Design Time at Run Time - Power to the end user

Agility is the name of the game. The ability to quickly respond to changing requirements - with minimal cost, risk and time-to-market. In a dynamic world where hypes emerge in no time at all, today's stars are tomorrow's have-hads and social media fuel viral marketing that give unprecedented influence to consumers that vote with their mouse, it is simply unacceptable for applications to evolve in a 3 to 6 month release schedule with no room for intermediate changes whatsoever.

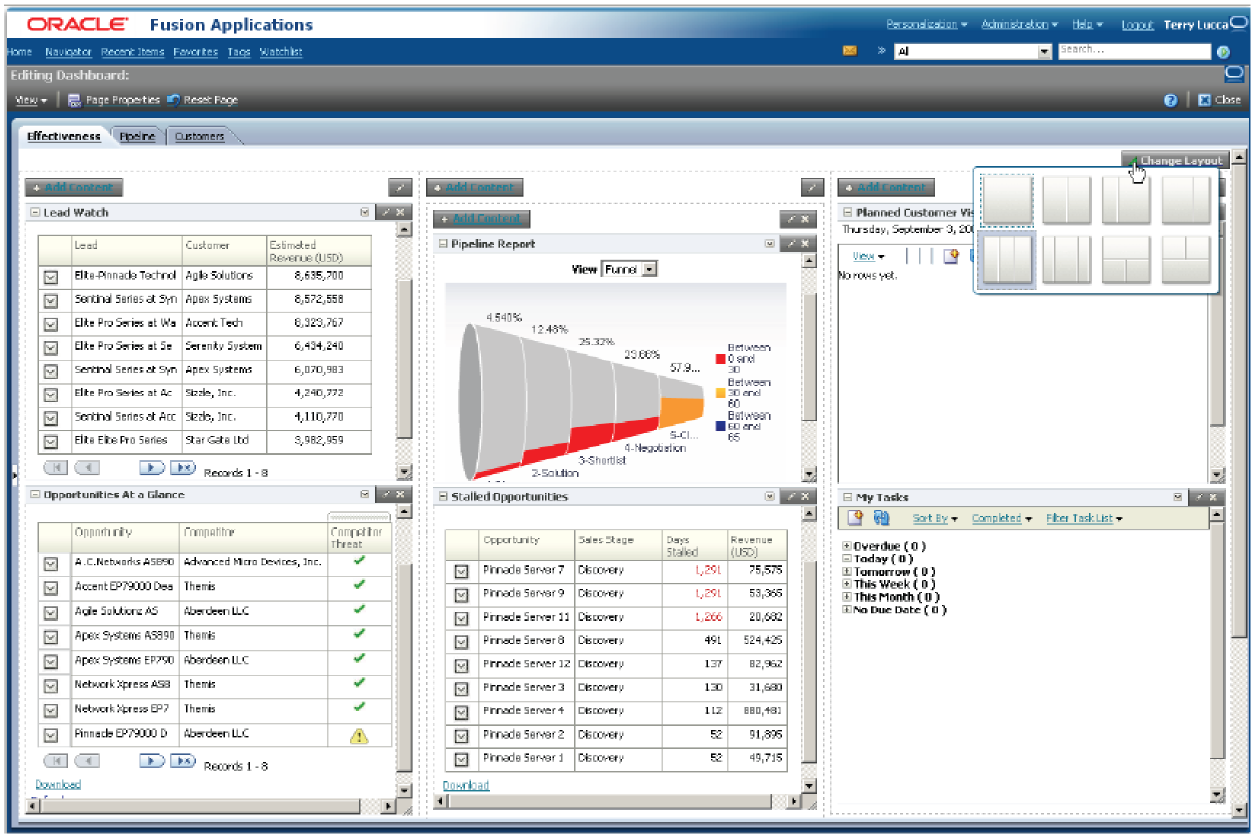
Websites always were somewhat volatile, open to rapid change especially in terms of content and style. With enterprise applications exposing self service facilities to partly external user communities including consumers, the same requirements increasingly apply as well. Applications that are only used by professional users inside the organization also meet with ever more demanding users - the same users that surf the internet on their iPad outside business hours - as well as a high pace of business change.

Oracle is catering for agility in the Fusion Middleware stack in several ways. One aspect is support for agile software engineering practices, such as facilities for automated build, testing and deployment as well as integration with source code control systems. Increased development productivity and an infrastructure that promotes reuse is another. Both of these, while valuable, will only help to speed up development-test-deployment cycles. Agility can be bolstered to an even higher degree by cutting out these stages in their entirety. Every requirement that can be met by run time actions only is most likely much faster and far more cheaply implemented than modifications that need to go through the full release cycle. The notion of "design time at run time" is this idea turned reality: across the Fusion Middleware stack, run time facilities are provided that allow manipulation of the user experience, static content, business logic, security aspects and process flow of applications.

This installment of the Fusion Column looks into several design time @ run time composers in various FMW components - and discusses the impact this concept of 'edge development' will have on IT organizations and procedures. It also hints how you can build 'design time @ run time' facilities into your own applications, even without leveraging those built-in FMW capabilities. This column also muses on the role that is introduced into organizations through DT@RT and the impact this notion will have on the software engineering process.

Web Center Composer

The probably first and most explicit design time at run time facility in Fusion Middleware is the WebCenter Page Composer that can be used to enhance ADF Faces web pages. This run time page composer makes it possible to switch pages to *page editing* mode. In this mode, for example the layout of the page can be changed - for example to a different number of columns or with a different ordering of the panels in the page.



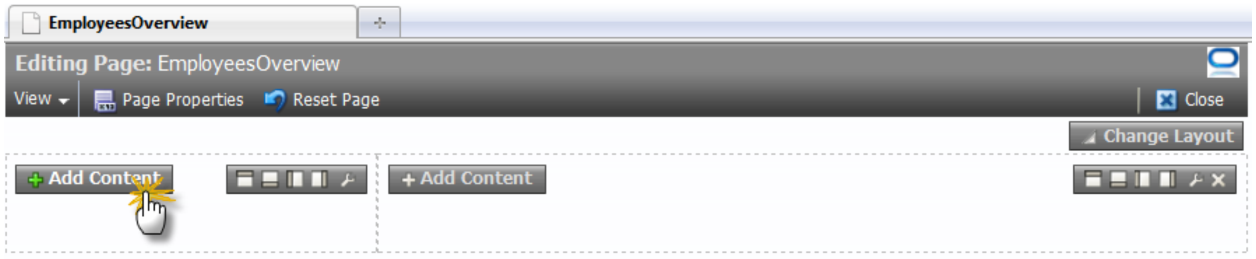
Lead	Customer	Estimated Revenue (USD)
Elite-Pinnacle Technol	Agile Solutions	8,635,700
Sentinal Series at Syn	Apex Systems	8,572,558
Elite Pro Series at Wa	Accent Tech	8,323,767
Elite Pro Series at Se	Serenity System	6,434,240
Sentinal Series at Syn	Apex Systems	6,070,983
Elite Pro Series at Ac	Sizzle, Inc.	4,240,772
Sentinal Series at Acc	Sizzle, Inc.	4,110,770
Elite Elite Pro Series	Star Gate Ltd	3,982,959

Opportunity	Competitor	Competitor Threat
A.C.Networks AS890	Advanced Micro Devices, Inc.	✓
Accent EP79000 Dea	Themis	✓
Agile Solutions AS	Aberdeen LLC	✓
Apex Systems AS890	Themis	✓
Apex Systems EP790	Aberdeen LLC	✓
Network Xpress AS8	Themis	✓
Network Xpress EP7	Themis	✓
Pinnacle EP79000 D	Aberdeen LLC	⚠

Opportunity	Sales Stage	Days Stalled	Revenue (USD)
Pinnacle Server 7	Discovery	1,291	75,575
Pinnacle Server 9	Discovery	1,291	53,365
Pinnacle Server 11	Discovery	1,266	20,682
Pinnacle Server 8	Discovery	491	524,425
Pinnacle Server 12	Discovery	137	82,962
Pinnacle Server 3	Discovery	130	31,680
Pinnacle Server 4	Discovery	112	880,481
Pinnacle Server 2	Discovery	52	91,895
Pinnacle Server 1	Discovery	52	49,715

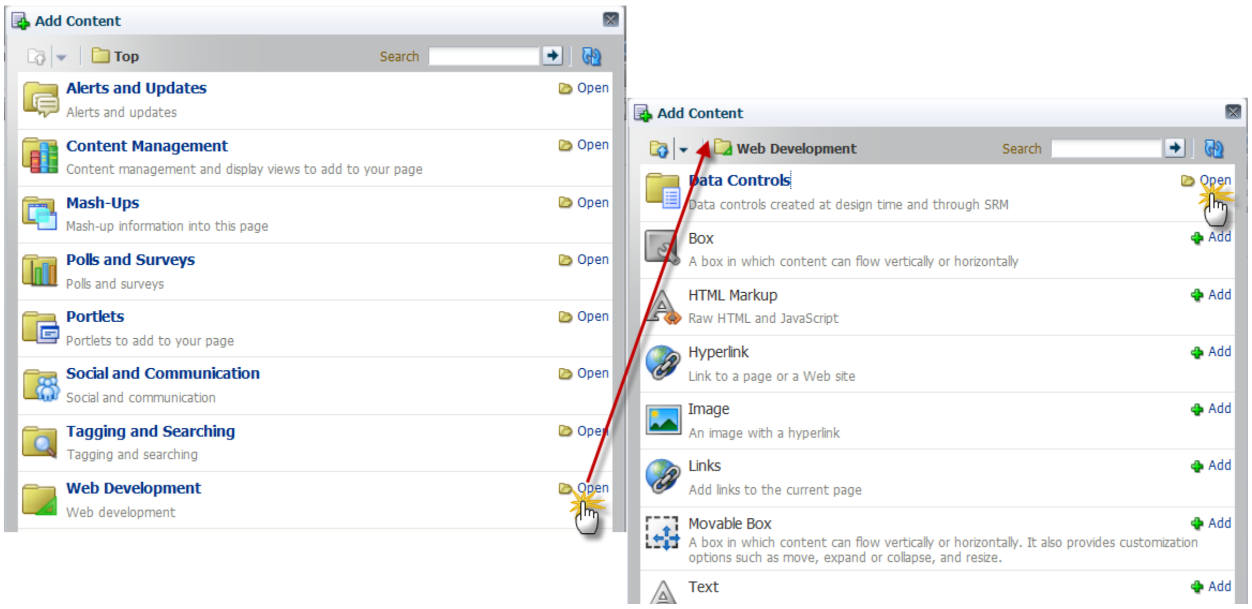
Screenshot: an example of the WebCenter Page Composer, used in Fusion Applications to allow run time manipulation of the layout of a page

Additionally, the page composer supports removing components from and adding components to a page. Newly added components can be predefined task flows, images or run-time defined rich text items, WebCenter Service components or dynamically published portlets.



