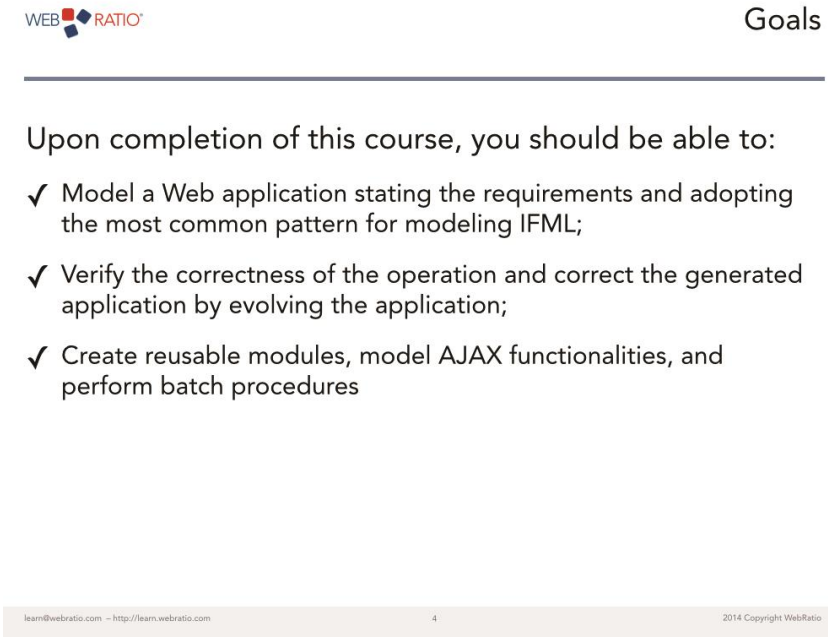
WebRatio Platform Modeler

University Kit - Student Guide

About this course

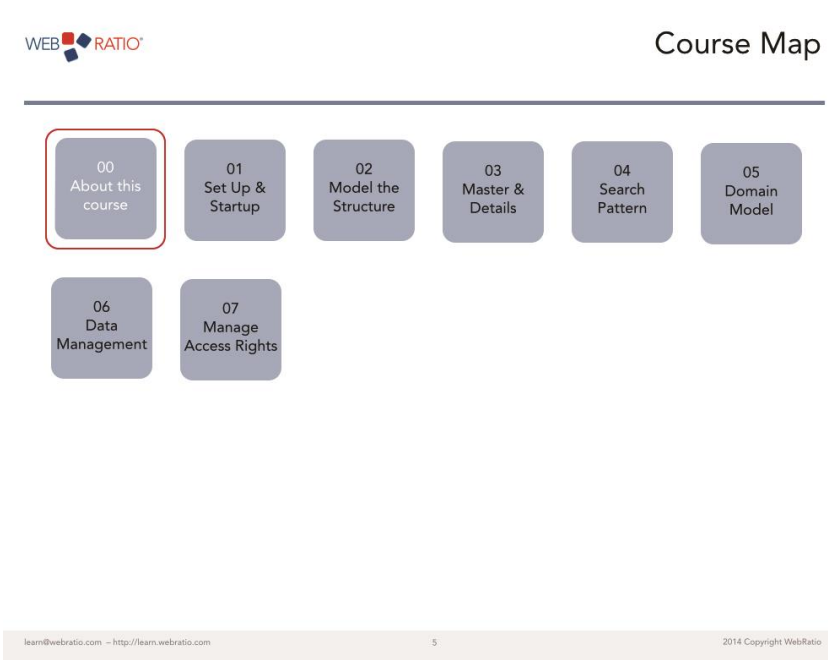
Goals

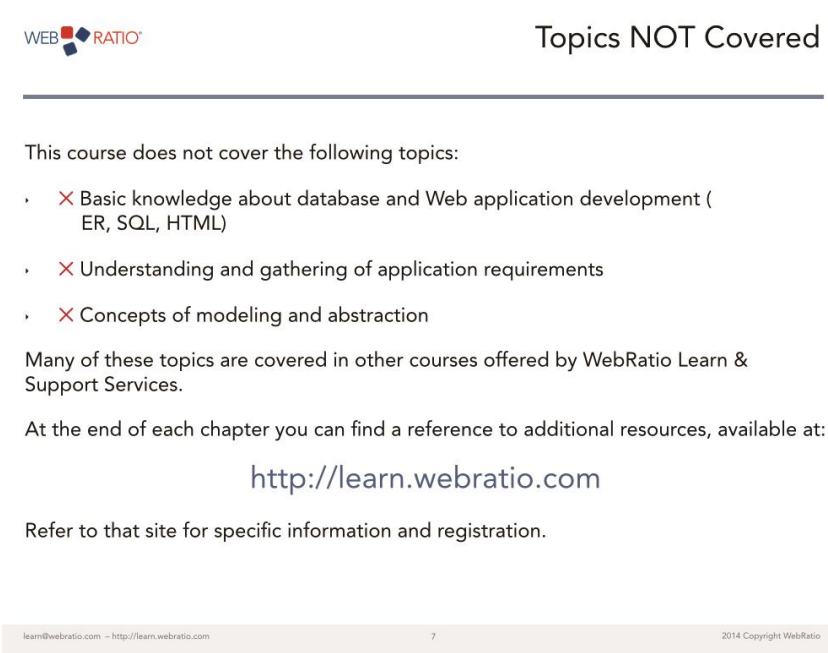
The WebRatio Modeler course enables you to model a Web application that complies with customer requirements using the most common IFML patterns. You will not only work on the model specification; you will also learn how to verify the model's correctness by navigating the generated Web application. In this way you can be sure that the model meets its requirements, and you can easily change the model if you find something that is not working or that is different from its initial requirements.



Course Map

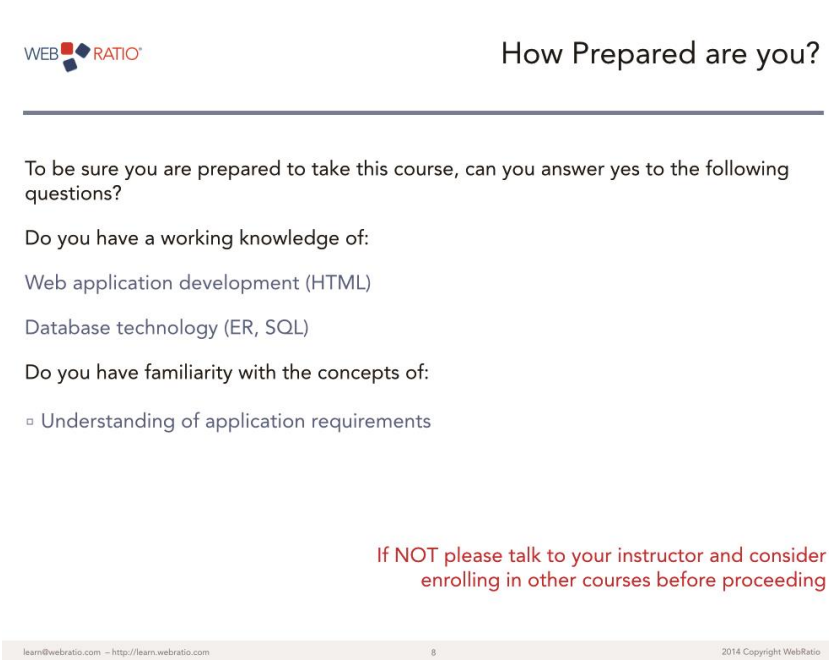
This course is composed of 8 lessons, each one focusing on a specific modeling topic.
This map is shown at the beginning of each module to remind you where we are in the complete course schedule.





Topics NOT Covered

This course does not cover the basic knowledge about database and Web application development, such as the Entity-Relationship notation, the SQL and HTML languages.
You will not learn how to gather and understand the application requirements but you will see how to transform a requirement into an abstract model. You will not learn general concepts about modeling and abstraction.
All these topics are considered as prerequisite. You can reference to other courses to learn more about them.
For the course topics you will find a reference to additional resources at the end of each lesson.

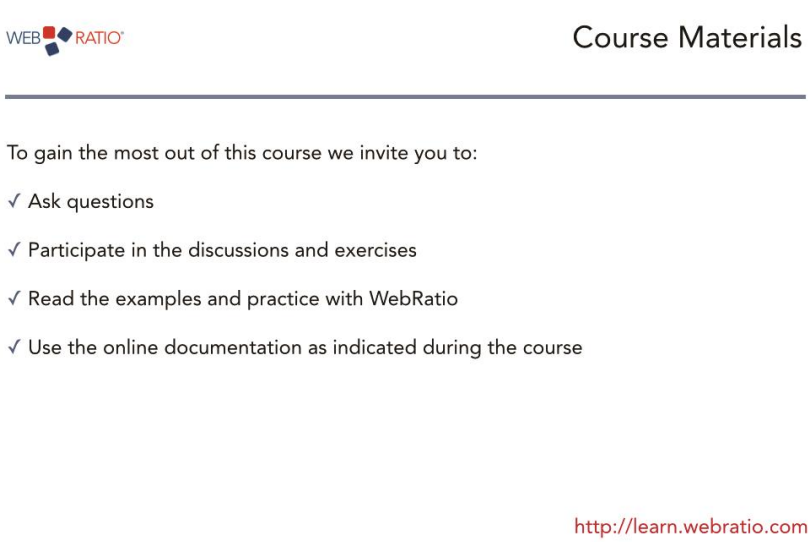


How Prepared are you?

Since some topics are not covered but they are assumed as prerequisite, if you cannot answer "Yes" to the proposed questions, please consider enrolling in other courses before proceeding.

