



**Joget DX**

# Integrating With External System

 <http://facebook.com/jogetworkflow>

 <http://twitter.com/jogetworkflow>

# Prerequisites

---

1. Understand Joget components in depth and able to make full use of its components.
2. Avid software developer with vast know-hows of web application technologies.

# Content

---

1. Introduction
2. JSON API
3. JSON API Authentication
4. JavaScript API
5. Single Sign On (SSO)
6. Embedding Task Inbox into External System
7. Embedding Userview Page in an iFrame
8. JSON Tool
9. SOAP Tool
10. Integrating with External Form
11. Using API Builder



# Chapter 1

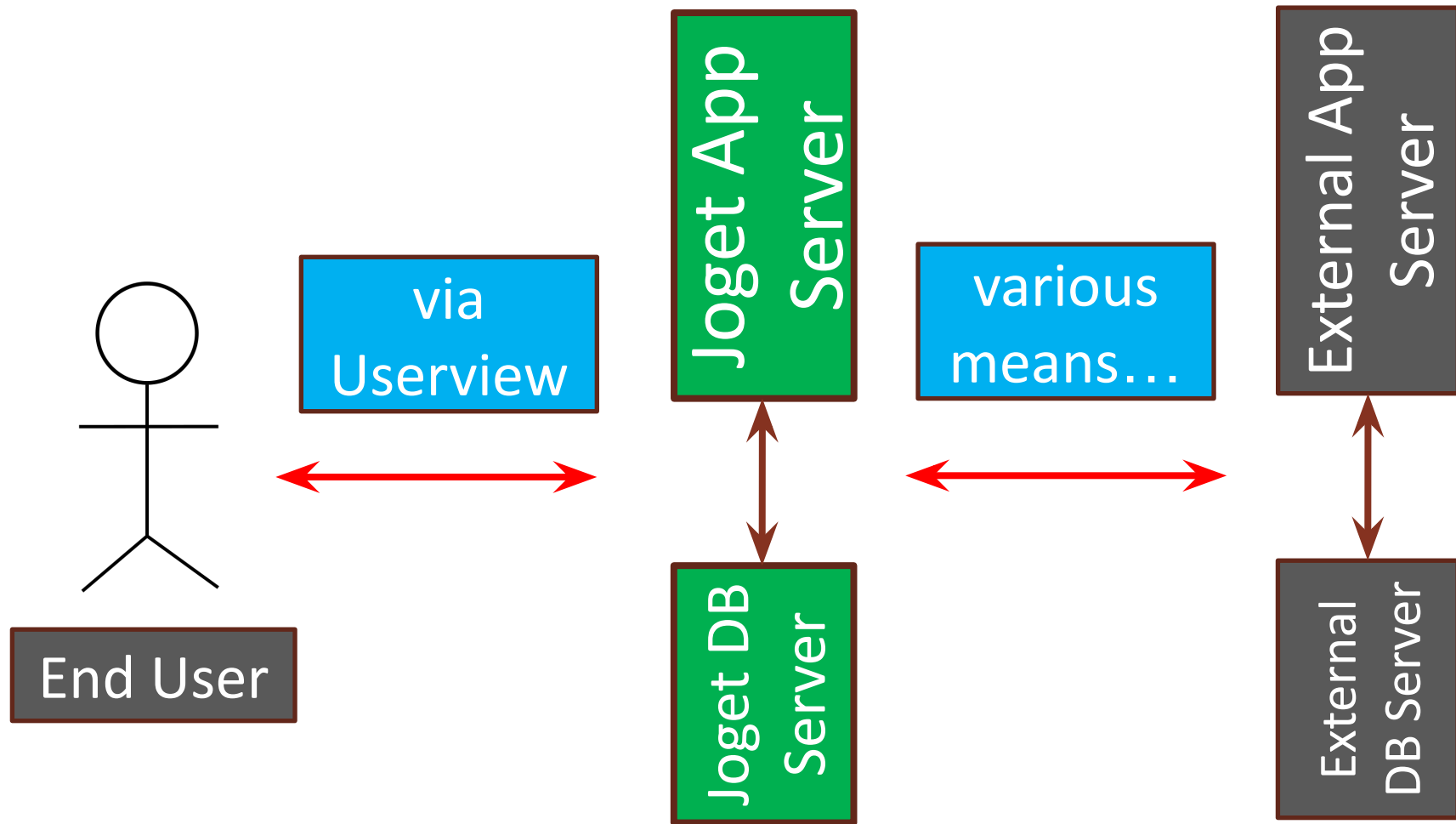
## Introduction

# Introduction

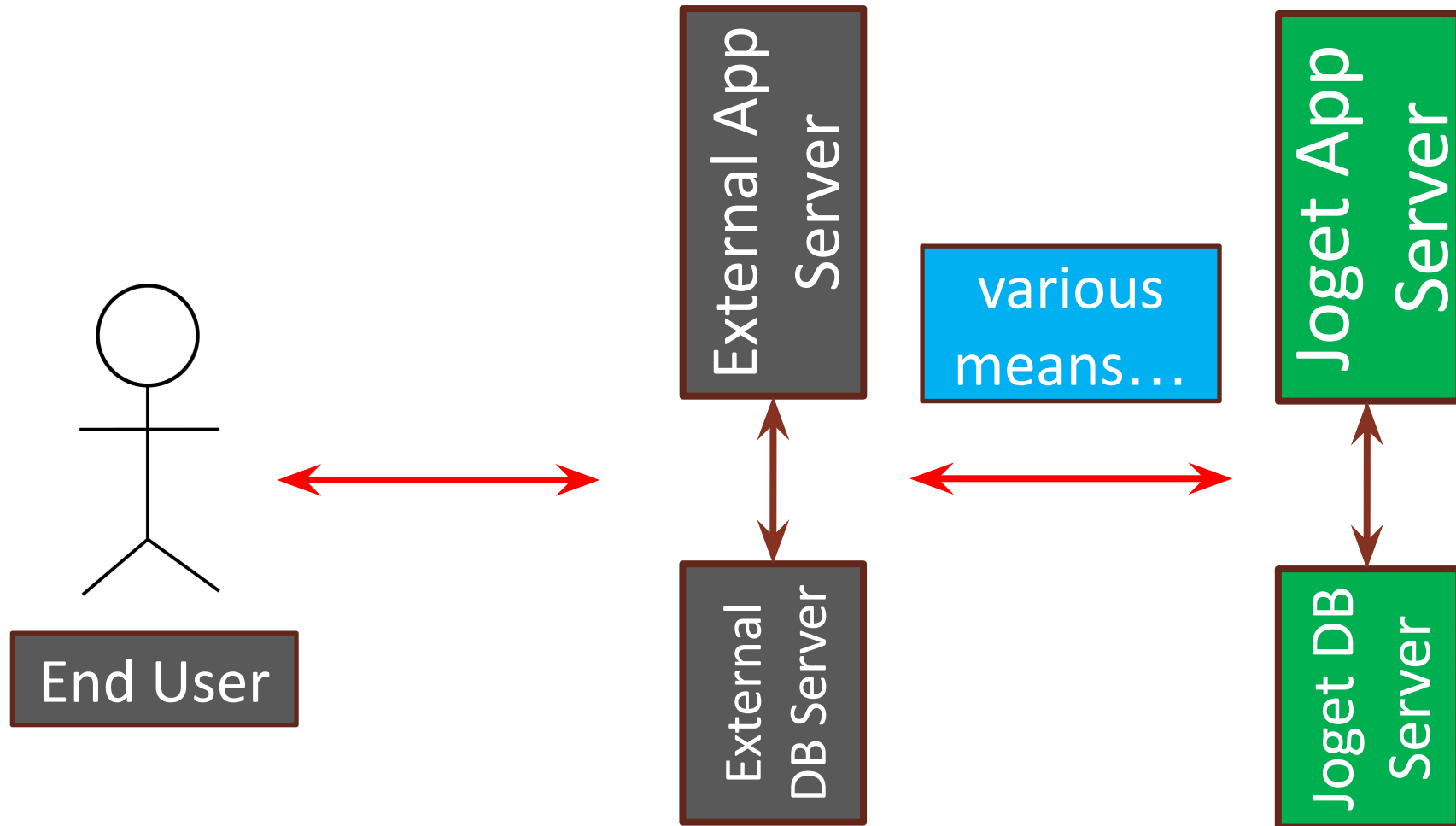
---

- There are various ways to integrate Joget with external applications.
- Typically, there are 4 main components and the client user to be considered about.
  1. Joget Application Server
  2. Joget Database Server
  3. External Application Server
  4. External Application Database Server
  5. Client user

# Joget as the Front-end

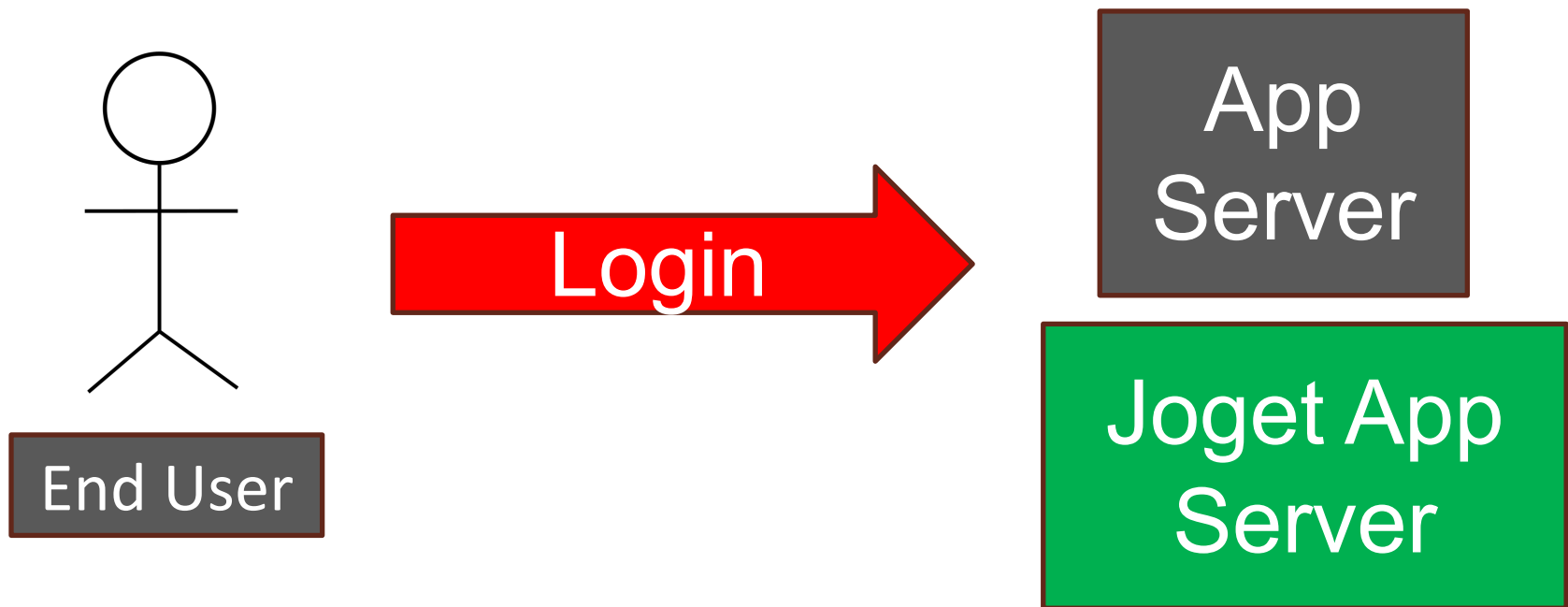


# External App as the Front-end



# User Directory

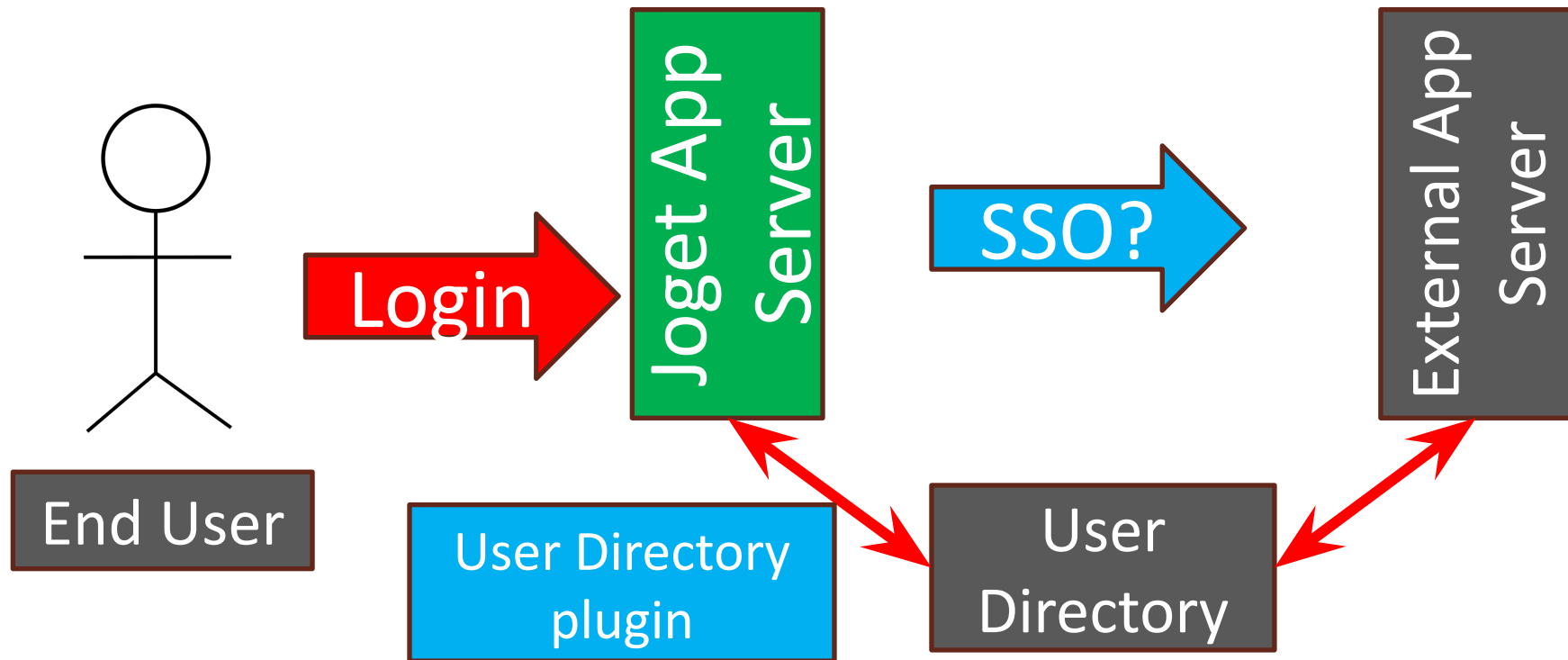
- Is User session important?
- How do you want to handle the user session between the servers (Joget server and external application server)?





