

## Configure parameters using the ‘Helper’ Button (Customization.XML)

### *Making migration to Java uncomplicated for the Oracle Forms Developer*

Migrating Oracle Forms is a task normally envisioned as something that the Java developer has to perform. We wanted to design a process where specifics of the Forms application can be easily added into the new Java application by the Forms developer who understand the application and its functionality better.

The customization of **TRANSFORM** process using the ‘Helper’ button available on the **TRANSFORM** UI, or creation of the “Customization.XML” was designed to deal with exactly this.

### **Customization is performed after the “1st pass” of the migration.**

Using Customization.XML (“Helper” button on the TRANSFORM migration page) allows Forms developers to apply parametric behavior, event based triggers and refresh dependencies while using a GUI interface rather than do the coding in Java. Accordingly, the Forms developer may wish to make changes in the Forms module and migrate the new structure across to Java.

The idea is to make and/or apply forms specific behavior in the application by using Forms Builder and TRANSFORM customization processes, instead of making changes manually to the Java code.

In essence Forms developer can now use the already familiar 4GL Forms builder tool to make changes and then use TRANSFORM’s powerful structure building to incorporate those behaviors into Java, rather than trying to code them in Java later. This is possible using the customization.XML file.

Customization.XML file currently manages the behaviors for Block & LOVs. And can be easily extended to incorporate additional such behaviors.

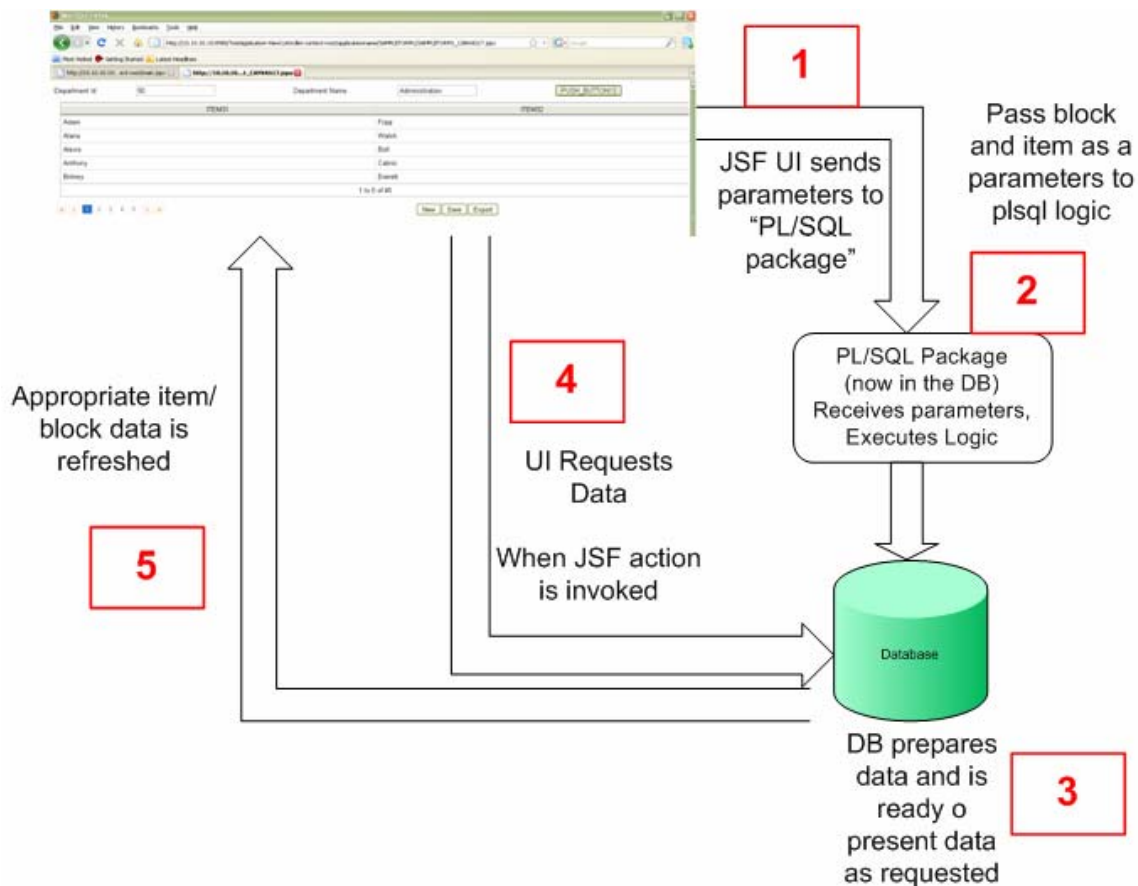
Benefit to the Forms Developer: This is beneficial since now, the Forms developer can ideally deal with a basic training on Java-JSF to migrate the application. Over a period of time, the developer can enhance his/her knowledge of Java-JSF rather than being forced to learn first and then perform migration.