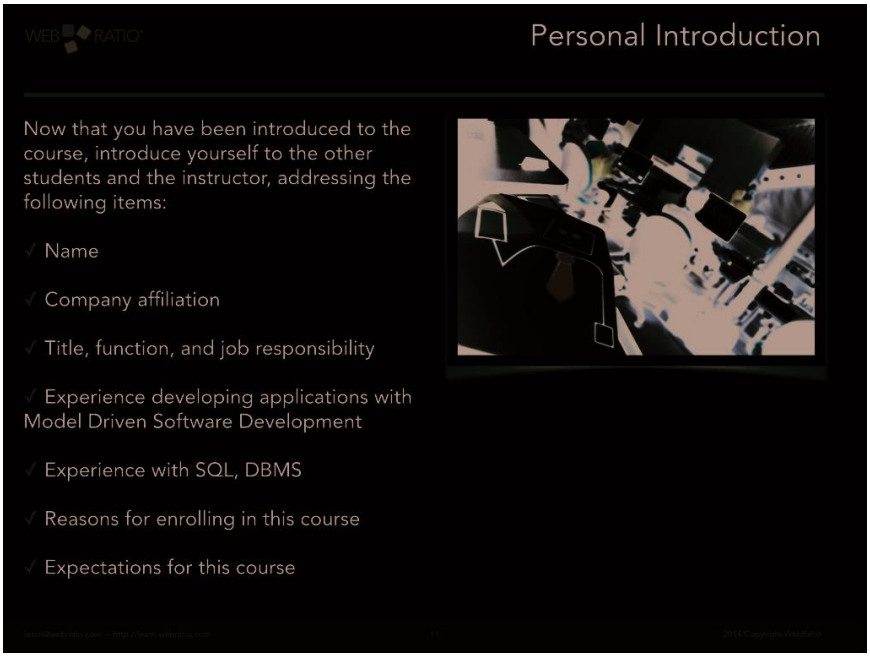
Course Materials

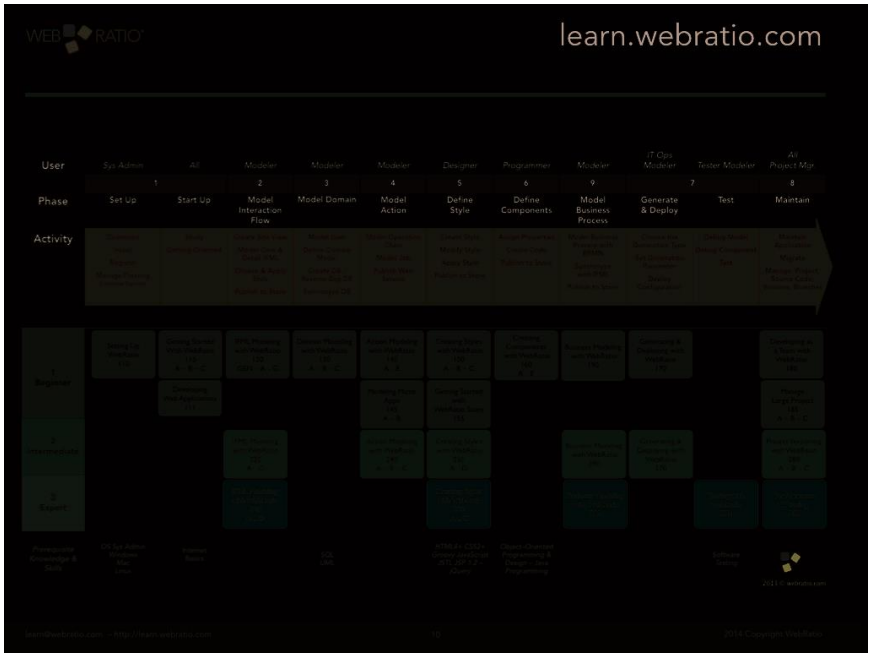
This course aims at giving you all the instruments in order to be able to model a full Web application. Please feel free to ask questions at any time. Participate in discussions and exercises to test yourself during the course. Go through all the provided documentation to gain the most out of this course.



Personal Introduction

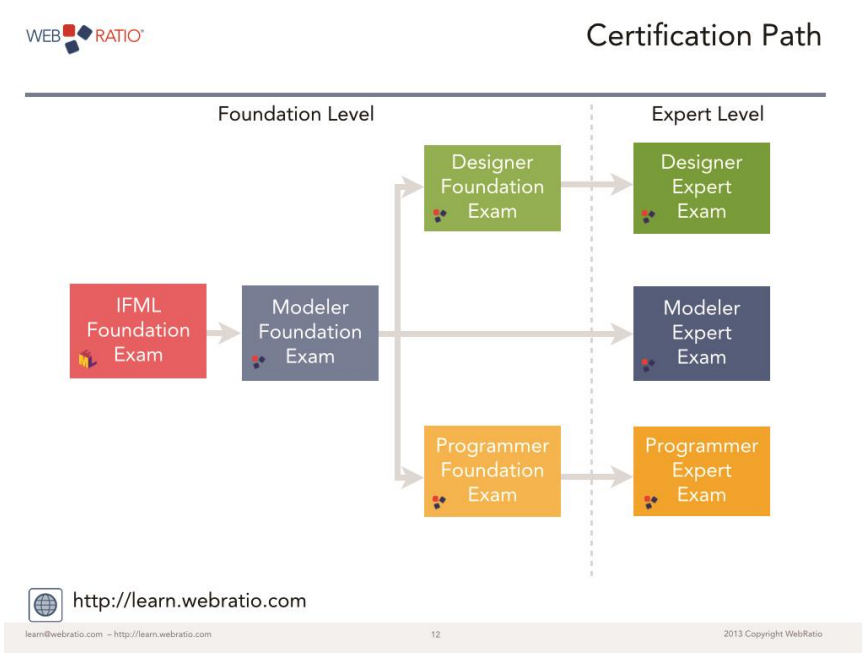
The online documentation is organized considering the web application development phases and the user role. For each phase and role you can find a set of online lessons that can be used to review a topic or to view a procedure about how to do something on WebRatio when you do not remember it.





learn.webratio.com

Now think about your expectation about this course and the reason why you are enrolled. Then introduce yourself to the other students saying your name and job responsibility.

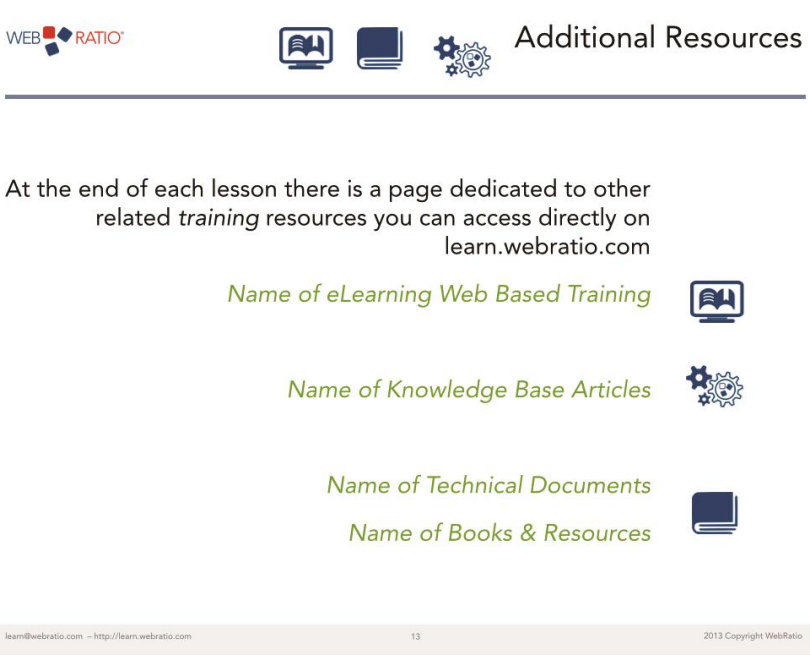


Certification Path

This slide is repeated at the end of each lesson and contains useful material to deepen your knowledge of the presented topic.
Resources are organized depending on the type; online lesson, knowledge base article; technical documents.

Additional Resources

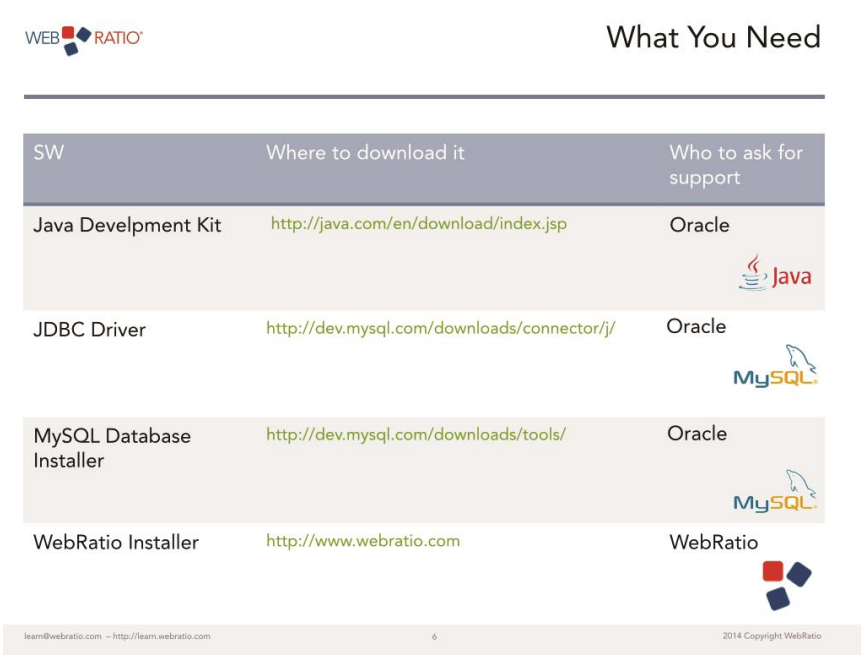
This slide is repeated at the end of each lesson and contains useful material to deepen your knowledge of the presented topic.
Resources are organized depending on the type; online lesson, knowledge base article; technical documents.