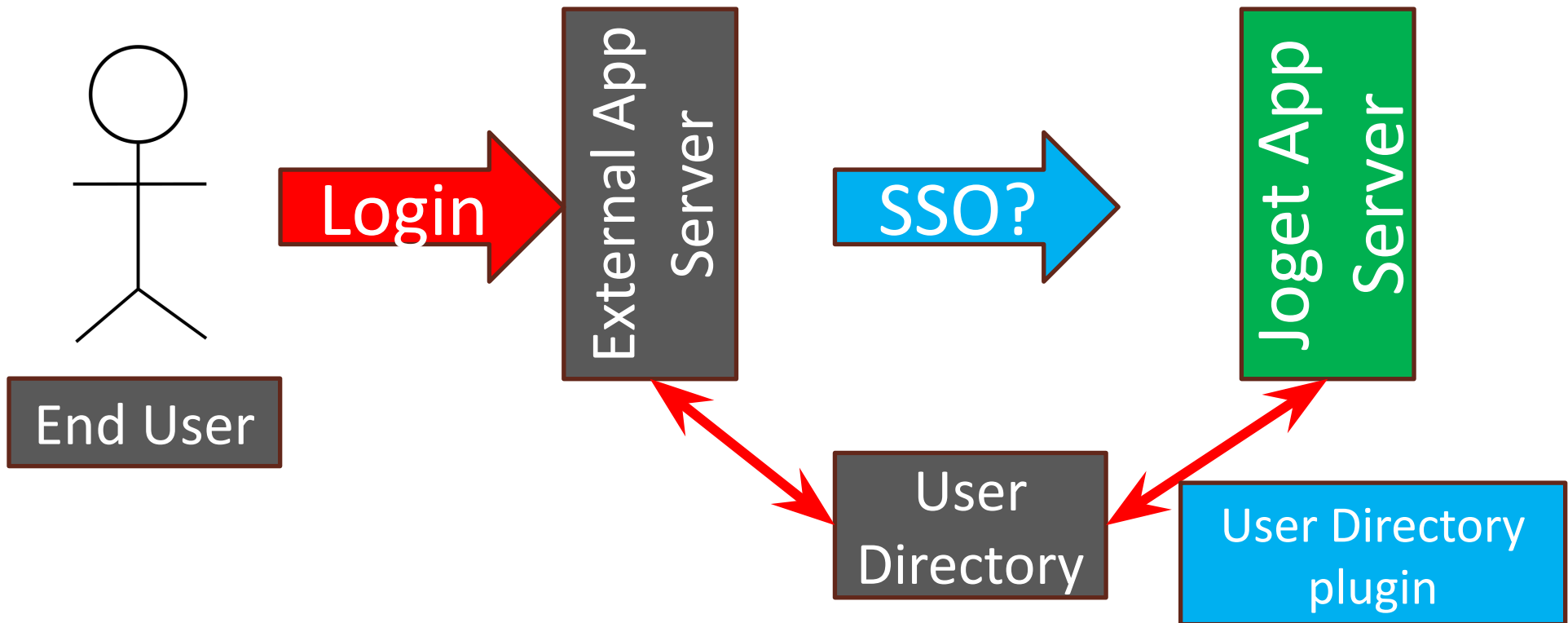
# User Directory – Joget as Front-end

- Is the identity of the End User important to the External App Server?



# User Directory – Ext App as Front-end

- Is the identity of the End User important to the Joget App Server?



# Chapter Review

---

- Understand various ways of integration with Joget and external application server.

---

# Chapter 2

## JSON API

# JSON API

---

- Joget provides a comprehensive list of APIs for Process related tasks.
- Full list of the APIs can be obtained from <https://dev.joget.org/community/display/DX7/JSON+API>


# Sample List of APIs

---

- [web/json/monitoring/running/process/list](#)
- [web/json/workflow/currentUsername](#)
- [web/json/workflow/assignment/completeWithVariable/\(:activityId\)](#)
- [web/json/workflow/assignment/complete/\(:activityId\)](#)

# Before We Start!

- Ensure you have configured the **API Domain Whitelist** in **General Settings** to allow JSON API requests.
- If a request is from a non-whitelisted domain, the response will be a HTTP 400 Bad Request



The screenshot shows the 'JOGET DX ENTERPRISE' interface. On the left is a navigation menu with the following items: System Settings, General Settings (highlighted), Datasource & Profile Settings, Directory Manager Settings, Manage Plugins, and Manage Messages. The main content area is titled 'General Settings' and contains three input fields: 'API Domain Whitelist (Separated by ';')' (highlighted with a red box), 'API IP Whitelist (Separated by ';')', and 'Glowroot API URL'. Below the 'Glowroot API URL' field, the text 'default: http://localhost:4000' is displayed.

# web/json/monitoring/running/process/list

- **URL:** /web/json/monitoring/running/process/list
- **Method:** HTTP GET/POST
- **Description:** Retrieve running process list
- **Parameters**
  - packageId - (Optional) package id
  - processId - (Optional) process definition id without version
  - processName - (Optional) process name
  - version - (Optional) process version
  - sort - (Optional) column name to be sorted
  - desc - (Optional) Boolean value to determine whether to sort by ascending or descending order (true equals to descending)
  - start - (Optional) where rows start from
  - rows - (Optional) number of rows per page

## Sample Result

```
{
  "total":2,
  "desc":false,
  "sort":"name",
  "start":0,
  "data":
  [
    {"id":"3724_mdec_v1002_mdec_wp1",
      "serviceLevelMonitor":"<span
class=\"dot_red\"></span>",
      "name":"mdec_wp1","state":"open.running",
      "due":"Fri Mar 20 14:01:27 SGT 2009",
      "startedTime":"Fri Mar 20 13:51:27 SGT
2009","version":"2"},
    {"id":"3725_mdec_v1002_mdec_wp1",
      "serviceLevelMonitor":"<span
class=\"dot_red\"></span>",
      "name":"mdec_wp1",
      "state":"open.running",
      "due":"Fri Mar 20 14:03:16 SGT 2009",
      "startedTime":"Fri Mar 20 13:53:16 SGT 2009",
      "version":"2"}
  ]
}
```



# web/json/workflow/currentUsername

---

- **URL**

/web/json/workflow/currentUsername

**Sample Result**

```
{ "username": "admin" }
```

- **Method**

HTTP GET/POST

- **Description**

Get current logged in user's username

- **Parameters**

- callback - (Optional) a function (in JavaScript) to call back after invoking this method

## `/web/json/workflow/assignment/completeWithVariable/(:activityId)`

- **URL**  
`/web/json/workflow/assignment/completeWithVariable/(:activityId)`
- **Method** HTTP POST
- **Description**  
Set activity variable  
Variables can be passed as parameters with the **var\_** prefix
- **Parameters**
  - **callback** - (Optional) a function (in JavaScript) to call back after invoking this method
  - **activityId** - activity id
  - **var\_(workflow variable id)** - (Optional) set workflow variable value

**/web/json/workflow/assignment/completeWithVariable/(:activityId)**

---

### Sample Result

```
{  
  "activityId": "1079_563_crm_process1_approve_proposal",  "assignment":  
  "admin",  
  "nextActivityId": "1093_563_crm_process1_send_proposal",  
  "processId": "563_crm_process1",  
  "status": "completed"  
}
```

# Chapter Review

---

- Understand and able to retrieve the list of available APIs.



# Chapter 3

## JSON API Authentication

# Using JSON API

- Can be called using AJAX call (front-end/web) or through server-backend to retrieve data from or to perform action to Joget System.
- Return response in JSON format.
- Example:

web/json/workflow/assignment/list/count

{"total":11}

# Passing Parameters

{Host}/jw/web/json/workflow/assignment/list/count

This JSON API is accessed as  
**anonymous**

{Host}/jw/web/json/workflow/assignment/list/count?**j\_username=admin**  
&**j\_password=admin**

{Host}/jw/web/json/workflow/assignment/list/count?**j\_username=admin**  
&**hash=14ACD782DCFEB2BCDE2B271CCD559477**

By using **j\_username** and **j\_password** or **hash** parameter.  
This JSON API is accessed as **admin**

# Password Hashing

---

- Used in JSON API authentication and JavaScript Single Sign ON (SSO).
- Prevents a user's password from being directly exposed during authentication.
- **Formula**
  - `md5(username + "::" + md5Base16(password))`
  - E.g. Assuming that the username is “admin” and the password is “admin”, the resulting hash should be **“14ACD782DCFEB2BCDE2B271CCD559477”**.
- Online reference:  
<http://dev.joget.org/community/display/DX7/Hashed+Password>



# Password Hashing Sample Java Code

```
public static String md5(String content) {
    try {
        MessageDigest m = MessageDigest.getInstance("MD5");
        byte[] data = content.getBytes();
        m.update(data, 0, data.length);
        BigInteger i = new BigInteger(1, m.digest());
        return String.format("%1$032X", i);
    } catch (Exception ex) {}
    return "";
}

public static String md5Base16(String content) {
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] bytes = md.digest(content.getBytes());
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < bytes.length; i++) {
            byte b = bytes[i];
            String hex = Integer.toHexString((int) 0x00FF & b);
            if (hex.length() == 1) {
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString();
    } catch (Exception e) {}
    return "";
}
```

# Master Login

---

- Master Login Username and Password is a set of credential that can be used to login to Joget system as different user.
- This is particularly catered to integration needs:-
  - Server-backend to Joget server where individual's credential is not required to perform calls on behalf.
- Reference:  
<http://dev.joget.org/community/display/DX7/JSON+API+Authentication>

# JSON API Authentication – Master Login

## System Administration Settings

Master Login Username

master

Master Login Password

.....

master login hash:0b244463ce522dc21bc58b42d13abcf0

In System Setting >  
General Setting

{Host}/jw/web/json/workflow/assignment/list/count?j\_username={master-login-username}&j\_password={master-login-password}&loginAs={username}

{Host}/jw/web/json/workflow/assignment/list/count?j\_username={master-login-username}&hash={master-login-hash}&loginAs={username}

By using **Master Login** feature, user's password is not exposed in JSON API authentication.

# Basic Http Authentication

---

- A set of user credential encoded to login to Joget system.
- Credentials are passed in the request header.
- Username and password not exposed in clear text
- **Formula:**
  - "Basic" + base64(username + ":" + password)
  - E.g. Assuming that the username is "admin" and the password is "admin", the result should be "**Basic YWRtaW46YWRtaW4=**".
- Reference:  
<https://dev.joget.org/community/display/DX7/JSON+API+Authentication#JSONAPIAuthentication-BasicHttpAuthentication>

# Caution

---

- Do **NOT** expose clear text password in the DOM
- Ensure that "API Domain Whitelist" & "API IP Whitelist" is configured according to principle of least privilege

# Chapter Review

---

- Understand how to authenticate with Joget's JSON APIs.

---

# Chapter 4

## JavaScript API

# Introduction

- Javascript API is a helper/utility with a set of methods to ease integration calls

```
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-1.9.1.min.js"></s
cript>
<script type="text/javascript"
src="http://localhost:8080/jw/js/json/util.js" ></script>

<script type="text/javascript" >
[put your code here]
</script>
```

- util.js source code: `\wflow-consoleweb\src\main\webapp\js\json\util.js`  
(GitHub link: <https://github.com/jogetworkflow/jw-community/blob/7.0-SNAPSHOT/wflow-consoleweb/src/main/webapp/js/json/util.js> )
- Reference: <http://dev.joget.org/community/display/DX7/JavaScript+API>



# List of Available Methods

---

- [ConnectionManager.post\(url, callback, params\)](#)
- [ConnectionManager.ajaxJsonp\(url, callback, params\)](#)
- [ConnectionManager.get\(url, callback, params, xss\)](#)
- [AssignmentManager.getCurrentUser\(baseUrl, callback\)](#)
- [AssignmentManager.login\(baseUrl, username, password, callback\)](#)
- [AssignmentManager.loginWithHash\(baseUrl, username, hash, callback\)](#)
- [AssignmentManager.logout\(baseUrl\)](#)
- [AssignmentManager.completeAssignment\(baseUrl, activityId, redirect\)](#)
- [AssignmentManager.completeAssignmentWithVariable\(baseUrl, activityId, variableData, redirect\)](#)
- [UrlUtil.encodeUrlParam\(url\)](#)
- [getUrlParam\(paramName\)](#)

And many more...

# Sample usage – Start New Process Instance

- Starting a new process by calling the JSON API

Code:

```
url =
"http://localhost:8080/jv/web/json/workflow/process/start/purchaseRequest:latest:purchaseRequestProcess";

callback = {
  success: function(response){
    console.log(response);
  }
};

params = {var_status : 'test' };
ConnectionManager.post(url, callback, params);
```

Replace accordingly

JSON API to start a new process instance  
 Reference:  
<http://dev.joget.org/community/display/DX7/JSON+API>

Result:

```
[processId=6_purchaseRequest_purchaseRequestProcess,
processDefId=purchaseRequest#1#purchaseRequestProcess,
participantId=applicant, next user=[admin]]
```

- Sample coding can be obtained at 18.4.1.txt

# Sample Usage – Get Current User

- Finding out the current logged in user by using the `AssignmentManager.getCurrentUser()` method.