### **Understanding the Customization.XML**

Before you start editing or inputting the customization.XML, it is important to understand the process flow, so do take a few minutes to understand this diagram. If you have

questions, please contact us ([info@release3.com](mailto:info@release3.com)), and we'll be glad to walk you through it.

We've also included simple examples with the installation kit, so you could try customization step-by-step yourself and understand the process fully to be able to leverage this simple to use yet powerful process.

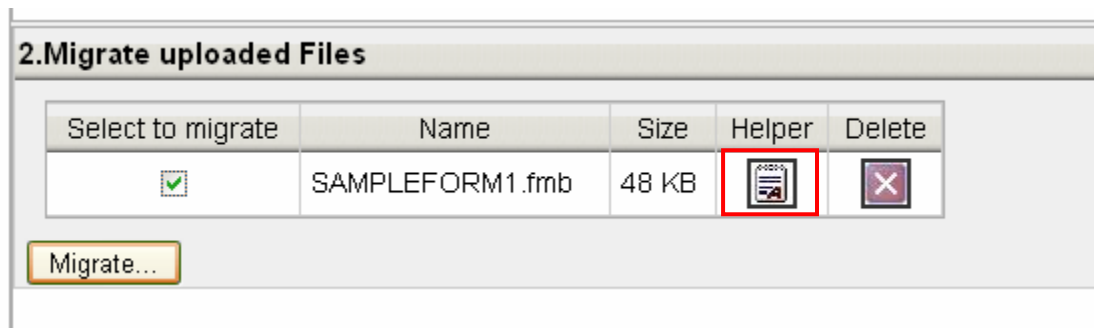


## Modifying Customization.XML

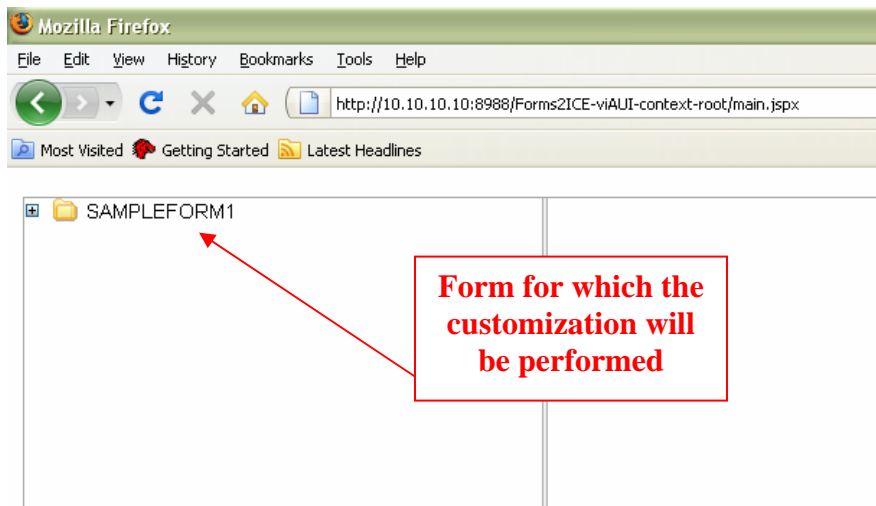
Modifying customization.XML is a task made very easy with this UI provided with TRANSFORM. Once you understand how to make changes and how TRANSFORM picks up these customizations, you will be able to quickly create your own customization files to reduce time required for migration.

- This tutorial uses the '**SAMPLEFORM1.fmb**' (provided along with the toolkit).
- The toolkit already contains the licenses required to migrate this form.
- Before moving to the next step, please perform Pass 1 of migration

1) Click on 'Helper' button



2) A window will be launched which uses a GUI interface to prepare the Customization.XML. This window consists of the Oracle Forms objects



### 3) Expand Structure Navigator

When you expand the Form name on the left side of the screen-panel in this window, you will see the structure of the Form, similar to what you see in Forms Builder