Set up and start up

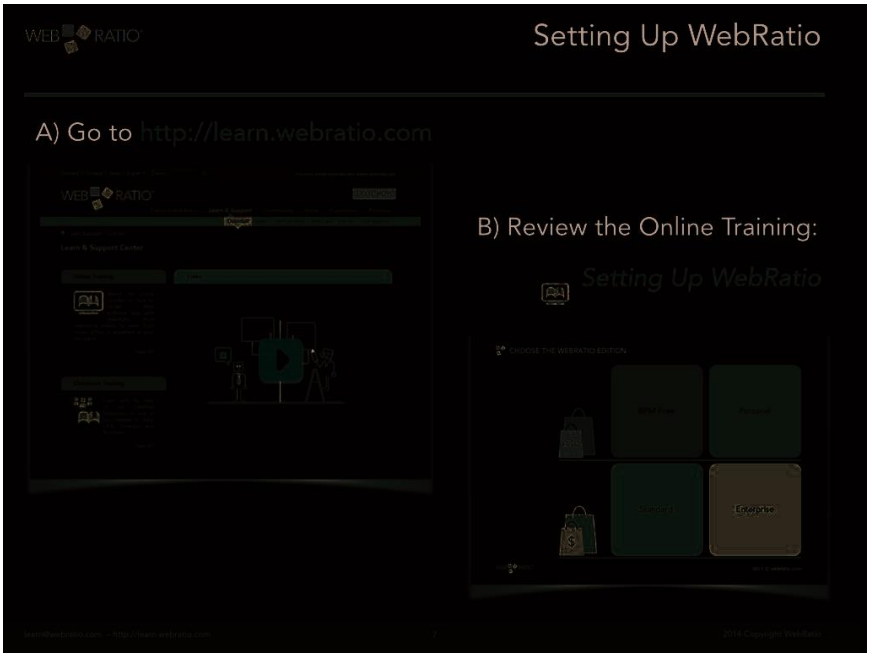
What You Need

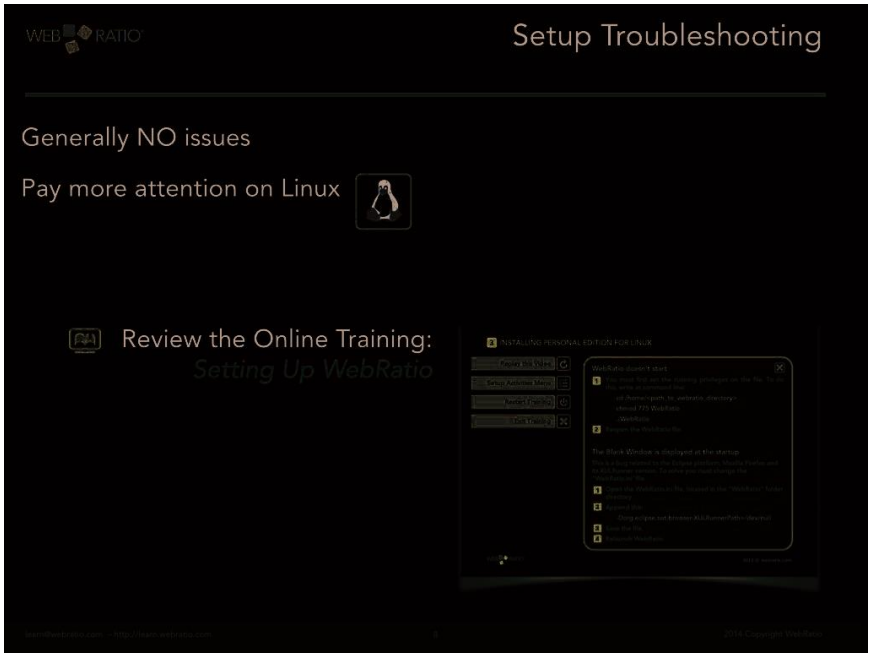
To participate in the course and perform exercises, you need to first set up your computer or laptop with the listed software. You need an Internet connection to get the installation files and to activate WebRatio. Refer to the official software documentation to learn how to install it. You can use any operating system.



Setting Up WebRatio

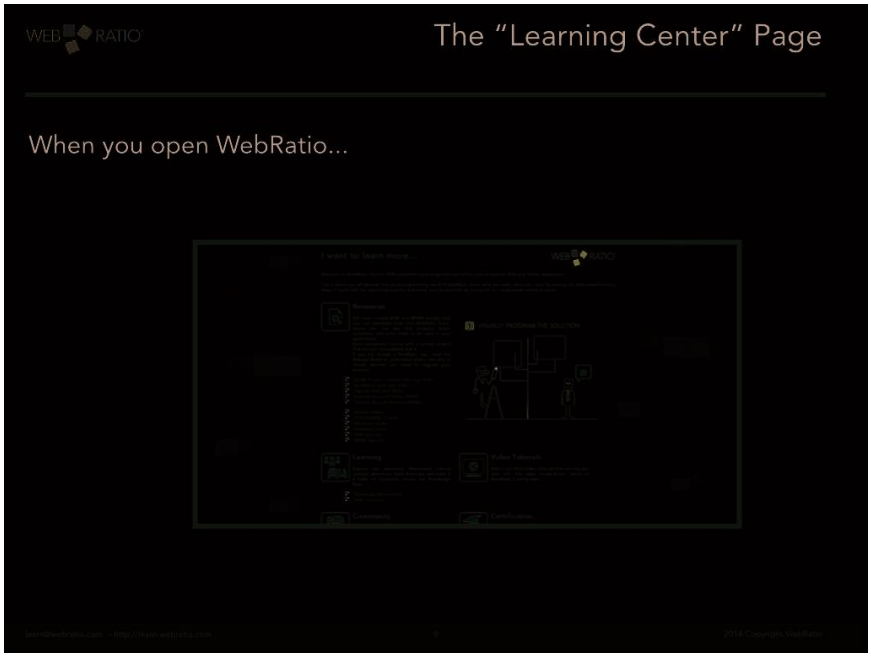
To see how to set up WebRatio, watch the dedicated online lesson on http://learn.webratio.com. Just remember at the beginning to select your WebRatio edition from the proposed ones.





Setup Troubleshooting

The setup procedure usually ends successfully, but pay attention when installing on Linux. There are known issue about Eclipse running on Linux. Please refer to the section in the online lesson about the WebRatio Setup. If the problem persists please email support@webratio.com to get help.

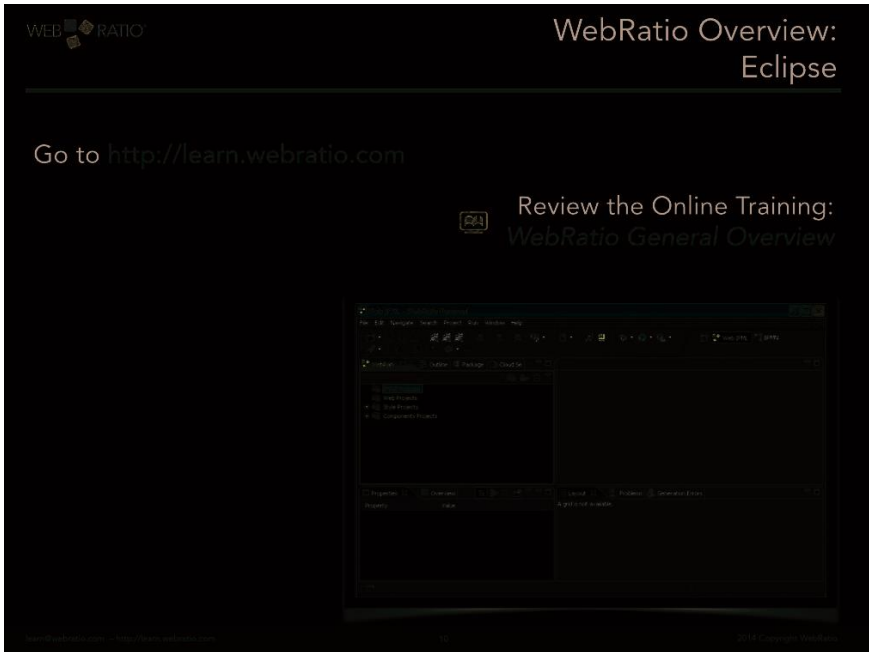


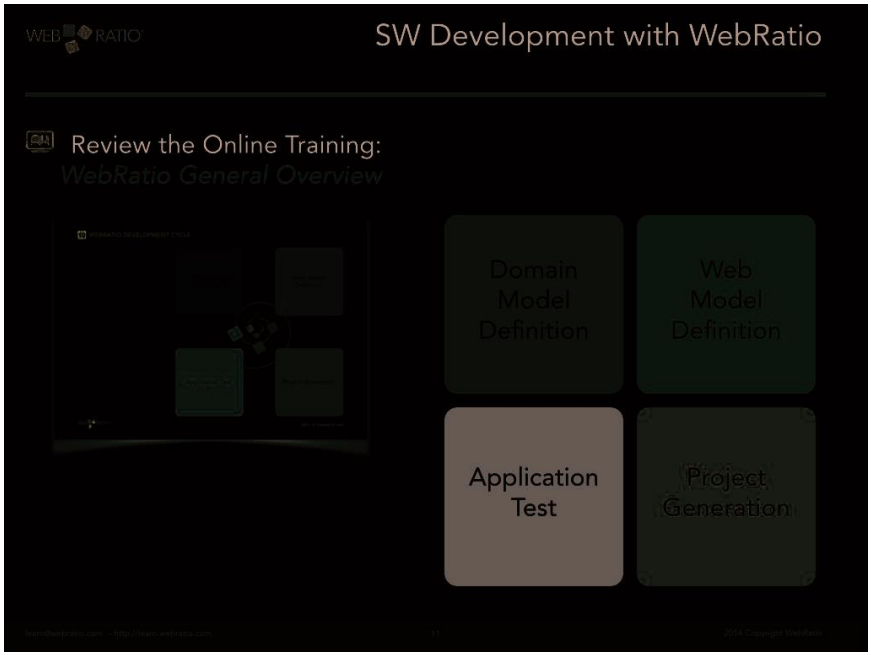
The "Learning Center" Page

At the first startup, WebRatio shows you the Learning Center page, which is where you can access educational material, such as sample projects, reference to the online documentation, technical documents, and so on. You can close or open the page when you want by using the dedicated button in the main toolbar.

WebRatio Overview: Eclipse

WebRatio is a modeling environment built on the top of the Eclipse platform. Eclipse is a well-known tool for Java developers. To get started with WebRatio, please watch the online lesson “WebRatio General Overview” on http://learn.webratio.com to learn how the WebRatio interface is organized.





