

How to Create, Enhance and Adapt Floorplan Manager Applications on SAP NetWeaver 7.0 EhP2



Applies to:

SAP NetWeaver 7.0 EhP2, SAP Business Suite 7 i2010, SAP ECC 6.0 EhP5

Summary

Since Business Suite 7 Innovations 2010 most of the new UIs provided by SAP are built using FPM (Floorplan Manager for Web Dynpro ABAP). These UIs are very flexible and easy to adapt to the needs of the individual customers. This document presents an overview about the different options that FPM provides for customer adaptations (configuration, customizing, personalization) and discusses their advantages and drawbacks.

Author: Christian Guenther, Simon Hoeg, SAP TIP Core UI Development

Company: SAP AG

Created on: 17 July 2011

Author Bio



Christian Günther is working as development architect for the Floorplan Manager since the project started. Before he was part of several application development projects in the Financials area (CO Reporting, Manager Self-Service, Express Planning).



Simon Hoeg joined SAP in year 2002 and contributes as software developer to the Floorplan Manager since several years. He was involved in various projects before, such as: Manager Self Service, Corporate Performance Management and Internal Service Request (Adobe Interactive Forms). The current focus of his work is on the development of the Floorplan Manager Design Time.

Table of Contents

Important SAP Notes (corrections) that should be applied via SNOTE up to SAP NetWeaver EhP2 SP7	3
Floorplan Manager for Web Dynpro ABAP.....	3
Component Configurations	5
A Detailed Look at the Structure of an FPM-Based Application	6
Which Objects have to be Adapted?.....	9
Options for Adaptations - Overview	10
Copy Application	10
Enhancement	10
Customizing	10
Modification	11
Options for Adaptations – Details	12
Copy Application	12
Starting the Application	13
Adapting the Header	13
Adapting the Floorplan Configuration.....	14
Adapting the Form	14
Customizing	15
Adapting the Floorplan Configuration.....	15
Adapting the Header	15
Adapting the Form	16
Enhancement	16
Adaptation – Best Practices.....	16
Related Content.....	18
Copyright.....	19

Important SAP Notes (corrections) that should be applied via SNOTE up to SAP NetWeaver EhP2 SP7

- [1561807](#): FPM Configuration Editor: Page header text for customizing
- [1563020](#): In place Navigation to GUIBBs does not work
- [1566584](#): WDA Changes in configuration at runtime
- [1574297](#): Creation of UIBBs on customizing level

Floorplan Manager for Web Dynpro ABAP

The Floorplan Manager (FPM) is a framework based on Web Dynpro for ABAP.

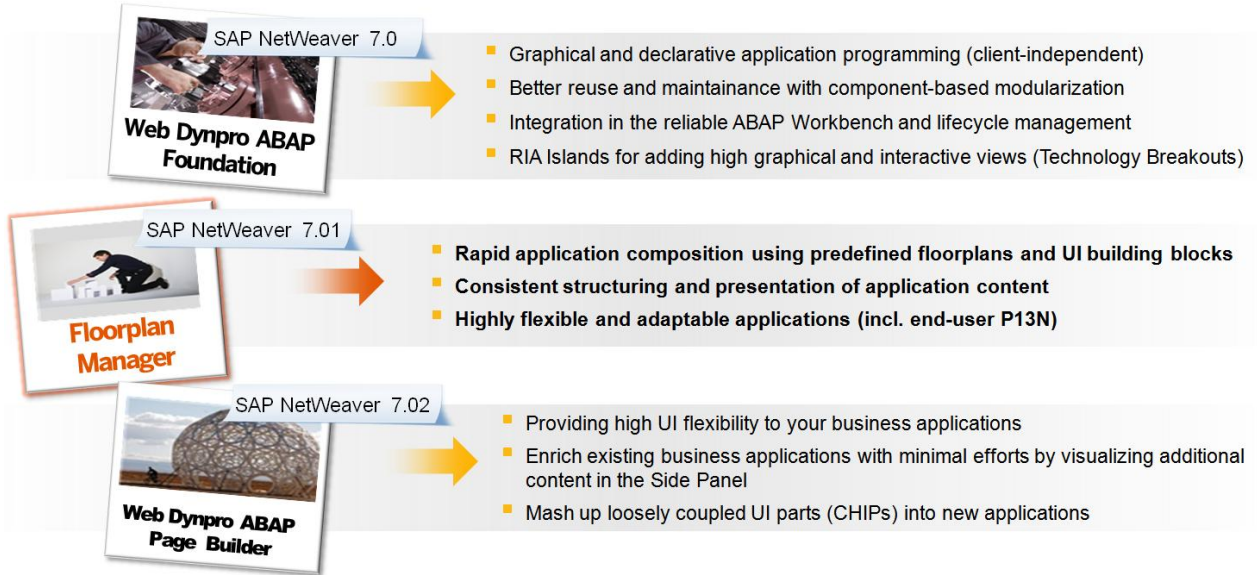


Figure 1: Web Dynpro ABAP Family including FPM

There are no specific adaptation concepts used by FPM which aren't available for freestyle¹ Web Dynpro ABAP applications as well (configuration, customizing and personalization).

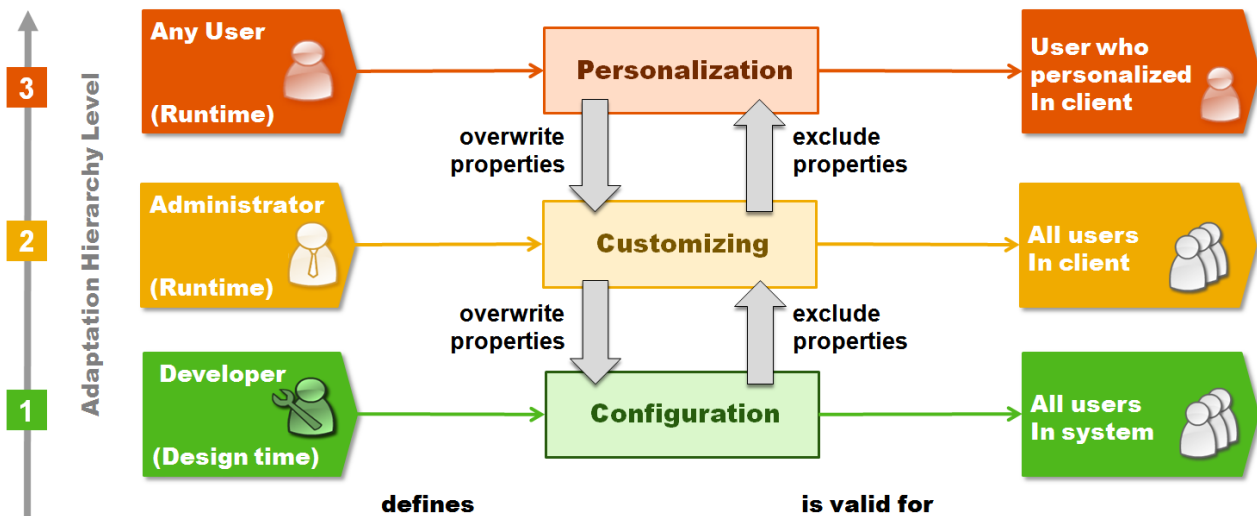


Figure 2: Hierarchy of Web Dynpro ABAP Adaptation Techniques

¹ Web Dynpro ABAP applications built without the use of FPM

However, due to the specific way FPM uses Web Dynpro ABAP, special considerations are needed.