**Customization.XML**

JSF events: actionListener | PL/SQL procedure

Parameters: Refresh blocks

Block: 0 to 0 of 0

Buttons: New, Save, Export

**Object Navigator:**

- Forms
  - SAMPLEFORM1
    - Triggers
    - Alerts
    - Attached Libraries
    - Data Blocks
      - NODDBLOCK
        - Triggers
        - Items
          - ITEM31
            - Triggers
          - ITEM32
            - Triggers
        - Relations
          - DEPARTMENTS
            - Items
              - DEPARTMENT\_ID
              - DEPARTMENT\_NAME
            - Relations
              - BLOCK12
                - Triggers
                  - WHEN-BUTTON-PRESSED
                - Items
                  - BUTTON
          - Canvases
            - CANVAS17
          - Editors
          - LOVs
          - Object Groups
          - Parameters
          - Popup Menus

**Yellow Box Text:**

For Customization, you need to select Block or Item for which the triggers need to be mapped to events

In our example, We are going to change Trigger : “WHEN-BUTTON-PRESSED”

Action: Select “BLOCK”>”BUTTON”

**4) Look at the PL/SQL package file to determine what parameters need to be passed to the DB**

```

create or replace
PACKAGE BODY SAMPLEFORM1 AS
PROCEDURE itm_BLOCK12_BUTTON( sessionId NUMBER, a number) IS
BEGIN
delete from NONDBLOCK where SESSION_ID = sessionId and form_id = 'SAMPLEFORM1' and BLOCK_ID='NODDBLOCK';
insert into NONDBLOCK(REC_ID,SESSION_ID,BLOCK_ID,FORM_ID,FIELD1,FIELD2)
select REC_ID_SEQ.NEXTVAL,sessionId,'NODDBLOCK','SAMPLEFORM1', FIRST_NAME, LAST_NAME
from EMPLOYEES
WHERE DEPARTMENT_ID = a;
END;
END SAMPLEFORM1;
  
```

**We determine from the PL/SQL statement that**

- Department\_ID is being passed as parameter
- The related block is: 'Department'
- Related item is: 'DEPARTMENT\_ID'

## 5) Bind JSF and PL/SQL Package

On the corresponding **right Panel**, from Drop Down, click and select desired JSF event. These events match triggers in the Forms environments to the ones available in JSF.

**Object Navigator**

- Forms
  - SAMPLEFORM1
    - Triggers
    - Alerts
    - Attached Libraries
    - Data Blocks
      - NODBBLOCK
        - Triggers
        - Items
          - ITEM31
            - Triggers
          - ITEM32
            - Triggers
        - Relations
      - DEPARTMENTS
        - Triggers
          - WHEN-BUTTON-PRESSED
        - Items
          - DEPARTMENT\_ID
          - DEPARTMENT\_NAME
        - Relations
      - BLOCK12
        - Triggers
        - Items
          - BUTTON
            - Triggers
        - Relations
    - Canvases
      - CANVAS17
      - Editors
      - LOVs
      - Object Groups
      - Parameters
      - Popup Menu

**Sample Form Window**

Departments

|               |                 |             |
|---------------|-----------------|-------------|
| Department Id | Department Name | PUSH_BUTTON |
| 50            | Shipping        |             |

Mozhe Atkinson  
 Sarah Bell  
 Laura Bissot  
 Alexis Bull  
 Anthony Cabrio

**WHEN-BUTTON-PRESSED** Trigger is now a PL/Sql procedure in the DB renamed to "itm\_BLOCK12\_BUTTON"

Button maps to JSF event

This procedure name entered here should be the same as the name in the DB package

**Properties**

JSF events:  PL/SQL procedure:

Parameters: 0 to 0 of 0

Refresh blocks: 0 to 0 of 0

Buttons: New, Save, Export

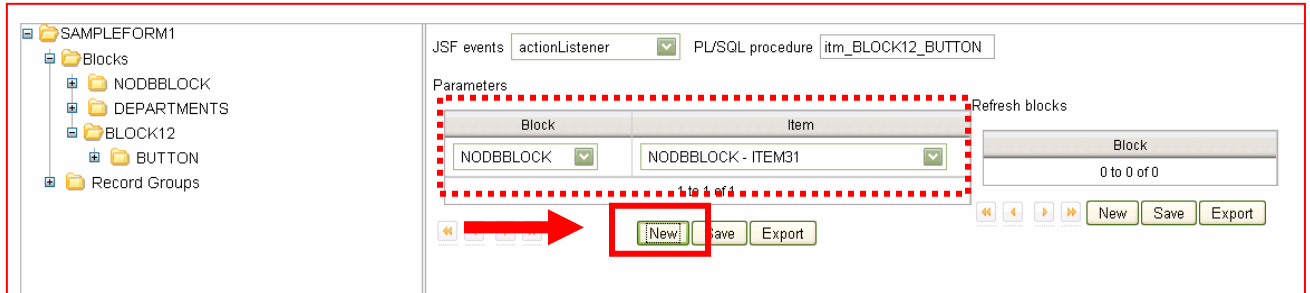
*The number of available events in JSF are far less than JSF, and so there may be many events that may not have an equivalent JSF behavior. In such a case, the user will have to choose the nearest event that matches the one in Forms Behavior*