SW Development with WebRatio

With WebRatio you can use a software development cycle that allows you to make several iterations during the Web application development. Each iteration is composed of the following activities:

Domain Model Definition.
Using the Entity-Relationship notation, you can define the data structure used by the Web application. You can define a portion of the model and refine it later.

Web Model Definition.
Using the IFML language, you can specify the user interaction for your Web application. Moreover, you can also specify the business logic of the Web application, encapsulating it inside an Action Definition. You can also define the model for a single feature and then enrich the model later with other features.

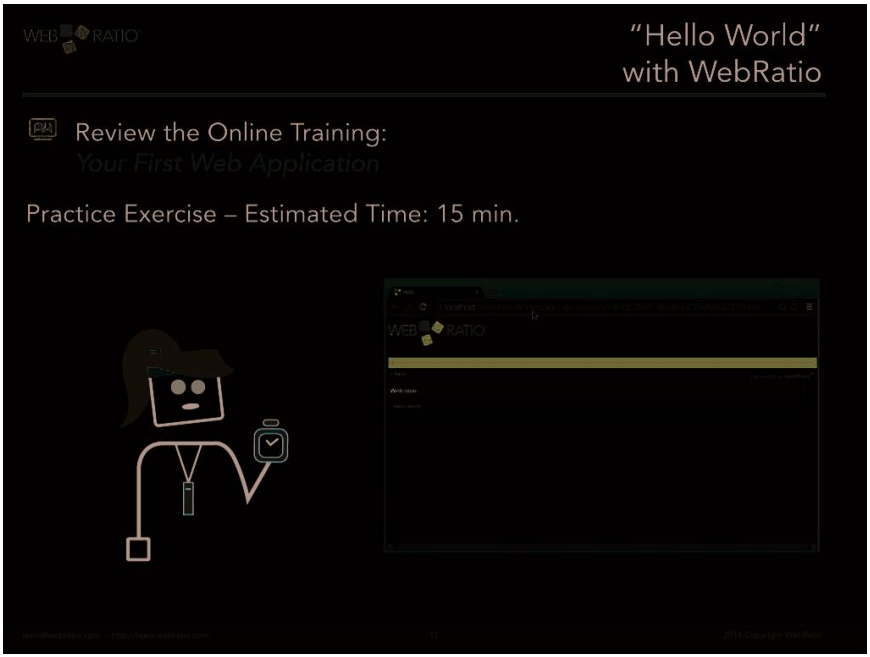
Project Generation.
Once you define the Domain Model and the Web Model for a specific feature, you can easily obtain the Web application implementing the modeled feature. You can generate the project as many times as you want.

Application Test.
When you have the Web application, you can test whether or not the feature is working as desired. Then you can decide to change the model and start a new iteration.

Now that you know a little bit how WebRatio works, it's time to create your first Web Application. Watch the dedicated online lesson on http://learn.webratio.com and try to do the same steps proposed in the lesson. It takes just 15 minutes of your time. Now that you know a little bit how WebRatio works, it's time to create your first Web Application. Watch the dedicated online lesson on http://learn.webratio.com and try to do the same steps proposed in the lesson. It takes just 15 minutes of your time.

"Hello World" with WebRatio

Now that you know a little bit how WebRatio works, it's time to create your first Web Application. Watch the dedicated online lesson on http://learn.webratio.com and try to do the same steps proposed in the lesson. It takes just 15 minutes of your time.



Additional Resources

Online Training

- http://www.webratio.com/learn/learningobject/setting-up-webratio
- http://www.webratio.com/learn/learningobject/developing-web-applications
- http://www.webratio.com/learn/learningobject/webratio-general-overview
- http://www.webratio.com/learn/learningobject/your-first-web-application
- http://www.webratio.com/learn/learningobject/organize-the-workspace (coming soon)

KB Articles

- http://www.webratio.com/learn/learningobject/setting-up-webratio-license-server

MySQL

- http://dev.mysql.com/doc/

Model the structure

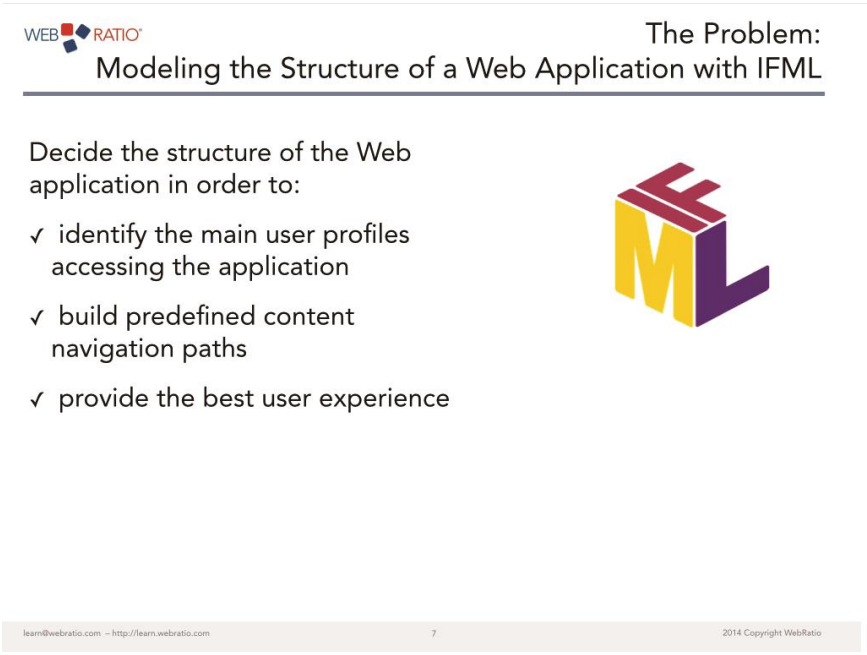
The Problem: Modeling the Structure of a Web Application with IFML

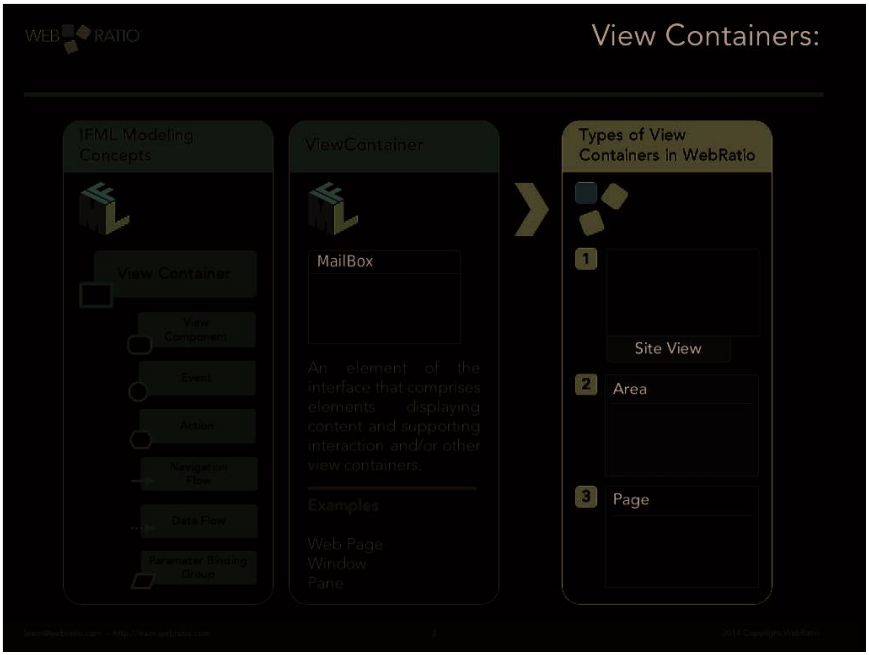
The very first thing to do when you start modeling a Web application using IFML and WebRatio is to think about the overall structure of the application itself. The overall structure is important because it influences how you model the user interaction.

You have to identify which user profiles will have access to the Web application. Depending on user needs, it may be necessary to specify different user interaction models for different user profiles. On the contrary, if all user profiles are quite similar, you can decide to model a unique user interaction model that contains a section dedicated to features belonging to specific users.

You can guide the user in the use of the web application by creating a set of predefined navigation paths. A navigation path is a sequence of user interactions that allows the user to reach a specific feature of the Web application. The most common example of navigation path is the Web application's main menu.

If you do a good job in the two previous points, you will have a Web application that provides the best possible user experience. An effective Web application is the one that is easy to use in all its aspects.





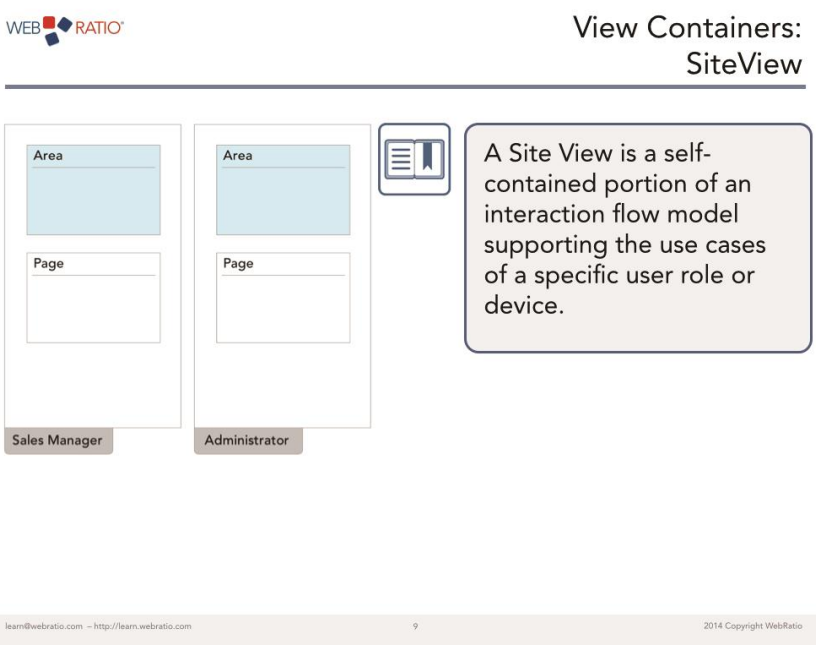
View Containers

IFML provides you with the concept of a View Container to organize your Web application. A View Container is an element of the interface that includes other elements that display content or support interactions.
WebRatio extends the concept of View Container in three different container types: Site View, Area and Page. Each container type has its own representation inside the Web Project. Using these View Containers, you are able to model the structure of the Web application exactly as you desire.