## Code:

```
var callback = {
    success : function(response) {
        console.log(response.username);
    }
}
AssignmentManager.getCurrentUser('http://localhost:8080/jw', callback);
```

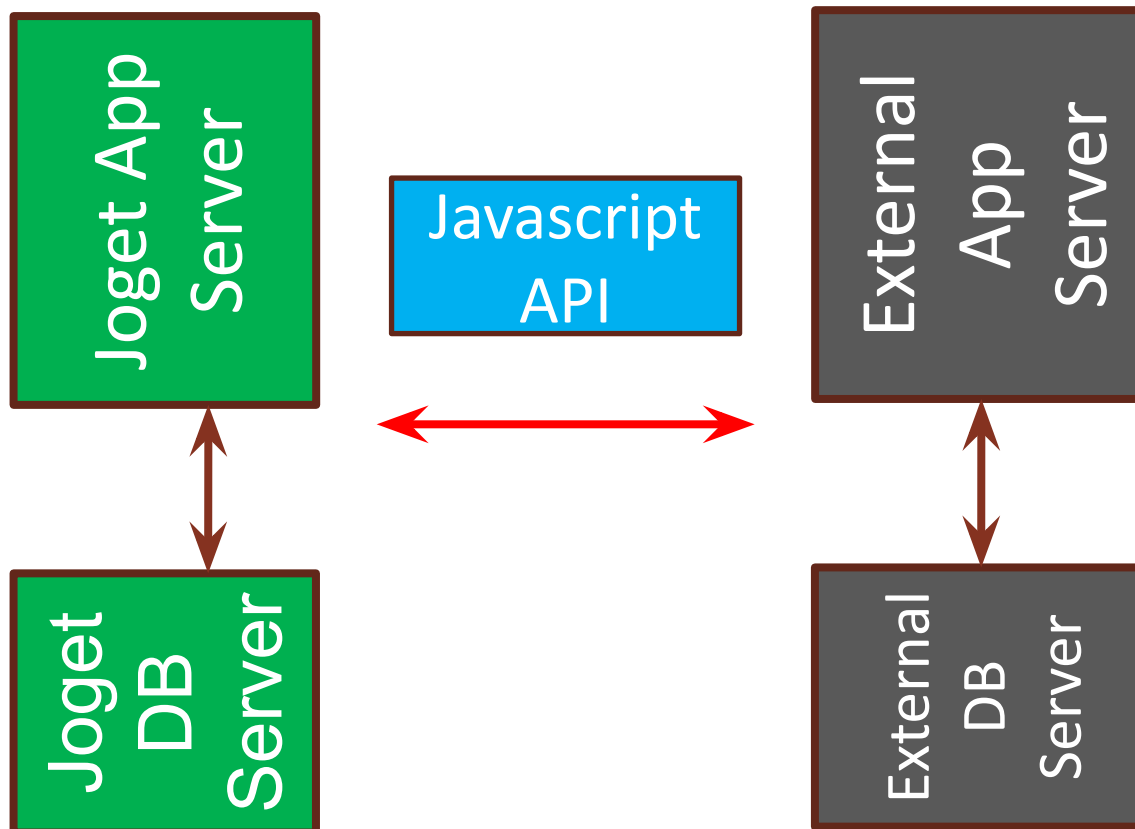
## Result:

```
admin
```

- Sample coding can be obtained at [18.4.2.txt](#)

# Setup Suggestion

- Can you now imagine the following setup?



# Chapter Review

---

- Understands the purpose and usages of the Javascript API.



# Chapter 5

## Single Sign On (SSO)

# Introduction

---

- User logs in to external system and implicitly gains access to Joget without being prompted to login again.
- Can be achieved via **Directory Manager plugins** and programmatically via Web Service plugin

Reference:

<https://dev.joget.org/community/display/DX7/Single+Sign+On+-+SSO>

# Various SSO Methods Available...

---

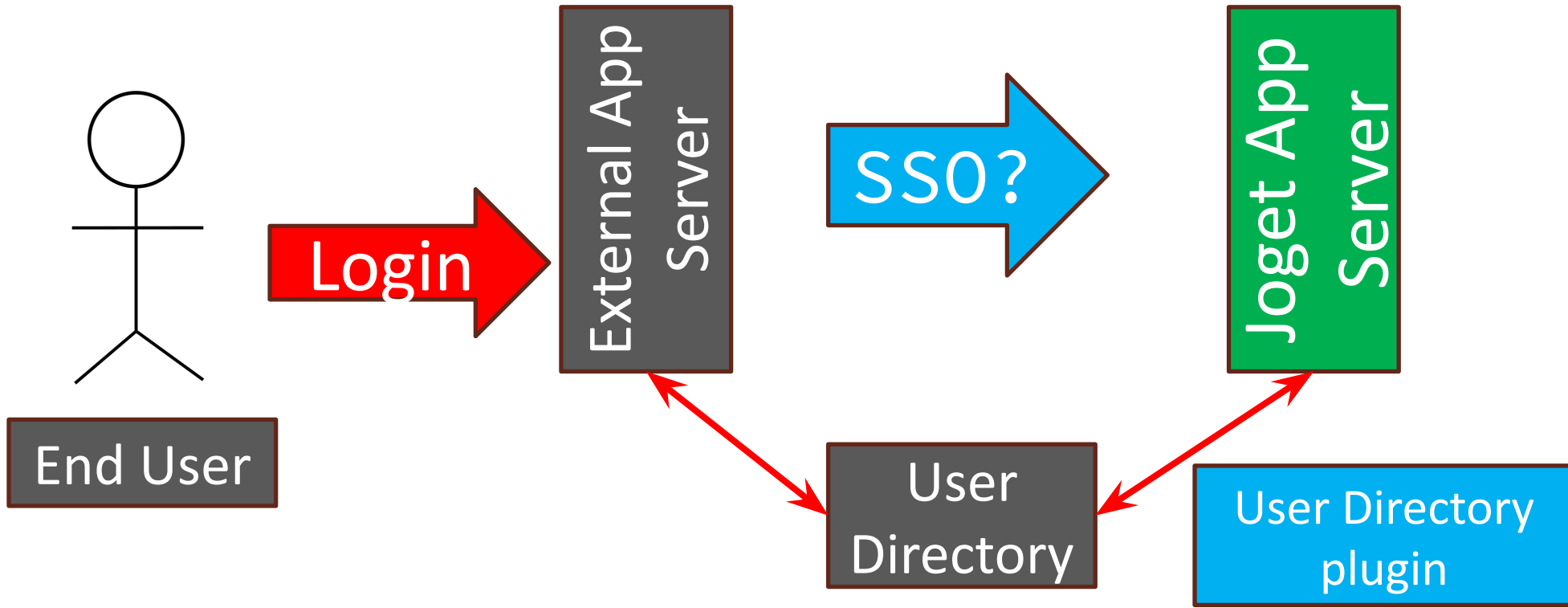
- G Suite
- Kerberos
- SAML (IDP initiated) (e.g.: Sharepoint, some ADs, etc.)
- OpenID Connect
- Programmatically via Web Service Plugin (Java)
- API Builder - SSO

See KB for detailed guides for all the above.



# Setup Suggestion

- Can you now imagine the following setup?



# Chapter Review

---

- Understand on how to use the SSO feature.


---

# Chapter 6

## Embedding Task Inbox into External System

# Embedding Task Inbox into External System

- Display Joget **task inbox** in other web applications, such as portal (SharePoint, Liferay) and content management system (Joomla!, WordPress, Drupal, Alfresco).

```
Embed Code <> RSS   
<link rel="stylesheet" type="text/css" href="http://localhost:8080/jw/css/portlet.css">  
<script type="text/javascript" src="http://localhost:8080/jw/js/jquery/jquery-1.9.1.min.js"></script>  
<script type="text/javascript" src="http://localhost:8080/jw/js/jquery/jquery-migrate-1.2.1.min.js"></script>  
<script type="text/javascript" src="http://localhost:8080/jw/js/json/util.js"></script>  
<div id="inbox1">  
  <center>  
  </center>  
</div>  
<script type="text/javascript">
```

# Embedding Task Inbox into External System

```

<link rel="stylesheet" type="text/css" href="http://localhost:8080/jw/css/portlet.css">
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-migrate-1.2.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/jw/js/json/util.js"></script>
<div id="inbox1">
  <center>
  </center>
</div>
<script type="text/javascript">
  $(document).ready(function() {

$.getScript('http://localhost:8080/jw/web/js/client/inbox.js?id=1&rows=5&divId=inbox1',
null);
  });
</script>

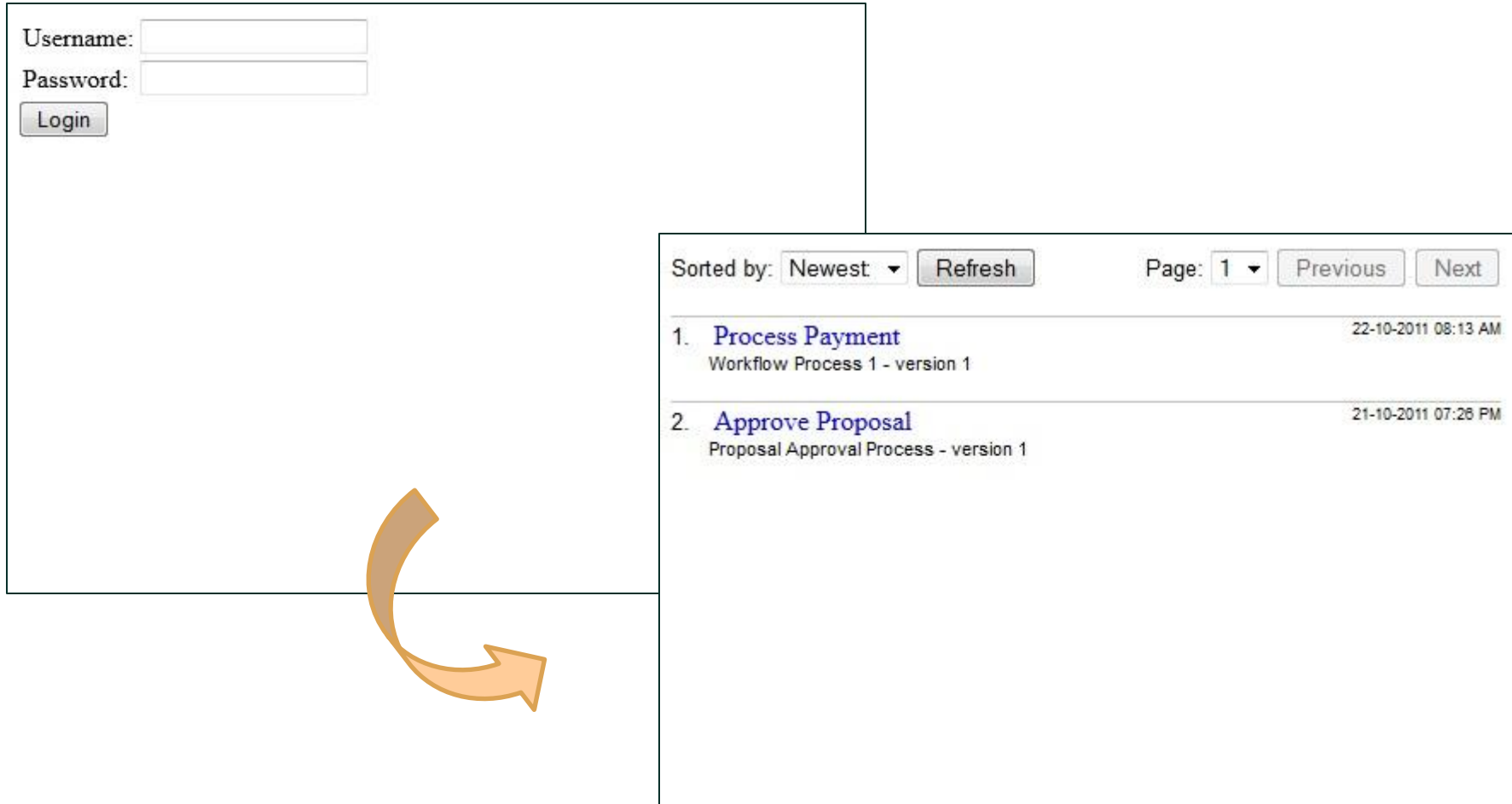
```

Unique id to identify the inbox instance

Number of rows to be displayed in this inbox instance

ID of the HTML DIV to display this inbox instance

# Embedding Task Inbox into External System



The diagram illustrates the embedding of a task inbox into an external system. On the left, a login form is shown with fields for Username and Password, and a Login button. On the right, a task inbox is displayed, showing a list of tasks with their titles, descriptions, and timestamps. The tasks are sorted by Newest, and the page is 1 of 1. A large orange arrow points from the login form to the task inbox, indicating the flow of data or the integration process.

Username:

Password:

Login


Sorted by: Newest ▾ Refresh

Page: 1 ▾ Previous Next

1. [Process Payment](#) 22-10-2011 08:13 AM  
Workflow Process 1 - version 1
2. [Approve Proposal](#) 21-10-2011 07:28 PM  
Proposal Approval Process - version 1

# Embedding Task Inbox into External System

Customize the look and feel of embedded task inbox by pointing to a new css file.



```
<link rel="stylesheet" type="text/css" href="http://localhost:8080/jw/css/portlet.css">
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-migrate-1.2.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/jw/js/json/util.js"></script>
<div id="inbox1">
  <center>
  </center>
</div>
<script type="text/javascript">
  $(document).ready(function() {

$.getScript('http://localhost:8080/jw/web/js/client/inbox.js?id=1&rows=5&divId=inbox1',
null);
  });
</script>
```