What is specific about FPM is the heavy usage of Web Dynpro configurations. So let's have a look at the typical structure of an FPM-based application.

The entry point you need for starting an application is the *application configuration*, which is tied to a single Web Dynpro application. The necessary information needed to start the application is divided between those two entities:

- **Web Dynpro ABAP Application:** Contains the information about the main component (described below) and window of the application
- **Web Dynpro ABAP Application Configuration:** Contains the information about the configuration used for starting the main component

There are only 3 different main components used in FPM-based applications. Each one corresponds to one of the supported floorplans:

- **OIF (Object Instance Floorplan):** component FPM_OIF_COMPONENT
- **GAF (Guided Activity Floorplan):** component FPM_GAF_COMPONENT
- **OVP (Overview Page Floorplan):** component FPM_OVP_COMPONENT

These components implement the common behavior which is required by the SAP UI guidelines for all applications. However, as hundreds of applications share these components, the application-specific settings obviously cannot be part of the components themselves. Therefore, you can think of these components as templates and all application-specific settings are stored within the configurations that you make of these components.

The configuration of the floorplan component contains the information about the structure of the application (e.g. which roadmap steps shall be displayed in GAF, which pages shall be offered in OVP, etc.) and which components shall be embedded in the content areas, which buttons are displayed in the toolbar, name of application-specific controller, etc.

The most important information within the floorplan configuration is the list of the embedded components used for displaying the application data. These are normal Web Dynpro components which implement a specific interface (IF_FPM_UI_BUILDING_BLOCKS). In order to distinguish them from arbitrary Web Dynpro components, they are called UIBBs (UI Building Blocks). There are two types of UIBBs:

- **Freestyle UIBBs:** These UIBBs mostly have their own views, which are more or less static and specific for the application they are used for. Configuration is not relevant or of minor importance for this type of UIBB.
- **Generic UIBBs (GUIBBs):** These UIBBs are provided by the FPM framework (e.g. Form UIBB or List UIBB). These GUIBBs again implement the common behavior defined by the UI Guidelines. However, everything that is application-specific is again stored in the component's configuration. The configuration contains the layout information for the UI as well as a link to the feeder class, which is the interface to the backend functionality.

And where is the application-specific code? So far, an FPM-based application is defined by a set of individual component configurations, but of course, there is specific coding involved in every application.

For the decision on how to adapt an FPM-based application, it is crucial to know what is done by coding and what is defined by configuration. Therefore, a list of places where coding is relevant is necessary:

- **Application Controllers:** These are used for dynamically changing, at runtime, the settings of the floorplan configuration. For example, the set of UIBBs within the content area might be configured statically within the floorplan configuration, or set at runtime.
- **GUIBB Feeder Classes:** These are used to provide the data displayed in GUIBBs. There is no option for feeders to modify the configured screen layout (except for disabling or hiding fields). So, in the case of GUIBBs, it is normally quite simple to decide where to adapt.
- **Freestyle UIBBs:** Freestyle UIBBs must be considered as coding.

To summarize, an FPM-based application can be described as a tree of configurations plus some code:

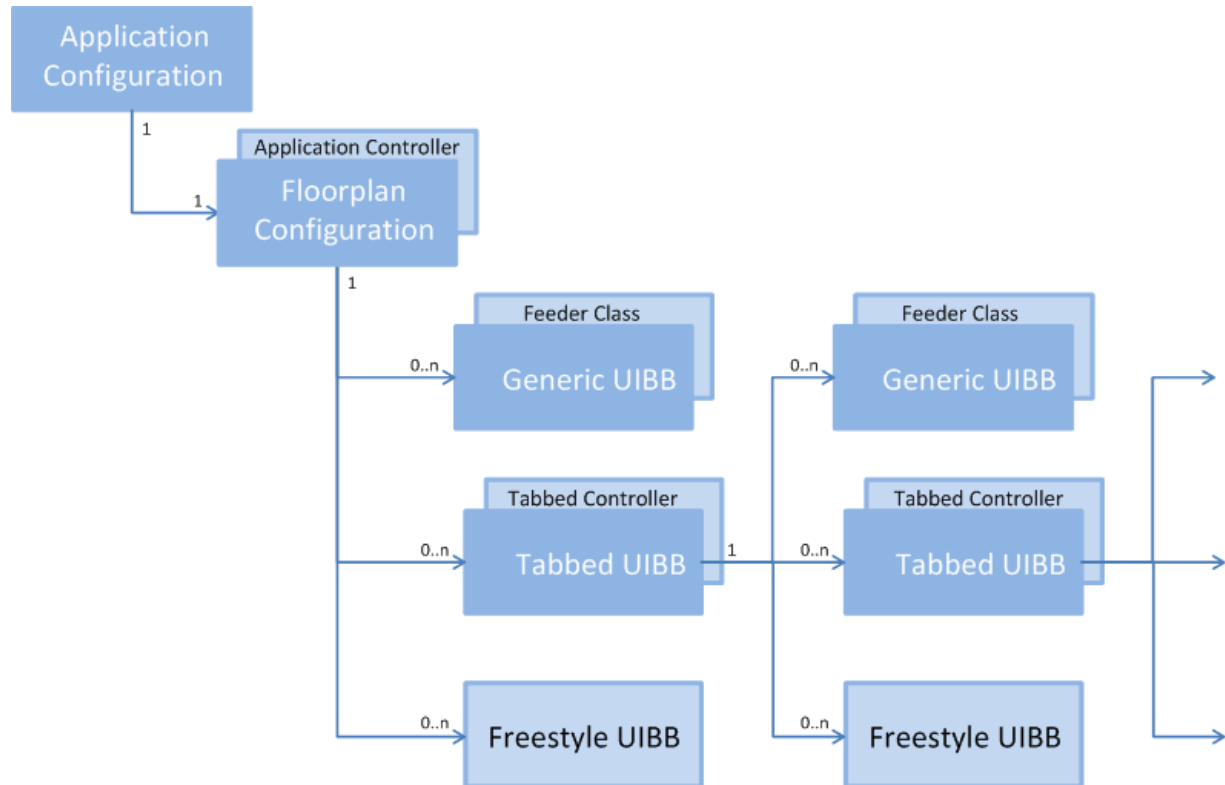


Figure 3

In Figure 3, the dark boxes represent configurations while the light blue boxes represent 'code'. There is always the option to dynamically change configurations at runtime; therefore there is a 'code' box behind each configuration to indicate this. Nevertheless, the configurations have precedence over the code. The Tabbed UIBB is included in this figure just to demonstrate how 'deep' an application hierarchy can be.