Site View

A Site View is a self-contained portion of an interaction flow model supporting the use cases of a specific user role or device. Examples of site views for a CRM Web application are the “Sales Manager” and the “Administrator” site views.

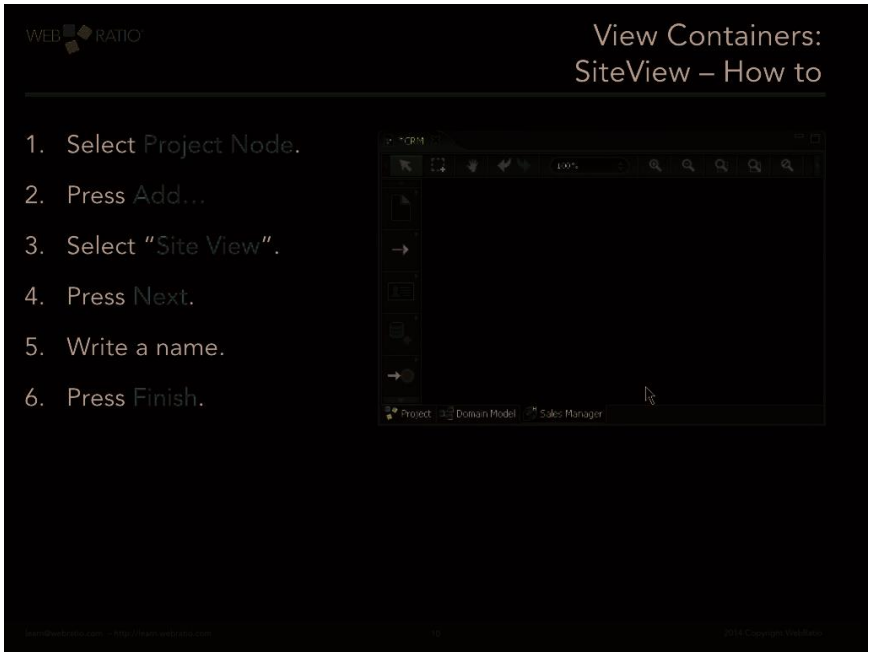
The “Sales Manager” site view contains the user interaction model specifying the features accessible by a sales manager; the “Administrator” site view contains the user interaction model specifying the features needed by the web application administrator in order to manage properly the content. An administrator and a sales manager come to the Web application having completely different goals and so the corresponding user interaction models are very different. That’s why the best choice, in this case, is to create two different site views.



There can be situations in which you have only one site view, for instance when you have two user profiles that share a lot of common functions. You can even mix the two options, so you can create different site views and dedicate one of these site views to multiple user profiles; for instance, if you have Administrators, Junior Administrators, Sales Managers and an Administrative Officer, you can create one site view for Administrators and set the user restrictions for Junior Administrators inside the site view itself.

A Site View is represented as a tab of the Web Project. You can set the name of each site view and create as many site views as you need; there is no limit on the number. A Site View corresponds to a specific section of the Web application that is reachable via the URL address. Since each site view contains an IFML diagram, it’s not possible to connect one site view to another.

A Site View is composed of other View Containers, such as areas and pages, which define the overall structure of the Web application.

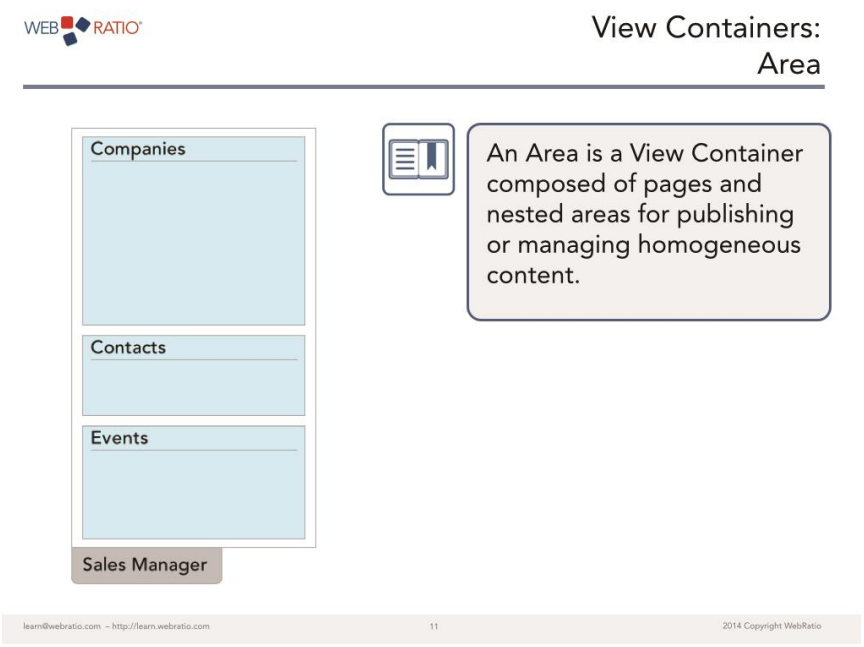


Site View - How to

Let's add site views to the CRM project we are developing as a case study.
Since the Site View represents a tab in the Web project, in order to be able to add it you need to move to the Project Overview, selecting the "Project" tab from the main work area.
Then press the "Add…" button next to the overall project structure. In the opening dialog, select the "Site View" option and press the "Next" button. Then give the Site View a meaningful name, for example "Sales Manager" and then press the "Finish" button to confirm.
Ignore for now the other properties presented in the dialog as we will talk about them later.
After you complete the procedure you see the Site View as a new tab in the main editor.
Repeat the same procedure to add all the site views required by the CRM Case Study.

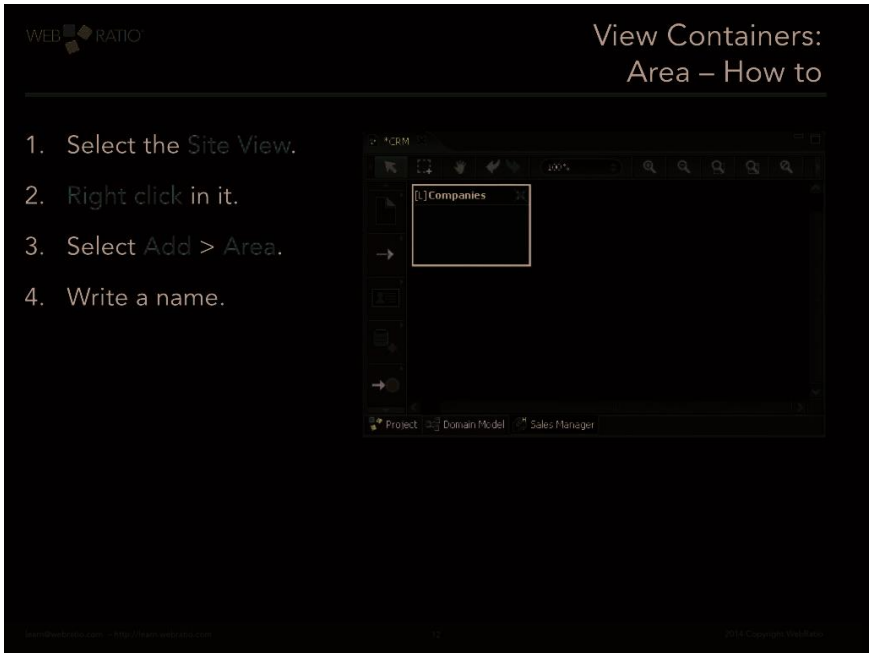
Area

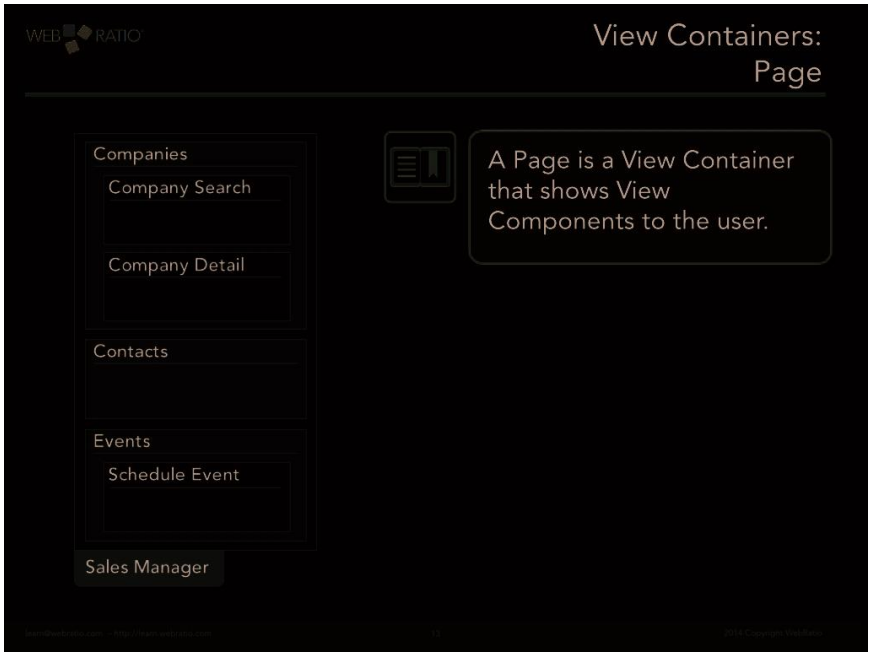
An Area is a View Container composed of pages and nested areas for publishing or managing homogeneous content. Examples of areas in a CRM Web application are “Companies”, “Employees”, and “Events”. In each area the user can find a set of features related to the same object. In the “Companies” area, there is the user interaction model needed to search and view the company information, while in the “Employee” area there is the user interaction model needed to search and view the employee information. An Area is represented with a light blue rectangle with the name at the top. You can create an area inside a Site View or inside another area. In this way you can nest areas one into the other without any limit on the number of nested levels.