# Embedding Task Inbox into External System with SSO

```

<link rel="stylesheet" type="text/css" href="http://localhost:8080/jw/css/portlet.css">
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-migrate-1.2.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/jw/js/json/util.js"></script>
<div id="inbox1">
  <center>
  </center>
</div>
<script type="text/javascript">
  $(document).ready(function(){
    var loginCallback = {
      success: function(){
$.getScript('http://localhost:8080/jw/web/js/client/inbox.js?id=1&rows=5&divId=inbox1',
null);
      }};
    AssignmentManager.login('http://localhost:8080/jw', 'admin', 'admin',
loginCallback);
  });
</script>

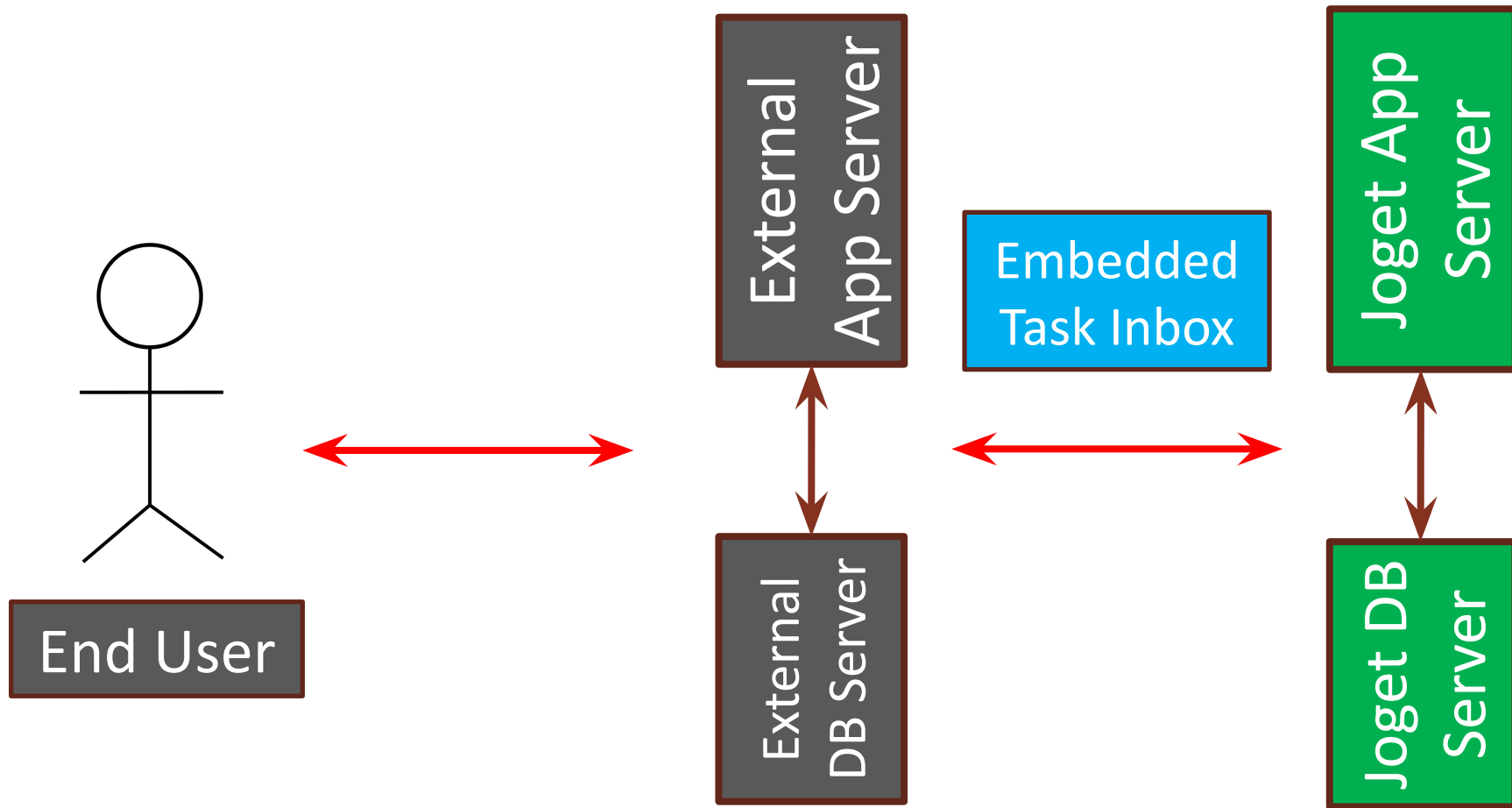
```

Load script after successfully logged in



# Setup Suggestion

- Can you now imagine the following setup?



# Chapter Review

---

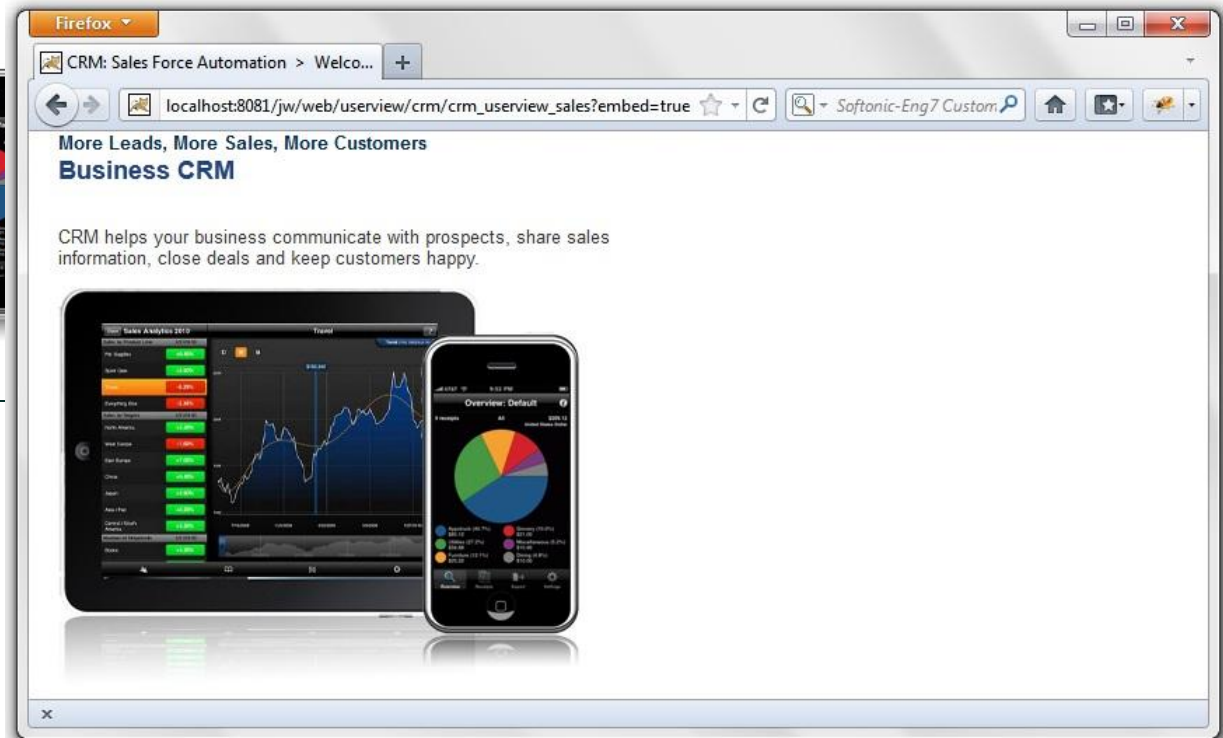
- Understand on how to embed Task Inbox into external app.

---

# Chapter 7

## Embedding Userview Page in an iFrame

# Embedding Userview Page in an IFrame



# Embedding Userview Page in an IFrame

---

- The header, menu & footer of Userview can be removed by following:
  - By adding parameter “**embed=true**” in the **URL** OR
  - By modifying the **URL**

From

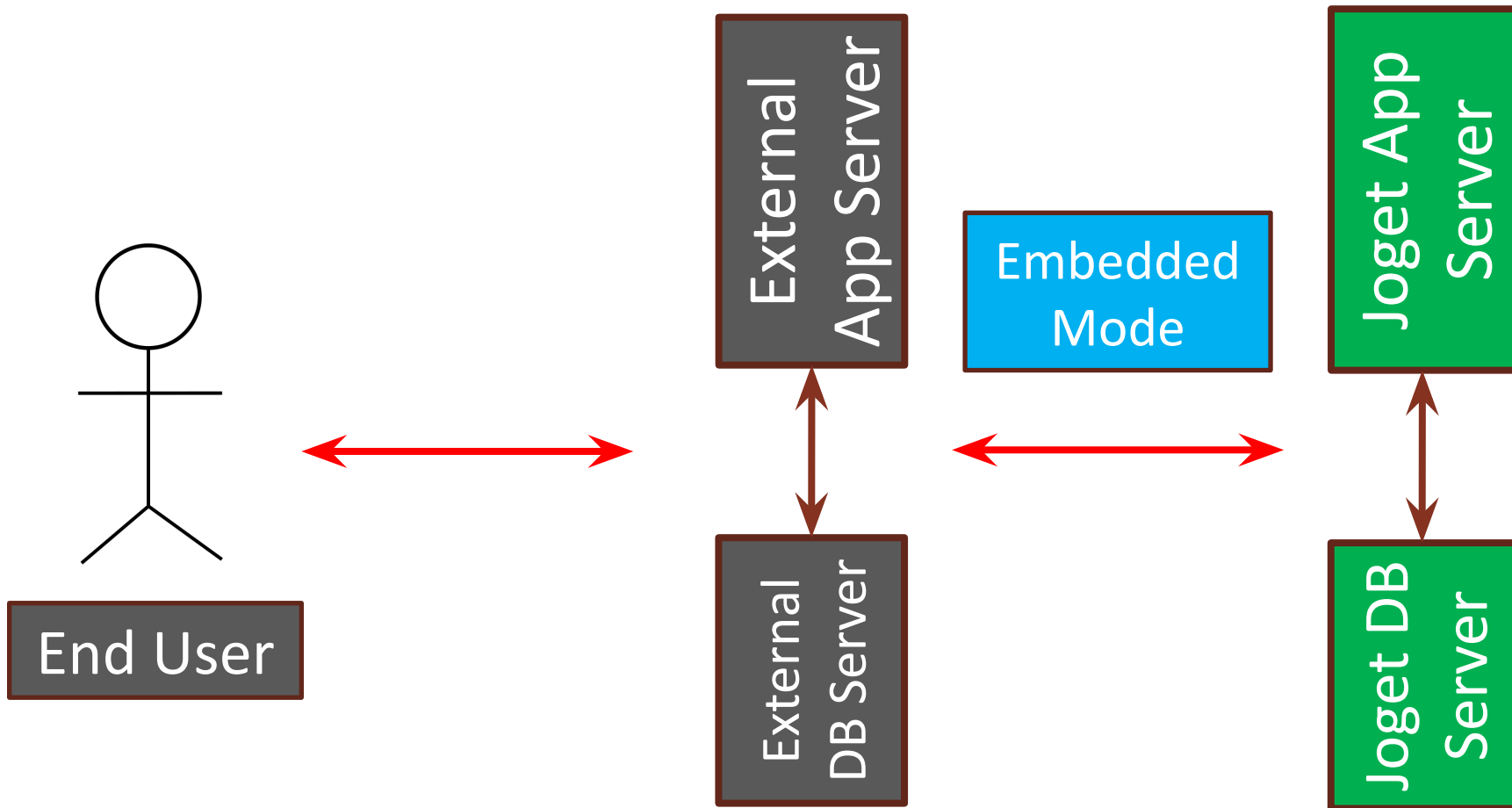
[http://localhost:8081/jw/web/userview/crm/crm\\_userview\\_sales](http://localhost:8081/jw/web/userview/crm/crm_userview_sales)

To

[http://localhost:8081/jw/web/\*\*embed\*\*/userview/crm/crm\\_userview\\_sales](http://localhost:8081/jw/web/embed/userview/crm/crm_userview_sales)

# Setup Suggestion

- Can you now imagine the following setup?



# Exercise

---

- Choose a Userview Page.
- Add in the necessary parameter to turn it into the Embedded mode.
- Use iFrame to display it in your custom app.

# Chapter Review

---

- Understands the purpose of using Embedded mode of Userview pages.



---

# Chapter 8

## Using the JSON Tool

# Introduction

---

- The JSON Tool enables one to issue a JSON web service call, and to save the returned data into Joget's form data and/or into the process's workflow variable.

- Reference:

<https://dev.joget.org/community/display/DX7/JSON+Tool>

# How it works?

- The JSON Tool will call the JSON API endpoint, along with any required parameters.

### Configure JSON Tool ?

Configure JSON Tool > Store To Form > Store To Workflow Variable

JSON URL \*

Call Type

Request Headers

NAME	VALUE
<span>+</span>	

No Response Expected

Debug Mode ?

# How it works?

---

- Data will be returned in JSON format.

```
INFO 28 Jul 2021 04:27:26 org.joget.apps.app.lib.JsonTool - GET :  
https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT  
  
INFO 28 Jul 2021 04:27:26 org.joget.apps.app.lib.JsonTool -  
https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT returned with status :  
200  
  
INFO 28 Jul 2021 04:27:26 org.joget.apps.app.lib.JsonTool -  
{"symbol":"BTCUSDT","price":"38069.47000000"}
```


```
{"symbol":"BTCUSDT","price":"38069.47000000"}
```

# How it works?

- Data can be then stored into Form table or Workflow Variable.

**Store To Form**

Configure JSON Tool > **Store To Form** > Store To Workflow Variable

Form  ✕ ▾ 

Base JSON Object Name for Multirow Data

Field Mapping

FIELD NAME	JSON OBJECT NAME
<input type="text" value="name"/> <span>✕ ▾</span>	<input type="text" value="symbol"/>

+ ⬆ ⬇ ✕

# Exercise

---

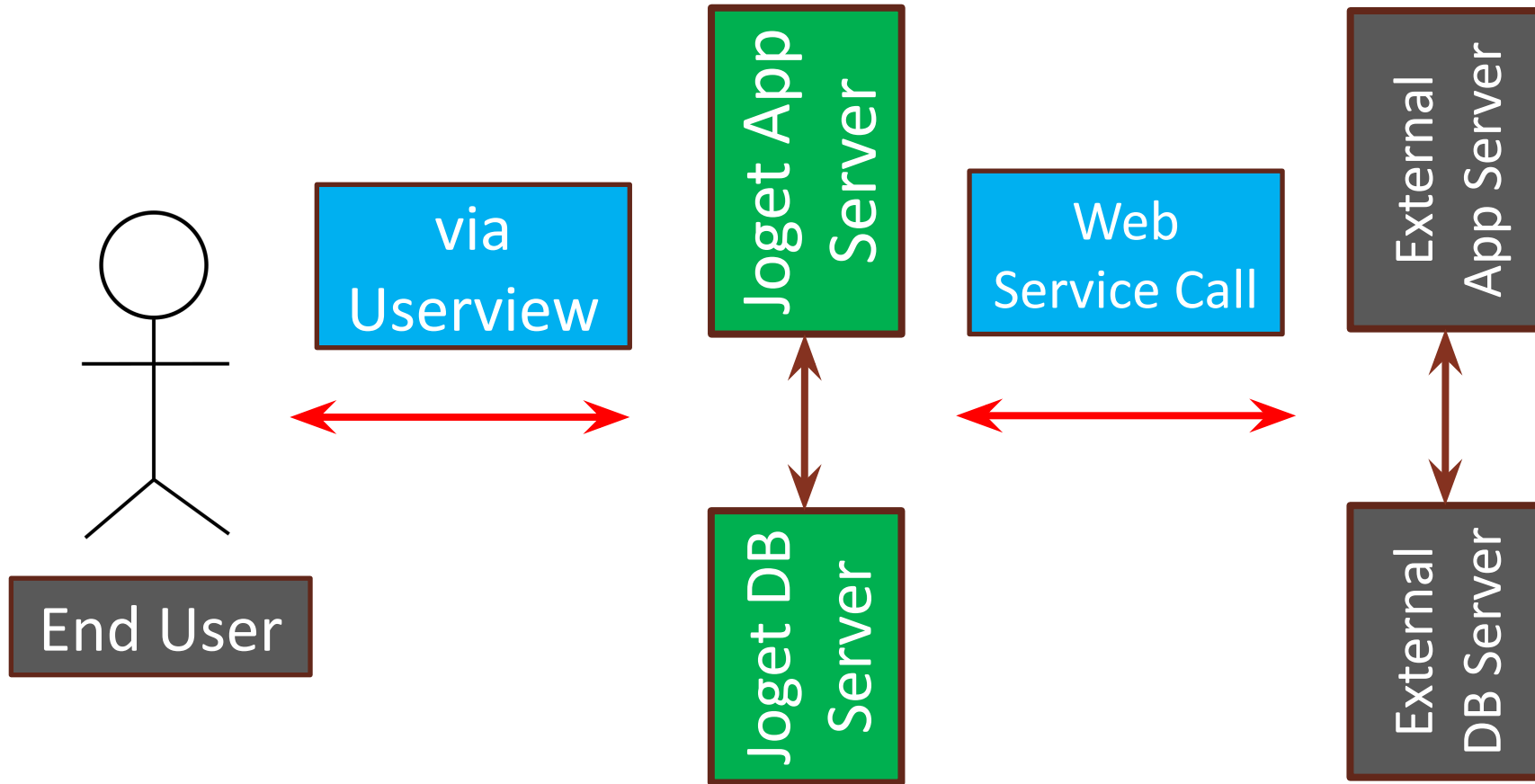
- Try this JSON API endpoint:  
<https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT>
- Create a new Joget App with a **Activity -> Tool -> Activity** process.
- Map the Tool to a **JSON Tool**, configure accordingly.