Screenshot: Adding content to pages at run-time through Page Composer

After pressing the *Add Content* button, the Page Composer presents the Resource Catalog. This catalog presents a library of components that the page editor can choose from to add to the page. Note that an application can ship with multiple resource catalogs; which one is presented depends on privileges and context, governed by dynamic conditions. Application developers define these conditions at design time and also determine the content of each catalog.



Add Content

Top Search

Alerts and Updates Alerts and updates	Open
Content Management Content management and display views to add to your page	Open
Mash-Ups Mash-up information into this page	Open
Polls and Surveys Polls and surveys	Open
Portlets Portlets to add to your page	Open
Social and Communication Social and communication	Open
Tagging and Searching Tagging and searching	Open
Web Development Web development	Open

Add Content

Web Development Search

Data Controls Data controls created at design time and through SRM	Open
Box A box in which content can flow vertically or horizontally	Add
HTML Markup Raw HTML and JavaScript	Add
Hyperlink Link to a page or a Web site	Add
Image An image with a hyperlink	Add
Links Add links to the current page	Add
Movable Box A box in which content can flow vertically or horizontally. It also provides customization options such as move, expand or collapse, and resize.	Add
Text	Add

Select a component to add to a page from the Resource Catalog

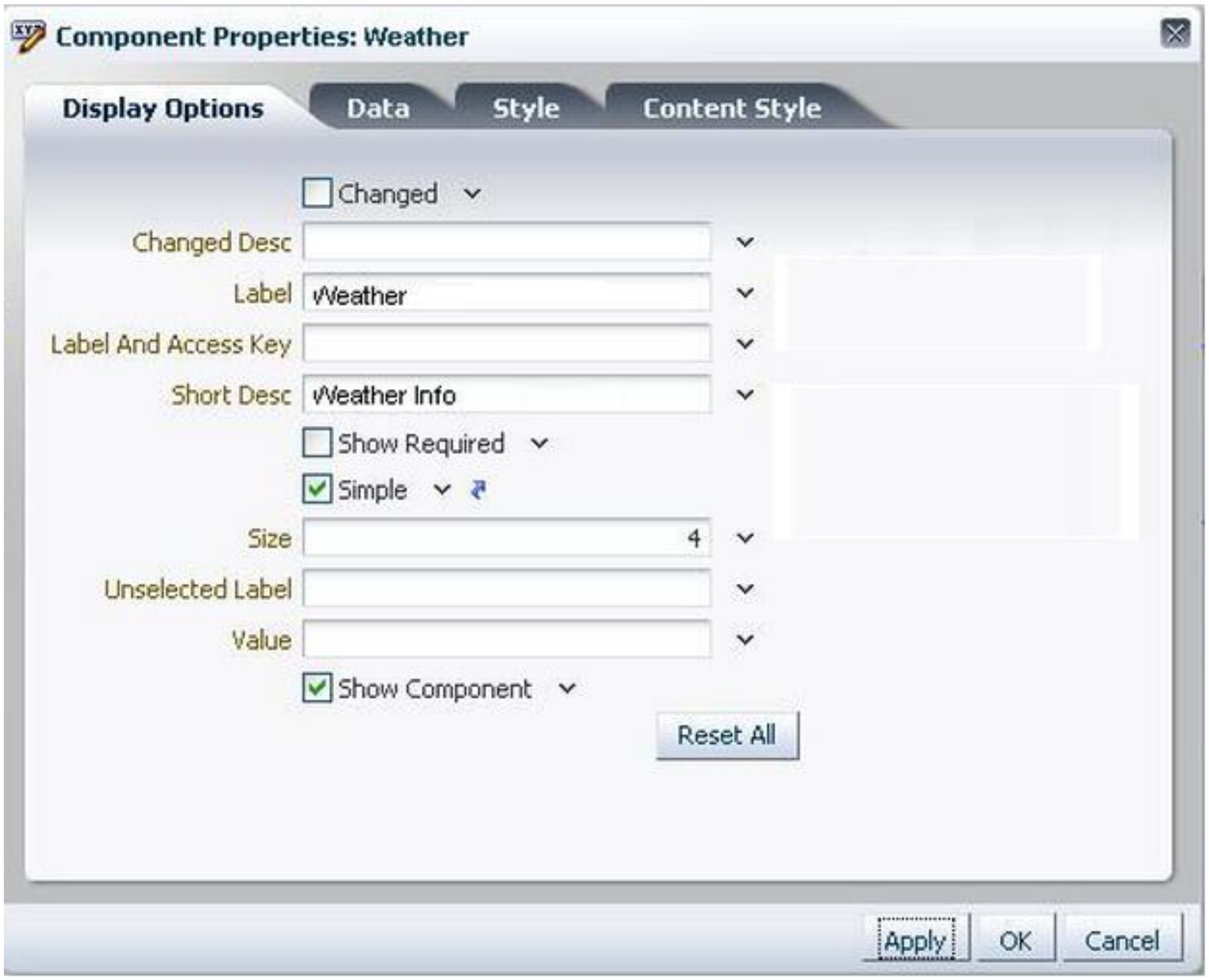
Typical components in the Resource Catalog are the front end for WebCenter Services - like Polls, Tagging, Search, RSS Feed Reader, Activity Stream and Content Items from a Content Repository - or pre defined common Web Development elements like rich text elements, images and hyperlinks. Custom ADF task-flows and static HTML components can also be included in the Resource Catalog by the application developers. Portlets from portlet producers that were registered with the ADF application will usually also be shown in the Resource Catalog.

One of the default Portlets shipped by Oracle is the WebClipping Portlet - an golden oldie dating back to the early days of Oracle Portal. This Portlet can be added to a page and subsequently be configured to 'clip' an external web page - rendering part of the live content of that external page inside the portlet area in the edited page.

Most components will be configured in the Composer after they have been added to the page, to determine what exactly they will display.

Web Center Composer also allows us to edit the majority of attributes of many items in the page, using static values or even dynamic EL expressions. We can for example change style, prompt or maximum value for the fields in a form - at run time. Items can also be hidden and reordered. Tabs can be added, reordered or removed and the contents of tabs can be edited.

Note that all changes made at run time can either apply to all users under all circumstances - the default behavior - or only apply under specific conditions such as selected user groups, day of the week or a user preference. This is governed by the ADF customization framework - more on that in a later installment of this column.


Component Properties: Weather

Display Options | Data | Style | Content Style

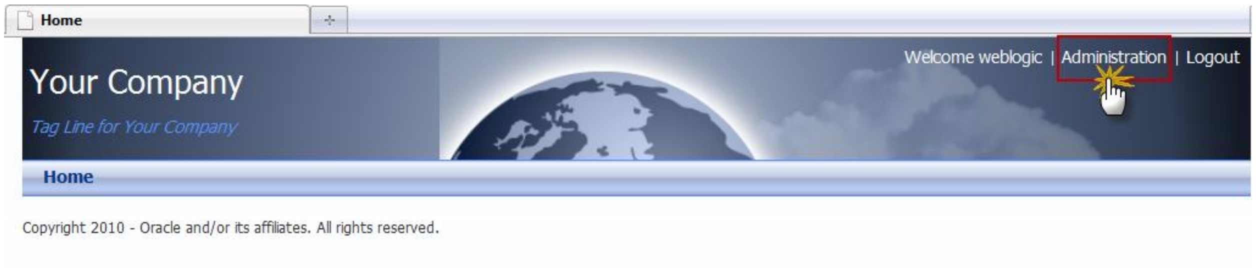
[ ] Changed
Changed Desc:
Label: Weather
Label And Access Key:
Short Desc: Weather Info
[ ] Show Required
[✓] Simple
Size: 4
Unselected Label:
Value:
[✓] Show Component

Screenshot: Editing the properties of a dropdown component in the Composer

Obviously for a run time application editor to be able to change application characteristics through specifying EL expressions on component attributes, requires some fairly advanced technical skills, way beyond those ordinarily found in content editors.

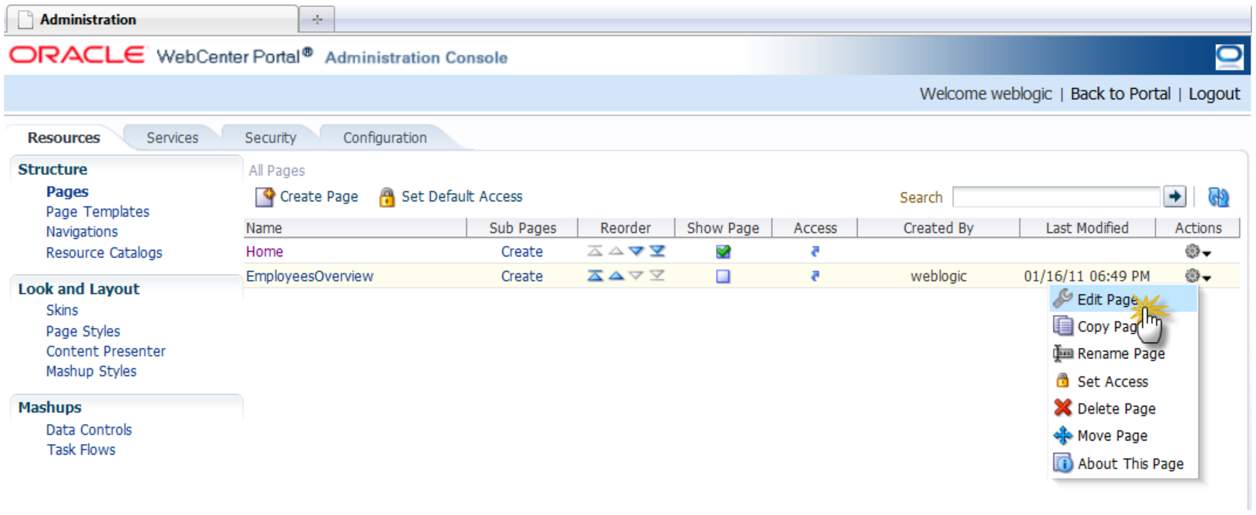
WebCenter Application Administration Console for Run Time Application Editing

WebCenter Patch Set 3 extended the range of application aspects that can be edited at run time quite dramatically. The Administration Console that can be opened by authorized users provides access to many different characteristics of the ADF web application. This includes: the ability to create (and delete) entire pages at run time, functionality to manage resource catalogs, a skin style editor, management of the navigation model for the application and editing the application's security model, creation of Data Controls and definition of Data Visualizations like tables, forms and charts.