Component Configurations

As outlined in the previous section, an FPM-based application is fundamentally defined by a tree of configurations. A configuration that belongs to a Web Dynpro component is created and edited by developers using Web Dynpro application `CONFIGURE_COMPONENT`. Like all workbench objects, the component configuration is found in all clients and it can be copied, versioned, enhanced and transported. Furthermore, it consists of an explicit and implicit part:

- The **explicit part** results from an explicitly-defined, also known as **component-defined**, configuration context. All the design templates (floorplans, Generic UIBBs) that are included in the Floorplan Manager are based on an explicit configuration context. They can be edited by using a WYSIWYG editor, the FPM Configuration Editor (see figure below).
- The **implicit part** results from a generically created configuration context. UI elements that have been statically created by Web Dynpro can be edited in the Configuration Editor via the view dropdown list option **Web Dynpro built-in**. Normally, implicit configuration can be ignored when using FPM.



Figure 4

A Detailed Look at the Structure of an FPM-Based Application

In this section we will have a detailed look at the structure of an FPM-based application. As an example, we will use application FPM_DEMO_PLAYER with application configuration FPM_DEMO_PLAYER. This demo application is part of SAP_ABA 7.02 SP 9. If your system is not yet on this support package level, the demo application is created by implementing [SAP Note 1590173](#).

When you start the application, the screen looks like the screenshot below:

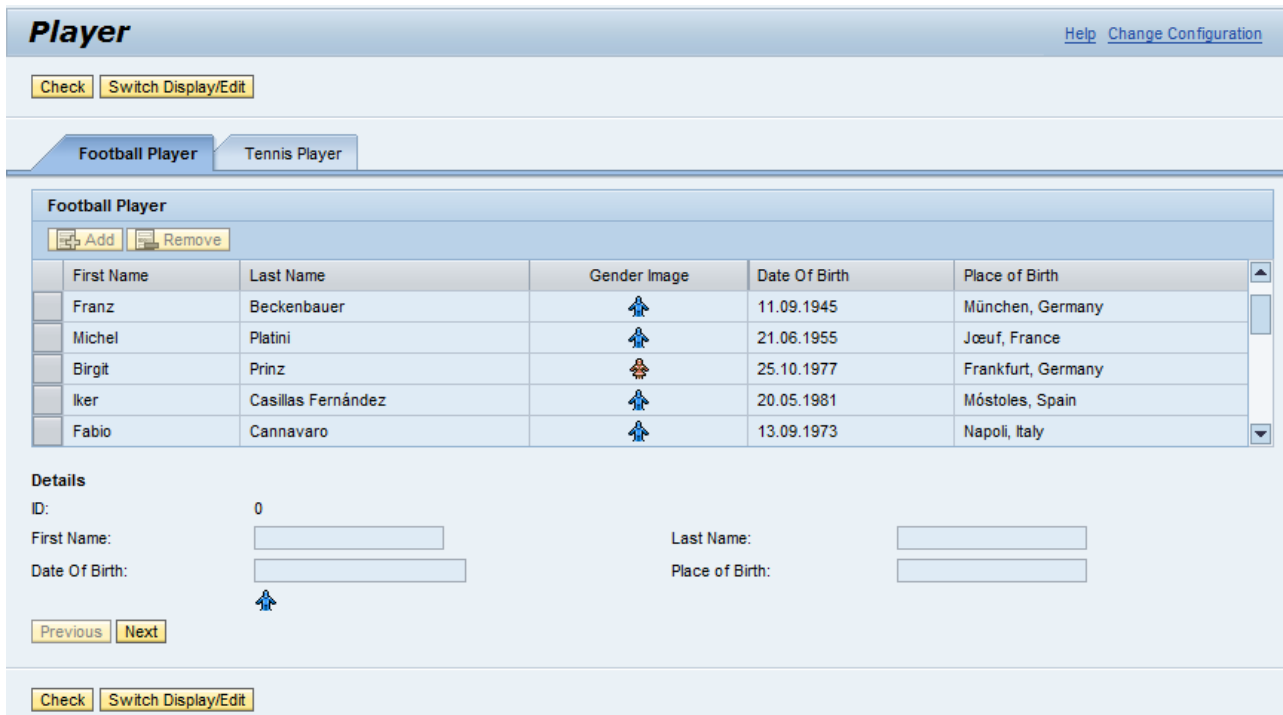


Figure 5

To get an overview of the application's structure, launch Web Dynpro application FPM_CFG_HIERARCHY_BROWSER in transaction SE80 in package APB_FPM_CONF. Enter the application configuration on the initial screen and choose 'Start'. You are directed to the FPM 'Application Hierarchy Browser' which has 2 modes: the browser and the deep-copy mode. To get an overview of the application, we will use only the browser mode. Choose 'Expand All'; the screen should look like Figure 4, displaying the application's structure in a hierarchical way:

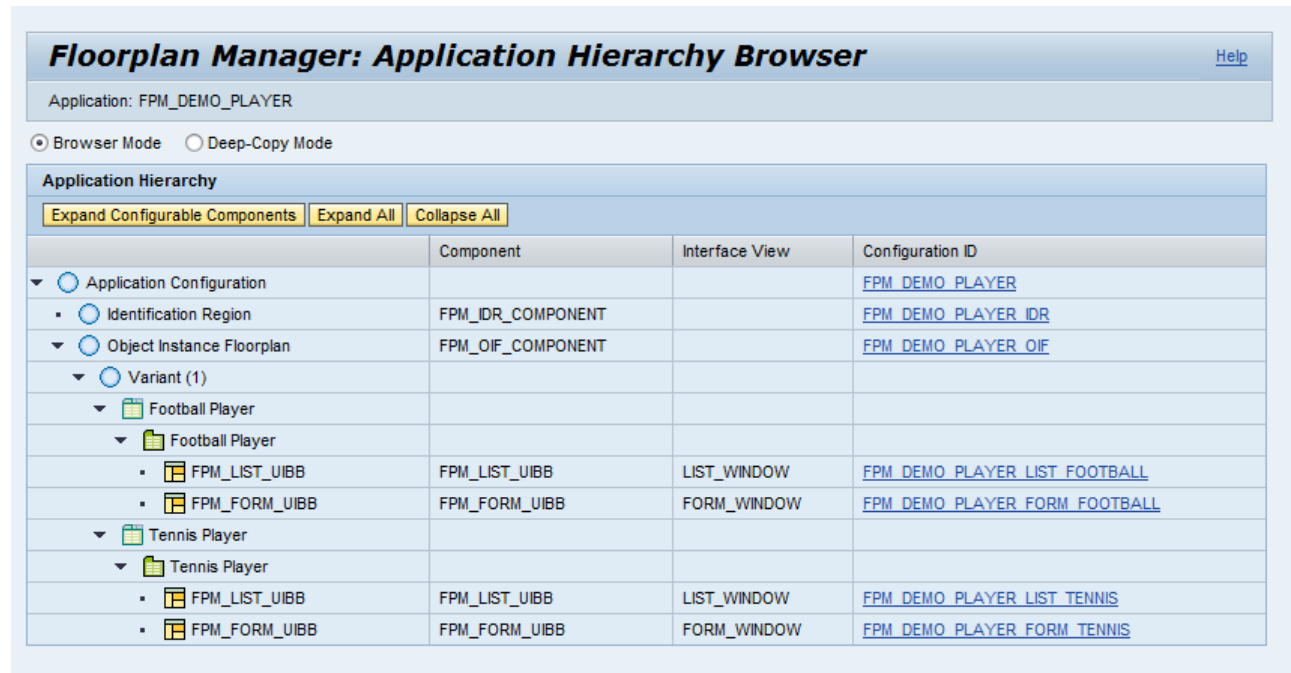


Figure 6

Now let's try to map what you see in this schematic view of the application with the screen of the application.