Reference:

<https://binance-docs.github.io/apidocs/spot/en/#symbol-price-ticker>

# Setup Suggestion

- Can you now imagine the following setup?



# Chapter Review

---

- Be able to make calls from Joget to other JSON web services by using the **JSON Tool**.



---

# Chapter 9

## Using the SOAP Tool

# Introduction

---

- The SOAP Tool allows one to invoke web service for integration purpose to return useful information from external sources into the process instance.
- Reference:  
<http://dev.joget.org/community/display/DX7/SOAP+Tool>

# How it works?

- The SOAP Tool will call the Web Service configured, passes the set of parameters set.

### Configure SOAP Tool

Configure SOAP Tool > Store To Form > Store To Workflow Variable > Advanced





WSDL URL \*

Operation Name \*

Username

Password

Parameters

Value
<input type="text" value="8.8.8.8"/>   


# How it works?

- Data will be returned in JSON format.

```
INFO 07 Jun 2013 10:54:37 SoapTool - <ns1:GetGeoIPResult
xmlns:ns1="http://www.webservices.net/"><ns1:ReturnCode>1</ns1:ReturnCode><ns1:IP>8.
8.8.8</
ns1:IP><ns1:ReturnCodeDetails>Success</ns1:ReturnCodeDetails><ns1:CountryName>United
States</ns1:CountryName><ns1:CountryCode>USA</ns1:CountryCode></n s1:GetGeoIPResult>
INFO 07 Jun 2013 10:54:37 SoapTool - {"GetGeoIPResult":{"CountryName":"United
States","ReturnCodeDetails":"Success","ReturnCode":"1","IP":"8.8.8.8",
"CountryCode":"USA"}}
```

```
{
  "GetGeoIPResult": {
    "CountryName": "United States",
    "ReturnCodeDetails": "Success",
    "ReturnCode": "1",
    "IP": "8.8.8.8",
    "CountryCode": "USA"
  }
}
```

# How it works?

- Data can be then stored into Form table or Workflow Variable.

**Store To Form**

Configure SOAP Tool > Store To Form > Store To Workflow Variable > Advanced

Form Data Form ▼

Base XML Object Name for Multirow Data

Field Mapping

Field ID	XML Object Name	
package_name	GetGeolIPResult.CountryName	↑ + ×
package_id	GetGeolIPResult.CountryCode	+ ↓ ×

+

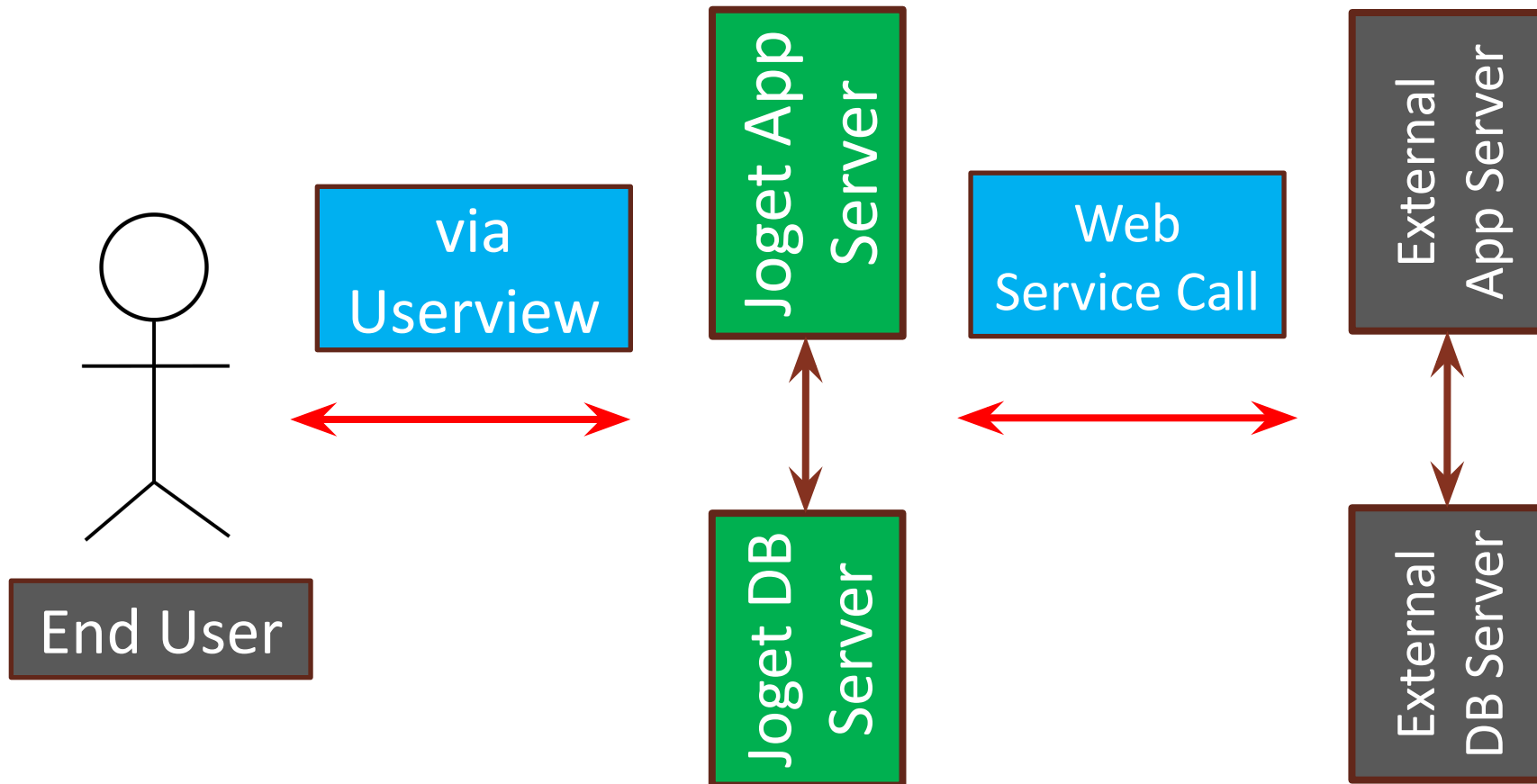
# Exercise

---

- Go to <https://documenter.getpostman.com/view/8854915/Szf26WHn#33a2b225-11a6-48d3-a695-fb0989cc4971>
- Select a service, e.g.: "List of Countries by Name"
- Create a new Joget App with a **Activity -> Tool -> Activity** process.
- Map the Tool to a **SOAP Tool**, configure accordingly.

# Setup Suggestion

- Can you now imagine the following setup?



# Chapter Review

---

- Be able to make calls from Joget to other web services by using the **SOAP Tool**.





# Chapter 10

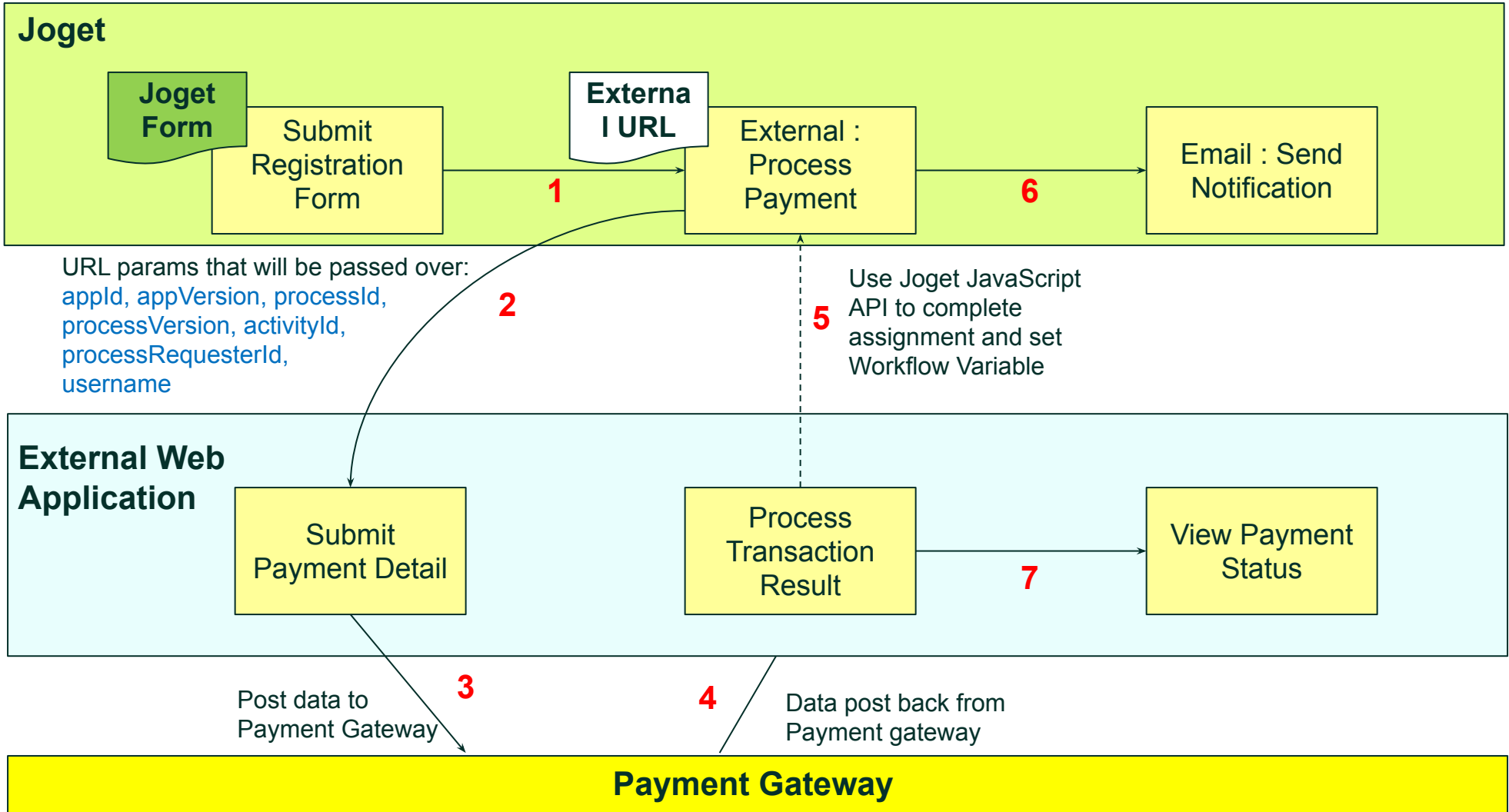
## Integrating with External Form

# Integrating with External Form

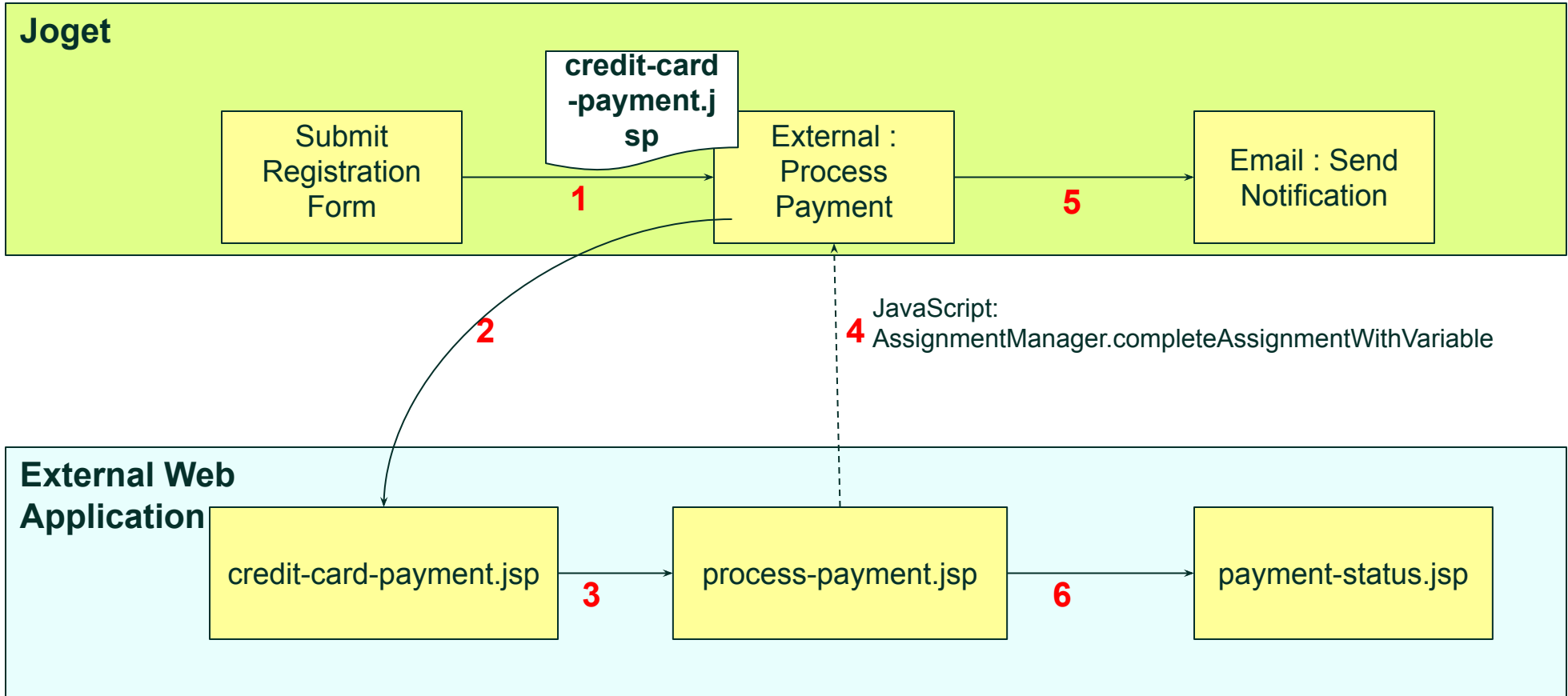
---

- To perform (form) activities that Joget is not designed to do.
- To integrate with web data form built in external system, regardless of platform.
- To allow data to be submitted into external system within a process.