Home
Your Company
Tag Line for Your Company
Welcome weblogic | Administration | Logout
Home
Copyright 2010 - Oracle and/or its affiliates. All rights reserved.

Screenshot: Enter the Application Administration console in the ADF Web application

The Administration Console is available only to users with appropriate privileges. These users have a very special role - somewhere between content editor and application developer.



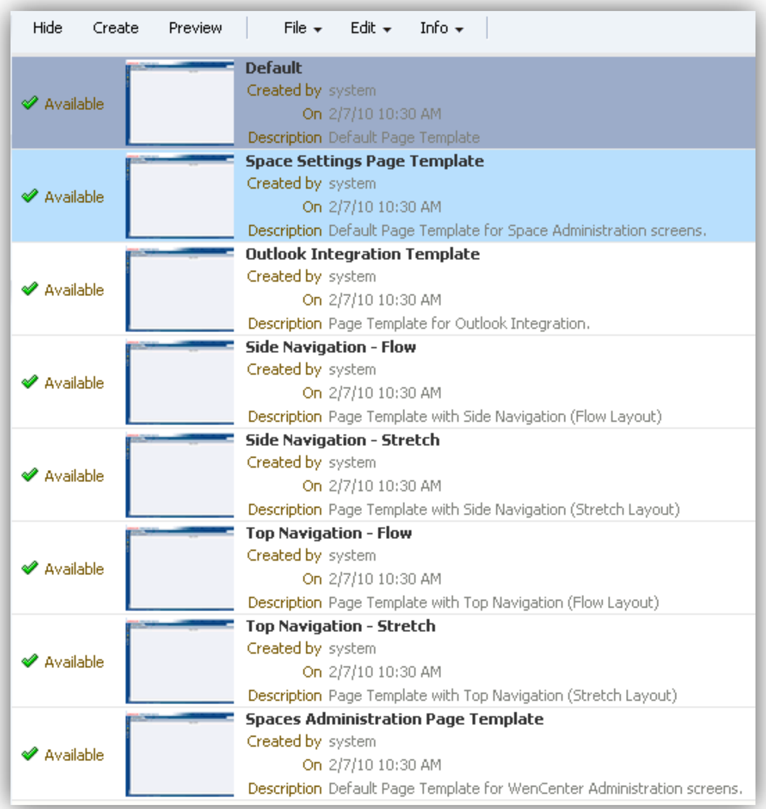
Name	Sub Pages	Reorder	Show Page	Access	Created By	Last Modified	Actions
Home	Create		✓				
EmployeesOverview	Create		☐		weblogic	01/16/11 06:49 PM	

Screenshot: overview of the Application Administration Console available in ADF applications that leverage WebCenter Services 11g PS3

The Page Service - also available in earlier 11g releases for Web Center - allows the creation at run time of new pages. A new page is based on a selected Page Template - see below - and can be organized using the Page Composer that we saw before. Components can be added from the Resource Catalog in addition to the content that is inherited from the template. Navigation to newly created pages can be added in existing pages through page links and is also available through the default navigation model. Fine grained access privileges for the new pages can be manipulated through the Administration Console.

Pages, newly created at run time or created at design time by the application developers, can be based on a page template. The structure and predefined content of the page template is an important aspect in determining a consistent over all look and feel for the application. Note that page templates, like so many other aspects of FMW Web Applications, can be dynamically selected based on the current context.

Page Templates can be created and edited at run time. Changes to an existing Page Template may have a large impact, depending of course on the change and the number of pages that is based on the template being modified. It is clear that through Page Template editing, a useful tool for design time at run time is provided.

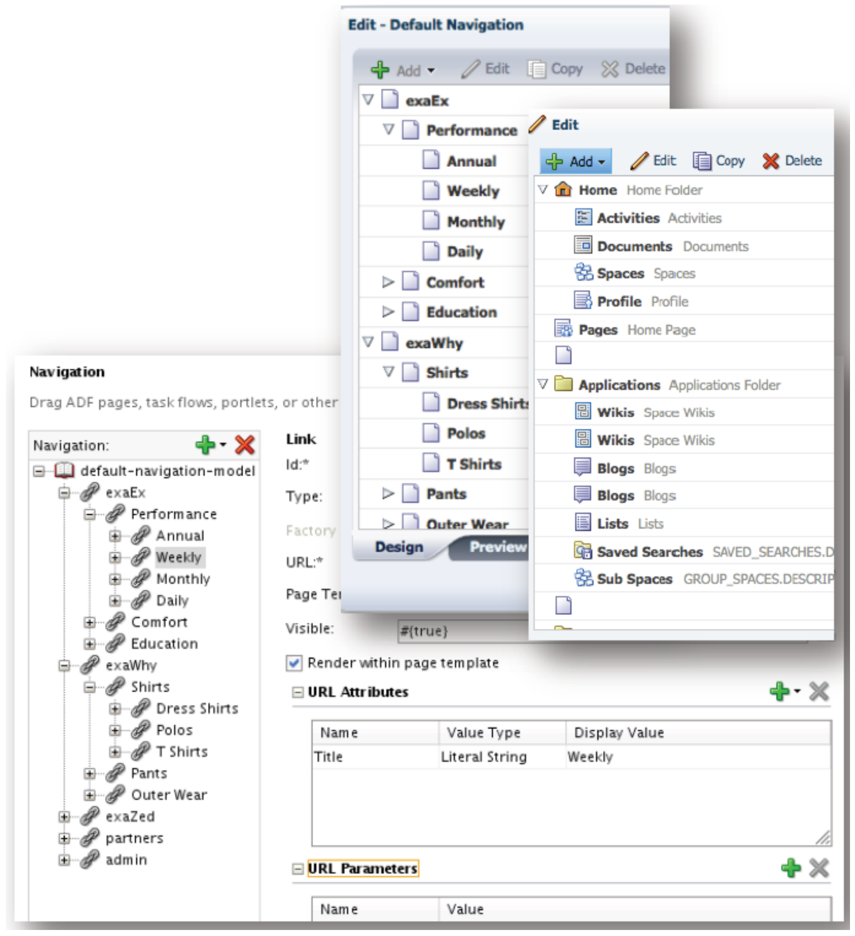


✓ Available	**Default** Created by system On 2/7/10 10:30 AM Description Default Page Template
✓ Available	**Space Settings Page Template** Created by system On 2/7/10 10:30 AM Description Default Page Template for Space Administration screens.
✓ Available	**Outlook Integration Template** Created by system On 2/7/10 10:30 AM Description Page Template for Outlook Integration.
✓ Available	**Side Navigation - Flow** Created by system On 2/7/10 10:30 AM Description Page Template with Side Navigation (Flow Layout)
✓ Available	**Side Navigation - Stretch** Created by system On 2/7/10 10:30 AM Description Page Template with Side Navigation (Stretch Layout)
✓ Available	**Top Navigation - Flow** Created by system On 2/7/10 10:30 AM Description Page Template with Top Navigation (Flow Layout)
✓ Available	**Top Navigation - Stretch** Created by system On 2/7/10 10:30 AM Description Page Template with Top Navigation (Stretch Layout)
✓ Available	**Spaces Administration Page Template** Created by system On 2/7/10 10:30 AM Description Default Page Template for WenCenter Administration screens.

Screenshot: run time management of Page Templates in the Application Administration console

Another important aspect of web [site like] applications is the navigation around the application. Through tabs, menu-trees, pull down menus and scattered hyperlinks as well as functional drill down options, users can find their way through the pages of the application. The navigation models that developers create at design time consist of static or dynamic, query based links to pages, portlets, task flows and content items from content management repositories . These models can be rendered, wholly or partially, in various ways - as menu, link list, tree, tabs or some custom representation.

The navigation models can be edited at run time through the application administration console, allowing immediate reorganization of the web application site structure and the user experience. New pages, content items and portlets can be inserted in the model, existing navigation items removed or rearranged.