The root node corresponds to an application configuration named FPM_DEMO_PLAYER which is a configuration for application FPM_DEMO_PLAYER. If we click on the link in the 'Configuration ID' column, the editor for the application configuration is launched (see Figure 5). There you can see three views ('Attributes', 'Structure' and 'Application Parameters') – the most important one is the 'Structure' view.

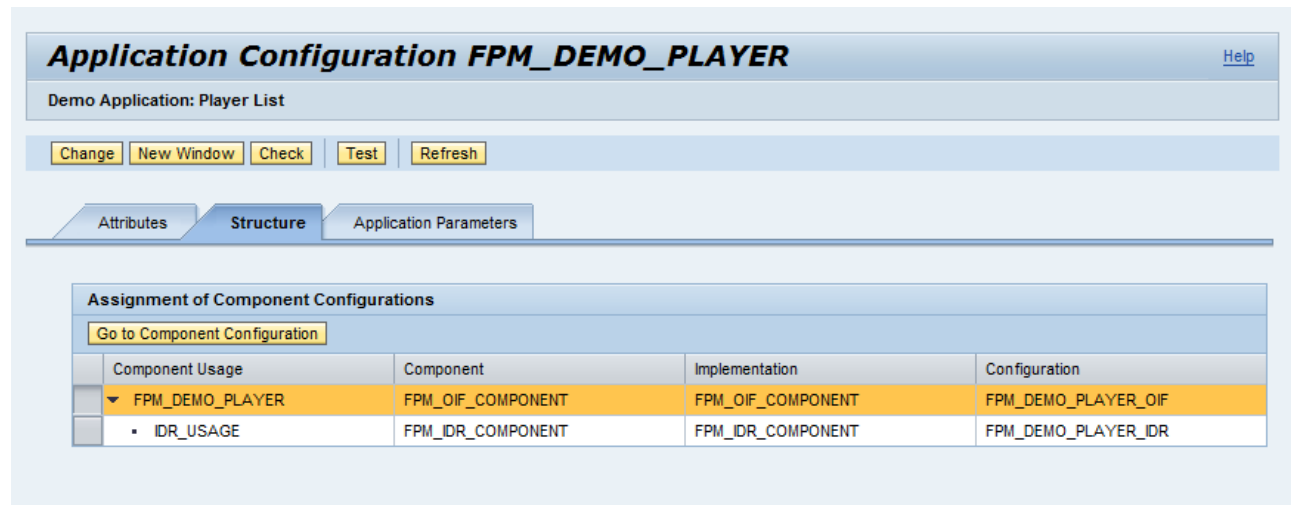


Figure 7

Application FPM_DEMO_PLAYER is using component FPM_OIF_COMPONENT as the start component, therefore the first line corresponds to that component. Here, it is specified that FPM_OIF_COMPONENT will start with component configuration FPM_DEMO_PLAYER_OIF. As component FPM_OIF_COMPONENT is

the component providing the floorplan's functionality and layout², we will use the term 'floorplan component' for it and the term 'floorplan configuration' for the configuration used to start it.

In the second, subordinate, line (IDR_USAGE) you find the configuration ID of the IDR (Identification Region or 'header area'). In the screenshot above, it is FPM_DEMO_PLAYER_IDR. The header area is configured in its own configuration, allowing you to re-use header configurations across different applications. You can specify the title in this configuration ('Player' in Figure 3) and the ticket area (which is not used in this example).

Going back to the hierarchy browser, we can find the same information again, in the second and third lines.

Below the Object Instance Floorplan node there is a Variant node. This is a technical node which is never represented on the UI. It allows specifying multiple application layouts within one application configuration, but there is always only one visible at any point in time during runtime³.

The next two levels below the variant node are specific for the OIF. These are the nodes for the main views ('Football Player' and 'Tennis Player') and, below them, the nodes for the sub-views (here there is one sub-view for each main view with the same name). You can see these nodes at runtime as part of a tab strip⁴. However, as there is only one sub-view per main view the sub-views aren't displayed.

Below the sub-view nodes there are the UIBB nodes. These are the places where the application-specific content is displayed. These UIBBs can be configurable or non-configurable. In this example each sub-view contains two UIBBs: a list and a form UIBB. The list is used for displaying football or tennis players. If there is an entry selected in the list, the form displays the details of the selected player.

The UIBBs in this demo application are generic UIBBs, which have to be configured. Therefore, in the rightmost column of the hierarchy you can find the configuration IDs used for running these UIBBs. For non-configurable UIBBs, this column would be empty.

The following figure shows the application at runtime. The colored sections indicate the different component configurations within the application.

The screenshot shows the runtime application for 'Player' (FPM_DEMO_PLAYER_IDR). It has a header bar with 'Player' and 'FPM_DEMO_PLAYER_IDR' and a 'Help Change Configuration' link. Below the header is a 'Check' and 'Switch Display/Edit' button. The main content area is titled 'OIF: FPM_DEMO_PLAYER_OIF' and contains a tab strip with 'Football Player' and 'Tennis Player'. The 'Football Player' tab is active, showing a list of players titled 'List: FPM_DEMO_PLAYER_FORM_FOOTBALL'. The list has columns for First Name, Last Name, Gender Image, Date Of Birth, and Place of Birth. Below the list is a 'Details' section titled 'Form: FPM_DEMO_PLAYER_FORM_FOOTBALL' with fields for ID, First Name, Last Name, Date Of Birth, and Place of Birth. At the bottom, there are 'Previous', 'Next', 'Check', and 'Switch Display/Edit' buttons.

First Name	Last Name	Gender Image	Date Of Birth	Place of Birth
Franz	Beckenbauer		11.09.1945	München, Germany
Michel	Platini		21.06.1955	Jœuf, France
Birgit	Prinz		25.10.1977	Frankfurt, Germany
Iker	Casillas Fernández		20.05.1981	Móstoles, Spain
Fabio	Cannavaro		13.09.1973	Napoli, Italy

Figure 8

² For each supported floorplan there exists exactly one component which must be used as floorplan component

³ Variants and the separate header area (IDR) are only available in the Object Instance Floorplan (OIF) and in the Guided Activity Floorplan (GAF). In the Overview Page Floorplan (OVP), which was introduced with SAP NetWeaver 7.02, different concepts are available.

