

---

# Joget DX

## Improving Your Form Design & Presentation

 <http://facebook.com/jogetworkflow>

 <http://twitter.com/jogetworkflow>

# Prerequisites

---

1. Good understanding on the basic functionality of the Form Builder.

# Content

---

1. Introduction
2. Grid
3. Form Grid
4. Multirow Form Binder
5. List Grid
6. CRUD
7. Custom HTML
8. Using Advanced Tool's Permission



# Chapter 1

## Introduction

# Introduction

---

- Learning about more Joget plugins/elements to improve your form design and presentation.
- In this module, we will be covering the following elements.
  1. Grid (Form Element)
  2. Form Grid (Form Element)
  3. Multirow Form Binder (Form Binder)
  4. List Grid (Form Element)
  5. CRUD (Userview Menu)
  6. Custom HTML (Form Element)

# List of Available Elements

---

- Check out

<https://dev.joget.org/community/display/DX7/Form+Builder> for the list of **Form** related elements (Form Element, Form Validator, Form Binder, Form Options Binder).

- Check out

<https://dev.joget.org/community/display/DX7/Userview+Builder> for the list of **Userview** related elements.



# Chapter 2

## Grid

# Grid

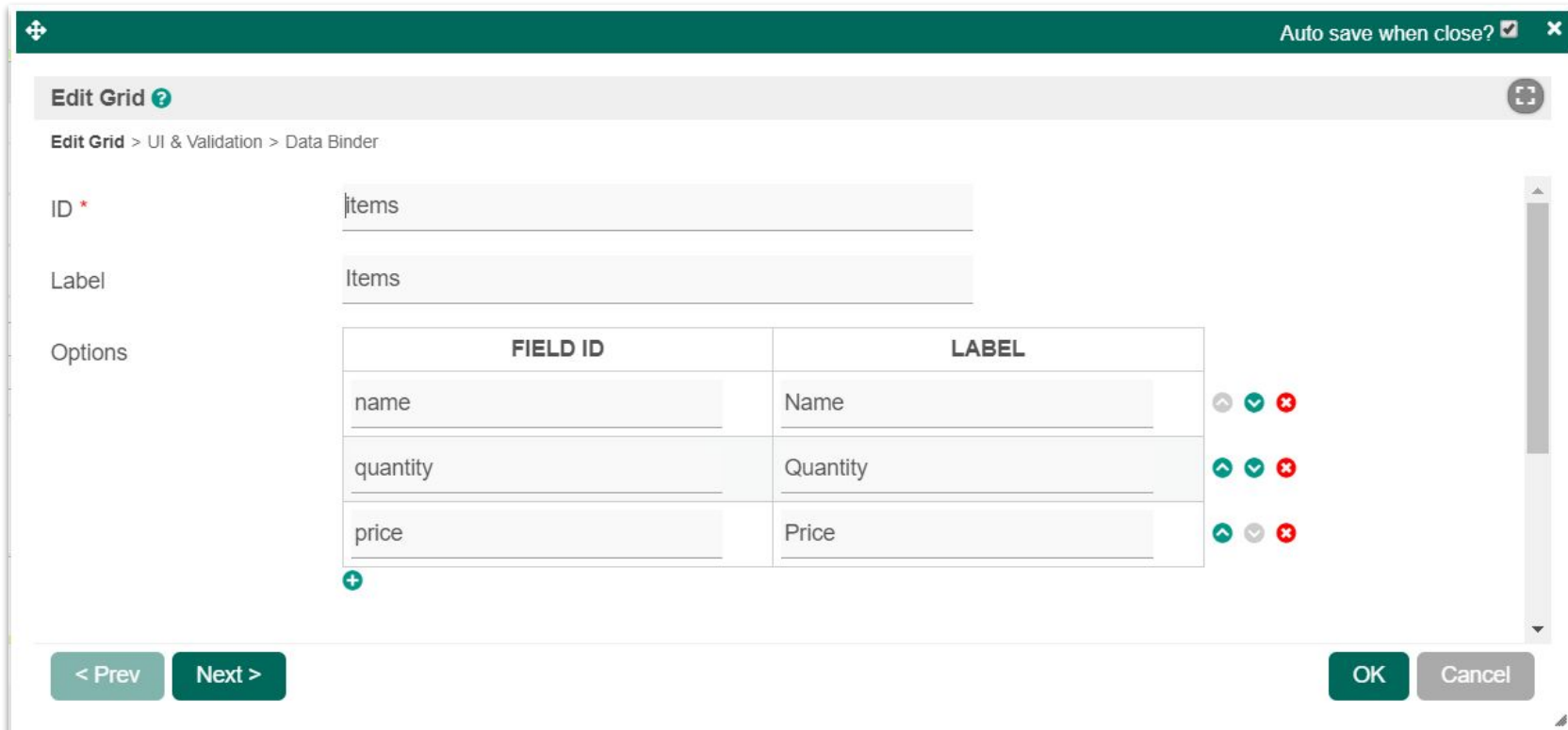
---

- Grid is the most basic element available in the Form Builder in capturing multi-row data.
  
- Reference: <http://dev.joget.org/community/display/DX7/Grid>
- For your information: The other grid-like element available in Form Builder is Form Grid.



# Refresh

- Refresh your memory on what you did back in module 5 - Designing your first Form
- Take a look at your “items” field element



Auto save when close?

**Edit Grid** ?

Edit Grid > UI & Validation > Data Binder

ID \*

Label

Options

| FIELD ID                              | LABEL                                 |  |
|---------------------------------------|---------------------------------------|--|
| <input type="text" value="name"/>     | <input type="text" value="Name"/>     | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/> |
| <input type="text" value="quantity"/> | <input type="text" value="Quantity"/> | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/> |
| <input type="text" value="price"/>    | <input type="text" value="Price"/>    | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/> |

< Prev    Next >

OK    Cancel

# Just In Case...

---

- You can download the base Purchase Requisition app - **13.jwa** to start ‘playing around’ with it.
- Completed Form Definition for “1-Submit Request” can be obtained from **13.2.1.txt**.

# Chapter Review

---

- Able to use the Grid element.