Visible: #{true}

[✓] Render within page template

URL Attributes

Name	Value Type	Display Value
Title	Literal String	Weekly

URL Parameters

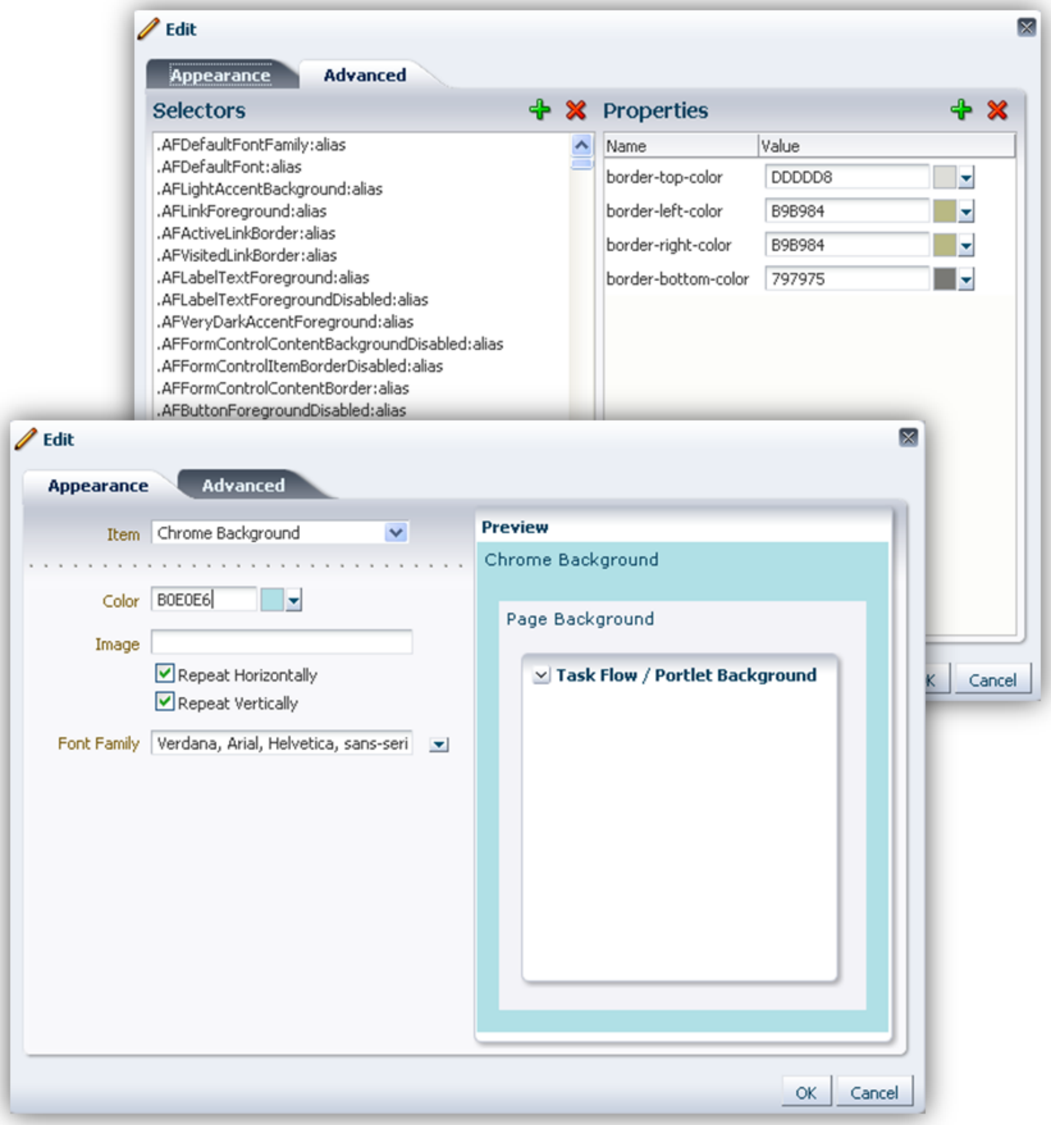
Name	Value

Screenshot: manipulating the Navigation Model for the application

Skins management

The visual appearance of ADF applications is governed by so called skins - based on CSS 3.0 stylesheet technology. Skins determine the styles that should be applied to all visual components, such as boilerplate text, buttons, input components and container elements like boxes, tabs, trees and accordions. Developers create one or more skins to be deployed with the application. The active skin can be dynamically selected based on conditions expressed through EL expressions.

The WebCenter Application Administration console makes it possible for application editors to manipulate the application skins at run time: style properties can be changed, previewed and applied from within the browser.



.AFDefaultFontFamily:alias
.AFDefaultFont:alias
.AFLightAccentBackground:alias
.AFLinkForeground:alias
.AFActiveLinkBorder:alias
.AFVisitedLinkBorder:alias
.AFLabelTextForeground:alias
.AFLabelTextForegroundDisabled:alias
.AFVeryDarkAccentForeground:alias
.AFFormControlContentBackgroundDisabled:alias
.AFFormControlItemBorderDisabled:alias
.AFFormControlContentBorder:alias
.AFButtonForegroundDisabled:alias

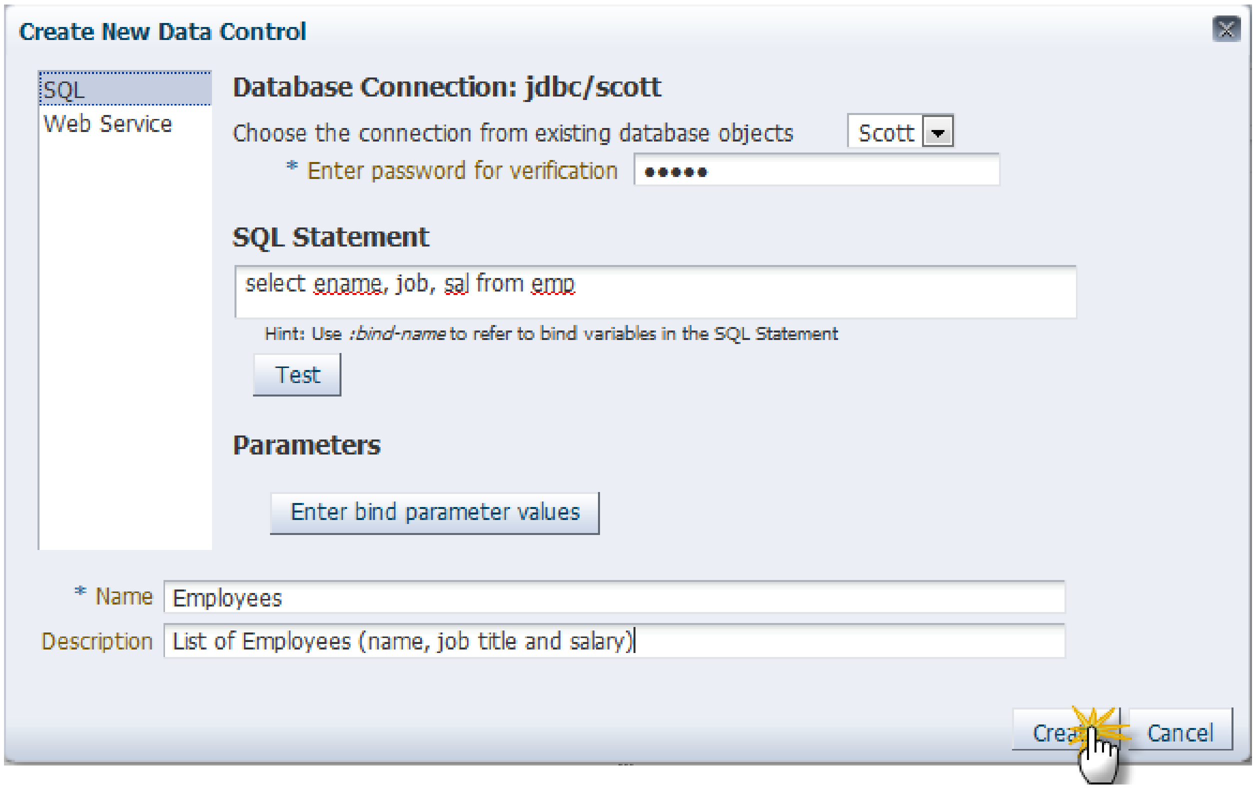
Name	Value
border-top-color	DDDDD8
border-left-color	B9B984
border-right-color	B9B984
border-bottom-color	797975

Item	Chrome Background
Color	B0E0E6
Image	
	☑ Repeat Horizontally
	☑ Repeat Vertically
Font Family	Verdana, Arial, Helvetica, sans-seri

Screenshot: the Skin editor in the WebCenter Application Administration console that allows for changing and skin style properties and live previewing the effects of changes

Creating data control and data visualization

A fairly spectacular piece of functionality in the whole palette of design time @ run time features available in WebCenter as of Patch Set 3 is the ability to create a Data Control at run time. Either by specifying a SQL query against tables and views found in one of the predefined Data Sources or by accessing a WebService, the run time application editor can configure a Data Control. The data produced by the Data Control can be filtered, sorted and subsequently visualized in various ways in the Web Application. Note: readers who have used the Oracle Portal OmniPortlet will see strong similarities here.