⁴ This is strictly speaking not a tab strip, but a view switch, which is represented by the UI control Horizontal Contextual Panel

The root of the configuration hierarchy is the application configuration, which has no direct visual representation in the running application. However, it defines which components and configurations will be embedded directly into the application. These are the IDR component configured by FPM_DEMO_PLAYER_IDR (orange overlay) and the OIF with configuration FPM_DEMO_PLAYER_OIF (red overlay).

Embedded in the OIF are several UIBBs. The two which are currently visible are marked with the blue overlay.

Which Objects have to be Adapted?

As you can see, an FPM-based application is composed of multiple objects: applications, application configurations, component configurations and code (where code may mean feeder classes, application controllers, and Freestyle UIBBs). It is not easy to determine which objects must be adapted in order to achieve the intended changes. Therefore, the following table of use-cases is intended to help you:

Intended Change	Where to Make the Change
Change the header (e.g. replace 'Player' with a different title) in an OIF or GAF based application	Adapt the header configuration (FPM_DEMO_PLAYER_IDR in the example)
Add or remove a main or sub-view in an OIF based application	Adapt the floorplan configuration (FPM_DEMO_PLAYER_OIF in the example)
Rename a main or sub-view (e.g. replace 'Football Player' with 'Soccer Player') in an OIF based application	Adapt the floorplan configuration (FPM_DEMO_PLAYER_OIF in the example)
Change the toolbar (e.g. add or remove a button)	Adapt the floorplan configuration (FPM_DEMO_PLAYER_OIF in the example)
Add, remove or replace a UIBB	Adapt the floorplan configuration (FPM_DEMO_PLAYER_OIF in the example)
Add or remove a roadmap step in a GAF based application	Adapt the floorplan configuration (not applicable in this example)
Add or change a roadmap sub-step in a GAF based application	Adapt the floorplan configuration (not applicable in this example)
Add, remove or rearrange fields in a form	Adapt the form UIBB configuration (FPM_DEMO_PLAYER_FORM_FOOTBALL in the example)
Add, remove or rearrange columns in a list	Adapt the list UIBB configuration (FPM_DEMO_PLAYER_LIST_FOOTBALL in the example)

As you see, there are a lot of use-cases which can be covered by adapting component configurations and there are very powerful options to do this. Nevertheless, there are some adaptation use-cases where it is not enough to adapt only the configurations. The following table details some adaptation use-cases where 'code' must be adapted:

Intended Change	Reason for Code Change
Manipulate the data displayed in a form or list	The data extraction logic is part of the feeder classes of the list and form UIBBs
Dynamically change the floorplan configuration at runtime	This is a very common use-case; based on the application state, the toolbar must be adjusted or a different set of UIBBs must be displayed, etc. This is possible using APIs.

Options for Adaptations - Overview

Copy Application

The simplest option is to copy a delivered application. With the help of the [Application Hierarchy Browser for Floorplan Manager](#) it is quite simple to copy the whole configuration tree of an FPM application. It is also possible to keep some branches of the original tree and copy only the part you need to adapt.

Advantages:

- ➕ It is easy to understand what is happening and there is absolutely no interference between your copied entities and later deliveries of the original application.

Drawbacks:

- ➖ Corrections and improvements of configurations delivered by SAP which affect the original application will not reach the copied entities of your application.

Enhancement

The general [NetWeaver Enhancement Framework](#) works with FPM-based applications too, as all necessary entities can be enhanced (a detailed description of the Enhancement Framework in Web Dynpro ABAP applications can be found in this [SAP Help documentation](#)).

Advantages:

- ➕ It is the only option that allows you to combine both code changes and configuration adaptations.

Drawbacks:

- ➖ The enhancement of a component configuration is technically nothing other than a system-managed copy of the configuration. Therefore, it is not possible to enhance only parts of a configuration. Enhanced configurations are separated from corrections or upgrades delivered in the future. Additionally, it is not recommended to have more than one enhancement per component configuration, as only one enhancement is ever active.

Customizing

Component configurations can be adapted on the basis of [two additional layers](#):

- **Customizing Layer:** This is for adaptations carried out by the administrator at the customer's site. Adaptations are valid for all users in the corresponding client and can be transported. Customizing can be created and edited both at the design-time (Web Dynpro application CUSTOMIZE_COMPONENT) and runtime (URL parameter SAP-CONFIG-MODE = X, plus context menu entry 'Settings for Current Configuration'). For any FPM application that has been started in the so-called *Administrator Mode* (URL parameter SAP-CONFIG-MODE = X) a link 'Adapt Configuration' is provided in the application header that points to Web Dynpro application CUSTOMIZE_COMPONENT.
- **Personalization Layer.** This contains individual settings for each individual user. A personalization can be made only during the runtime (context menu entry 'User Settings'). This layer is mentioned only for completeness – it is not relevant for the topic of this document and is not discussed further.

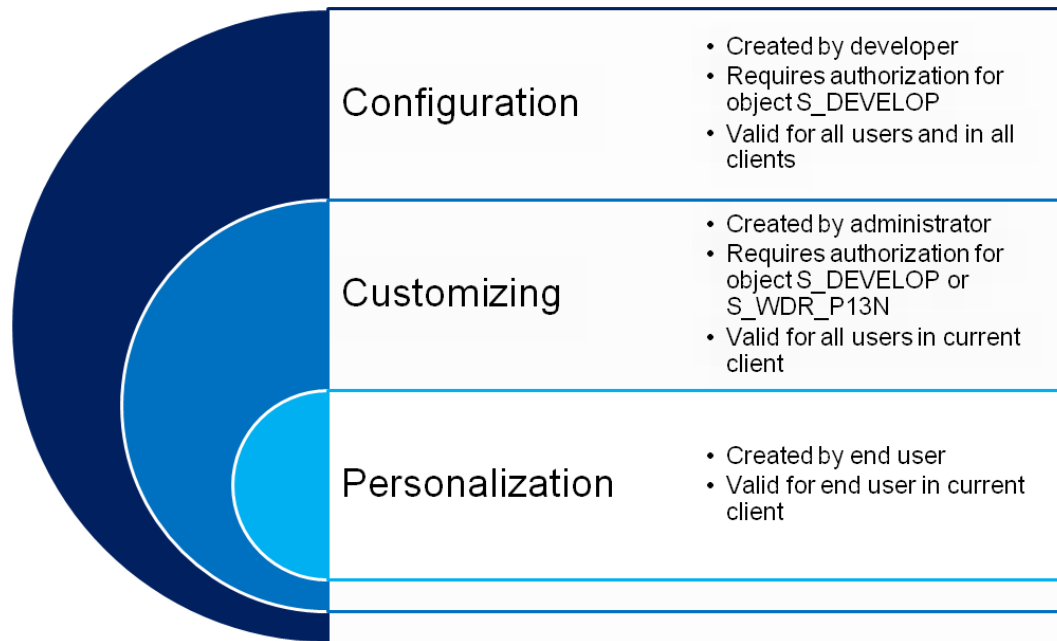


Figure 9

The smart thing about customizing is that it only contains the changed parts of the application (so-called *delta-handling*). At runtime, the original configuration is merged with the changes in the customizing layer.

