Channel
SlackNotification.sendTestMessage.success=Test message sent.
SlackNotification.sendTestMessage.fail=Fail to sent test message. Error:
SlackNotification.testMessage=Test Message
SlackNotification.viewAssignment=View Assignment
```

e. Felix

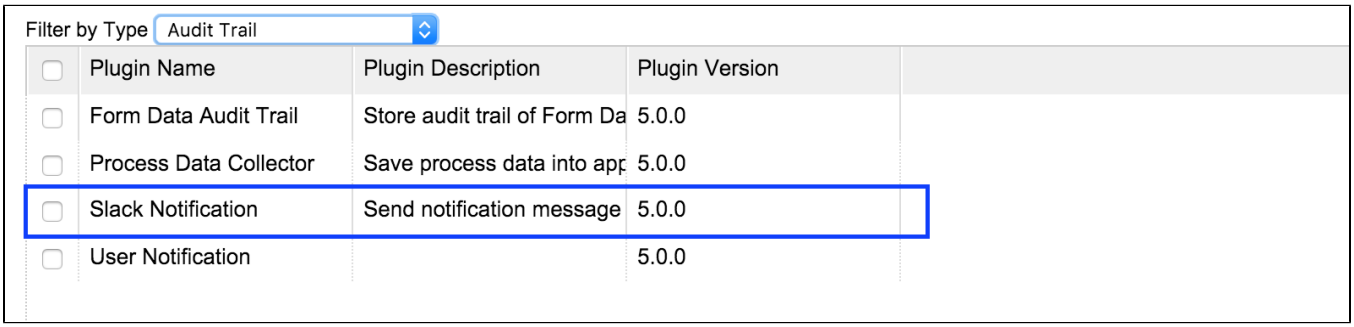
ActivatorFelix

```
    public void start(BundleContext context) {
        registrationList = new ArrayList<ServiceRegistration>();
        //Register plugin here
        registrationList.add(context.registerService(SlackNotification.class.getName(), new SlackNotification(),
null));
    }
```

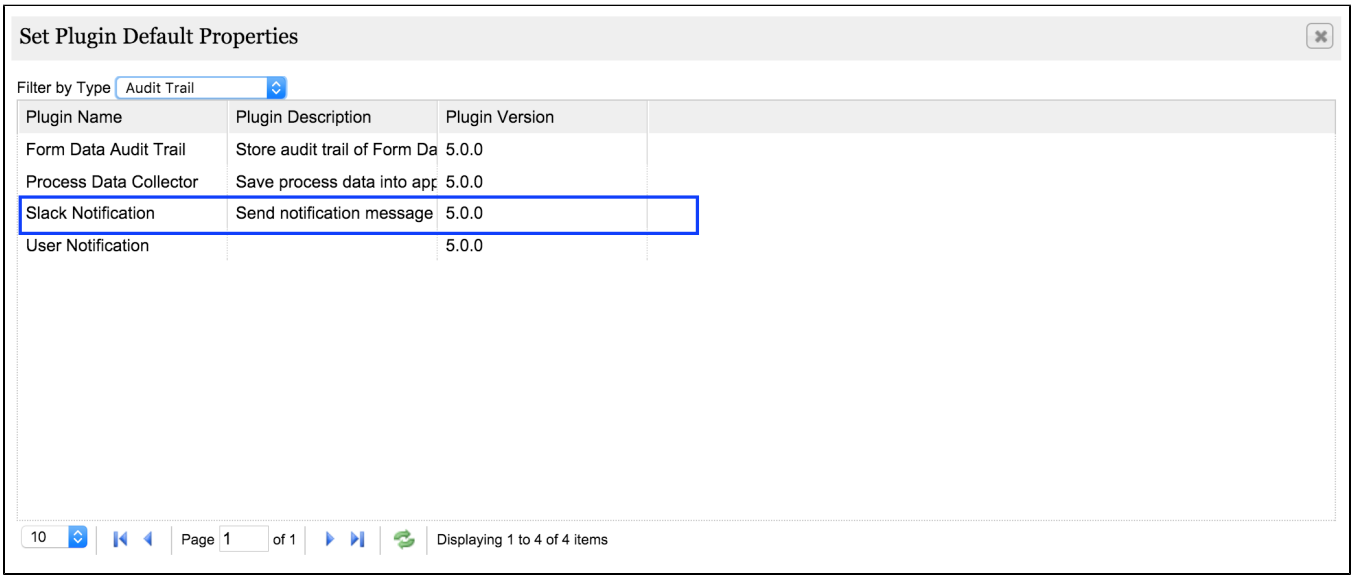
f.