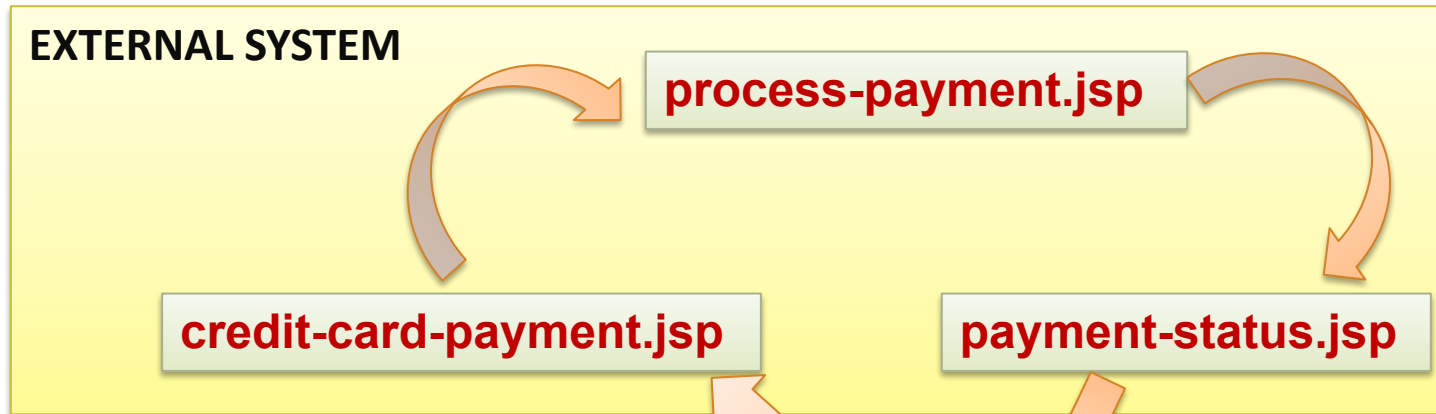
# Scenario – Integrate with Payment Gateway



# Example

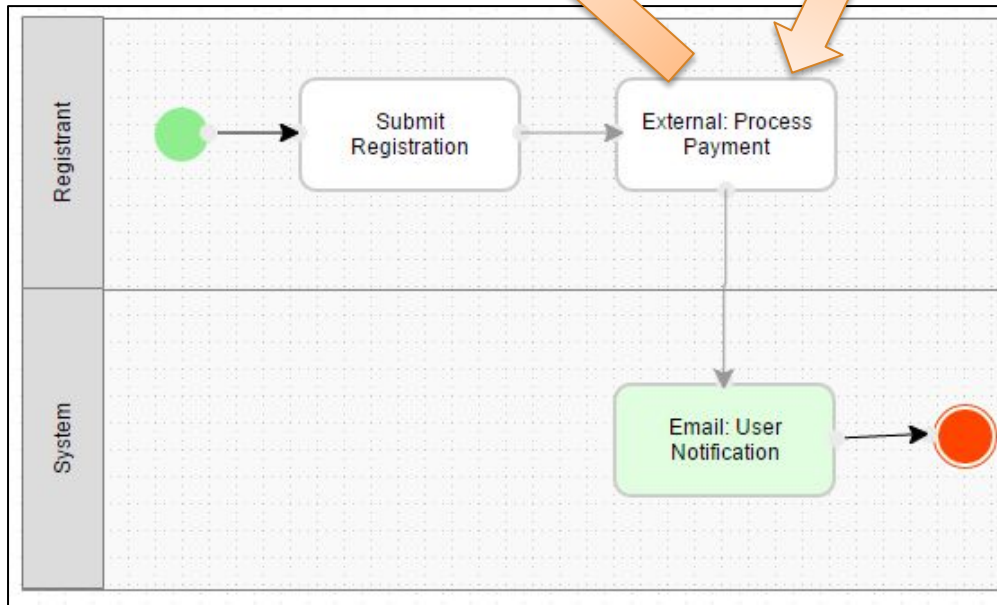


# Example – Process

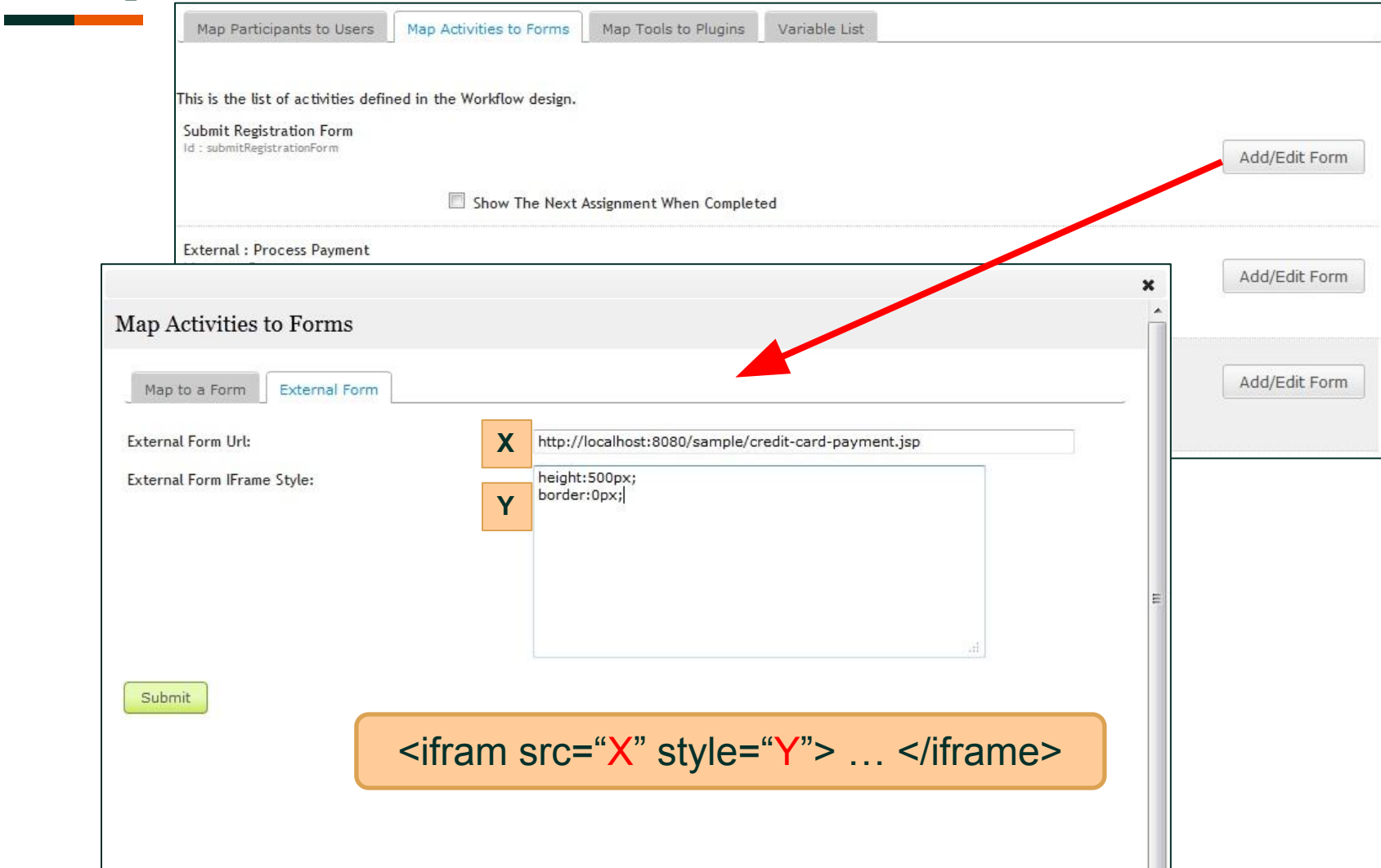


External Form Mapping

JavaScript API



# Map to an External Form



Map Participants to Users | **Map Activities to Forms** | Map Tools to Plugins | Variable List

This is the list of activities defined in the Workflow design.

Submit Registration Form  
Id : submitRegistrationForm

Show The Next Assignment When Completed

External : Process Payment

Map Activities to Forms

Map to a Form | **External Form**

External Form Url: **X**

External Form IFrame Style: **Y**

Submit

`<iframe src="X" style="Y"> ... </iframe>`

# credit-card-payment.jsp

---

- Parameters that will be passed into external form from Joget:
  - appld
  - appVersion
  - processId
  - processVersion
  - activityId
  - processRequesterId
  - username
- **credit-card-payment.jsp** is the first external URL called, but it can call to any other web pages in the series, before getting back to Joget activity.

# credit-card-payment.jsp

```
<h1>Credit Card Payment</h1>
```

```
<p>Parameters passed into this page:
```

```
<ul>
```

```
  <li>appId: <%=request.getParameter("appId")%></li>
```

```
  <li>appVersion: <%=request.getParameter("appVersion")%></li>
```

```
  <li>processId: <%=request.getParameter("processId")%></li>
```

```
  <li>processVersion: <%=request.getParameter("processVersion")%></li>
```

```
  <li>activityId: <%=request.getParameter("activityId")%></li>
```

```
  <li>processRequesterId :
```

```
<%=request.getParameter("processRequesterId")%></li>
```

```
  <li>username: <%=request.getParameter("username")%></li>
```

```
</ul>
```

```
</p>
```

```
<form method="POST"
```

```
  action="process-payment.jsp?activityId=<%=request.getParameter("activityId")%>
  &redirect=payment-status.jsp">
```

```
  <p>Assuming you have completed all payment details, now let's submit this
  form to process-payment.jsp</p>
```

```
  <input type="submit" value="Submit" />
```

```
</form>
```



# process-payment.jsp

- Process the payment details submitted from credit-card-payment.jsp, and make a Ajax JavaScript call to Joget, to
  - Complete the “External: Process Payment” assignment
  - Update workflow variable – “**transaction\_no**”
- Redirect to payment-status.jsp to display summary of the transaction.
- Once the “External: Process Payment” is completed, subsequent workflow activities will be continued and performed in Joget.

# process-payment.jsp

```
<script type="text/javascript"
src="http://localhost:8080/jw/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/jw/js/json/util.js"></script>

<script type="text/javascript">
    var callback = {
        success : function(response) {

AssignmentManager.completeAssignmentWithVariable("http://localhost:8080/
jw",
            getUrlParam("activityId"),
            "var_transaction_no =TN00001",
            escape(getUrlParam("redirect")));
        }
    }
    AssignmentManager.login("http://localhost:8080/jw", "admin", "admin",
callback);
</script>
```

# payment-status.jsp

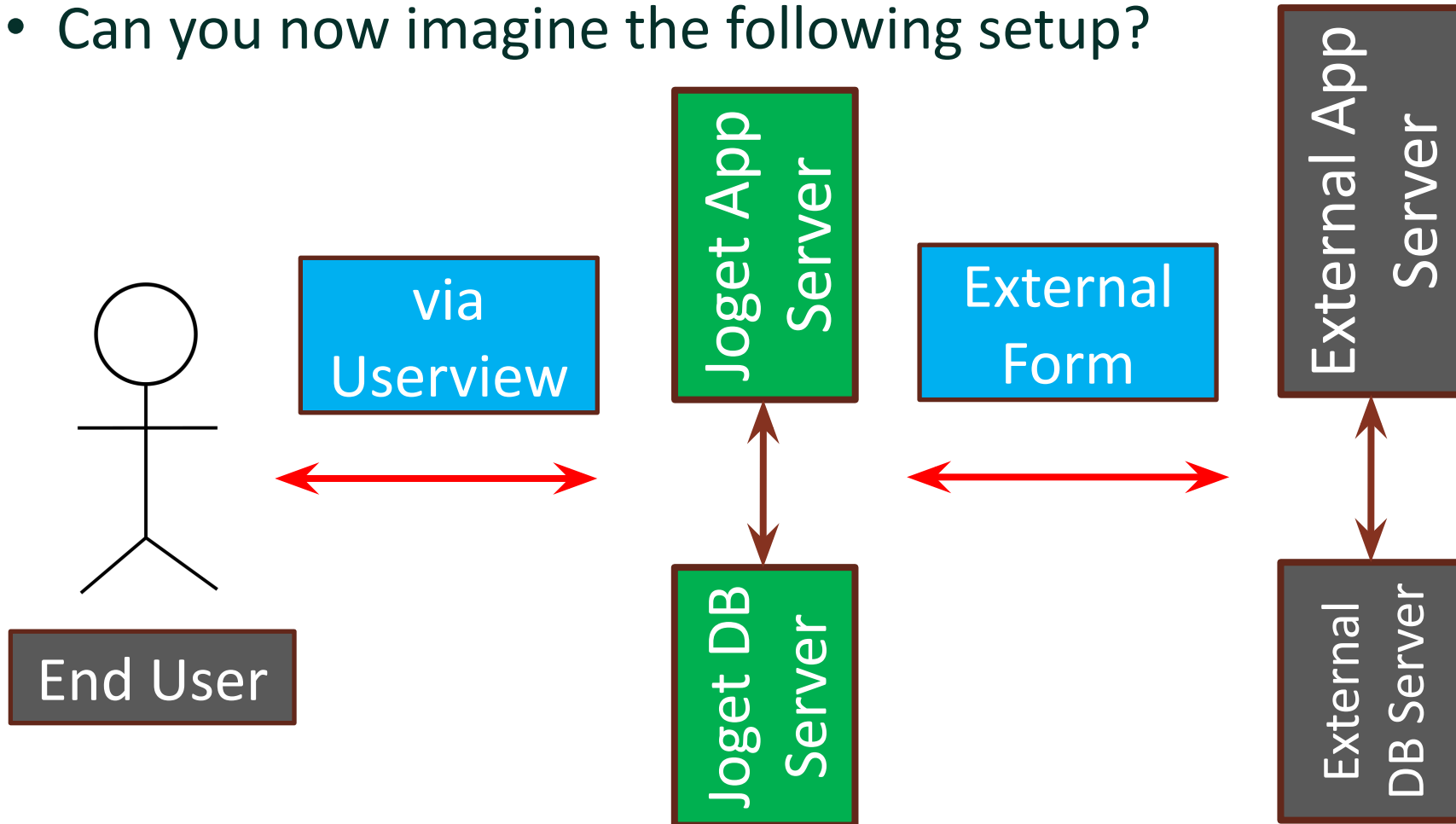
- Display to user synchronously without interrupt subsequent workflow activities.

```
<h1>Payment Status</h1>
```

```
<p>Yeh~~~ payment transaction is completed.</p>
```

# Setup Suggestion

- Can you now imagine the following setup?



