PING MANUAL

Use it

August 2017

SOMMAIRE

CONTENT

1. Introduction

This document explains how to use the Ping Page.

Bonita portal has a huge function: it accepts to deploy some custom page. There are two way to build a custom page in Bonita

- Using the UI Designer
- Using the Groovy page architecture

With the UI Designer, you design your page using the standard and custom widget. You can call the Bonita Rest API, and you can call some custom REST API.

You face theses limitation with the UI Designer

- No control on the AngularJS controller. Then, when you create a JavaScript variable, the UI Designer page call this script every time: you add a character in the Input? your script is executed again. Same, if you have two different scripts, you don't know which script is called first.
- No access to the HTML. If you need a special control, then you are supposed to build a Custom Widget. Then, for simple usage like have a button to call a JavaScript, it's quite impossible to do that with the UI Designer
- Deployment issue. If you page need 3 or 4 custom RESTAPI, you must build them separately, and deploy it separately. You have no control to verify that the RESTAPI has the correct version for your page.

But the UI Designer is wonderful to build 80% of your custom page, because it's a WHYISWIG tool, and not required any development.

If your need is most complex, then the Bonita Portal offer you the second level: the groovy page architecture. The Ping page is a tutorial to explain you how to deploy and use it. It contains too some tools to help you to build some powerful page using Bonita Portal. To use this way, you should know:

- HTML to build the page
- AngularJS to be able to add special behavior on the page
- Java or Groovy for the server side

2. Principe

A Groovy Bonita page has different components:

- The HTML
- The AngularJS
- The Server Site (Index.groovy)

2.1. Html

The HTML part use some Angular world

```
<body ng-app="pingmonitor">
```

```
<h1><center>Ping the server</center></h1>


<div class="container-fluid" id="accordion">
  <div class="row">
    <div class="component col-xs-12"  ng-controller="PingControler as pingctrl">
  <button ng-click="pingctrl.ping()" ng-disable="pingctrl.inprogress" class="btn btn-info
btn-xs">Collect informations</button>
```

Then the Angular controller is load in the page

```
<script
src="pageResource?page=custompage_ping&location=pingmonitor.js&t=@_CURRENTTIMEMILIS_@"></scr
ipt>
```

Tip: include the " t" parameters implied that the URL will be different each time and bypass the Bonita Cache policy, which is very large. This is necessary for the development, not for the production (the JS part should not change).

2.2. Angular

The Angular JS part defines all the control for the page

```
(function() {


var appCommand = angular.module('pingmonitor', ['googlechart', 'ui.bootstrap','ngSanitize',
'ngModal', 'ngMaterial']);


// ----------------------------------------------------------------------
//
// Controler Ping
//
// ----------------------------------------------------------------------

// Ping the server
appCommand.controller('PingControler',
  function ( $http, $scope,$sce,$filter ) {

  this.pingdate='';
  this.pinginfo='';
  this.listevents='';
  this.inprogress=false;

  this.ping = function()
  {

    this.pinginfo="Hello";

    var self=this;
    self.inprogress=true;
    // 7.6 : the server force a cache on all URL, so to bypass the cache, then create a
different URL
    var d = new Date();

    $http.get( '?page=custompage_ping&action=ping&t='+d.getTime() )
       .success( function ( jsonResult ) {
```

```
        console.log("history",jsonResult);
        self.pingdate                = jsonResult.pingcurrentdate;
        self.pinginfo                = jsonResult.pingserverinfo;
        self.listprocesses    = jsonResult.listprocesses;
        self.listusers               = jsonResult.listusers;
        self.listevents              = jsonResult.listevents;

        $scope.chartObject                  = JSON.parse(jsonResult.chartObject);

        self.inprogress=false;


    })
    .error( function() {
     alert('an error occure');
     self.inprogress=false;
     });

}
```

When you need some information to the server, use the Angular module $http. Call the same page (custompage_ping) and set an action (here action = “ping”).