# Chapter 3

## Form Grid

# Form Grid

---

- Form Grid works similarly like the basic Grid.
- Instead of editing data row inline, editing is done on a full fledged Form that opens up in a dialog.
- Able to reuse validation and formatting from the selected Form.
- Reference:  
<http://dev.joget.org/community/display/DX7/Form+Grid>

# Exercise

---

- Re-import the base app “**13.jwa**” into your copy of Joget, OR delete the “items” Grid element.
- Create a new Form with the following details.

| CREATE NEW FORM |                        |
|-----------------|------------------------|
| FORM DETAILS    |                        |
| Form ID         | items                  |
| Form Name       | Items                  |
| Table Name      | app_fd_ purchase_items |

# Exercise

---

- Add 3 text fields with the following details:-
  - ID: name, Label: Name
  - ID: quantity, Label: Quantity
  - ID: price, Label: Price

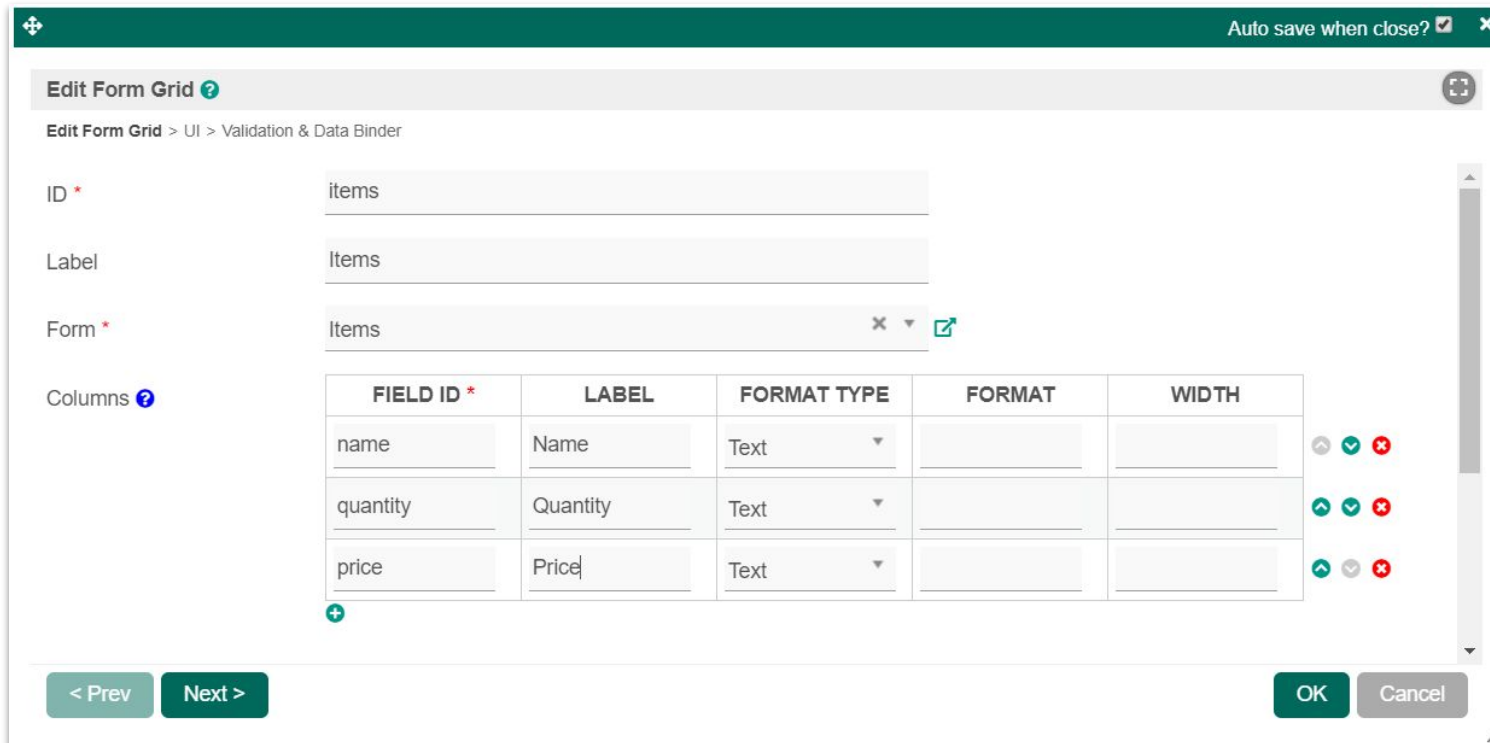


The screenshot shows a form titled "Items" with a header bar. Below the header, there are three text input fields. The first field is labeled "Name" and has a blue border. The second field is labeled "Quantity" and the third is labeled "Price". A dashed border surrounds the entire form area. The text "Drag This Column" is visible in the top right corner of the form area.

- Save the form

# Exercise

- Edit the “1-Submit Request” form.
- Add a “Form Grid” wherever relevant.
- Configure accordingly.



Auto save when close?

Edit Form Grid ?

Edit Form Grid > UI > Validation & Data Binder

ID \*

Label

Form \*  ✕ 🔗

Columns ?

| FIELD ID *                            | LABEL                                 | FORMAT TYPE                       | FORMAT               | WIDTH                |
|---------------------------------------|---------------------------------------|-----------------------------------|----------------------|----------------------|
| <input type="text" value="name"/>     | <input type="text" value="Name"/>     | <input type="text" value="Text"/> | <input type="text"/> | <input type="text"/> |
| <input type="text" value="quantity"/> | <input type="text" value="Quantity"/> | <input type="text" value="Text"/> | <input type="text"/> | <input type="text"/> |
| <input type="text" value="price"/>    | <input type="text" value="Price"/>    | <input type="text" value="Text"/> | <input type="text"/> | <input type="text"/> |

⬅ ✓ ✖

⬆ ✓ ✖

⬆ ⬇ ✖

+

< Prev Next > OK Cancel




# Exercise

- This is how your form design should look like.

**Request Details**


*Drag This Column*

Name

Request Date \*  

Category  ▼

**Items**

| Name  | Quantity | Price |
|---|----------|-------|
|  |          |       |

Remarks

# Exercise

- This is how the form should look like in runtime.