# Chapter Review

---

- Understand and being able to use External Form effectively.

---

# Chapter 11

## Using API Builder

# API Builder

---

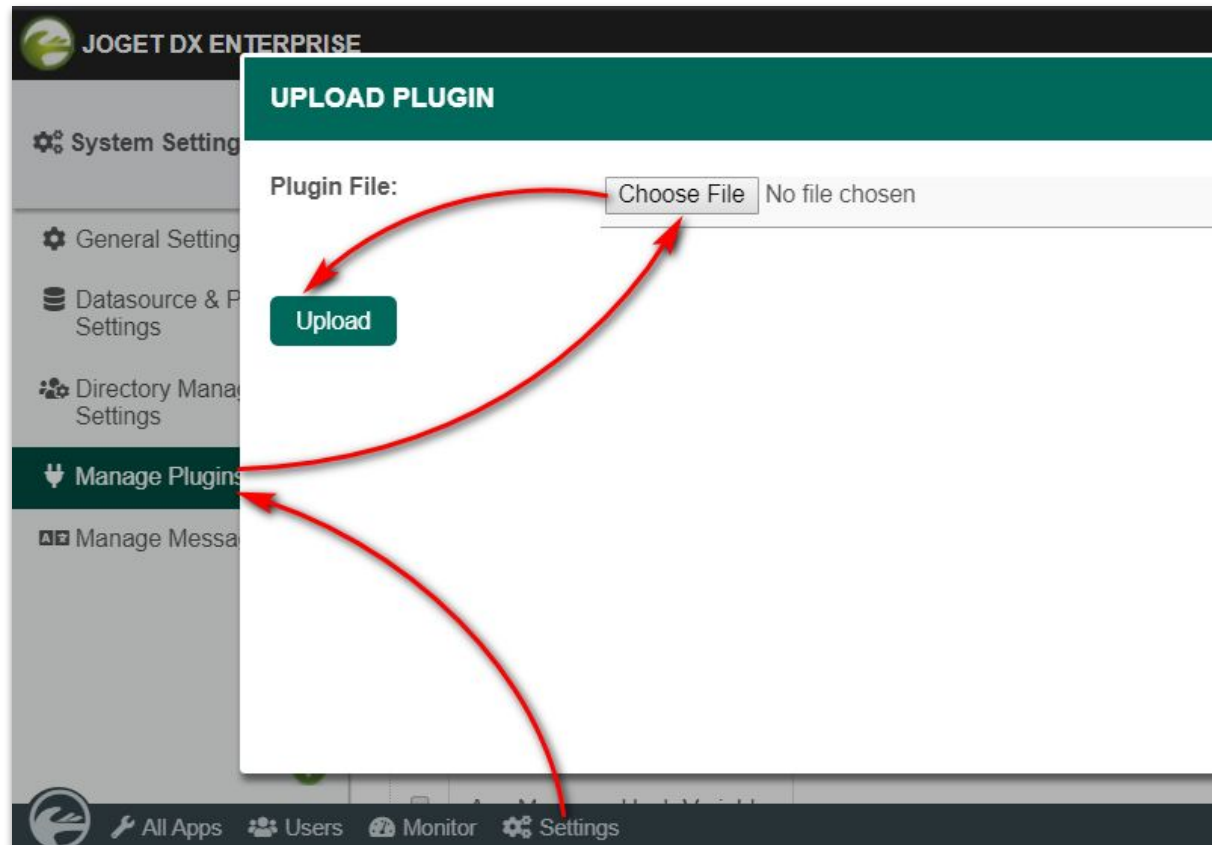
- API Builder is a new type of plugin called Custom Builder.
- Allows one to create your own customized JSON APIs for Joget Apps.

Reference:

<https://dev.joget.org/community/display/marketplace/API+Builder>

# Installing API Builder

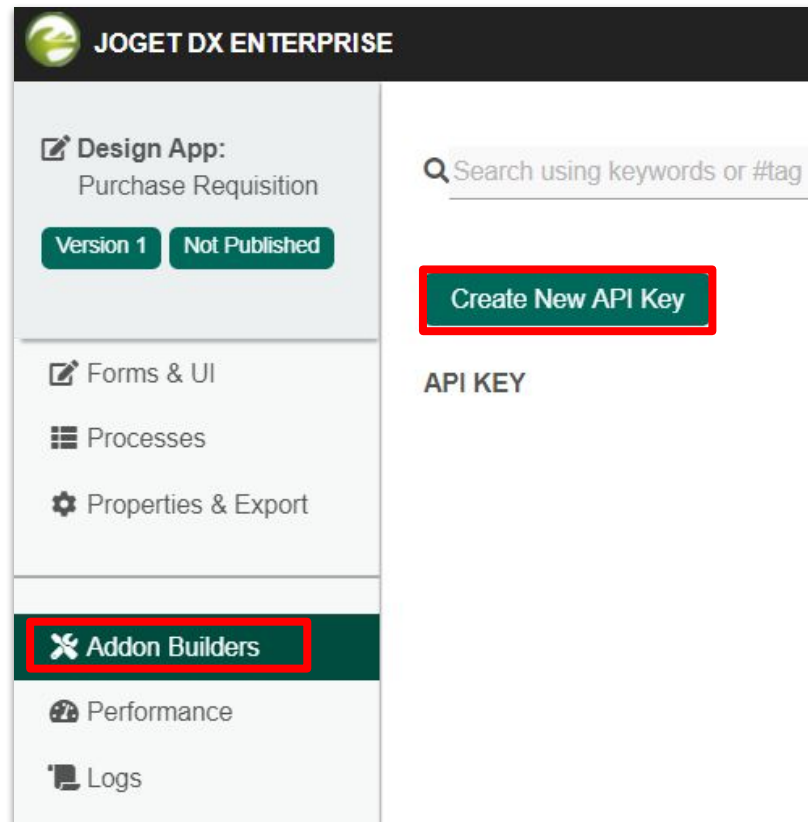
- Download the plugin from Joget Marketplace and import it into your Joget platform via Manage Plugin.





# Exercise

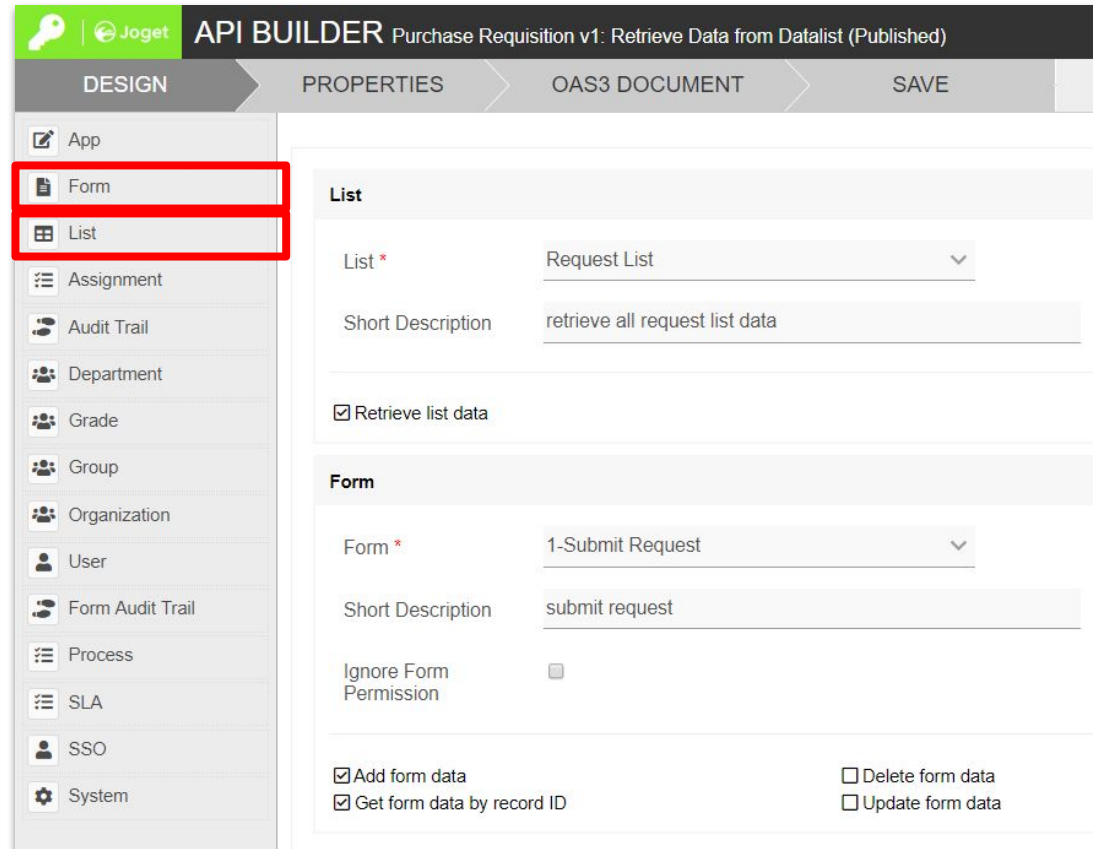
- Continue to use the app from the previous chapters.
- Click on the new menu Addon Builders and click on Create new API Key.



The screenshot displays the Joget DX Enterprise interface. The top header reads "JOGET DX ENTERPRISE". On the left sidebar, the "Addon Builders" menu item is highlighted with a red box. The main content area shows a search bar with the text "Search using keywords or #tag". Below the search bar, the "Create New API Key" button is highlighted with a red box. The sidebar also shows other menu items: "Design App: Purchase Requisition" (with "Version 1" and "Not Published" buttons), "Forms & UI", "Processes", "Properties & Export", "Performance", and "Logs".

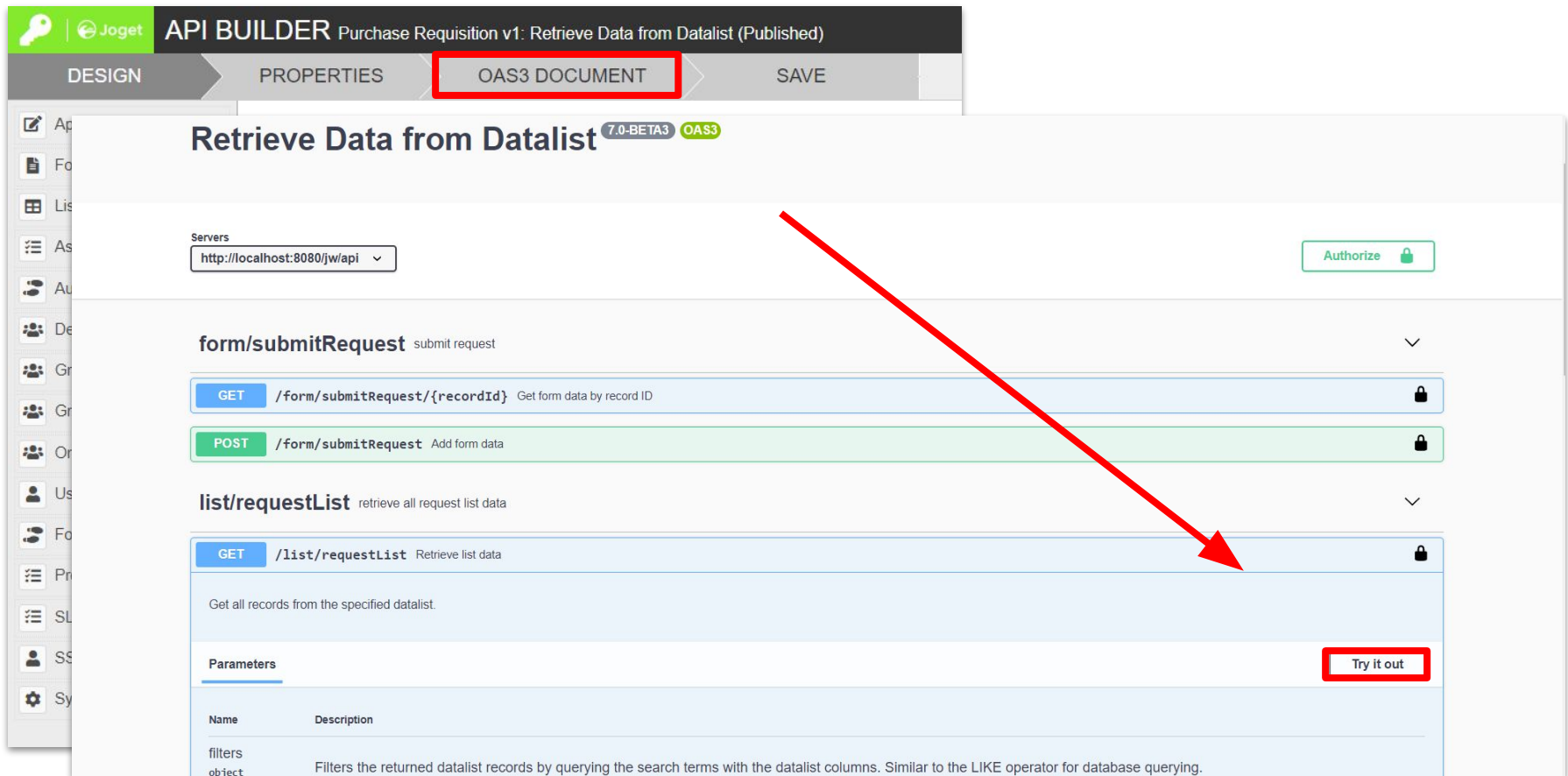
# Exercise

- Try retrieving
  - all Request data from the list,
  - add form data to Submit Request and,
  - retrieve form data by record ID.