Note: the Custom page mechanism support only the GET protocole. In Tomcat V8, the URL parameters must be encoded. When you have to parse a large amount of data, you have to split the data in multiple URL, to send data packet per packet. On the server side, collect the packet, save them in the Tomcat Session, then reassemble the complete data.

2.3. Groovy

For convenient method, Groovy is split in two files:

Index.groovy is the file called by Bonita Server. This call is done to get all files (Javascript, HTML…) and what is important, is to manage only the URL which contains a special mark: “action”. Then, all URL contains a action is send to the Actions.groovy file

Actions.groovy : here, define your REST CALL function. This call is private to your page

```
  public static Index.ActionAnswer doAction(HttpServletRequest request, String paramJsonSt,
HttpServletResponse response, PageResourceProvider pageResourceProvider, PageContext
pageContext) {

        // logger.info("#### PingActions:Actions start");
        Index.ActionAnswer actionAnswer = new Index.ActionAnswer();
        List<BEvent> listEvents=new ArrayList<BEvent>();
        Object jsonParam = (paramJsonSt==null ? null : JSONValue.parse(paramJsonSt));

        try {
            String action=request.getParameter("action");
            logger.info("#### log:Actions  action is["+action+"] !");
            if (action==null || action.length()==0 )
            {
                actionAnswer.isManaged=false;
                logger.info("#### log:Actions END No Actions");
                return actionAnswer;
            }
            actionAnswer.isManaged=true;

            APISession apiSession = pageContext.getApiSession();
            HttpSession httpSession = request.getSession();
            ProcessAPI processAPI = TenantAPIAccessor.getProcessAPI(apiSession);
            IdentityAPI identityAPI = TenantAPIAccessor.getIdentityAPI(apiSession);

            long tenantId = apiSession.getTenantId();
```

```
            TenantServiceAccessor tenantServiceAccessor =
TenantServiceSingleton.getInstance(tenantId);



    if ("ping".equals(action))
    {
      SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
```

You have to return a JSON information.

3. How to use the Ping Skeleton?

TO use the Ping Skeleton to create your own page, do the different step:

- Copy the page from Github https://github.com/Bonitasoft-Community/page_ping
- Change the name in the pom.xml

```
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.bonitasoft.custompage</groupId>
  <artifactId>Mypage</artifactId>
  <version>1.0.0</version>
```

- Rename the page in pages.properties

```
#The name must start with 'custompage_'
#Thu Dec 13 09:54:02 PST 2018
version=1.0
displayName=MyPage
name=custompage_mypage
description=Report and Search on BDM and case
```

- In the index.html, rename all the “page=custompage_ping” by “page=custompage_mypage”
- Search in the different file under resource, and rename all “custompage_ping” par “custompage_mypage”

Recompile the page via a mvn install. The maven:

- Rebuild all the JAVA file if you have in the folder src
- Download a install any librairy you need, and place them in the folder lib
- Create the page as a resource in target
- Deploy the page in your portal, in the profile BOTools (profile is created if it is not present). Deployment is done one Bonita portal, port 8080. You can adapt the port number in the pom.xml

4. How to debug a page?

The Studio launch a Tomcat server, with the Bonita application. It’s possible to run the Debugger.