Advantages:

- Only the changed parts of the configuration are stored. Therefore, later changes or corrections will be applied even to an adapted configuration. No additional effort is required compared to modifying the original object.
- Customizing is the best approach to fine-tuning an FPM application

Drawbacks:

- Customizing is restricted to configurations. There is no way to customize code.
- In contrast to component configurations, it is not possible to create versions or copies as a kind of 'backup'.
- Customizing can be created even if there is no underlying component configuration. This might lead to semantic data inconsistencies during the FPM runtime if a corresponding component configuration is created at a later point.
- If a configuration is deleted, the customizing delta remains in the data base. This might lead to semantic data inconsistencies during the FPM runtime, as the delta might contain only fragments of the information needed to render the FPM application.

Modification

Only for completeness sake and not further discussed in this document: Modifying the original object is another option (changing an SAP-delivered component at configuration level).

Advantages:

- Low initial effort and easy to understand.

Drawbacks:

- After upgrade of the application or a support package, there is the risk that the modification is overwritten and has to be re-implemented.

Options for Adaptations – Details

In this section, we will go step-by-step through the different adaptation options and we will make the following simple adaptations:

1. Change the title of the application from 'Player' to 'Football Player'.
2. Simplify the application by removing the 'Tennis Player' main view and the second toolbar below the application.
3. Rearrange the form layout; put all fields below each other and remove the Previous and Next.

These changes can all be made 'code-free' by adapting configurations. Let's see which configurations must be adapted:

- To change the title, we must adapt the header configuration (FPM_DEMO_PLAYER_IDR).
- To remove the 'Tennis Player' main view and the second toolbar, we must adapt the floorplan configuration (FPM_DEMO_PLAYER_OIF).
- To rearrange the form layout, we must adapt the form configuration (FPM_DEMO_PLAYER_FORM_FOOTBALL).

In the end, the application should look like Figure 8:

Football Player [Help](#) [Change Configuration](#) [Application Hierarchy](#)

Close | Check | Switch Display/Edit

	First Name	Last Name	Gender Image	Date Of Birth	Place of Birth
	Edison	Aranes do Nascimento (Pelé)		22.10.1940	Três Corações, Brazil
	Franz	Beckenbauer		11.09.1945	München, Germany
	Michel	Platini		21.06.1955	Jœuf, France
	Birgit	Prinz		25.10.1977	Frankfurt, Germany
	Iker	Casillas Fernández		20.05.1981	Móstoles, Spain

Details

ID: 2

First Name:

Last Name:

Date Of Birth:

Place of Birth:

Figure 10

Copy Application

The 'Application Hierarchy Browser' allows us to not only analyze the structure of an FPM-based application, but to copy the configuration hierarchy too. Therefore, let's start Web Dynpro application FPM_CFG_HIERARCHY_BROWSER again, enter the application configuration on the initial screen and choose 'Start'. However, this time let's switch to the 'Deep-Copy Mode'. This will give you two more columns; the 'Copy' checkbox column allows you to determine whether to copy a configuration or not, and the 'Target Configuration ID' allows you to enter a name for the copied configuration.

Floorplan Manager: Application Hierarchy Browser [Help](#)

Application: FPM_DEMO_PLAYER

Browser Mode Deep-Copy Mode

Application Hierarchy

Expand Configurable Components
Expand All
Collapse All
Change Affixes
Start Deep-Copy

	Copy	Component	Interface View	Configuration ID	Target Configuration ID
Application Configuration	<input checked="" type="checkbox"/>			FPM_DEMO_PLAYER	ZC_FPM_DEMO_PLAYER
Identification Region	<input checked="" type="checkbox"/>	FPM_IDR_COMPONENT		FPM_DEMO_PLAYER_IDR	ZC_FPM_DEMO_PLAYER_IDR
Object Instance Floorplan	<input checked="" type="checkbox"/>	FPM_OIF_COMPONENT		FPM_DEMO_PLAYER_OIF	ZC_FPM_DEMO_PLAYER_OIF
Variant (1)					
Football Player					
Football Player					
FPM_LIST_UIBB	<input type="checkbox"/>	FPM_LIST_UIBB	LIST_WINDOW	FPM_DEMO_PLAYER_LIST FOOTBALL	
FPM_FORM_UIBB	<input checked="" type="checkbox"/>	FPM_FORM_UIBB	FORM_WINDOW	FPM_DEMO_PLAYER_FORM FOOTBALL	ZC_FPM_DEMO_PLAYER_FORM FOOTBALL
Tennis Player					
Tennis Player					
FPM_LIST_UIBB	<input type="checkbox"/>	FPM_LIST_UIBB	LIST_WINDOW	FPM_DEMO_PLAYER_LIST TENNIS	
FPM_FORM_UIBB	<input type="checkbox"/>	FPM_FORM_UIBB	FORM_WINDOW	FPM_DEMO_PLAYER_FORM TENNIS	

Figure 11

For this example we will only copy the configurations we want to adapt and keep the original configurations for those we will not change. This is a good idea if you are sure that you will never adapt the other parts. If you think that you will adapt the others in the future too, it is easier to copy the configurations now rather than later.

After you choose the 'Start Deep-Copy' button, the system starts to create the new configurations. After you enter a package name in a dialog box, you will come back to the hierarchy, but now you will have links in the 'Target Configuration ID' column which will navigate you to the editors for these configurations.

Try to keep this window for the rest of this section open, as we will use it to launch the different configuration editors⁵.

Starting the Application

Firstly, we will start the copied application. Within the application hierarchy browser, click the link for the copied application configuration (which is in the first line). Another window will open, displaying the copied application configuration. In the toolbar there is a 'Test' button – pressing it will launch the application. It should look exactly like the original one.

Adapting the Header

In the Hierarchy Browser, click the link for the header configuration (in the hierarchy, the node is named Identification Region). This navigates you to the header's editor. Choose the 'Change' button and select the node 'IDR Basic' in the hierarchy on the left side of the window. Now you can change the application title from 'Player' to 'Football Player' (see Figure 10).

⁵ In case you mistakenly closed this window, you can re-launch it by starting Web Dynpro application FPM_CFG_HIERARCHY_BROWSER again and enter the name of the copied application configuration on the initial screen (in the example this is ZC_FPM_DEMO_PLAYER).



Figure 12

Press 'Save' and restart the application to see the changed header.

Adapting the Floorplan Configuration

Now let's launch the editor for the floorplan configuration by clicking on the corresponding link in the Hierarchy Browser. The floorplan's configuration editor is launched. Here, we want to remove the 'Tennis Player' main view.

Select the main view to be deleted (either in the hierarchy on the left side of the window, or in the preview in the main area), go to 'Change' mode and press the 'Delete' button which is found below the preview.

To remove the second toolbar, select the node 'Toolbar' within the hierarchy on the left side and uncheck the 'Duplicate Toolbar' Checkbox located in the Attributes area below the Preview.