## 6) Choose block and items to be passed as parameters

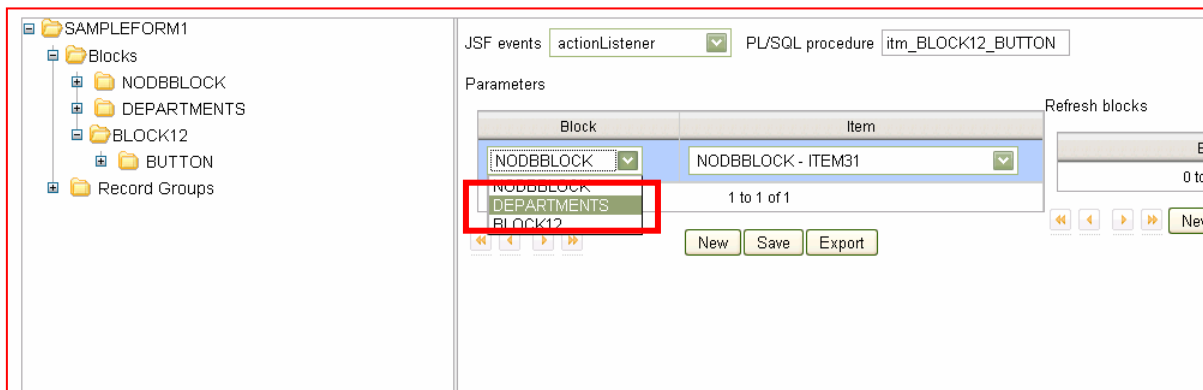
The parameters depend upon the logic in the PL/SQL now in the DB Package.

Since we saw in the package that have to pass **Block: Departments and Item:Department\_ID** as **parameters**, we will need to select those here.

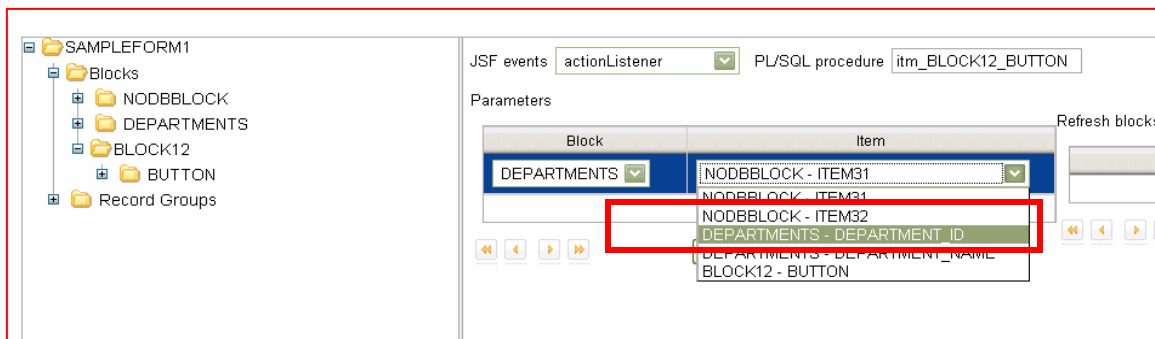
7) Click 'NEW' under Parameters, will provide you with the first set of parameters.



8) Click on the drop\_down list under Block, and select Block parameter = 'DEPARTMENTS'

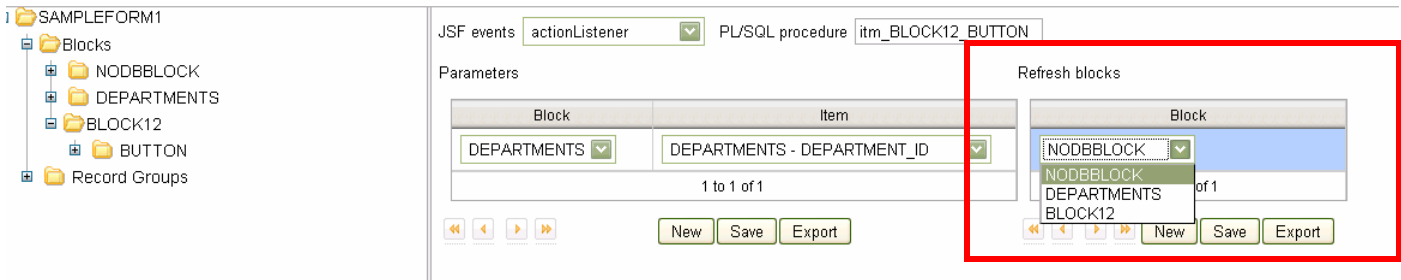


9) Click on the drop\_down list under Item, and select ITEM parameter = 'DEPARTMENT\_ID'