4.4. Setup the debug

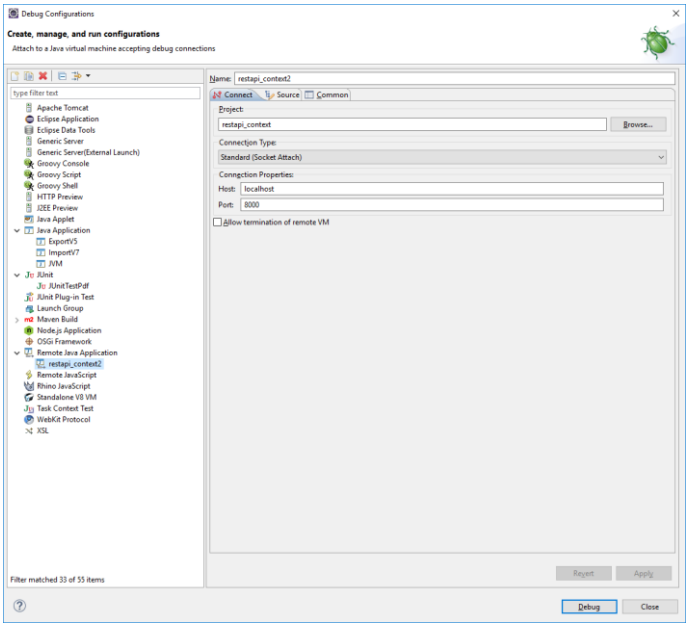
To allow the Tomcat attached to the Studio to accept any external debug call, edit the file “BonitaStudioSubscription64.ini” (depend on your platform) and add at the end the line

```
-Dtomcat.extra.params="-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8000 -
DnoCacheCustomPage=true"
```

Restart the studio.

4.5. Start a debug session

In eclipse, create a “Remote Java Application” configuration, and set up the host to “localhost” and the port to 8000.



Start the session: you can now place a breakpoint in Groovy or in Java, when you use it in the Portal, you will have a debug session

5. Events

Ping use the Bonita Event library (https://github.com/Bonitasoft-Community/BonitaEvent)

Goal of this library is:

- To describe any main information, any error, as an event. Each information got then a unique identifiant
- If the event is an error, then describe the title, description, consequence, action to solve it

```
  private static EVENT_USERS_FOUND = new BEvent("org.bonitasoft.custompage.ping", 1,
Level.INFO, "Number of users found in the system");
    private static EVENT_FAKE_ERROR  = new BEvent("com.bonitasoft.ping", 1,
Level.APPLICATIONERROR, "Fake error", "This is not a real error", "No consequence", "don't
call anybody"));
```

Send event (or list of events) on the page, in HTML

```
actionAnswer.responseMap.put("listevents",BEventFactory.getHtml( listEvents));
```

6. Properties

Ping use the Properties Event library (https://github.com/Bonitasoft-Community/BonitaProperties)

You can save and retrieve parameters via this library. BonitaProperties save information in the Bonita Engine database, in a new table. Information are then saved when the Bonita Engine database is backup.

This library manages group information in domain. The page Resource is used to select a specific domain, and then ensure than two pages does not use same properties set.

```
BonitaProperties bonitaProperties = new BonitaProperties( pageResourceProvider );
listEvents.addAll( bonitaProperties.load() );
bonitaProperties.setProperty( apiSession.getUserId()+"_firstname",
jsonParam.get("firstname") );

listEvents.addAll(  bonitaProperties.store());
```

7. Chart

8. Modal (Dialog)

9. Generate a Exel file in local

10. Download a file

Html :

On the page, place a link:

```
<a class="btn btn-info" href="?page=custompage_ping&action=downloadPdf"
target="blanck">Download a PDF</a>
```

Nota: using Bootstrat, the link look like a button

AngularJS :

Nothing

Groovy :

On the action “download”, you must send a document

```
} else if ("downloadPdf".equals(action))
    {
     logger.info("Download the PDF");

     // get the document which is saved in the Custom Page directory
     InputStream input = pageResourceProvider.getResourceAsStream("doc/Ping Manual.pdf");
     // we get the outpuStream in order to send the document as it
     OutputStream output = response.getOutputStream();
     byte[] buffer = new byte[1024];
        int bytesRead;
        while ((bytesRead = input.read(buffer)) != -1)
        {
         output.write( buffer, 0, bytesRead);
```

```
        }
        output.flush();
        output.close();
        // then add the name and the correct content type
                response.addHeader("content-disposition", "attachment; filename=\"Ping
Manual.pdf\"");
                response.addHeader("content-type", "application/pdf");
                return;
    }
```

Nota : don't call before the method

```
PrintWriter out = response.getWriter()
```

Because else Outstream is lock by the PrintWriter and then the method response.getOutputStream() will failed.