After saving and restarting the application, the screen should look much simpler. The tabstrip has completely disappeared (as it's only rendered if there is more than one main view and sub-view) and so has the bottom toolbar.

Adapting the Form

To rearrange the form fields, let's launch the configuration editor for the form by clicking on the corresponding link in the hierarchy browser. Our task here is to put all fields in one column. The left side of the screen displays the hierarchical view of the form. Here, we must select the only group and within the Attributes area, change the Group type to 'Full width, 1 column'.

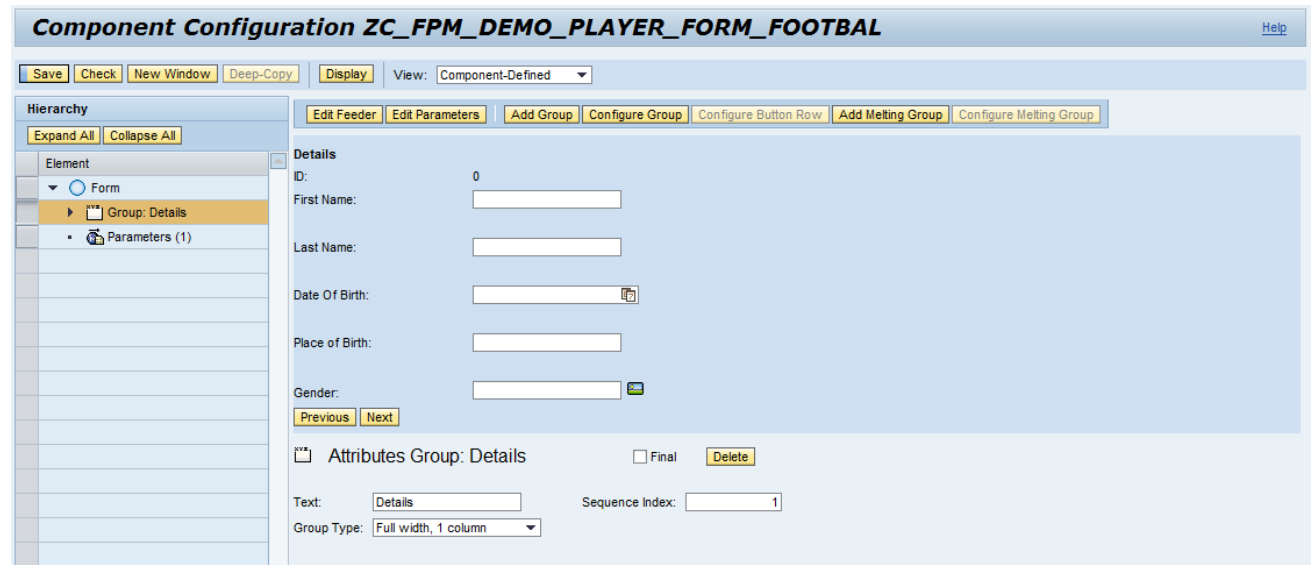


Figure 13

Then we must remove the Invisible Elements and the Buttons from the Group which are no longer necessary. Therefore, we have to launch the 'Configure Group' dialog box and remove all <space> elements as well as the Previous/Next Button Row.

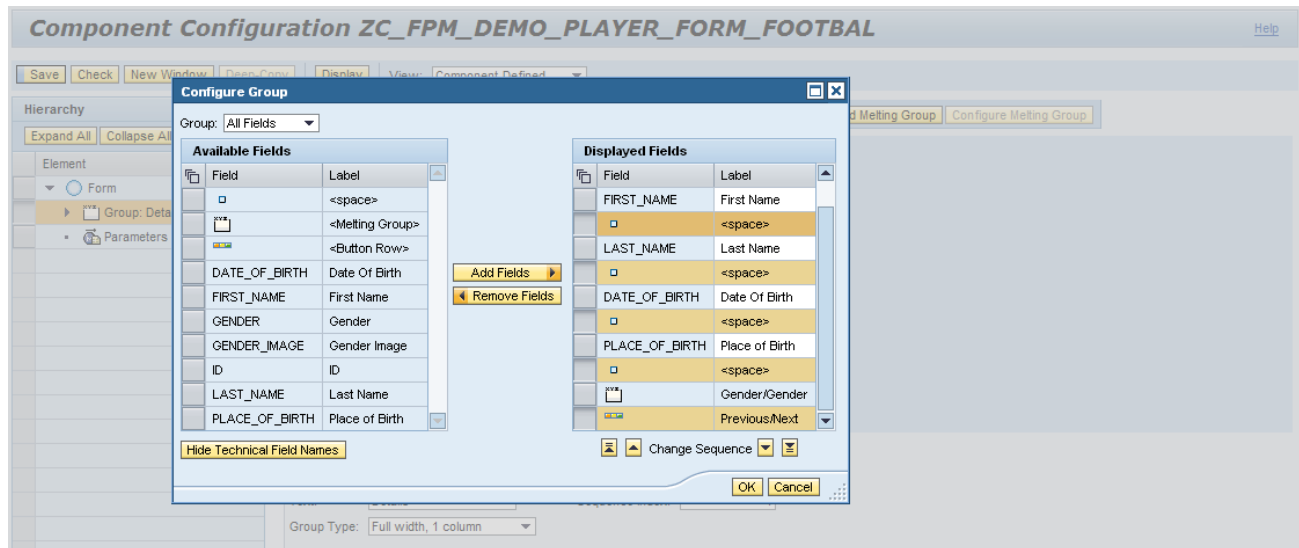



Figure 14

After saving, we have completed our task.

Customizing

Now let's do the same adaptations but this time we will use the 'Customizing' functionality. The main difference is the way in which we launch the editors. Once inside the configuration editors, the necessary changes are made in the same way.

So let's start by launching the application in customizing mode:

Go to transaction SE80 and, using the toolbar button 'Other object... Shift+F5' , select Web Dynpro Application Configuration FPM_DEMO_PLAYER. Start the application by selecting the following menu options in the main toolbar: Web Dynpro Configuration -> Test -> Execute in Administration Mode (Shift+F8). The application should look exactly the same as it as it always does but with one exception: there is an 'Adapt Configuration' link in the upper right corner of the application. We navigate to the customizing editor⁶ by choosing this link.

Adapting the Floorplan Configuration

If a dialog box informs you that no customizing exists, choose OK. In the next dialog box 'Create Customizing', choose OK. Now you can make the same changes in exactly the same way as you did for the copied floorplan configuration in the former section.

Adapting the Header

From the floorplan customizing editor, you can directly navigate to the header customizing editor by choosing the 'Configure IDR' button in the upper-right part of the window. Again, you might have to create a new customizing before you start to adapt the header (just as you did for the floorplan customizing).

Select IDR Basic in the hierarchy and change the title from 'Player' to 'Football Player'. After saving and restarting, this change should be visible in the running application.

⁶ The customizing editor is nothing more than the configuration editor started in a different mode. It looks completely the same as in configuration mode; the only difference is that on pressing 'Save', the changes are stored differently.