“slack_notification / target”“slack_notification-5.0.0.jar”

jar jar



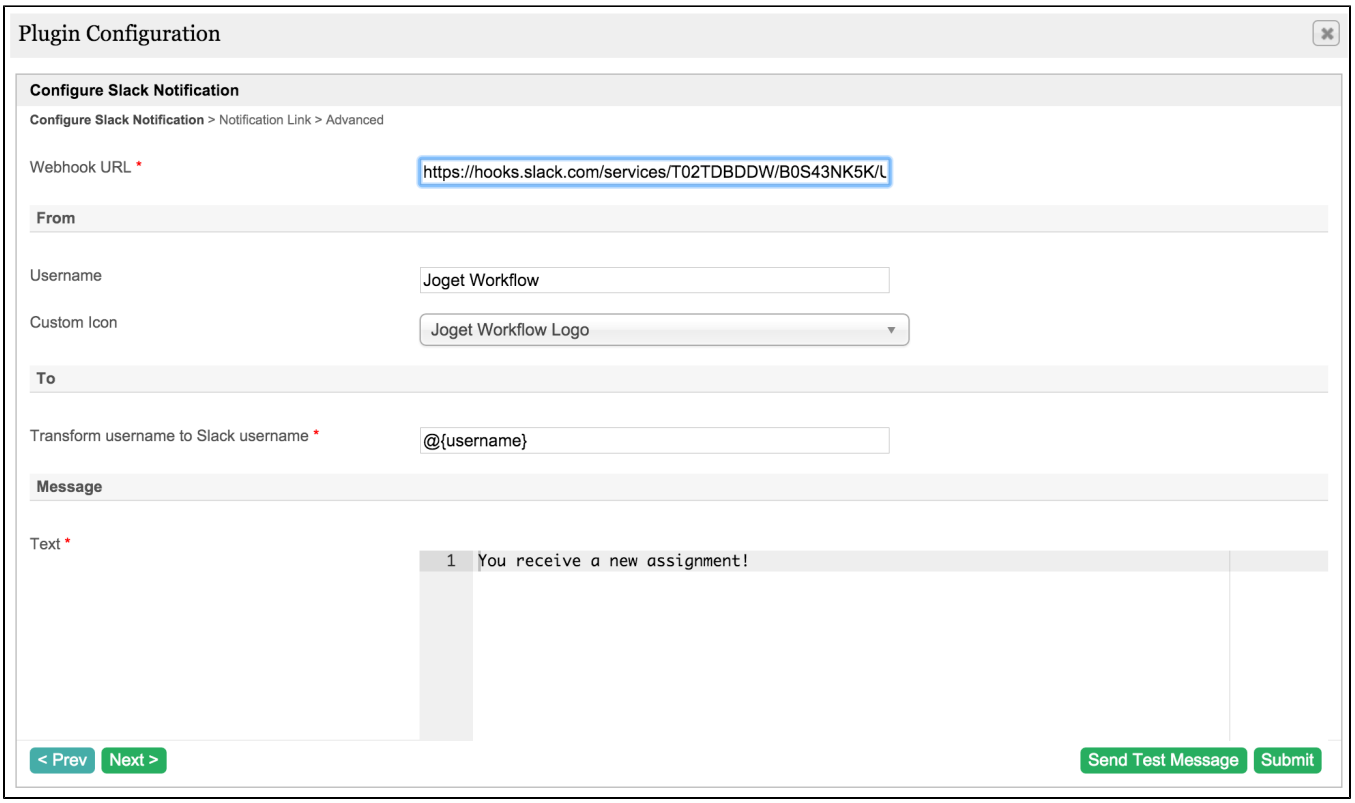
. Slack Notification

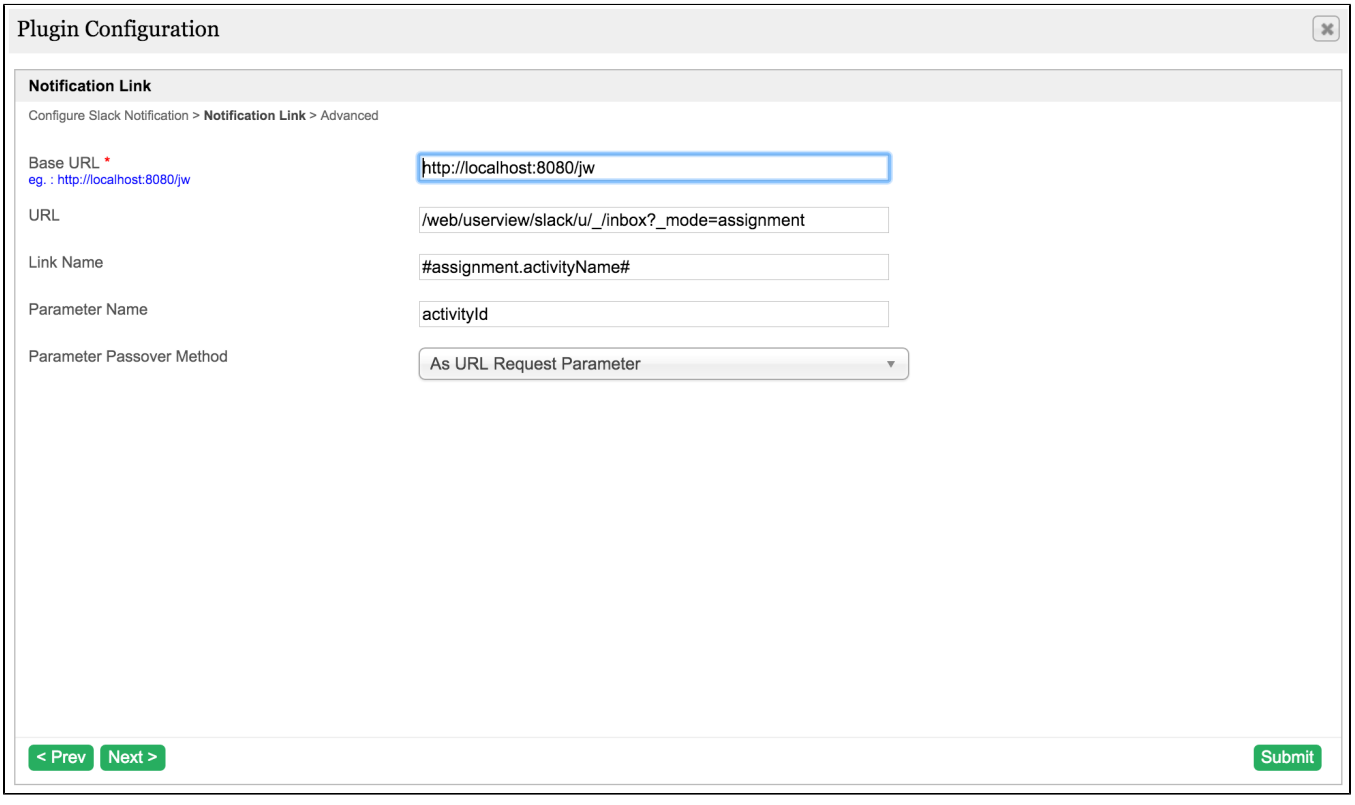


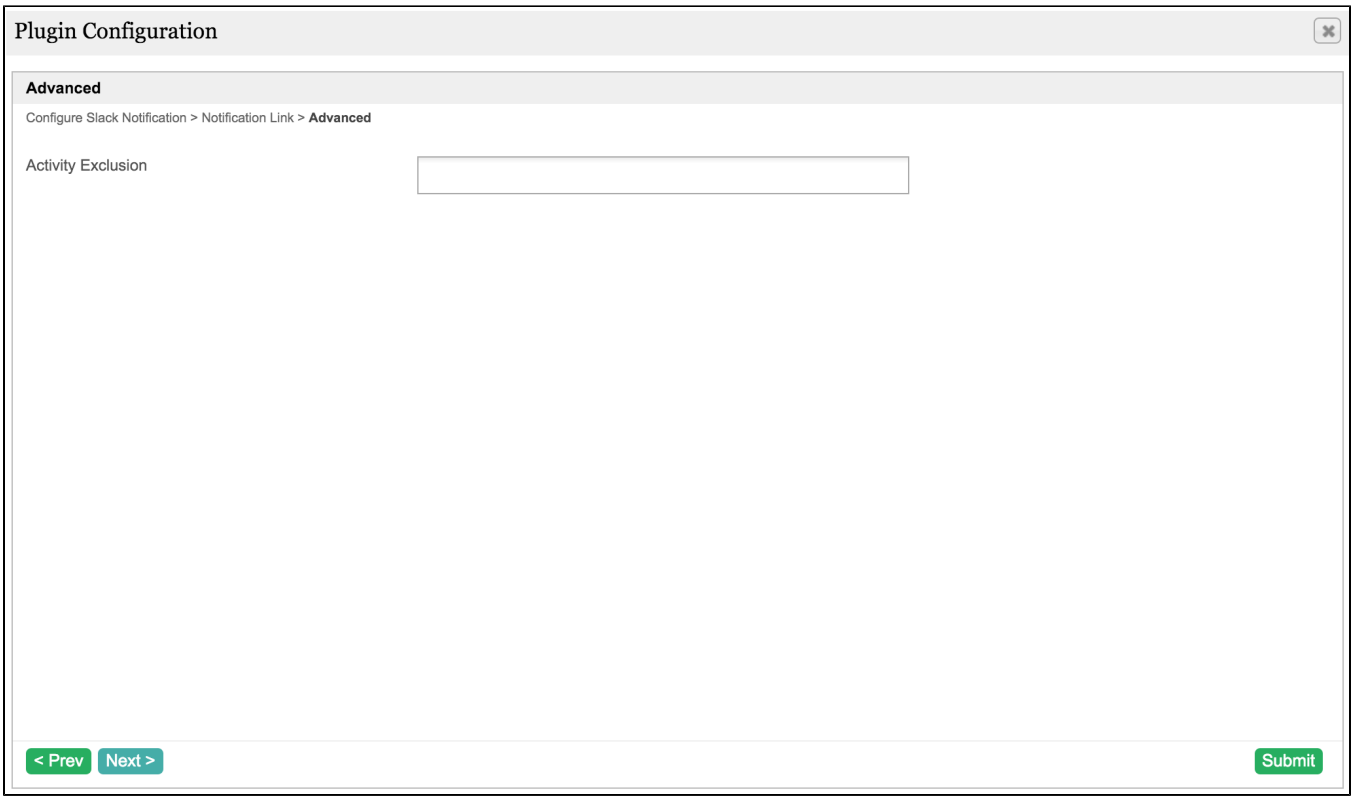
SlackIncoming Webhooks

1. *your_team.slack.com/services/new*
2. *WebHook* Add
3. Add Incoming WebHooks Integration
4. WebHook“ Webhook URL ”

Slack Notification


Plugin Configuration
Configure Slack Notification
Configure Slack Notification > Notification Link > Advanced
Webhook URL *
https://hooks.slack.com/services/T02TDBDDW/B0S43NK5K/L
From
Username
Joget Workflow
Custom Icon
Joget Workflow Logo
To
Transform username to Slack username *
@{username}
Message
Text *
1 You receive a new assignment!
< Prev
Next >
Send Test Message
Submit


Plugin Configuration
Notification Link
Configure Slack Notification > Notification Link > Advanced
Base URL *
eg. : http://localhost:8080/jw
http://localhost:8080/jw
URL
/web/userview/slack/u/_/inbox?_mode=assignment
Link Name
#assignment.activityName#
Parameter Name
activityId
Parameter Passover Method
As URL Request Parameter
< Prev
Next >
Submit



Slack

8.

slack_notification_src.zip.

jar http://marketplace.joget.org/. (Coming Soon)