Database Connection: jdbc/scott

Choose the connection from existing database objects: Scott

* Enter password for verification: •••••

SQL Statement

select ename, job, sal from emp

Hint: Use *:bind-name* to refer to bind variables in the SQL Statement

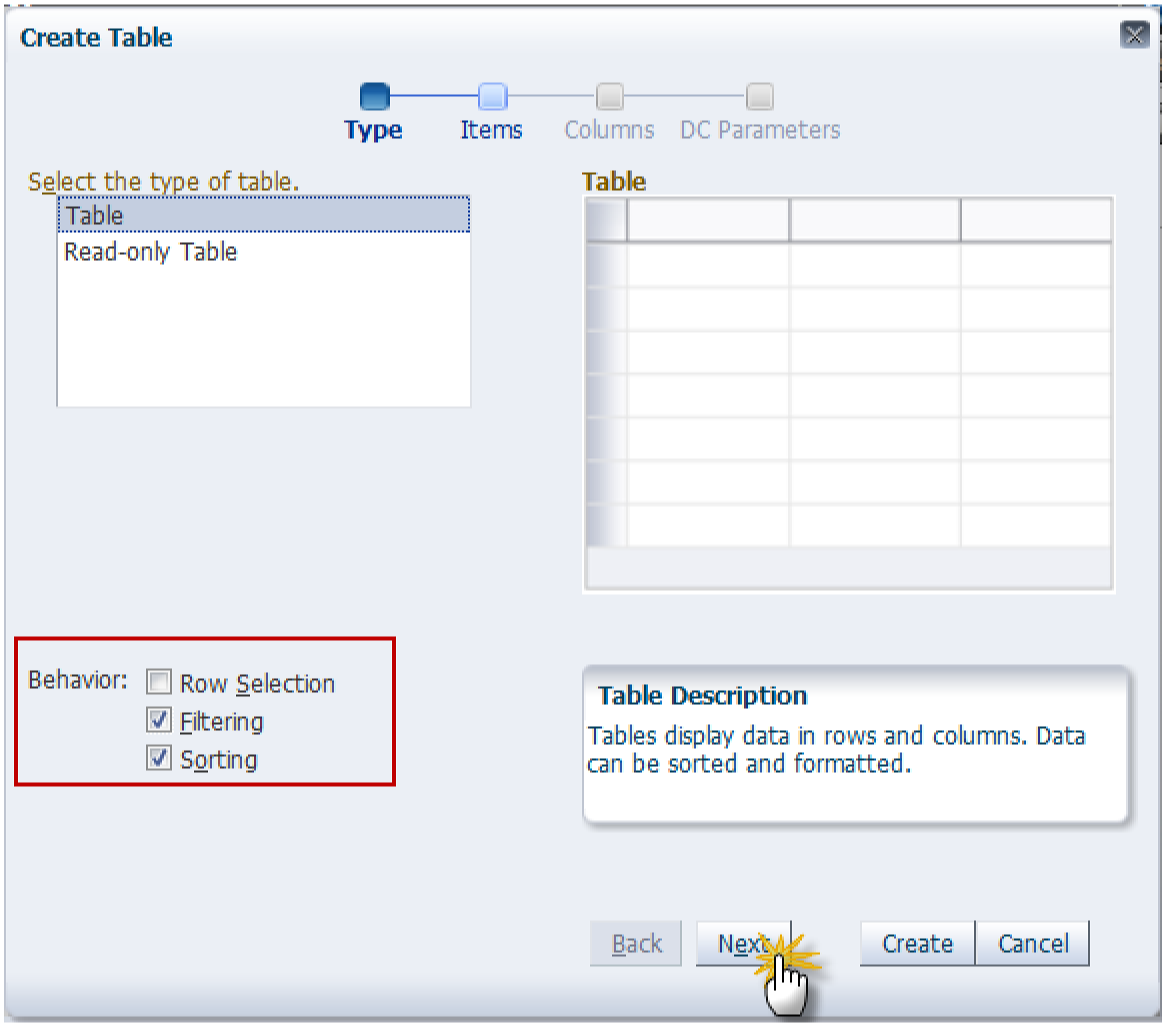
Parameters

* Name: Employees

Description: List of Employees (name, job title and salary)

Screenshot of the New Data Control wizard that can be launched from the WebCenter Application Administration Console - at Run Time

Data Controls make collections of data available that can subsequently be used in Data Visualizations that can be added to a page in *edit* mode. Content can be added to a page in edit mode, selected from the Resource Catalog. Data Controls that were dynamically created in the application can be found in the catalog and various visualizations options are available for a data control: table, form and graph. The data control's visualization can be fine tuned - by specifying visual attributes as well as through filters and sort criteria.



Select the type of table.

[✓ Table]
Read-only Table

Behavior:
☐ Row Selection
☑ Filtering
☑ Sorting

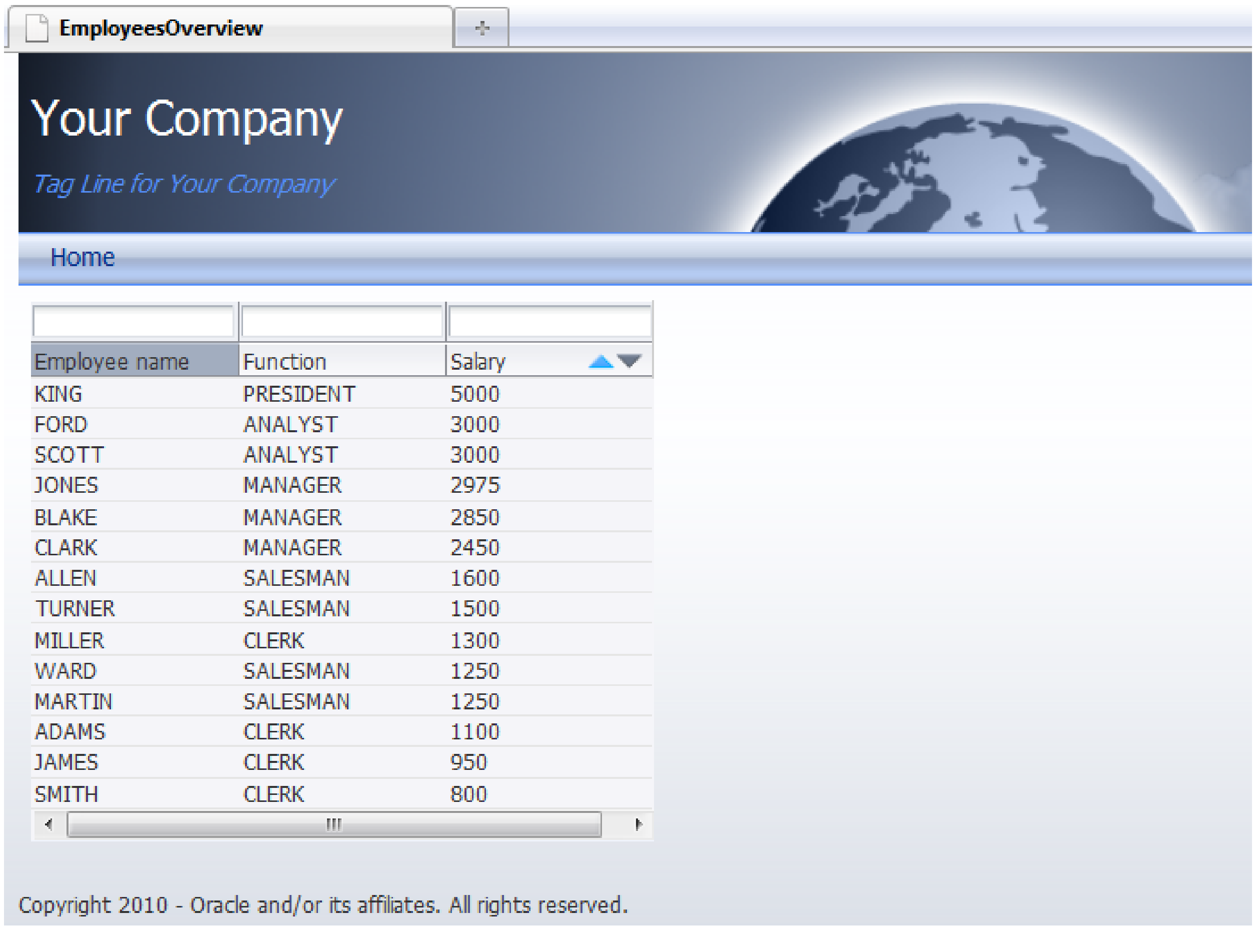
Table Description

Tables display data in rows and columns. Data can be sorted and formatted.

Screenshot of the Create Table wizard that was invoked from the Resource Catalog by opting to add a visualization of type Table based on the Employees collection exposed by the Employees Data Control

The wizard allows the editor to specify various display characteristics such as the display style, the column header and potentially the deeplink characteristics of the cell-values.

When the wizard is finished, the visualization of the data control is added to the page. It can be further refined in subsequent edit actions. Note that one data control can be the foundation for multiple visualizations. Also note that the data control as well as a specific visualization can expose bind parameters that can be set from other visualizations in the page or from other taskflows and portlets.



Employee name	Function	Salary
KING	PRESIDENT	5000
FORD	ANALYST	3000
SCOTT	ANALYST	3000
JONES	MANAGER	2975
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
MILLER	CLERK	1300
WARD	SALESMAN	1250
MARTIN	SALESMAN	1250
ADAMS	CLERK	1100
JAMES	CLERK	950
SMITH	CLERK	800

Copyright 2010 - Oracle and/or its affiliates. All rights reserved.

Screenshot of the Table style visualization of the Employees data control - fully created at run time

See http://technology.amis.nl/blog/10555/webcenter-11gr1-ps3-design-time-at-run-time-with-a-vengeance-introducing-run-time-data-controls-and-data-visualizations for a more detailed introduction of the Data Control functionality at run time.

In Line Content Editing based on UCM

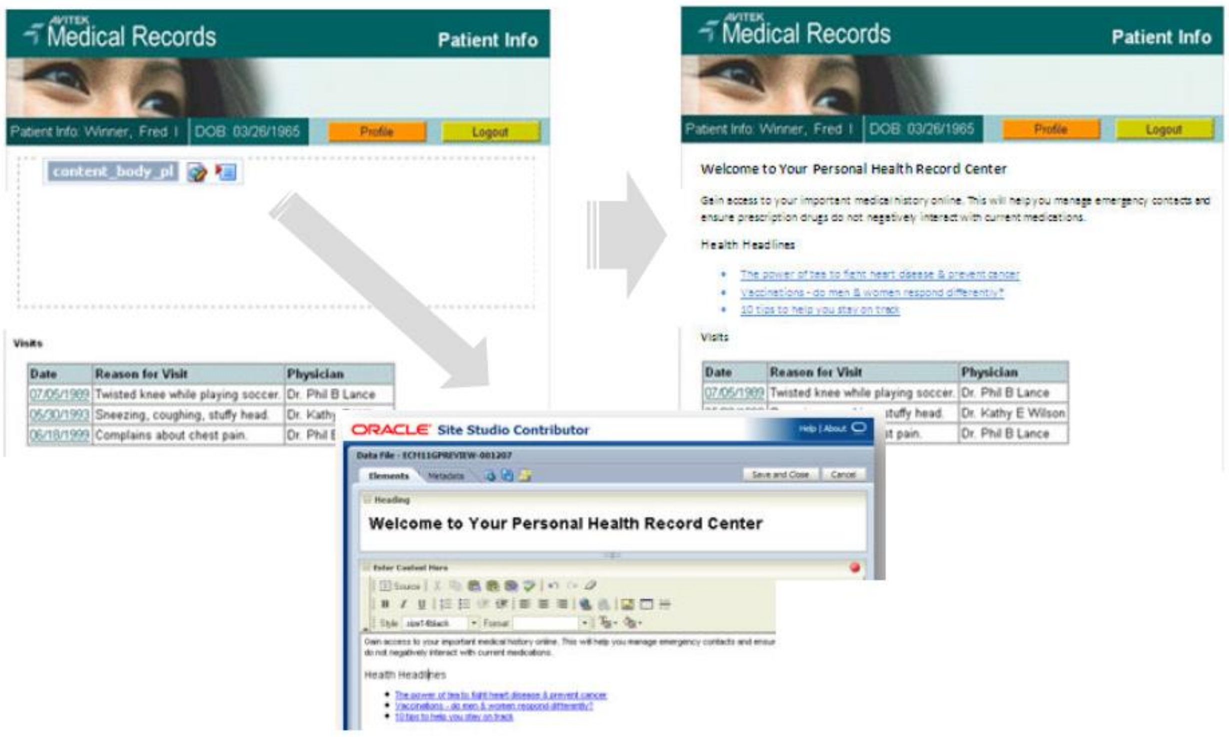
The elements presented in the pages of a Web Application can be part of the definition of the application - such as fields, buttons and tables - as discussed in the previous section. Other elements in the application can be retrieved from underlying enterprise resources, such as databases and web services. A special category of page content elements consists of semi-static components originating from a [web site] content management system such as Oracle Universal Content Manager. Such elements can be maintained by content editors and marketing staff or business representatives - outside of the context of the application - yet immediately influence the pages exposed by the web application.