# Generate OAS3 Document

- Click on OAS3 Document.
- Click on an API method and test it out.



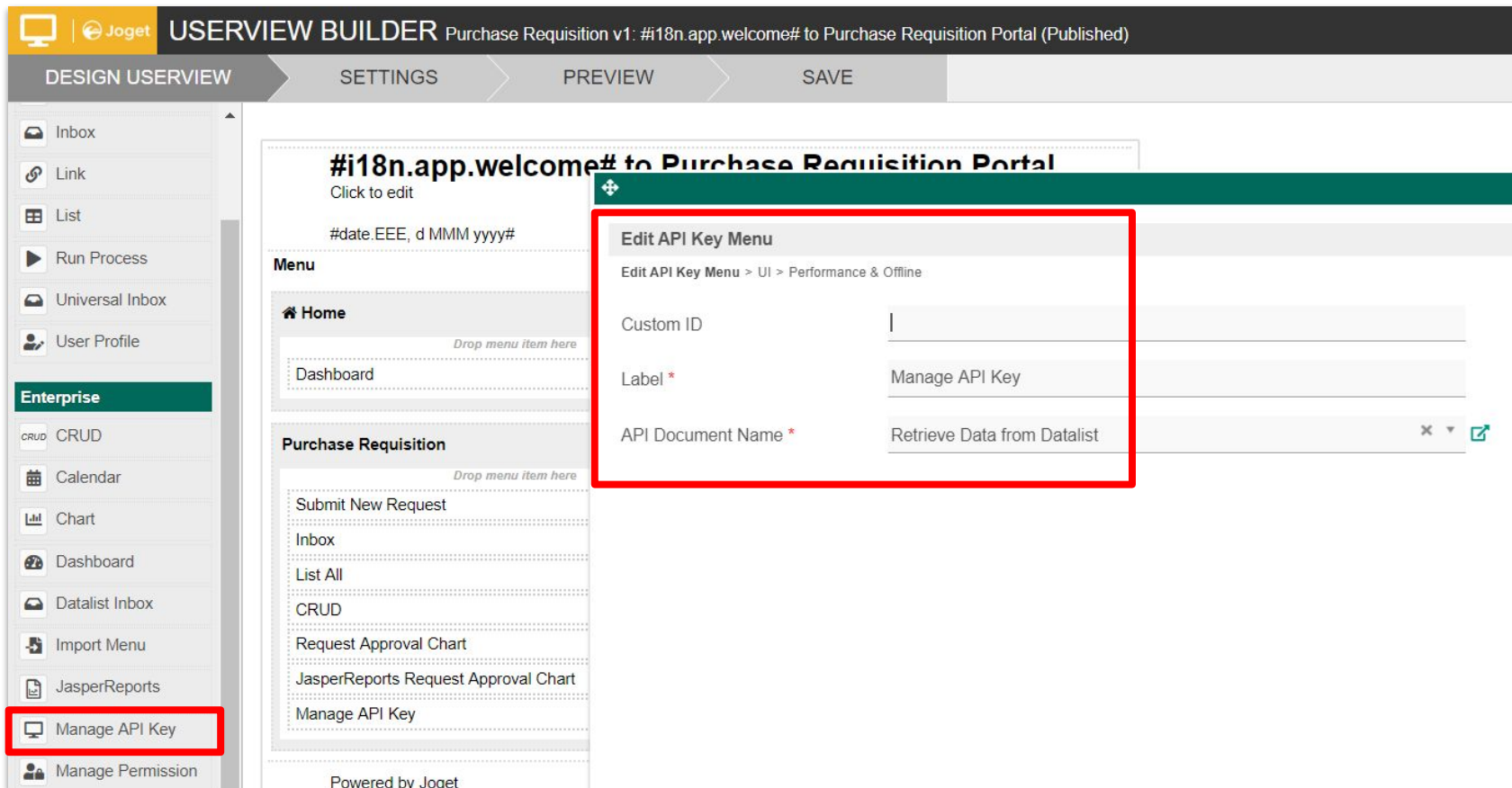
The screenshot shows the Joget API Builder interface for a service named "Purchase Requisition v1: Retrieve Data from Datalist (Published)". The "OAS3 DOCUMENT" tab is selected and highlighted with a red box. The main area displays the API specification for "Retrieve Data from Datalist" (7.0-BETA3 OAS3). A red arrow points from the "OAS3 DOCUMENT" tab to the "list/requestList" endpoint. The "list/requestList" endpoint is expanded to show a "GET" method with the description "Retrieve list data". A "Try it out" button is highlighted with a red box at the bottom right of the endpoint details.

**API Builder Interface:**

- Navigation:** DESIGN, PROPERTIES, **OAS3 DOCUMENT**, SAVE
- Service:** Retrieve Data from Datalist (7.0-BETA3 OAS3)
- Servers:** http://localhost:8080/jw/api
- API Endpoints:**
  - form/submitRequest** (submit request)
    - GET /form/submitRequest/{recordId} (Get form data by record ID)
    - POST /form/submitRequest (Add form data)
  - list/requestList** (retrieve all request list data)
    - GET /list/requestList (Retrieve list data)
- Endpoint Details (list/requestList GET):**
  - Description: Get all records from the specified datalist.
  - Parameters: filters, object
  - Try it out button

# Manage API Key

- Userview element to control access to API methods created with API Builder.



The screenshot displays the Joget Userview Builder interface for a 'Purchase Requisition v1' application. The main workspace shows a menu structure with a 'Purchase Requisition' section containing items like 'Submit New Request', 'Inbox', 'List All', 'CRUD', 'Request Approval Chart', 'JasperReports Request Approval Chart', and 'Manage API Key'. A red box highlights the 'Manage API Key' element in the left sidebar. A modal window titled 'Edit API Key Menu' is open, showing configuration fields: 'Custom ID' (empty), 'Label \*' (set to 'Manage API Key'), and 'API Document Name \*' (set to 'Retrieve Data from Datalist'). The breadcrumb path is 'Edit API Key Menu > UI > Performance & Offline'.

# Exercise

- Create an API key.
- Try calling the API methods with and without the API key.

Home > Purchase Requisition > Manage API Key

API Key	45548ee182a8412388b6ab3c62899b30
Authentication Method	API Key
Domain Whitelist	*
	<i>One domain in one line. Wildcard (*) is allowed.</i>
IP Address Whitelist	
	<i>One IP Address in one line. Wildcard (*) is allowed.</i>
Remark	API Key test

# Manage API Key Log

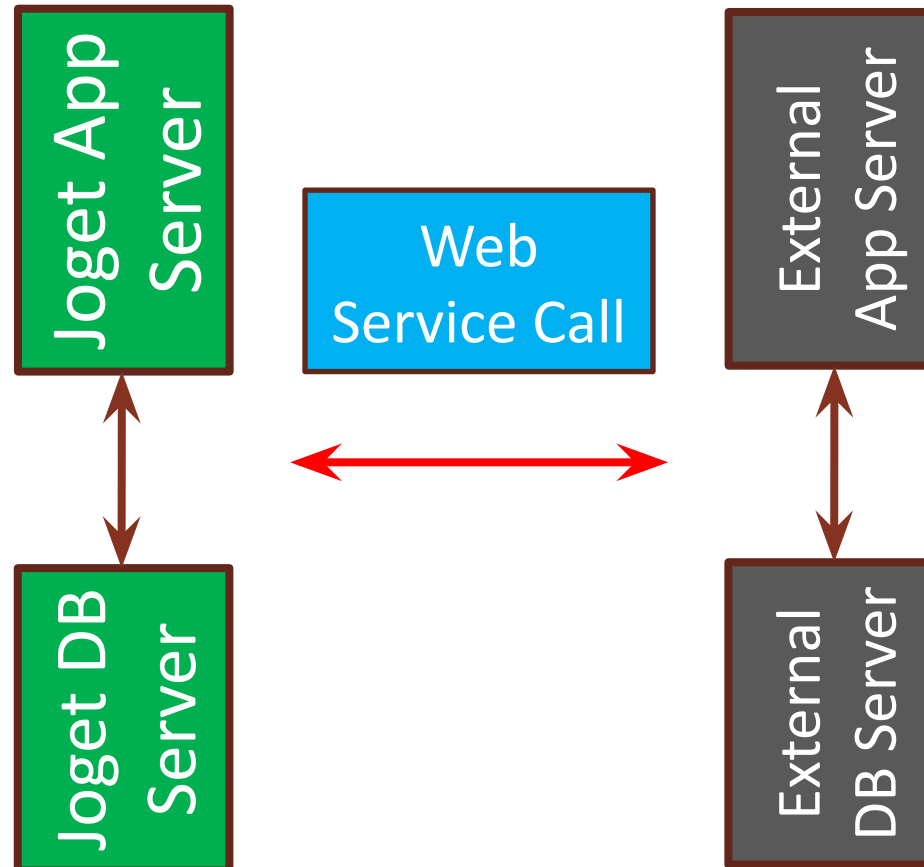
- There's also a log with details on each attempt.

DATE	API ID	API KEY	STATUS	METHOD	USER AGENT	SOURCE IP	MESSAGE	EXECUTE TIME (MS)
16-01-2020 04:00 PM	API-9aac3b80-2ff4-4384-aa43-f6ba158724fe	45548ee182a8412388b6ab3c62899b30	200	list/requestList	PostmanRuntime/7.21.0	0.0.0.0:0.0:1		828
16-01-2020 04:00 PM	API-9aac3b80-2ff4-4384-aa43-f6ba158724fe	45548ee182a8412388b6ab3c62899b30	200	list/requestList	PostmanRuntime/7.21.0	0.0.0.0:0.0:1		754
16-01-2020 04:00 PM	API-9aac3b80-2ff4-4384-aa43-f6ba158724fe	45548ee182a8412388b6ab3c62899b30	200	list/requestList	PostmanRuntime/7.21.0	0.0.0.0:0.0:1		328
16-01-2020 04:00 PM	API-9aac3b80-2ff4-4384-aa43-f6ba158724fe	45548ee182a8412388b6ab3c62899b30	200	list/requestList	PostmanRuntime/7.21.0	0.0.0.0:0.0:1		2195

4 items found, displaying all items.  
[CSV](#) | [Excel](#) | [XML](#) | [PDF](#)

# Setup Suggestion

- Can you now imagine the following setup?



# Chapter Review

---

- Understand and being able to use API Builder effectively.



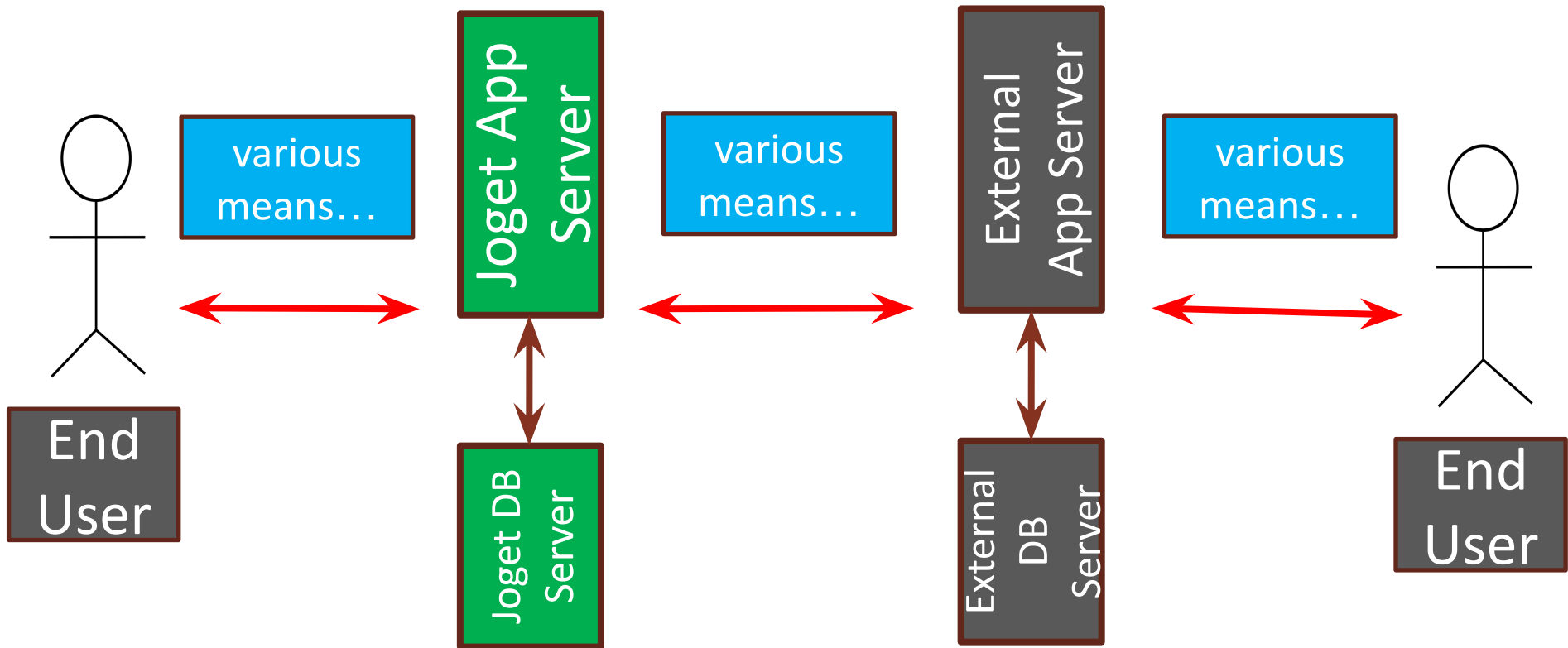
# Module Review

---

1. Introduction
2. JSON API
3. JSON API Authentication
4. JavaScript API
5. Single Sign On (SSO)
6. Embedding Task Inbox into External System
7. Embedding Userview Page in an iFrame
8. JSON Tool
9. SOAP Tool
10. Integrating with External Form
11. Using API Builder

# Module Review

- Can you now picture various setup scenarios to integrate Joget with your external app?



# Stay Connected with Joget

---

- <http://www.joget.org>
- <http://community.joget.org>
- <http://twitter.com/jogetworkflow>
- <http://facebook.com/jogetworkflow>
- <http://youtube.com/jogetworkflow>
- <http://slideshare.net/joget>