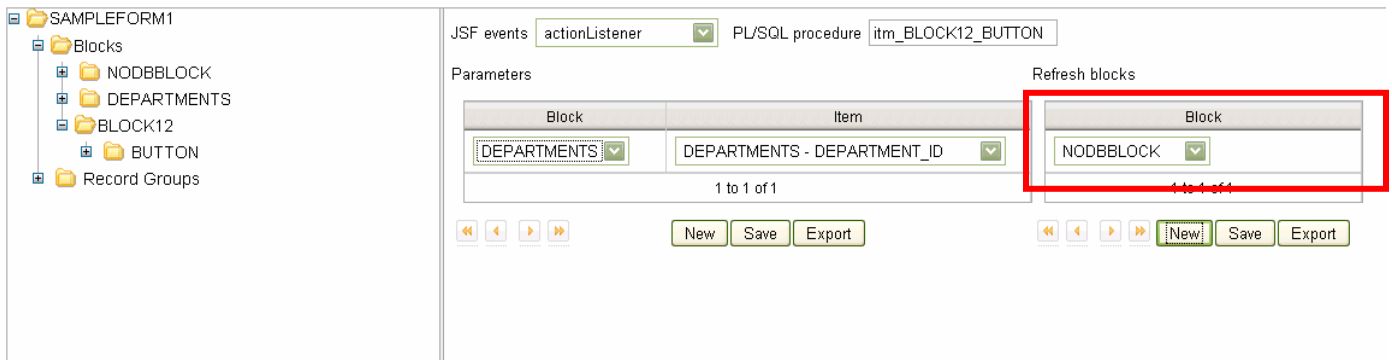


10) Select the resultant data-block or item that will be refreshed when the PL/SQL receives parameters.  
In our case the block non\_DB\_Block needs to be refreshed.

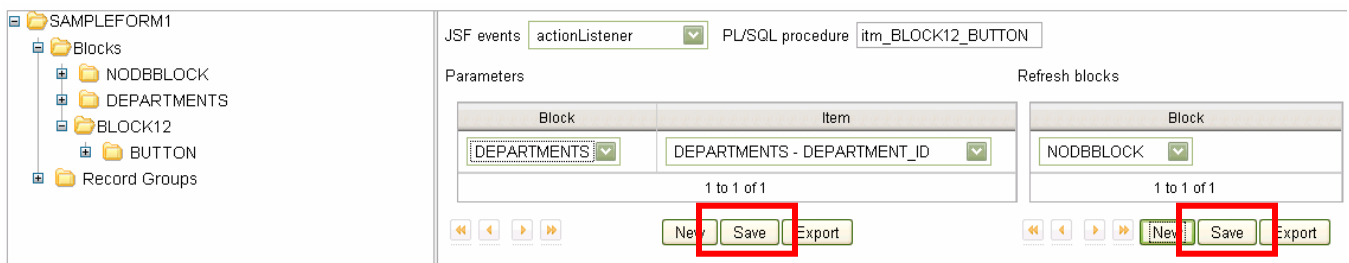
Click on the drop down list under section: “Refresh Blocks”



Select NODDBLOCK, as desired (or the block that you wish to refresh with the data from the database).



12) Click on “SAVE”



Click on “CLOSE”

The resultant Customization file will look like this

```
<?xml version='1.0' ?> <R3>
  <Trigger Block="BLOCK12" Item="BUTTON" Jsfattr="actionListener" plsqli="itm_BLOCK12_BUTTON">
    <Parameter Block="DEPARTMENTS" Item="DEPARTMENT_ID" Type="in"/>
  <Refresh Block="NODDBLOCK"/>
</Trigger>
</R3>
```

### **Note for Advanced Users:**

## **Creating your ‘customization.xml’ file manually**

As you gain experience, you may create your customization.XML yourself and place it in the appropriate folder. TRANSFORM will then automatically pick up the customization file during migration without the need to perform the iterative PASS01 and PASS02 of migration.

*In case of any clarifications or questions, please contact:*

**Release3 Inc.**

**[info@release3.com](mailto:info@release3.com)**

**[www.release3.com](http://www.release3.com)**