Site Studio [Contributor] for External Applications and Open WCM (Web Content Management) tags allow integration of content items as well as region templates from UCM Content Server in web applications. Content items can be images or fragments of rich (HTML) text that are presented in a web application page, documents such as PDF, Word, Excel or Powerpoint that are dynamically be converted into rich HTML representation that is embedded in a page or the items can be presented as hyperlinks or thumbnails for files to be downloaded or opened in a popup window.

When a content item is revised in whatever way - for example programmatically by an application through the UCM APIs or edited and checked in from MS Word through UCM Desktop Integration - the revised item will be visible as soon as it is approved and available. This means that text and images displayed in web applications as well as documents available for download from applications can be changed at run time by content editors with no specific technical knowledge whatsoever. Note that there is no limit on how small a content item could be - even a single logo, icon or title could be defined through a content item in UCM, making it a candidate for easy run time manipulation.

The selection of content items can be specified through EL expressions that are dynamically evaluated. That means that by changing tags or meta attributes on content items or by altering other aspects that impact the outcome of the EL expression, other content items can make their appearance in the web application - changing the information presented to the end user. For example the 'hot top three' query against UCM - based on meta data specified against content items - may result in a new selection of images or documents after a marketing specialist has changed some tags in UCM.

Additionally, web pages - such as ADF Faces applications - can be switched to content editing mode. In this mode, authorized users can edit content items and their appearance in line in the web page. Any changes are saved directly to the underlying UCM Content Server. This provides a very accessible, WYSIWYG way of manipulating 'static' content in web applications.



Welcome to Your Personal Health Record Center

Gain access to your important medical history online. This will help you manage emergency contacts and ensure prescription drugs do not negatively interact with current medications.

Health Headlines

- The power of tea to fight heart disease & prevent cancer
- Vaccinations - do men & women respond differently?
- 10 tips to help you stay on track

Visits

Date	Reason for Visit	Physician
07/05/1989	Twisted knee while playing soccer.	Dr. Phil B Lance
05/30/1993	Sneezing, coughing, stuffy head.	Dr. Kathy E Wilson
06/18/1999	Complains about chest pain.	Dr. Phil B Lance

Screenshot with web application leveraging Site Studio Contributor that allows in-line editing of the content item by privileged user. On the left you see the page as seen by the content editor, on the right the result displayed to the end user.

WebCenter and UCM

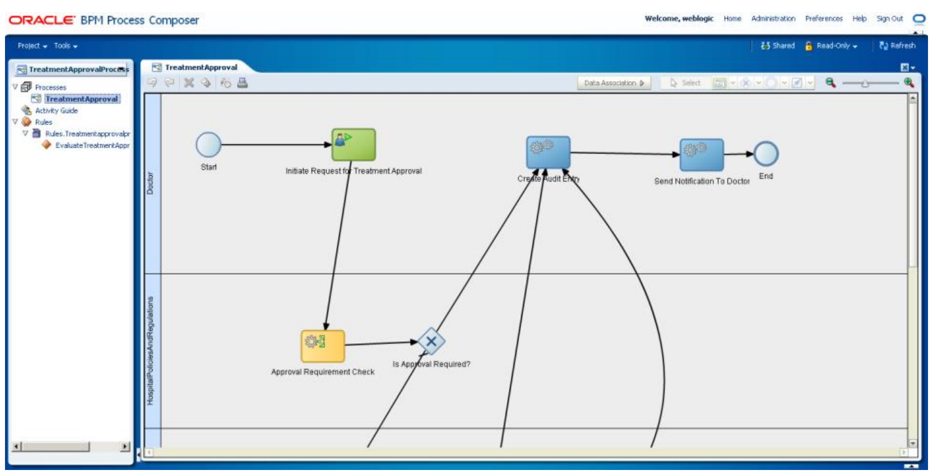
Content items like images, rich text fragments and documents can also be integrated into FMW web applications using the Document Services in ADF applications extended with WebCenter. These services can be added to an application page at design time in the form of a task flow or they can be introduced at run time in the Page Composer from the Resource Catalog. The Document Services provide various ways of exposing content from and interacting with the underlying Content Repository - which is typically UCM Content Server but can also be Sharepoint, Oracle Portal or a JCR-1/JSR-170 compliant content management system. When a Document Service is added to a page - at design or run time - it needs to be associated with a content repository and it subsequently needs to be configured with the specific content item(s) or folders it should expose.

Through the Document Explorer, the contents of a subset of folders from the content repository is exposed in the Web Application, very much like the Windows Explorer. Files can be opened, renamed, moved, deleted, tagged and added through upload. The Content Presenter services enables you to precisely customize the selection and presentation of content from the Oracle UCM Content Server. It can present content in various ways and formats, leveraging the conversion capabilities - for example Word to HTML or PowerPoint to PDF - in Content Server. Another example of a Document Service is the Document Viewer task flow that displays a preview of a file, or file properties for files that do not support a preview.

Note that all Document Services can be added and edited at run time from the Composer.

BPM Composer

In addition to BPM Workspace - for tracking running and completed process instances - and Process Spaces - a WebSpaces environment for collaborating on process instances and process definitions - there is a third browser based, run time tool available with Oracle BPM 11g. The BPM Composer is used to publish BPMN based process definitions. Furthermore, business analysts and others can use BPM Composer to also edit a process definition. In more or less the same way as in BPM Studio (JDeveloper), a business process can be modified, extended and annotated. Some changes requires additional implementation - such as the introduction of entirely new process activities. The process definition will need to be handed over to BPM Studio (aka JDeveloper) for implementation by the developer and no immediate run time impact is wrought.



Screenshot: editing a business process in the browser based BPM Composer

However, many other changes, like modifications in flow conditions, the removal of steps or introduction of gateways and flows , can be deployed to the run time environment from within BPM Composer, without interference of a developer in a design time environment. That means that through BPM Composer, potentially non-technical staff can modify the logic of business processes at run time.

In a near future release of Oracle BPM, it will be possible to also run simulations in the BPM Composer. Additionally, BPM Composer will provide compare and merge facilities for various revisions of a process definition and will make it possible to apply changes to already running instances of the changed process.

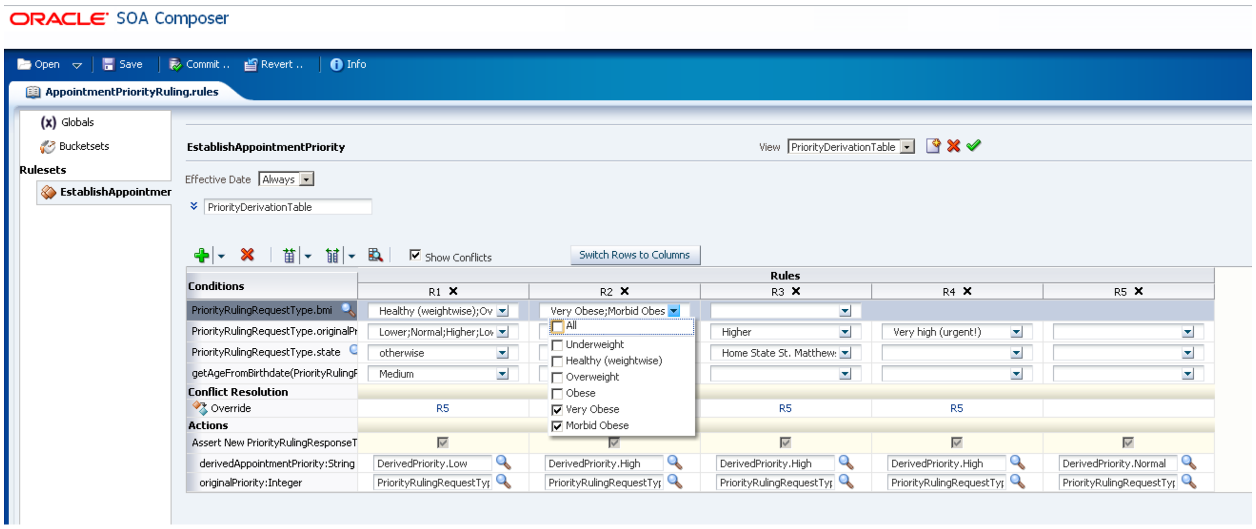
SOA Composer

The SOA Suite comes with yet another composer: the SOA Composer. This too is a run time application, installed along with the SOA Suite run time environment. The SOA Composer supports the run time manipulation of three resources: Business Rules, Domain Value Maps and Human Task definitions. Changes made in the SOA Composer can initially be saved - across sessions but without run time effect - and when complete subsequently committed. When they are committed, they have an immediate effect on new and running instances - any reference made to the resource from that moment on will deal with the changed definition.

The SOA Composer is a business user friendly tool. Without deep technical insights or steep learning curves, business application managers will be able to edit the behavior of various aspects of composite applications.

A useful service component that helps implement decision logic - including calculations, validations, derivations and rulings - is the Business Rule. A Business Rules is a solidified piece of business logic, expressed using dynamic data facts and static conditions and formulas. Examples of business rule are the determination of a hospital appointment's priority for a patient, the selection of the warehouse an order should be shipped from, deciding on the outcome of a game of *rock, paper and scissors* or the derivation of the sell by date of a dairy product.