**PURCHASE REQUEST PROCESS - SUBMIT REQUEST**


**Request Details**

Name Admin Admin

Request Date \* MM/DD/YYYY

Category Stationery

**Items**

| Name  | Quantity |
|---|----------|
|  |          |

Remarks

**Submit Request Modal**

**Items**

Name \_\_\_\_\_

Quantity \_\_\_\_\_

Price \_\_\_\_\_

**Submit**

**Save As Draft** **Complete**

# Exercise

---

- Run a new “Submit New Request” process, and submit the form to observe.

# Materials

---

- "1-Submit Request" form definition can be obtained from **13.3.1.txt**
- "Items" form definition can be obtained from **13.3.2.txt**

# Chapter Review

---

- Being able to use the Form Grid element.
- PS: Check out Advanced Grid  
(<http://dev.joget.org/community/display/DX7/Advanced+Grid>)  
Form Element that performs similarly as Form Grid.



# Chapter 4

## Multirow Form Binder

# Multirow Form Binder

---

- Multirow Form Binder is a Store/Load Form Binder that is designed to treat multi-row data for grid form element.
- Rather than storing as traditional JSON data format in a single column cell, the Multirow Form Binder saves the data into its respective tables.
- This would make data retrieval easier for sorting, statistics, and indexing/performance purpose.
- Reference:  
<http://dev.joget.org/community/display/DX7/Multirow+Form+Binder>

# Exercise

---

- Continue to use the application from the previous chapter OR import app from **13.4.1.jwa**.
- Edit the “Items” form.
- Add a Hidden Field to the form.
- Configure accordingly.

### Edit Hidden Field

Edit Hidden Field > Advanced Options

|               |   |
|---------------|---|
| ID *          | <input type="text" value="request_id"/> |
| Default Value | <input type="text"/>                    |



# Exercise

---

- This is how your “Items” form should look like.

| Items      |  |
|------------|--|
|            | <i>Drag This Column</i>                        |
| request_id | <input type="text"/> <span>HIDDEN FIELD</span> |
| Name       | <input type="text"/>                           |
| Quantity   | <input type="text"/>                           |
| Price      | <input type="text"/>                           |

# Exercise

- Edit the "1-Submit Request" form.
- Configure the Form Grid element to utilize the Multirow Form Binder in Data Binder.

**Validation & Data Binder**

Edit Form Grid > UI > **Validation & Data Binder** > Load Binder (Multirow Form Binder) > Store Binder (Multirow Form Binder)

**Validation**

|  |                        |
|--|------------------------|
| Validator                              | <input type="text"/>   |
| Unique Column <a href="#">?</a>        | <input type="text"/>   |
| Min Number of Row Validation (Integer) | <input type="text"/>   |
| Max Number of Row Validation (Integer) | <input type="text"/>   |
| Error Message                          | Invalid number of rows |

**Data Binder**


|              |   |
|--------------|---|
| Load Binder  | Multirow Form Binder <input type="text"/> |
| Store Binder | Multirow Form Binder <input type="text"/> |

# Exercise

- Click next to configure the Binder.
- Configure accordingly.


**Configure Multirow Form Binder** ?

Edit Form Grid > UI > Validation & Data Binder > **Configure Multirow Form Binder** > Store Binder (Multirow Form Binder)

|               |            |     |   |
|---------------|------------|-----|---|
| Form *        | Items      | x ▾ |  |
| Foreign Key * | request_id | x ▾ |   |

**Configure Multirow Form Binder** ?

Edit Form Grid > UI > Validation & Data Binder > Load Binder (Multirow Form Binder) > **Configure Multirow Form Binder**

|               |            |     |   |
|---------------|------------|-----|---|
| Form *        | Items      | x ▾ |  |
| Foreign Key * | request_id | x ▾ |   |

# Exercise

---

- Run a new “Submit New Request” process, and submit the form to observe.

# Exercise - Optional

- Inspect the database table of “Items”, you will notice that rows of data is now being saved into this table rather than the parent table as JSON.

| jwdb.app_fd_purchase_requests: 1 rows total (approximately) |                     |                     |           |               |            |                |             |                |         |  |
|---|---------------------|---------------------|-----------|---------------|------------|----------------|-------------|----------------|---------|--|
| id  | dateCreated         | dateModified        | createdBy | createdByName | modifiedBy | modifiedByName | c_name      | c_request_date | c_items |  |
| a7380ecd-72eb-4eea-8560-a28aae74ba6e                        | 2019-12-27 15:03:43 | 2019-12-27 15:03:43 | admin     | Admin Admin   | admin      | Admin Admin    | Admin Admin | 12/27/2019     | (NULL)  |  |

| jwdb.app_fd_purchase_items: 1 rows total (approximately) |                     |                     |           |               |            |                |            |         |        |                                      |
|--|---------------------|---------------------|-----------|---------------|------------|----------------|------------|---------|--------|--------------------------------------|
| id   | dateCreated         | dateModified        | createdBy | createdByName | modifiedBy | modifiedByName | c_quantity | c_price | c_name | c_request_id                         |
| 832103ee-df79-41ac-969a-2cb2870fd872                     | 2019-12-27 15:03:43 | 2019-12-27 15:03:43 | admin     | Admin Admin   | admin      | Admin Admin    | 5          | 20      | Pencil | a7380ecd-72eb-4eea-8560-a28aae74ba6e |

# Materials

---

- "1-Submit Request" form definition is available at **13.4.2.txt**
- "Items" form definition is available at **13.4.3.txt**
- Complete app is available at **13.4.4.jwa**

# Chapter Review

---

- Understand the use case of the Multirow Form Binder and its benefits.



# Chapter 5

## List Grid



# List Grid

---

- **List Grid** is a grid table that populates its data from a Datalist.
- It behaves similarly like a Grid (Chapter 2) but new rows are added from a specified Datalist instead.
- It also behaves similarly like a Form Grid that allows one to open up a Form for editing.
- Reference: <http://dev.joget.org/community/display/DX7/List+Grid>

# Sample Use Case

**Leave Process - Submit Leave**

**Leave Application Details**

Name \* Admin Admin

Start Date \*

End Date \*

Reason \*

**Add Entry** [close]

10

| <input type="checkbox"/> | name  | contact_no |
|--------------------------|-------|------------|
| <input type="checkbox"/> | Julia | 123        |
| <input type="checkbox"/> | Jude  | 124        |

2 items found, displaying all items. 1

**Emergency Contacts**

| Contact Name         | Contact No           |
|----------------------|----------------------|
| <input type="text"/> | <input type="text"/> |

# Chapter Review

---

- Understand the List Grid element and be able to think of use cases of it.
- Able to differentiate Grid, Form Grid, and List Grid.