11. Drag and drop and upload a file

The idea is to have a HTML zone where user can drip a file. Then the page will ommediately send the page to the server,and you can open it

Html :

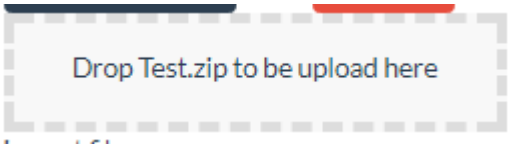
In the Header, add

```
<link rel="stylesheet"
href="pageResource?page=custompage_meteor&location=style/dropbox.css">
```

Place in your HTML

```
<div ng-file-drop ng-model="importfiles" class="drop-box ng-isolate-scope ng-valid ng-dirty"
 drag-over-class="{accept:'dragover', reject:'dragover-err', delay:100}"
 multiple="true" allow-dir="false">Drop Test.zip to be upload here</div>
<div ng-no-file-drop class="drop-box" style="display: none;">File Drop is not supported for
this browser</div>
```

With the result



And add in the bottom

```
 <script src="pageResource?page=custompage_ping&location=js/angular-file-upload-
shim.min.js"></script>
 <script src="pageResource?page=custompage_ping&location=js/angular-file-
upload.min.js"></script>
```

The different files (angular-file-upload-shim.js, angular-file-upload.min.js, dropbox.css has to be place.

AngularJS :

Add the angularFIleUpload in the angular.module

```
var appCommand = angular.module('pingmonitor', ['googlechart', 'ui.bootstrap',
'ngSanitize','ngModal','angularFileUpload']);
```

Add in the function the module $upload

```
appCommand.controller('PingControler',
      function ( $http, $scope,  $sce, $interval, $timeout, $upload ) {
```

and watch the drop down area

```
      var me = this;
            $scope.$watch('importfiles', function() {
```

```
                    if ($scope.importfiles)
                    {
                          for (var i = 0; i < $scope.importfiles.length; i++) {
                                var file = $scope.importfiles[i];
                                // V6 : url is fileUpload
                                // V7 : /bonita/portal/fileUpload
                                $scope.upload = $upload.upload({
                                      url: '/bonita/portal/fileUpload',
                                      method: 'POST',
                                      data: {myObj: $scope.myModelObj},
                                      file: file
                                }).progress(function(evt) {
      //                                console.log('progress: ' + parseInt(100.0
* evt.loaded / evt.total) + '% file :'+ evt.config.file.name);
                                }).success(function(data, status, headers,
config) {
                                      console.log('file ' + config.file.name +
'is uploaded successfully. Response: ' + data);
                                      me.fileIsDropped(data);
                                });
                          }
                    } // end $scope.importfiles
              });
```

Write the method “fileIsDropped”:

```
this.fileIsDropped = function( testfileimported ) {
              self.configwait=true;
              $http.get(
'?page=custompage_meteor&action=import&filename='+testfileimported )
              .success( function ( jsonResult ) {
                    self.config.list                = jsonResult.configList;
                    self.listeventsconfig           =
jsonResult.listeventsconfig;
                    self.configwait=false;
              });
```

Groovy :

On the action you have the file name, let get the file itself.

The point is this location change on the different BonitaVersion, so you have to explore different path.

HEADQUARTERS

PARIS, FRANCE
76 boulevard de la République
92100 Boulogne-Billancourt

EMEA, ASIA & LATIN AMERICA

GRENOBLE, FRANCE
32, rue Gustave Eiffel
38000 Grenoble

NORTH AMERICA

SAN FRANCISCO, USA
44 Tehama Street
San Francisco, CA 94105

NEW YORK, USA
33 Nassau Avenue
Brooklyn NY 11222

Bonitasoft

www.bonitasoft.com