Business Rules evaluate one or multiple conditions given the current values of the facts involved, and produce the outcome using expressions. Through SOA Composer, the static elements of the rules - the conditions and outcome expressions - can be changed at run time. This enables a run time application editor to change the logic in a SOA application, through a simple browser based tool, without redeployment.



EstablishAppointmentPriority

View: PriorityDerivationTable

Effective Date: Always

PriorityDerivationTable

☑ Show Conflicts

Conditions	R1	R2	R3	R4	R5
PriorityRulingRequestType.bmi	Healthy (weightwise);Ov	Very Obese;Morbid Obes			
PriorityRulingRequestType.originalPr	Lower;Normal;Higher;Lov		Higher	Very high (urgent!)	
PriorityRulingRequestType.state	otherwise		Home State St. Matthew:		
getAgeFromBirthdate(PriorityRulingF	Medium				
Conflict Resolution					
Override	R5		R5	R5	
Actions					
Assert New PriorityRulingResponseT	✓	✓	✓	✓	✓
derivedAppointmentPriority:String	DerivedPriority.Low	DerivedPriority.High	DerivedPriority.High	DerivedPriority.High	DerivedPriority.Normal
originalPriority:Integer	PriorityRulingRequestTyp	PriorityRulingRequestTyp	PriorityRulingRequestTyp	PriorityRulingRequestTyp	PriorityRulingRequestTyp

☐ All
☐ Underweight
☐ Healthy (weightwise)
☐ Overweight
☐ Obese
☑ Very Obese
☑ Morbid Obese

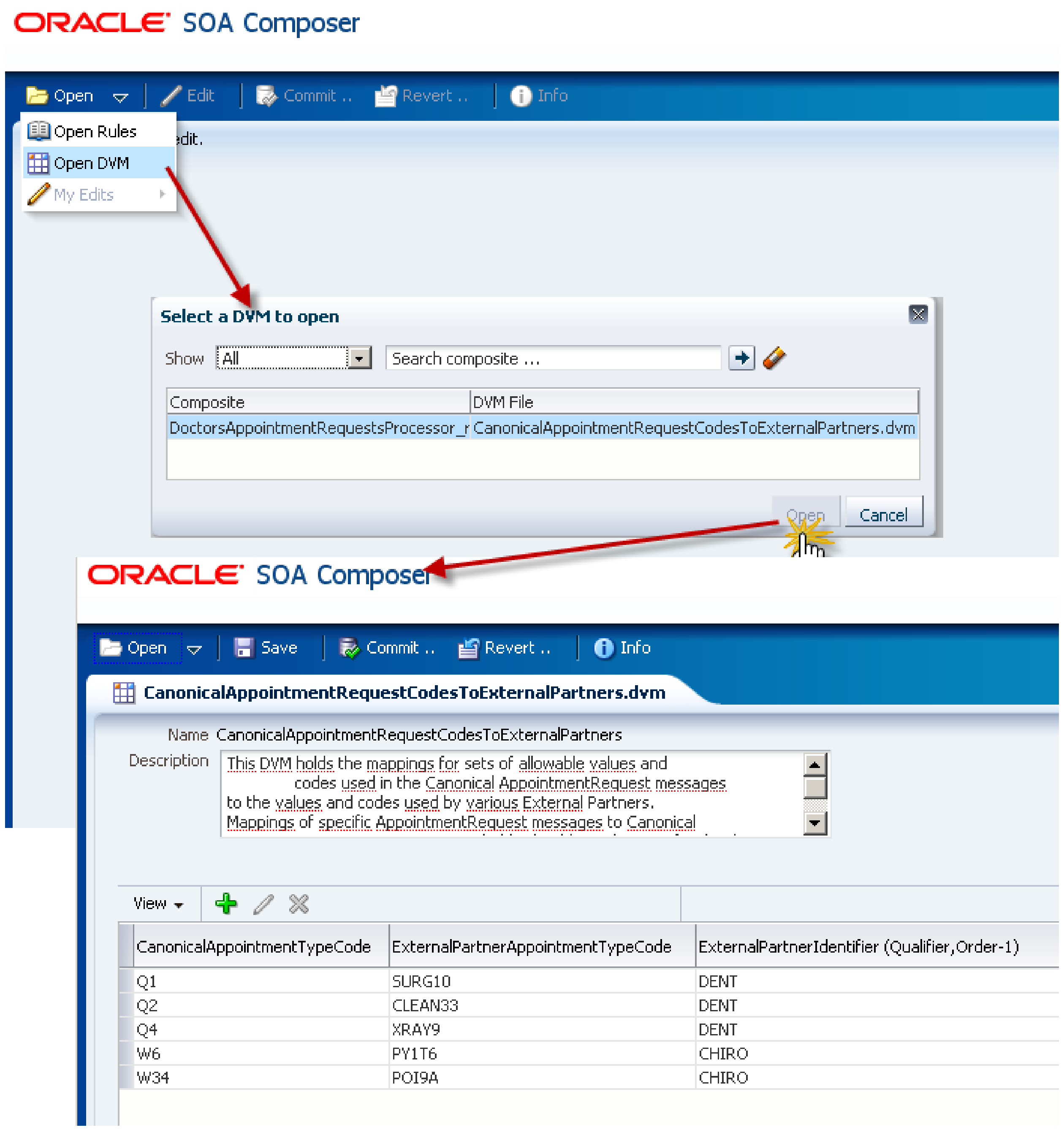
Screenshot: Editing a Decision Table style Business Rule in the SOA Composer run time tool

Enterprise Manager SOA Console also provides facilities for run time management of certain properties of SOA Composite applications. The end points of remote services can be set for example, and adapter binding properties such as File System Directory or Polling Interval can be edited, as well as custom BPEL properties. Attaching Policies - shipped or custom built - to WebService Bindings is another way of run time editing of SOA Composite applications, available through the Enterprise Manager console.

Run time changes made through SOA Composer or the Enterprise Manager SOA Console can be exported from the run time environment. The resulting service archives can be imported in JDeveloper and subsequently uploaded (and merged) into the source code control system.

Domain Value Maps

A very useful resource that can be edited in the SOA Composer is the Domain Value Map (DVM). DVMs are very simply put just straightforward key-value pairs. Domain Value Maps can be accessed from XPath expressions using the special Oracle XPath extension function. DVMs can be referenced for example in Transformations in BPEL, BPM and Mediator, in Mediator Filter expressions, BPEL Assign activities and BPM data associations. That means that whenever you may be tempted in any of these places to use a hard coded value, you could and probably should consider using a DVM reference instead - because that will then provide the run time modification option for free, through the SOA Composer!



Composite	DVM File
DoctorsAppointmentRequestsProcessor_r	CanonicalAppointmentRequestCodesToExternalPartners.dvm

Name CanonicalAppointmentRequestCodesToExternalPartners

Description This DVM holds the mappings for sets of allowable values and codes used in the Canonical AppointmentRequest messages to the values and codes used by various External Partners. Mappings of specific AppointmentRequest messages to Canonical

CanonicalAppointmentTypeCode	ExternalPartnerAppointmentTypeCode	ExternalPartnerIdentifier (Qualifier,Order-1)
Q1	SURG10	DENT
Q2	CLEAN33	DENT
Q4	XRAY9	DENT
W6	PY1T6	CHIRO
W34	POI9A	CHIRO

Screenshot: Editing a Domain Value Map definition in the run time SOA Composer tool

Design for Change

When designing your applications, you may want to consider behavioral aspects or look and feel characteristics that you want to allow to be managed at run time. Such a 'design for change' approach should be done explicitly as changeability does not come natural to an application. This typically means that during analysis and UI design activities, you need to include an additional task: in the user interaction design you need to include an indication for aspects whether or not they should be changeable. ADF applications that explicitly create customizable areas using WebCenter Composer have a lot of intrinsic run time change capacity, but in other applications such capability needs to be explicitly coded into the application.

Note: because run time changes can be applied specifically to selected user groups, under special conditions or for users with a special preference, this indication also governs the question whether specific characteristics are customizable.

Typical candidate changeable aspects are display properties like visible(Y/N), style, prompt, default value, hint and the positioning and order of elements. In addition to these visual aspects, many applications implement logic, for example for validations, calculations and decisions. Such logic may also be a prime suspect for run time modification. It is useful to identify and centralize such logic and parameterize the implementation as much as possible. In addition, the values of the parameters need to be editable at run time - for which the application requires an administration facility.

Note that many aspects of ADF Faces applications are defined through EL expressions. Such expressions are dynamically evaluated, making use of the actual context. By having the EL expressions based on settings the users or application composer can manipulate - indirect control over look & feel or behavior (whatever the EL expressions control) is made available.

ADF developers have another tool in their toolbox to implement run time editability into their applications- even if they do not make use of Web Center Composer - by leveraging the Change Persistence framework. This framework can be used at run time to (programmatically) apply changes to ADF Faces components, to persist these changes across sessions in the MDS Repository and to reapply these changes for a new session. Note that changes persisted in this way can be associated with specific users, user groups or dynamic conditions and can therefore be selectively applied to support run time customization and even personalization (fine tuning the application to the desires of individual users). Also be aware that ADF application developers need to create the administration facility or run time editing tool to gather and record the changes to the ADF Faces components that need to be persisted.

Custom Application facilities

Even though FMW offers many built-in composers which provide the design time at run time functionality, there are still many areas that are not [yet] covered by these editors. However, it can be quite simple to extend the run time editing to custom aspects of your application. Generic guidelines are not easy to provide, but some pointers may be useful:

- hard coded values in your applications are candidates, not just for turning into system parameters but more explicitly into adjustable parameters; Java applications can benefit from the JMX infrastructure in the JVM that allow objects to be published as MBeans that in turn can expose methods and attributes that can be invoked or manipulated through consoles and from external code; the behavior or functionality that these parameters influence are thus open to change at run time. See for example http://blog.frankel.ch/tag/ehcache for some examples of using JMX to expose application configuration at run time

- SOA applications can define properties - for example in BPEL and Mediator components - that can be used in the application to guide behavior (filters, transformations, validations, ...) and that can be set through the Enterprise Manager SOA Console, immediately impacting the deployed applications; you can engrain run time changeability into your SOA applications by expressing behavior using these properties - in addition to the standard binding and component properties such as service end-points.
- not yet supported by any FMW design time at run time facilities (although likely to be supported in the near future I believe) is the management of resource bundles, responsible for the rendering of boilerplate text - prompts, button labels, titles, error messages - in web applications, as well as the internationalization of these values; resource bundles are typically cached by the JVM itself - usually read from property files - upon the first access; however, several tricks are available to built 'resource bundle refreshability' into your web application. See for example this blog-article: http://technology.amis.nl/blog/1360/testing-i18n-features-of-jsf-applications-forcing-a-refresh-of-the-resource-bundles. Note that when resource bundles are deployed as property files, a change can only be accomplished by redeploying the web application after changing the resource bundle entries - definitely not a zero-downtime approach; you may therefore want to consider storing resource bundle entries in a database table, perhaps as a complement to the property files.