Area - How to

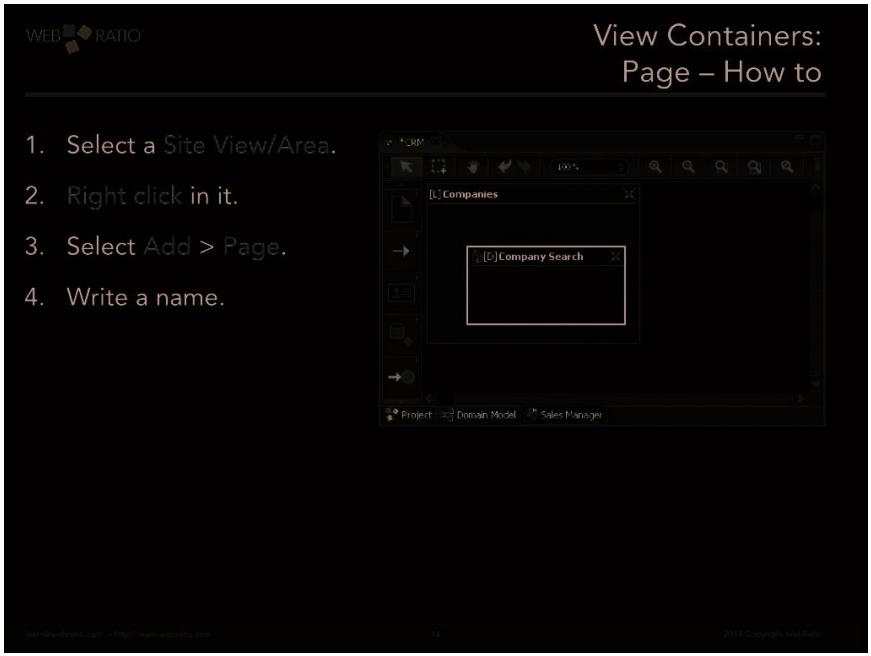
Let’s add to the “Sales Manager” Site View of the CRM project the areas mentioned previously. You just have to right-click on the container of the area - which can be another area or a site view and the select the “Add” and then the “Area” option.
You can change the name of the area by double-clicking on it in the work area and typing the desired name.
Repeat the same procedure to add all the areas required by the CRM Case.





Page

A Page is a View Container that shows View Components to the user. Examples of pages in a CRM Web application are “Company Search” and “Company Details”. The “Company Search” page lets the user search for companies with a predefined set of search criteria; the “Company Details” page, instead, shows the user all the information related to a specific company.
A Page is represented with a white rectangle with the name at the top. You can create a page inside a Site View or an Area.

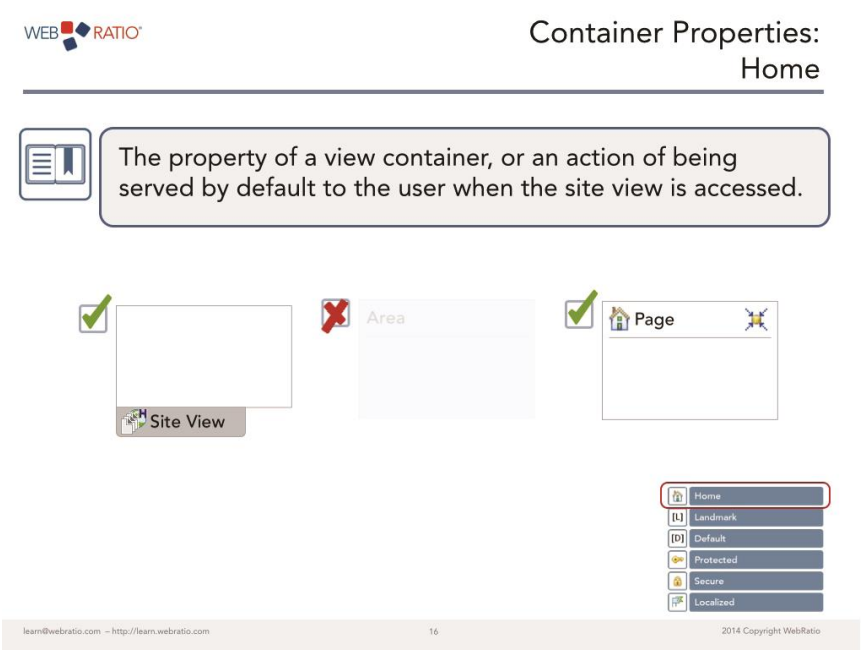


Page - How to

Let’s add to the “Companies” Area of the CRM project the pages mentioned previously.
You just have to right-click on the container of the page and select the “Add” and then the “Page” option. You can change the name of the page by double-clicking on it in the work area and typing the name.
Repeat the same procedure to add all the pages required by the CRM Case study.

Container Properties

Site View, Area and Page have a set of common properties that can be used to specify, in detail, the meaning of the container. These properties can be set in the Properties View of the desired element and include “Home”, “Landmark”, “Default”, “Protected”, “Secure” and “Localized”.



Home

The “Home” property is a property of a View Container, a module, or an action being served by default to the user when lands on the Web application. The View Container types having this property are Site View and Page. The “Home” property is shown as a “home” icon placed next to the element name.

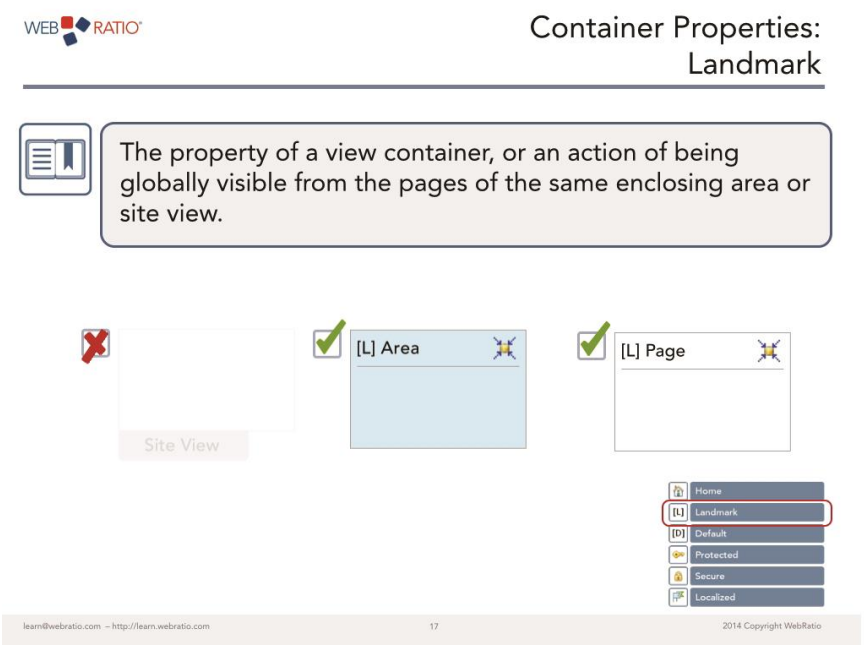
“Home” is the default Site View invoked by the Web application when a user lands on it. It’s mandatory to set a “Home” Site View inside the Web project.

The “Home” Page is the default page invoked by the Web application when a user accesses a specific Site View.

In some cases you may want the Web application to execute a piece of business logic before showing the “home” page to the user. What you have to do is to set as “Home” element the IFML element representing the business logic execution, which is an “Action”. In this case the home page is the page that is reached by the Action after its execution.

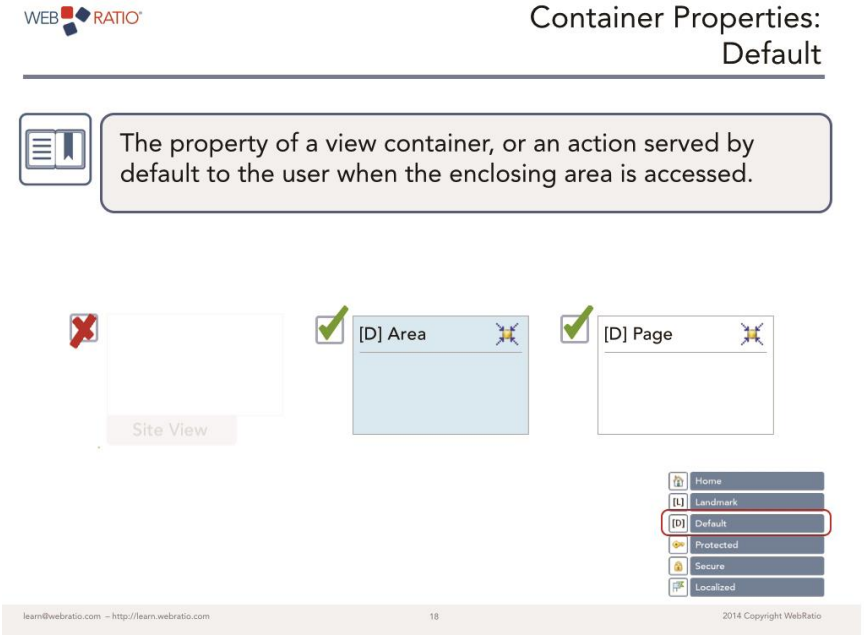
Landmark

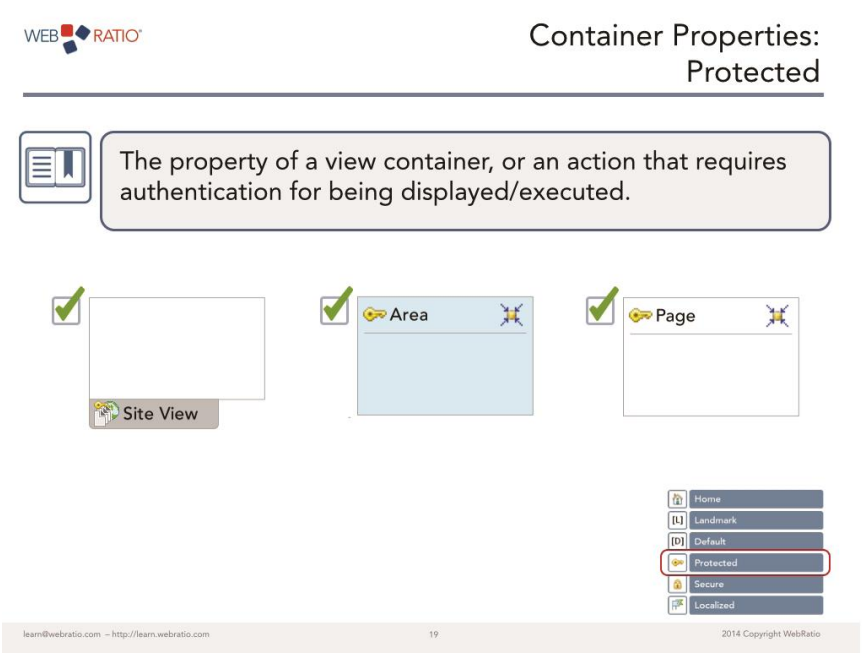
A Web application usually has a main menu from which the user chooses the section to see or the feature to use. You can create the menu structure directly in the IFML model, and you can create a menu for each Site View.
The "Landmark" property is the property specifying that the element is part of the main menu. The View Container types having this property are Area and Page. The "Landmark" property is shown as an "L" character placed next to the element name.