# Chapter 6

CRUD

# CRUD

---

- CRUD is a Userview Menu allows one to easily achieve the functionality of **C**reate, **R**etrieve, **U**ppdate, and **D**elate on a data entity.
- In short, manipulate records on a specified table.
  
- Reference: <http://dev.joget.org/community/display/DX7/CRUD>

# What Is Needed For CRUD To run?

---

- A Form entity
- A List of the same data entity as the form
- A Userview

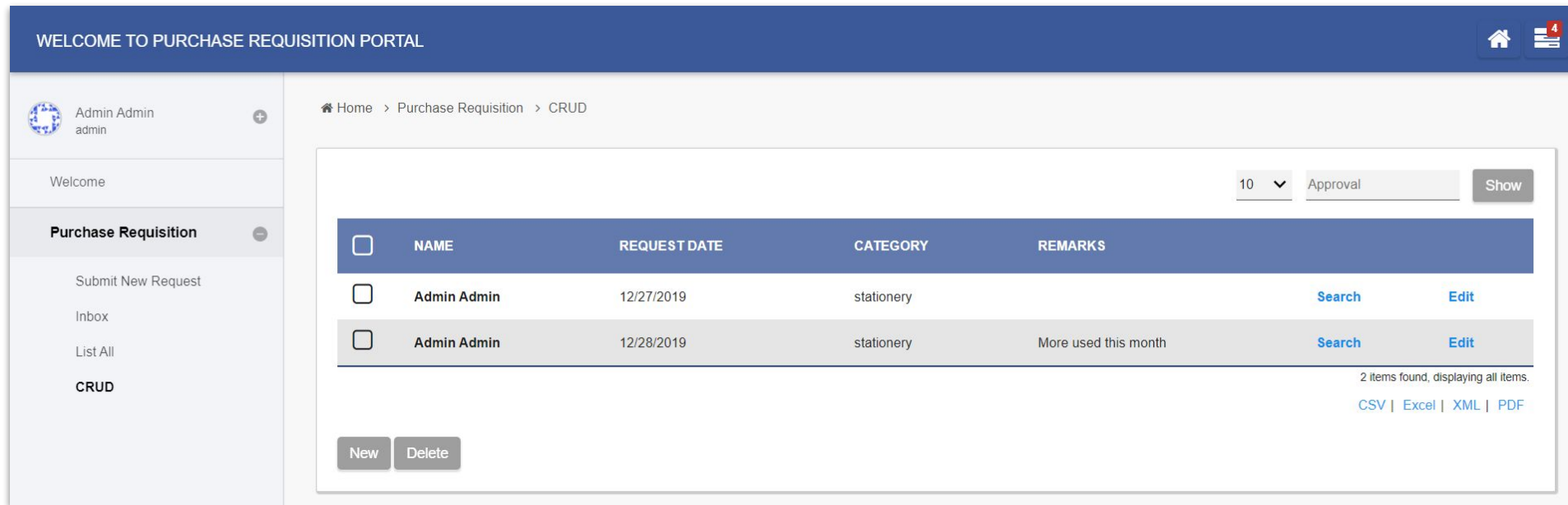
# Refresh

---

- You've already done it! Refresh what you did back in Module 8 - Designing your first Userview
- If you do not have the CRUD for "Request List", ask your colleague how to, or raise hand...

# How CRUD Looks Like...

- This is how the CRUD element would look like in runtime.



The screenshot displays a web application interface for a purchase requisition portal. The header includes a welcome message and navigation icons. The left sidebar contains a user profile for 'Admin Admin' and a menu with options like 'Submit New Request', 'Inbox', 'List All', and 'CRUD'. The main content area shows a breadcrumb trail 'Home > Purchase Requisition > CRUD' and a table of requisitions. The table has columns for 'NAME', 'REQUEST DATE', 'CATEGORY', and 'REMARKS'. Two rows are visible, both for 'Admin Admin' with dates 12/27/2019 and 12/28/2019, and category 'stationery'. The second row has the remark 'More used this month'. Each row has 'Search' and 'Edit' links. Below the table are 'New' and 'Delete' buttons. A summary at the bottom right indicates '2 items found, displaying all items.' with links for 'CSV', 'Excel', 'XML', and 'PDF'.

| <input type="checkbox"/> | NAME        | REQUEST DATE | CATEGORY   | REMARKS              |                        |                      |
|--------------------------|-------------|--------------|------------|----------------------|------------------------|----------------------|
| <input type="checkbox"/> | Admin Admin | 12/27/2019   | stationery |                      | <a href="#">Search</a> | <a href="#">Edit</a> |
| <input type="checkbox"/> | Admin Admin | 12/28/2019   | stationery | More used this month | <a href="#">Search</a> | <a href="#">Edit</a> |

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



# Chapter Review

---

- Able to use CRUD and understand the linkages.



# Chapter 7

## Custom HTML

# Custom HTML

---

**Custom HTML** in Form Builder can be used to achieve advanced form design by putting in any valid -

- **HTML**

E.g: `<b>this text is in bold</b>`

- **JavaScript** (jQuery is supported)

Remember to put in `<script type="text/javascript"></script>` block

- **CSS**

Don't forget to put in `<style type="text/css"></style>` block

Reference:

<http://dev.joget.org/community/display/DX7/Custom+HTML>

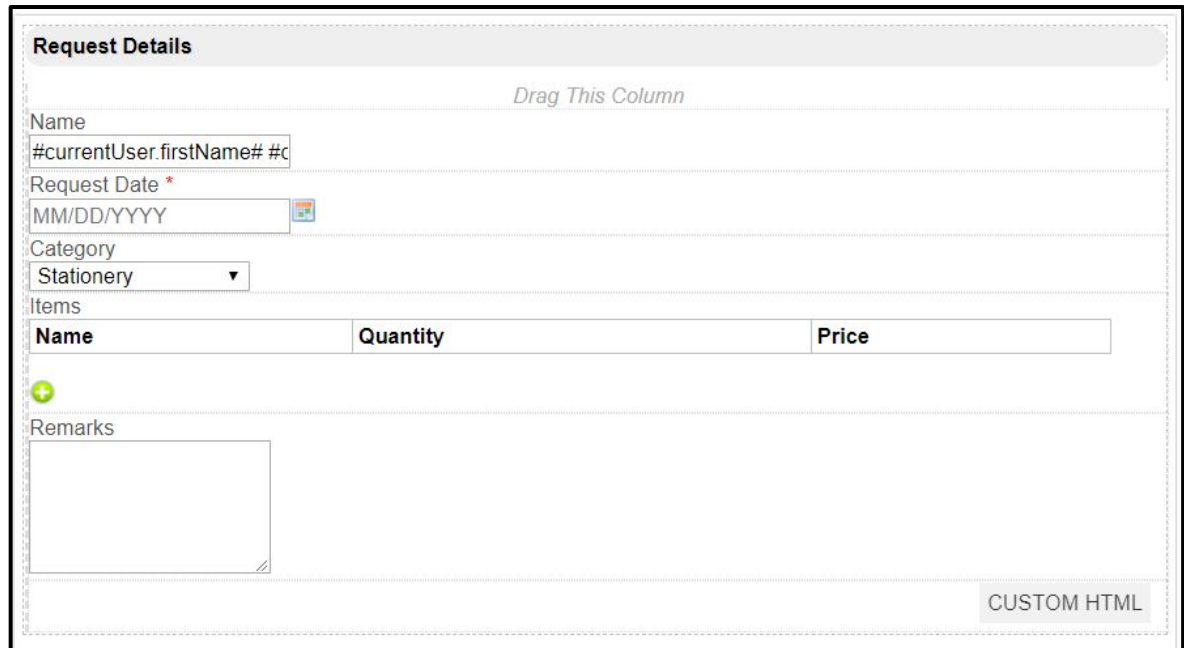
# Exercise - CSS (Optional)

- Customize the look and feel of how the form is rendered by modifying its CSS.
  - Add a **Custom HTML** form element into the bottom of the form.
  - Edit it, add the following code into **Custom HTML** property.

```
<style type="text/css">

.form-cell .label,
.subform-cell .label{
    width: 100%;
}

</style>
```



**Request Details**

*Drag This Column*

Name  
#currentUser.firstName# #c

Request Date \*  
MM/DD/YYYY

Category  
Stationery

Items

| Name | Quantity | Price |
|------|----------|-------|
| +    |          |       |

Remarks

CUSTOM HTML

# Exercise - Tooltip

- Make use of Advanced Tool's Tooltip to add hints into existing form "Submit Request" using Custom HTML also.

**Request Details**

Name i Key in your full name

Request Date \*  📅

Category  ▼

Items

| Name  | Quantity |
|---|----------|
| <span style="color: green; font-weight: bold; font-size: 1.2em;">+</span> |          |

Remarks

# Exercise - Color Picker (Optional)

- Create a text field “Color Code” with ID “color\_code”
- Create a Custom HTML. Make use of color picker library to turn text field into a color picker.



# Exercise - Color Picker - Materials

---

Do import these JS and CSS libraries into the app's resource from the materials folder:

- `colorPick.min.css`
- `colorPick.min.js`

# Exercise - Color Picker

---

```
<script src="#appResource.colorPick.min.js#"></script>
<link rel="stylesheet" href="#appResource.colorPick.min.css#">

<script type="text/javascript">
  $(function(){
    initialColor = FormUtil.getField("color_code").val();
    //console.log("initial " + initialColor);
    FormUtil.getField("color_code").colorPick({
      'initialColor' : initialColor,
      'onColorSelected': function() {
        //console.log("The user has selected the color: " + this.color);
        FormUtil.getField("color_code").val(this.color);
        this.element.css({'backgroundColor': this.color, 'color': this.color});
      }
    });
  });
</script>
```





# Chapter 8

## Using Advanced Tool's Permission

# Permission Control

- The Permission in Advanced Tools allows fine grain control up to field level.

**ADVANCED TOOLS**

Tree Viewer | **Permission** | Usages | Table | 118N | Tooltip | Diff Checker | JSON Definition

[+ Add Permission](#)

**Acknowledge Approval Activity**  
Bean Shell Script

**Submit Request Activity**  
Bean Shell Script

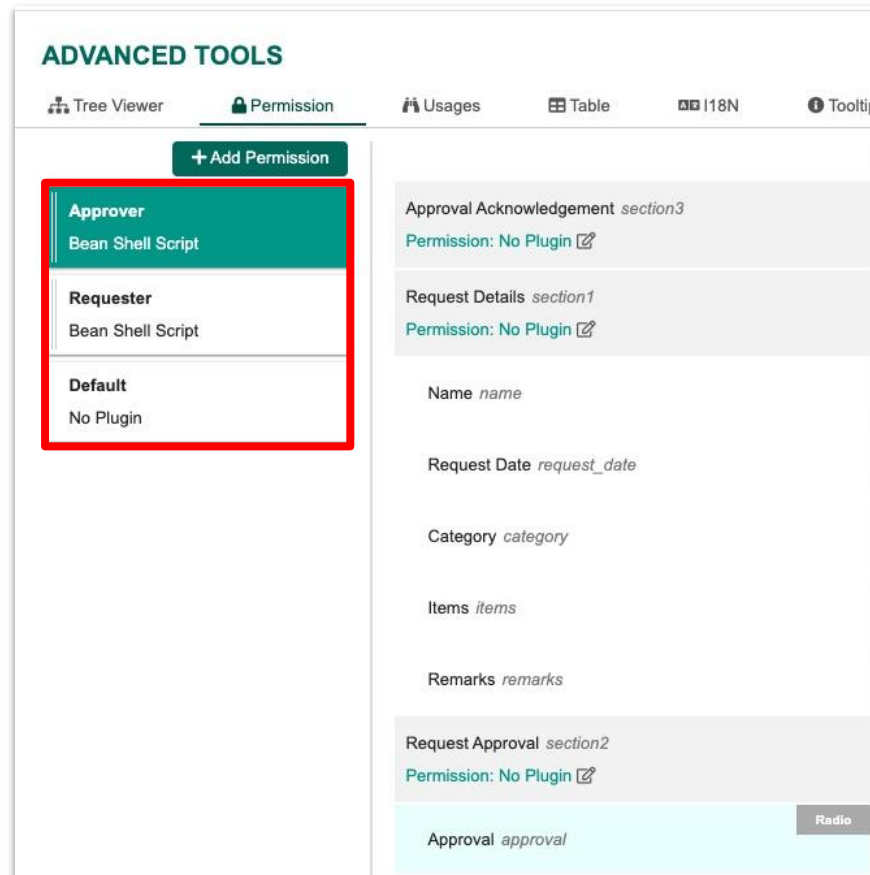
**Approval Activity**  
Bean Shell Script