Adapting the Form

Our last task is the adaptation of the form. From the header configuration editor, you can navigate back to the floorplan customizing editor via the breadcrumb in the upper part of the window.



Figure 15

Click the OIF link (OIF: FPM_DEMO_PLAYER_OIF) and you come back to the floorplan configuration editor again. There, you can choose the relevant (the lower) 'Configure UIBB' button within the preview and navigate to the form's customizing editor (after confirming the two dialog boxes again).

As this is basically the same editor as for the configuration, execute the same steps as described for [adapting the copied form configuration](#) before.

Enhancement

The third option is enhancement.

Again, we will adapt the same configurations FPM_DEMO_PLAYER_OIF, FPM_DEMO_PLAYER_IDR and FPM_DEMO_PLAYER_FORM_FOOTBALL.

Before creating enhancements, you should decide whether you want this enhancement switchable or not. Switchable means that you can activate or deactivate it whenever you like. In case you want to have it switchable, you should create your own package to put the enhancements in.

We launch the Web Dynpro Application CONFIGURE_COMPONENT for each of the three configurations to be adapted, enter the configuration ID, and via 'Other Functions' -> 'Create Enhancement', create the enhancement. While creating the enhancements, the system asks you in which package the enhancement should be put. Enter your switchable package here, if you have created one. Afterwards, you can open the editor for your enhancement and apply the changes the same way as before.

Adaptation – Best Practices

So far you have learned that there are three main ways to adapt an FPM application: Customize an application, copy or enhance it.

From an end-user perspective, all three possibilities lead to the same result on the user interface. However, to keep the overall effort to a minimum, it is worth applying the right adaptation technique to the situation on-site (see the figure below). In addition, we recommend avoiding a mix of different techniques wherever possible. It will help to keep the overall picture clearer.

As a rough rule, you may use customizing for all small adaptations (fine-tuning) that are valid in the corresponding client, whereas larger development efforts would be better invested in the configuration layer. There, you may decide between enhancing a delivered application and creating a new one.

The enhancement applies to a situation where only parts of an application need to be revised. By definition, this kind of adaptation corresponds to a modification-free development. The enhancement usually belongs to a package that is assigned to a switch, which can be activated by a business function.

When creating a new application, all the configurations may belong to the customer's name space. Before creating all of those on your own, consider the use of the deep-copy function of the FPM Application Hierarchy Browser (see example above).

Changing button texts, adding form fields, rearranging the UIBBs of a main view are typical examples of small adaptations (fine-tuning) that are most effectively done via customizing. Generally, those kinds of changes are done without any programming effort.

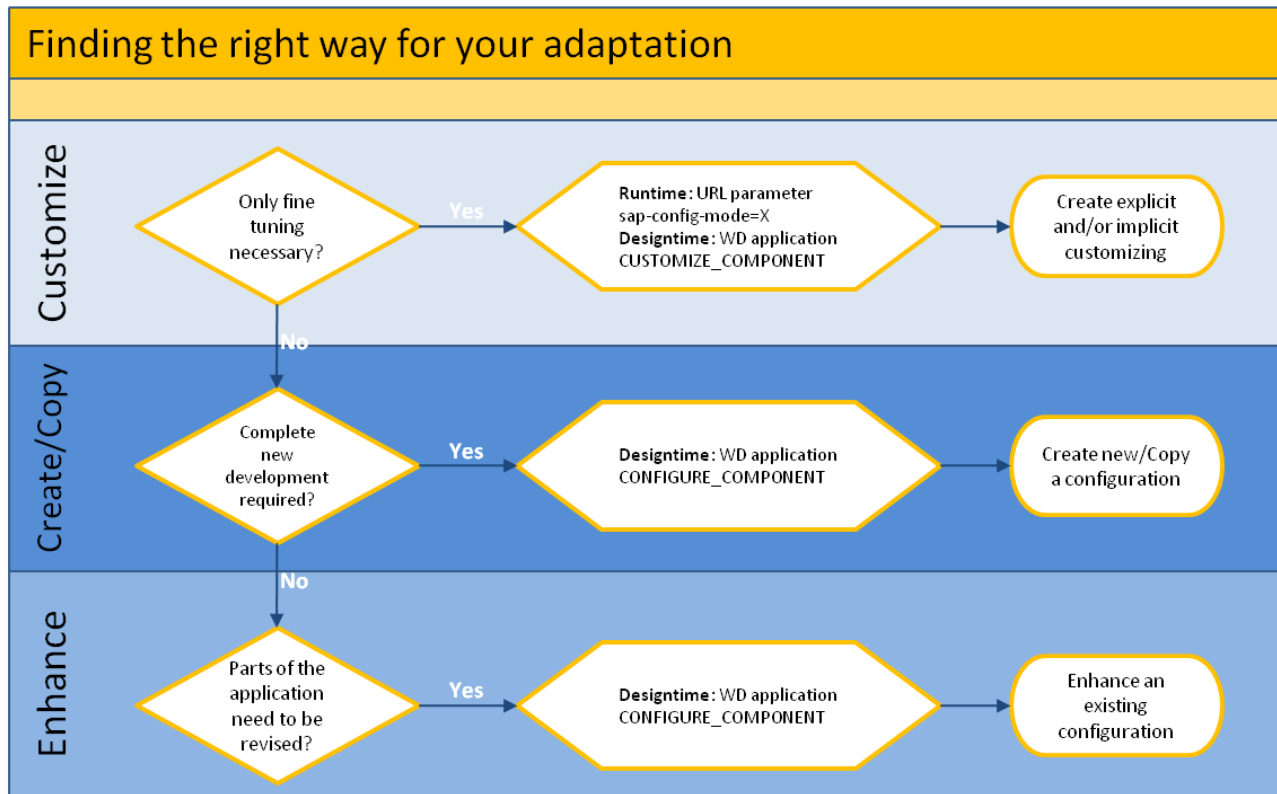


Figure 16

In contrast, adaptations that imply changes to ABAP code or creation of new code belong more appropriately to an enhanced or newly created application. Examples for this include the creation of a new OIF Variant that should be called at runtime from the OIF Initial Screen, or the creation of a new Feeder Class that is used to configure a new List UIBB.

Note: Be careful with objects that exist only on the customizing level, such as UIBBs that have been created in the Administrator Mode. At runtime, rendering may become impossible if a configuration of the same ID is created at a later point in time. Therefore, we recommend the creation of new UIBBs always on the configuration level.

Related Content

[SCN - Floorplan Manager \(FPM\) Web Dynpro ABAP](#)

[SCN – Floorplan Manager Developer’s Guide \(SAP NetWeaver 7.0 EhP2\)](#)

[SAP Online Help – SAP NetWeaver EhP2 – Floorplan Manager for Web Dynpro ABAP](#)

[SCN - Web Dynpo ABAP](#)

[SCN - New in Web Dynpro ABAP and FPM with SAP NetWeaver 7.0 EhP2](#)

Copyright

© Copyright 2011 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Oracle Corporation.

JavaScript is a registered trademark of Oracle Corporation, used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.