Default

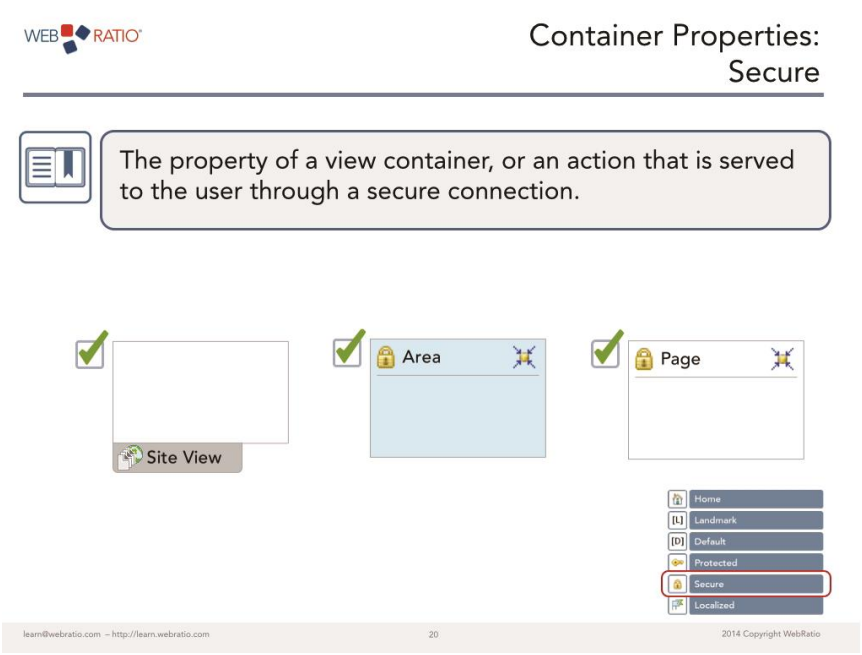
The "Default" property is a property of a View Container, a module, or an action served by default to the user when the enclosing area is accessed. The View Container types having this property are Area and Page.
The "Default" property is shown as a "D" character next to the element name. It's mandatory to set a "Default" element on any level of the area's hierarchy.





Protected

The "Protected" property is a property of a View Container, a module or an action that requires authentication for being displayed/executed. The View Container types having this property are Site View, Area and Page. The "Protected" property is shown as a key icon next to the element name.
When you set the "Protected" property on an element, everything contained in that object will be set as private and will be subject to the same access restrictions. Access rights are managed by the Web application. You will learn about access rights in another lesson.

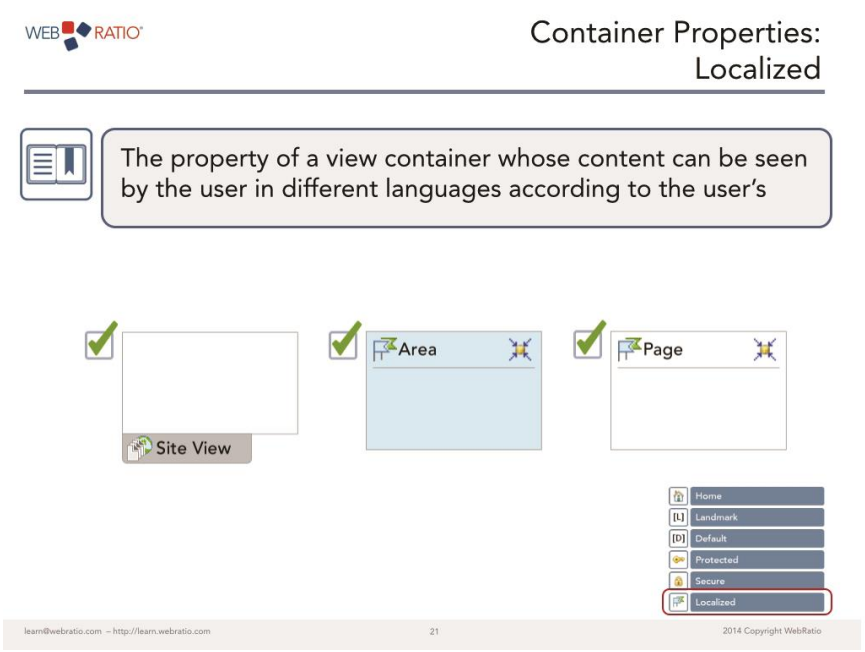


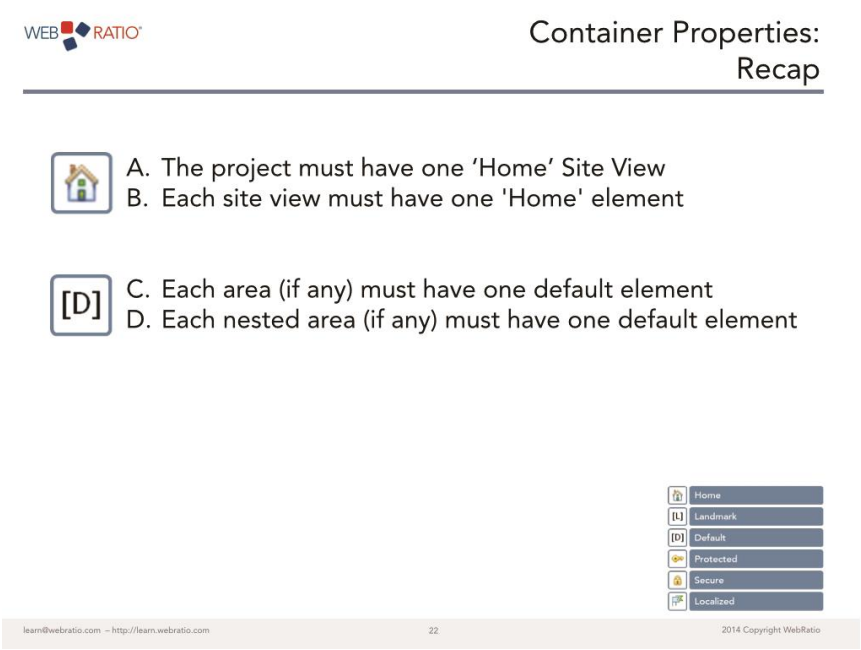
Secure

The "Secure" property is a property of a View Container, a module, or an action that is served to the user through a secure connection. The View Container types having this property are Site View, Area and Page. The "Secure" property is represented by a padlock icon next to the element name.
The secure property changes the protocol for the object web address from HTTP to HTTPS. This setting is inherited by each element contained by the object having this property enabled.

Localized

The “Localized” property is a property of a View Container whose content can be seen by the user in different languages according to the user’s locale.
The View Container types having this property are Site View, Area and Page. The “Localized” property is shown as a flag icon next to the element name.





Recap

Some View Container properties can be configured freely depending on the Web application requirements; others are subject to constraints. The constraints applied are the following:

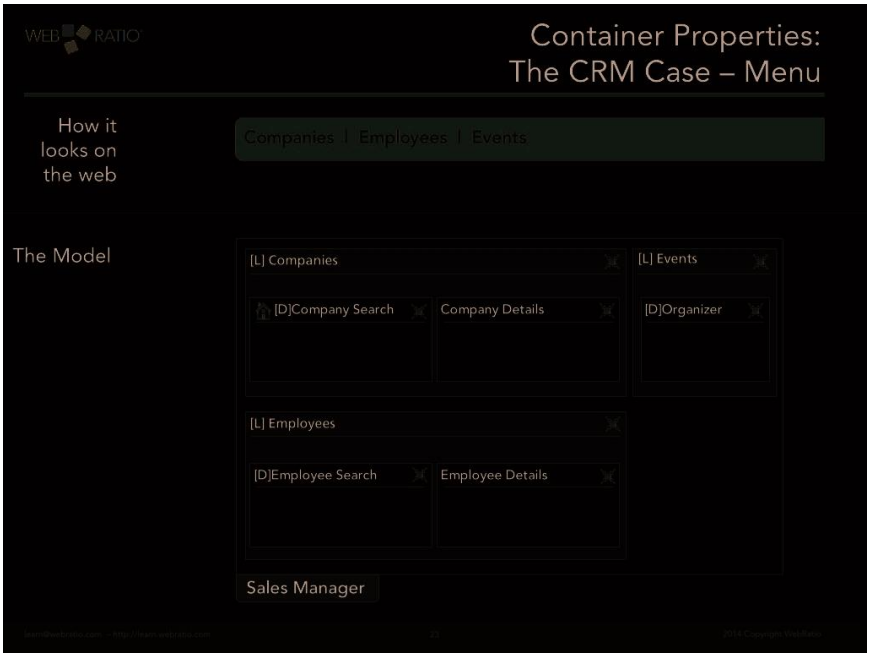
The Web project must have one "Home" Site View. WebRatio reports a warning in the Problems View when the "Home" Site View is missing and you execute the "Find Model Problems" command. The warning message is "The home site view is undefined".

Each site view must have one "Home" element. WebRatio reports an error in the Problems View when the "Home" element is missing and you execute the "Find Model Problems" command. The error message is "The home element for the site view …. is undefined".