**Default**  
No Plugin

|   | AUTHORIZED     |                 |               | UNAUTHORIZED |        |
|---|----------------|-----------------|---------------|--------------|--------|
| Approval Acknowledgement <i>section3</i><br>Permission: No Plugin <a href="#">🔗</a> | VISIBLE        | READONLY        | <b>HIDDEN</b> | READONLY     | HIDDEN |
| Request Details <i>section1</i><br>Permission: No Plugin <a href="#">🔗</a>          | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY     | HIDDEN |
| Name <i>name</i>  | VISIBLE        | <b>READONLY</b> | HIDDEN        | READONLY     | HIDDEN |
| Request Date <i>request_date</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY     | HIDDEN |
| Category <i>category</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY     | HIDDEN |
| Items <i>items</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY     | HIDDEN |
| Remarks <i>remarks</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY     | HIDDEN |
| Request Approval <i>section2</i><br>Permission: No Plugin <a href="#">🔗</a>         | VISIBLE        | READONLY        | <b>HIDDEN</b> | READONLY     | HIDDEN |
| Approval <i>approval</i>  | VISIBLE        | READONLY        | HIDDEN        | READONLY     | HIDDEN |

# Permission Control

- Typically, one would control **based on User Role** most of the time to show/hide fields.



The screenshot shows the 'ADVANCED TOOLS' interface with the 'Permission' tab selected. On the left, a table lists permissions for different roles, with the 'Approver' role highlighted in a red box:

| Role      | Permission        |
|-----------|-------------------|
| Approver  | Bean Shell Script |
| Requester | Bean Shell Script |
| Default   | No Plugin         |

On the right, a form displays various fields and sections:

- Approval Acknowledgement *section3*  
Permission: No Plugin [🔗](#)
- Request Details *section1*  
Permission: No Plugin [🔗](#)
- Name *name*
- Request Date *request\_date*
- Category *category*
- Items *items*
- Remarks *remarks*
- Request Approval *section2*  
Permission: No Plugin [🔗](#)
- Approval *approval*

# Permission Control

- If the form is used part of a process flow, it is also possible to exert permission **based on activity**, rather than user role.

**ADVANCED TOOLS**

Tree Viewer | **Permission** | Usages | Table | I18N | Tooltip | Diff Checker | JSON Definition

[+ Add Permission](#)

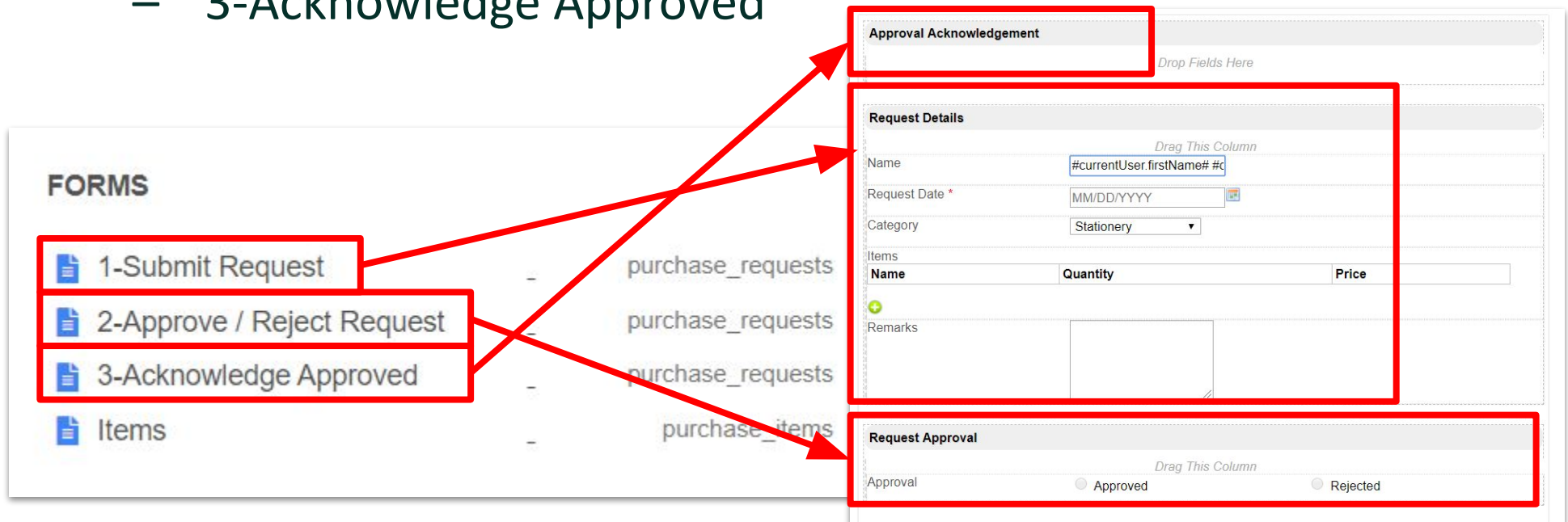
|  | AUTHORIZED     |                 |               | UNAUTHORIZED |        |
|--|----------------|-----------------|---------------|--------------|--------|
| <b>Acknowledge Approval</b><br>Activity<br>Bean Shell Script | VISIBLE        | READONLY        | <b>HIDDEN</b> | READONLY     | HIDDEN |
| <b>Submit Request Activity</b><br>Bean Shell Script          | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY     | HIDDEN |
| <b>Approval Activity</b><br>Bean Shell Script                | VISIBLE        | <b>READONLY</b> | HIDDEN        | READONLY     | HIDDEN |
| <b>Default</b><br>No Plugin                                  | VISIBLE        | READONLY        | HIDDEN        | READONLY     | HIDDEN |