The Run Time Application Editor

The software release process over time has evolved into a fairly well established routine: develop, test (in several stages) and deploy - followed by run time monitoring and administration. New or changed functionality is realized by developers who will then hand the changed software over to testers and eventually to administrators for deployment. With design time at run time, the game changes quite dramatically. No developers - or testers and administrators - need to be involved in order to realize changes in the application and no formal process is mandated by the technology stack.

Several questions arise: who makes these run time changes, how are they tested and released and how are they fed back to the developers to at least ensure that the next regular release does not undo these run time changes.

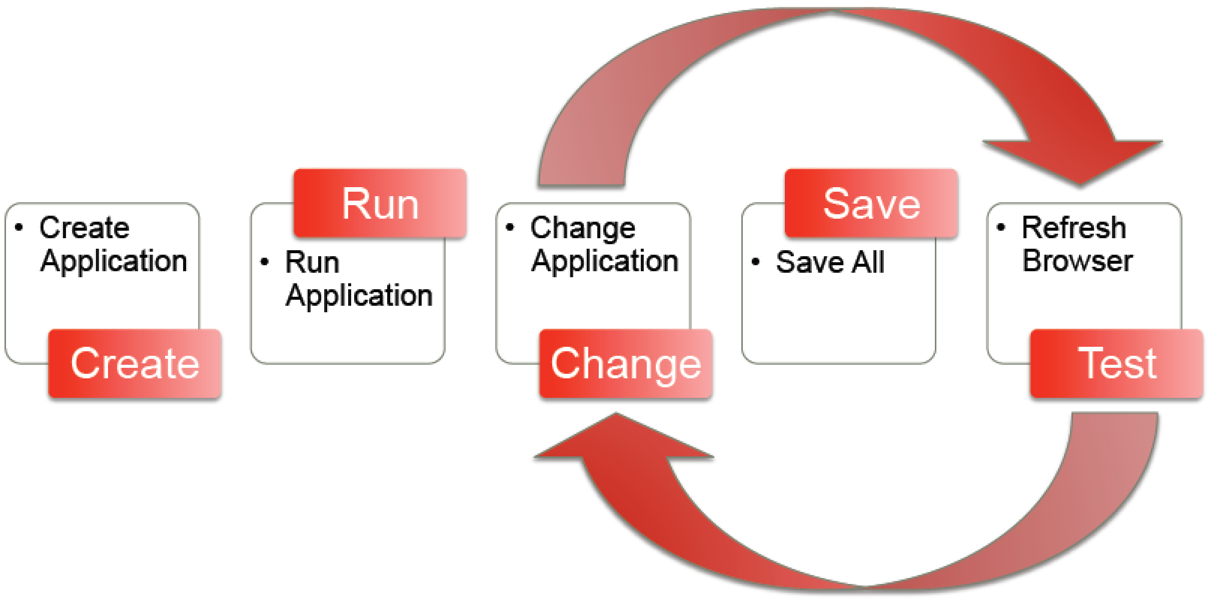


Figure: the software traditional develop, deploy and run loop will change because of design time at run time

The first question is quite an interesting one. Traditionally, the run time environment for enterprise application s has been the domain of system administrators monitoring infrastructure and application behavior at a very technical level. Application managers provide some functional support for end users, do some application configuration work, help resolve operational [data] issues (frequently using TOAD or some such tool to directly manipulate the database) and gather requirements for bug fixes and functional enhancements. Web sites typically also have editors who work on the content displayed in the site and usually the layout of the pages and the style of the site as well.

With the facilities offered by Fusion Middleware - and the demands from the business and the end users - there seems to be a new role to be defined and subsequently filled. We are looking for someone to liaise with users and business stakeholders - to spot and define changed requirements in terms of process flow, business logic, content and user experience. And that same someone to implement those requirements using the composer tools and other editing facilities available with FMW - requiring fairly deep technical skills on top of the broad and probably deep functional awareness. Finally, that person should also work together with the development organization to reconcile run time changes with development efforts. It probably requires a five legged sheep as the Dutch saying goes, a regular jack of all trades.

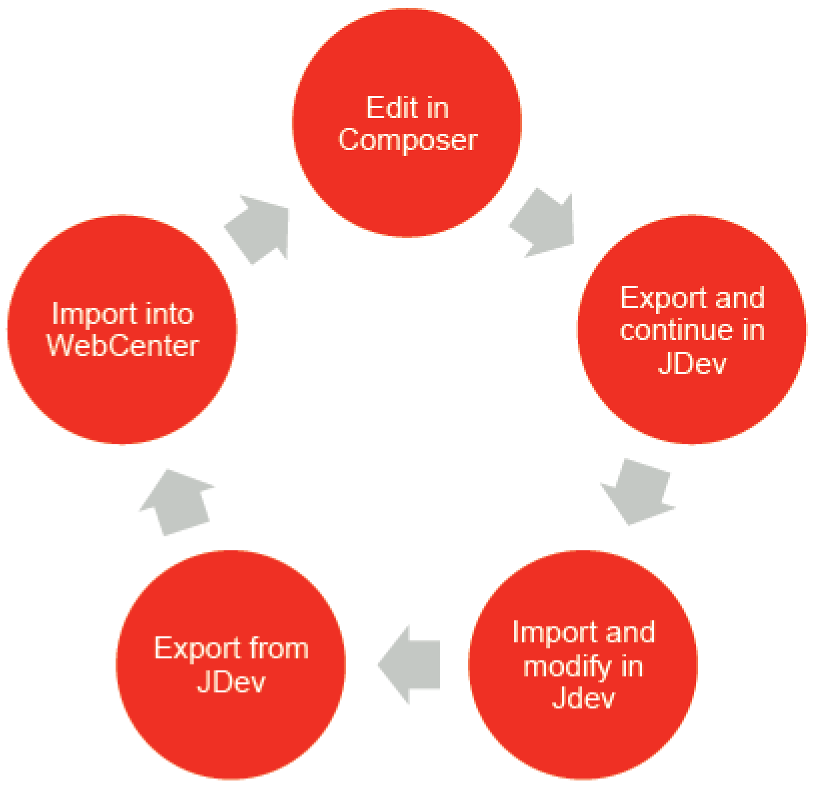


Figure: changes created at run time can be isolated, exported and passed to the development environment for reconciliation and further refinement; the results can be imported into the run time environment - outside of regular deployment procedures

To define an appropriate role title is somewhat of a challenge in itself. The design time at run time tools are predominantly called *composer* so Application Composer seems an obvious one. Considering the similarities with the web site *content editor* role, perhaps *application editor* is an option. The role is much more active than the current *application manager* role - perhaps *application administrator* is more apt? Or *run time developer*? I am wide open to suggestions. I believe it is important to clearly define the role - not just the title of course - as I strongly feel that the concept of changing important aspects of the behavior and look & feel of our systems is crucial for optimal agility. Clear definitions will help in establishing the role, adopting the way of working and leveraging the capabilities of today's technology.

Summary and Conclusion

It's the age of agile. Adapt or perish (or at least flourish less). That is the adage for many organizations. Competition is tough, pace of change is relentless and end users - especially consumers - are ever more demanding. In this climate, organizations need to continuously evolve their products and processes and typically both their internal and customer facing computer applications as well. Short time to market for small changes - and frequently for substantial modifications too - is critical.

Through agile software development methodologies and software engineering tools, application development has speeded up quite dramatically. However, a large category of requirements of change does not really warrant any development cycle at all: changing an image, hiding a field, adding a hyperlink, posting a fresh news item, publishing a new or changed document, altering the discount percentage, modifying the title of a page, readjusting the font for a specific text area - such changes are relatively small, are understood by business users, do not require very formal testing and should really be applied and published in minutes or hours at the most rather than days, weeks or even months.

The concept of *Design Time at Run Time* refers to changes in application appearance and behavior that are fairly small, superficial almost, and meaningful in terms of business value. That business users can apply, without too much technical knowledge. Through design time @ run time capabilities, such changes can be applied to an application in the live production environment, typically even in the page itself. Committing or publishing the change from a sandbox or preview mode, means bringing the change immediately live - without formal deployment process or acceptance testing procedures.

Content editor has been a common enough role in web sites that contain a lot of static content. This content could be edited at run time to continuously refresh parts of the site without deploying an entire new site for every little change. Design time at run time stretches this role to one that has potentially much more impact, not just on the content of a web site but on many behavior aspects of web applications and business processes. A role that is much closer to application developer than content editor ever was. A role that requires a lot more technical skills and is potentially very valuable for organizations in their quest for agility.

The software development process will have to be modified to cater for the design time at run time approach. For one, it means that changes to the application can originate not just in the design time development environment, but in the run time production environment as well. These changes need to be captured and fed back into the more formal development cycles. Additionally, to optimally benefit from design time at run time, applications need to be designed and implement for it: they must expose hooks for manipulating the behavior and look & feel of the application at run time.

Oracle Fusion Middleware ships with a range of design time @ run time facilities and tools, such as the Composers: WebCenter Page Composer, SOA Composer and BPM Composer. These provide built-in capabilities for ADF & WebCenter applications as well as for SOA Composite applications and BPM processes. ADF also provides the native Change Persistence framework that supports custom, programmatic design time @ run time features.

In short, design time at run time has the potential to provide a large degree of agility to organizations that have the courage and find a way to break out of the traditional approach to software development and run time application administration.