|   |                |                 |               |          |        |
|---|----------------|-----------------|---------------|----------|--------|
| Approval Acknowledgement <i>section3</i><br>Permission: No Plugin <a href="#">🔗</a> | VISIBLE        | READONLY        | <b>HIDDEN</b> | READONLY | HIDDEN |
| Request Details <i>section1</i><br>Permission: No Plugin <a href="#">🔗</a>          | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY | HIDDEN |
| Name <i>name</i>  | VISIBLE        | <b>READONLY</b> | HIDDEN        | READONLY | HIDDEN |
| Request Date <i>request_date</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY | HIDDEN |
| Category <i>category</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY | HIDDEN |
| Items <i>items</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY | HIDDEN |
| Remarks <i>remarks</i>  | <b>VISIBLE</b> | READONLY        | HIDDEN        | READONLY | HIDDEN |
| Request Approval <i>section2</i><br>Permission: No Plugin <a href="#">🔗</a>         | VISIBLE        | READONLY        | <b>HIDDEN</b> | READONLY | HIDDEN |
| Approval <i>approval</i>  | VISIBLE        | READONLY        | HIDDEN        | READONLY | HIDDEN |

# Exercise - Advanced Tool's Permission

- *One Master Form Concept* - Try combining the 3 forms in Purchase Requisition into a single form by using the Form Builder's Advance Tool Permission Tab:
  - 1-Submit Request
  - 2-Approve / Reject Request
  - 3-Acknowledge Approved



The screenshot shows a form builder interface. On the left, a 'FORMS' list contains four items: '1-Submit Request', '2-Approve / Reject Request', '3-Acknowledge Approved', and 'Items'. Each item is associated with a table name: 'purchase\_requests' for the first three and 'purchase\_items' for the last. On the right, a form titled 'Request Details' is shown. It has several sections: 'Approval Acknowledgement', 'Request Details', and 'Request Approval'. Red boxes highlight these sections, and red arrows point from the 'FORMS' list to them: '1-Submit Request' points to 'Request Details', '2-Approve / Reject Request' points to 'Request Approval', and '3-Acknowledge Approved' points to 'Approval Acknowledgement'.

**FORMS**

- 1-Submit Request - purchase\_requests
- 2-Approve / Reject Request - purchase\_requests
- 3-Acknowledge Approved - purchase\_requests
- Items - purchase\_items

**Request Details**

Name: #currentUser.firstName# #c

Request Date \*: MM/DD/YYYY

Category: Stationery

Items

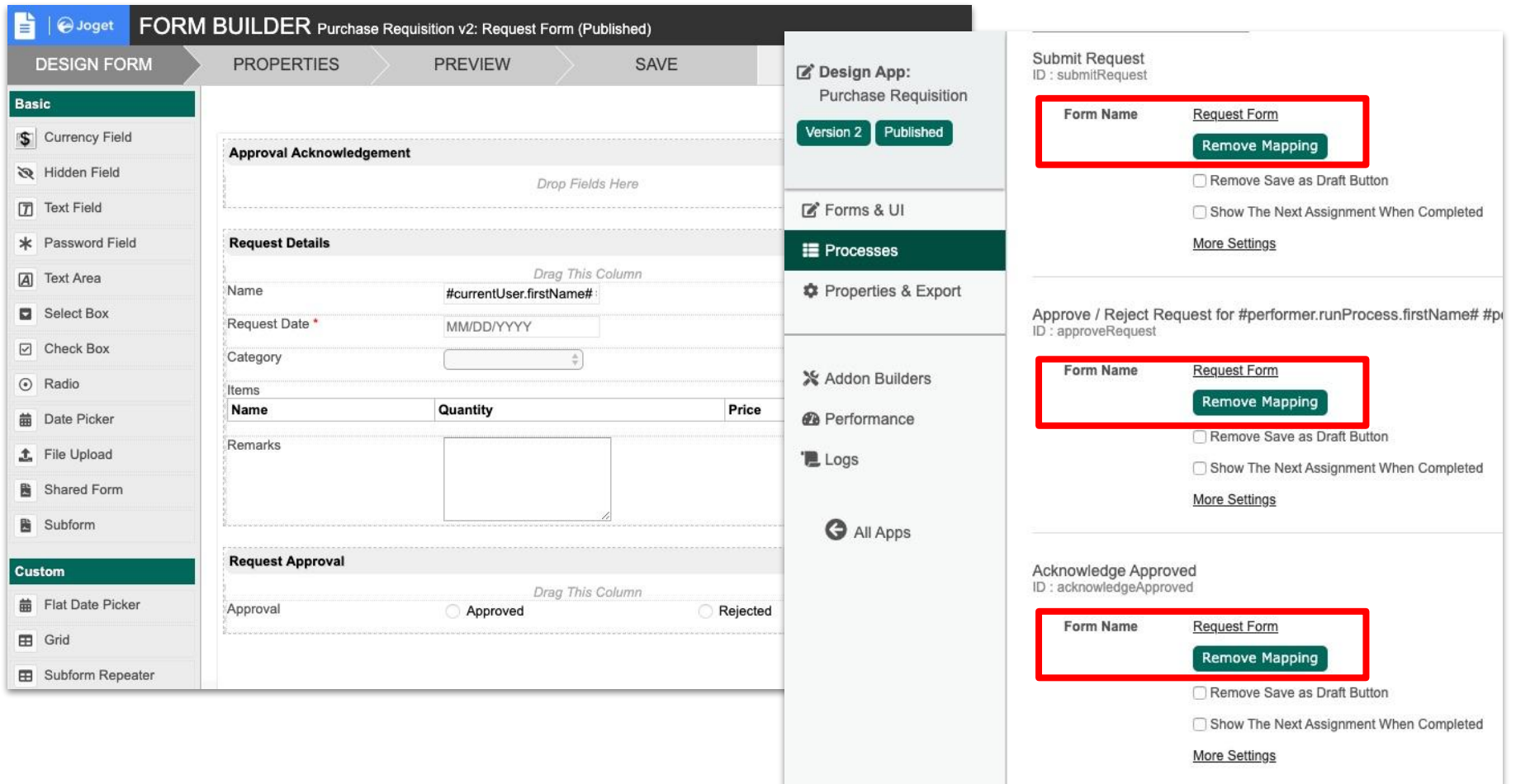
| Name    | Quantity | Price |
|---------|----------|-------|
| Remarks |          |       |

**Request Approval**

Approval:  Approved  Rejected

# Exercise - Advanced Tool's Permission

- The same form will be used in all activities.



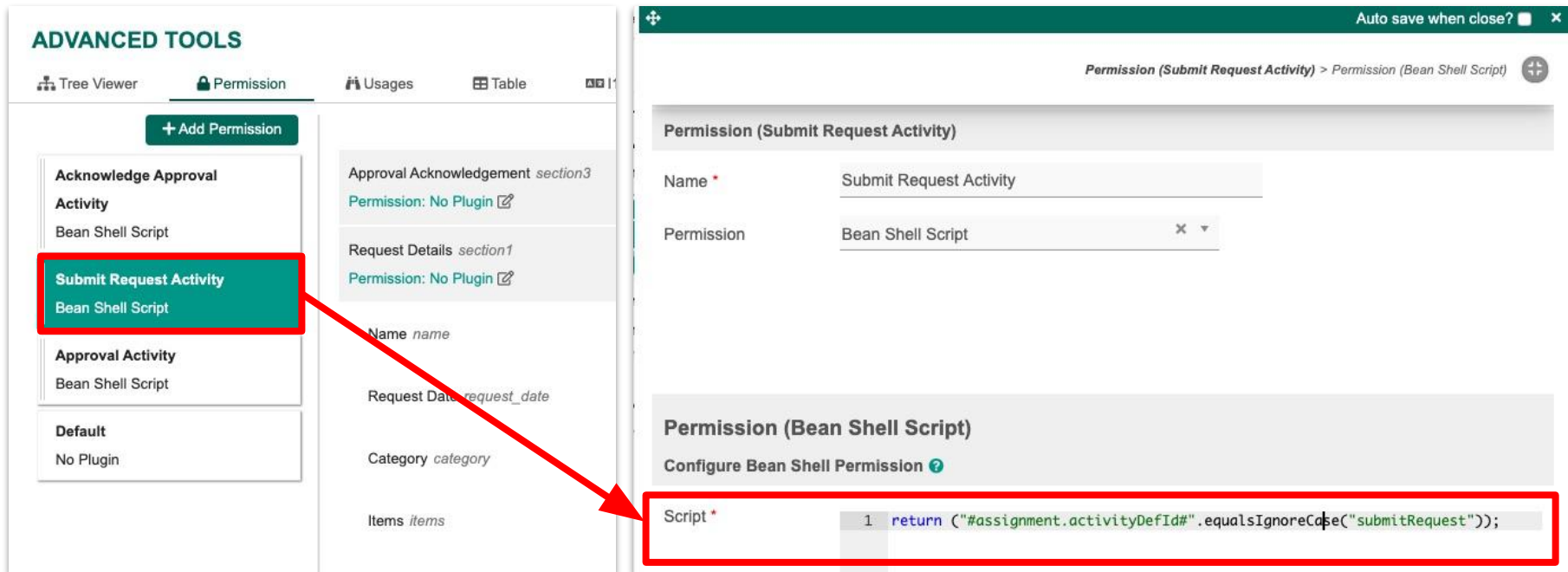
The screenshot displays the Joget Form Builder interface for a form titled "Purchase Requisition v2: Request Form (Published)". The interface includes a left sidebar with field types (Basic and Custom), a central design canvas, and a right sidebar with activity settings. The design canvas is divided into sections: "Approval Acknowledgement", "Request Details", and "Request Approval". The "Request Details" section contains fields for Name, Request Date, Category, and a table for Items. The "Request Approval" section has radio buttons for "Approved" and "Rejected".

On the right, three activity settings panels are shown, each with a red box highlighting the "Form Name" field set to "Request Form" and a "Remove Mapping" button:

- Submit Request** (ID: submitRequest)
- Approve / Reject Request for #performer.runProcess.firstName# #p** (ID: approveRequest)
- Acknowledge Approved** (ID: acknowledgeApproved)

# Exercise - Advanced Tool's Permission

- In the Permission tab, we can control on what to show based on current activity.



The screenshot shows the 'ADVANCED TOOLS' interface with the 'Permission' tab selected. On the left, a list of activities is shown, with 'Submit Request Activity' highlighted in a red box. A red arrow points from this box to the right-hand configuration window. The configuration window shows the 'Permission (Submit Request Activity)' settings, where the 'Name' is 'Submit Request Activity' and the 'Permission' is set to 'Bean Shell Script'. Below this, the 'Permission (Bean Shell Script)' configuration is shown, with a red box highlighting the 'Script' field containing the following code:

```
1 return ("#assignment.activityDefId#".equalsIgnoreCase("submitRequest"));
```

```
return ("#assignment.activityDefId#".equalsIgnoreCase("submitRequest"));
```

# Exercise - Advanced Tool's Permission

---

- Once you are done setting up, try to run through the whole flow and check if the form is showing up correctly as per the activity.



# Materials

---

- Complete app is available at **13.8.1.jwa**

# Module Review

---

1. Introduction
2. Grid
3. Form Grid
4. Multirow Form Binder
5. List Grid
6. CRUD
7. Custom HTML
8. Using Advanced Tool's Permission

# Learn More...

---

- Check out <https://dev.joget.org/community/display/DX7/Form+Builder> for the list of **Form** related elements (Form Element, Form Validator, Form Binder, Form Options Binder).
- Check out <https://dev.joget.org/community/display/DX7/Userview+Builder> for the list of **Userview** related elements.

# Stay Connected With Joget

---

- [www.joget.org](http://www.joget.org)
- [community.joget.org](http://community.joget.org)
- [twitter.com/jogetworkflow](https://twitter.com/jogetworkflow)
- [facebook.com/jogetworkflow](https://facebook.com/jogetworkflow)
- [youtube.com/jogetworkflow](https://youtube.com/jogetworkflow)