Each area must have one default element and the same applies to each nested area. WebRatio reports an error in the Problems View when the default element is missing and you execute the "Find Model Problems" command on the Site View. The warning message is "The default element for the '….' is undefined".

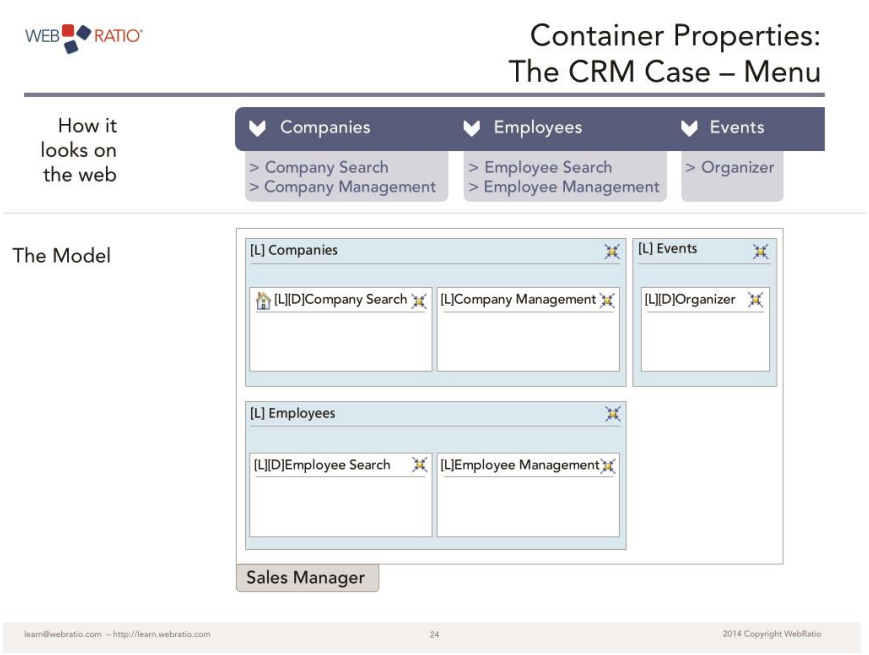
The CRM Case - Menu

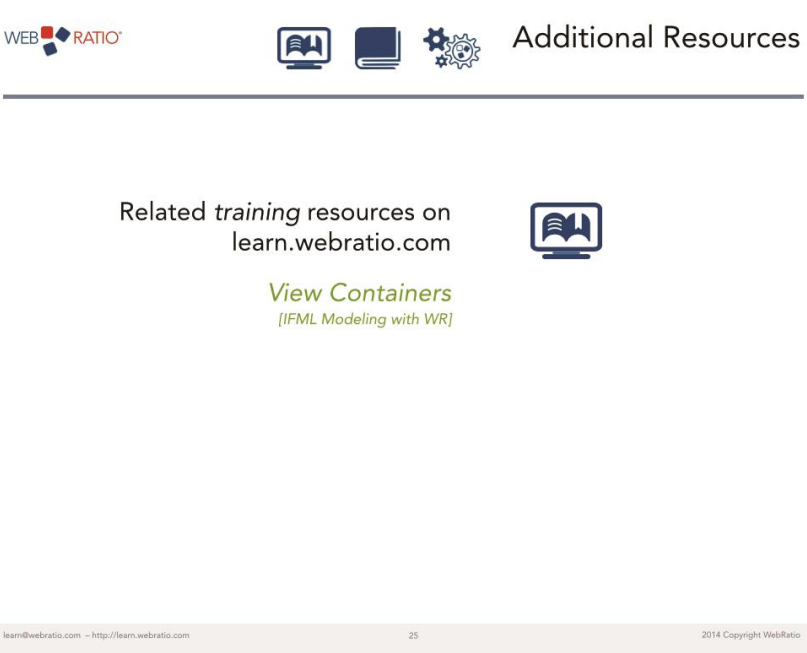
This is an example of a main menu for the CRM Web application. Let's suppose we want to obtain a menu showing these three items: "Companies", "Employees" and "Events".
There are different models in WebRatio that let you obtain the same menu. One option is to use a landmark area for each menu item. Each area contains different pages that will be connected one to the other through some other IFML concepts.
It's also possible to model the very same menu using a set of landmark pages. The results on the web page will be exactly the same, but the structure of the Web application is different because when using single pages you cannot group them.



The CRM Case - Menu

You can also model a structured menu that has several levels. In this example you can see a two-level menu. The first level is obtained through a set of landmark areas; the second level is obtained through a set of landmark pages.
You can create more complex menus, nested landmark areas, and pages. Be sure that the style project you are using is able to handle all the levels you modeled. If it isn't, then you need to create your own style project.





Additional Resources

Online Training

- http://www.webratio.com/learn/learningobject/view-containers

Master & Details pattern

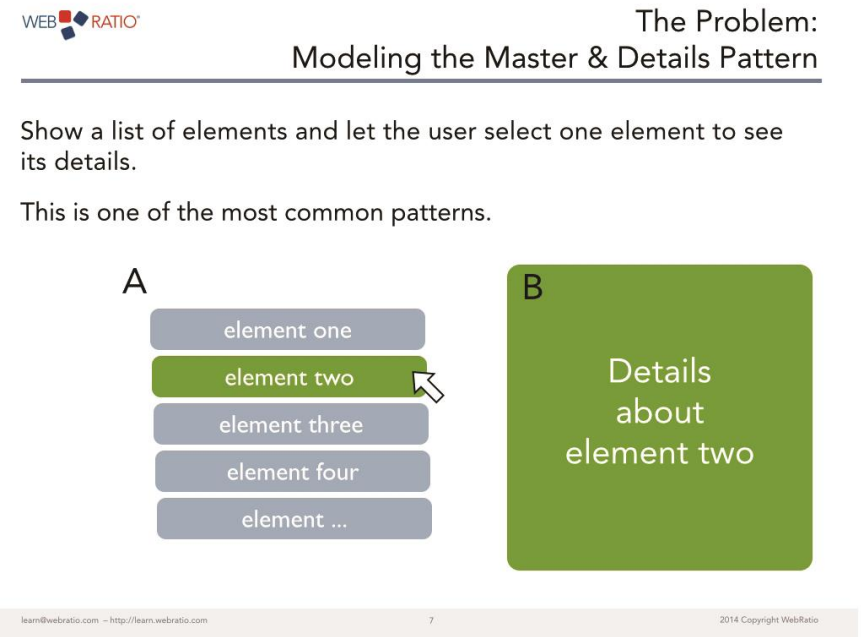
The Problem: Modeling the Master & Details Pattern

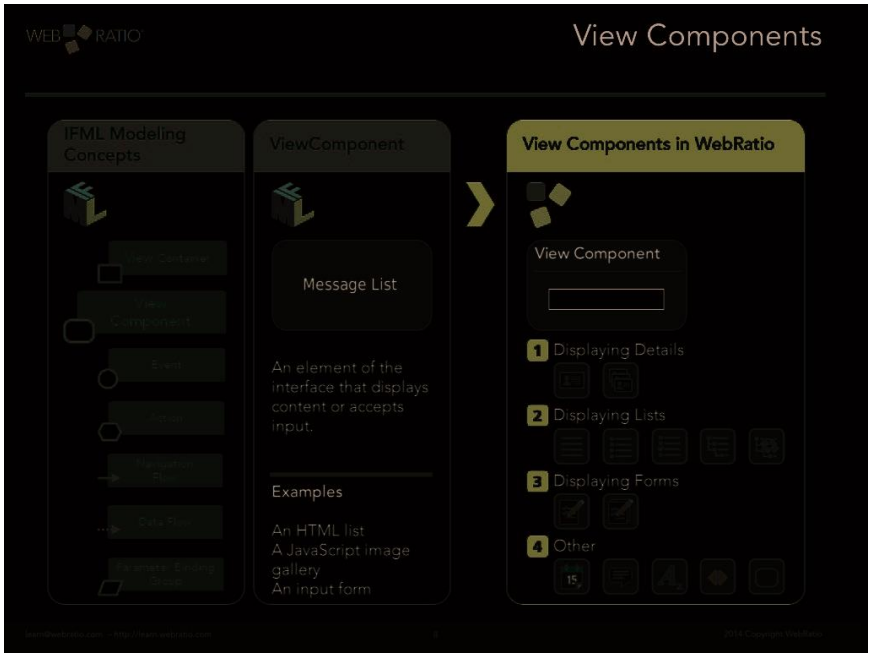
Showing a list of elements and letting the user select one of them to see the details is one of the key patterns of a Web application. It is crucial to show all the important information of an object. This modeling pattern is the Master & Details.

Since it is very simple and fundamental, this pattern is also the starting point for approaching IFML modeling in WebRatio. It is composed of at least two objects, which are the two sources of information: the Master and the Details.

The Master is the list of all the object instances displayed, which usually contains a large number of results.

The Details is the set of data related only to one instance of the list, and typically depends on the selection that the user makes on the list.





View Components

Before taking a look at the pattern in WebRatio, let's review what an IFML View Component is and the available View Component types in WebRatio.

The IFML definition for a view component defines it as an interface element that displays content or accepts inputs. Examples are an HTML List, a JavaScript Image Gallery, and an input form.

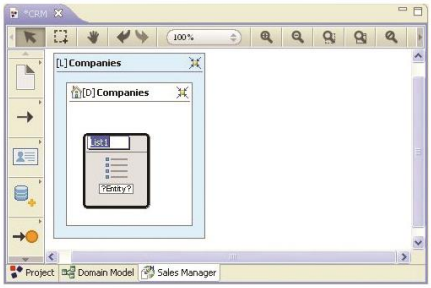
In WebRatio, the View Component has many different implementations and there are different View Components depending on the tasks they perform.

The set of available View Components includes four different categories: Details, Lists, Forms and Others.

WEB RATIO

View Components: How to Model

1. Right-click: on the page.
2. Select Add > View Components > List.
3. Write a name.



How to Model

Since View Components are made to show content, they must be placed inside a page. You can select a View Component from the 'View Components' section of the toolbar. For example, let's select the 'List' component from the toolbar and place it inside the page. Once a component is placed inside a page you can start to configure it using its Properties View.

Properties

The starting point for customizing the view component's properties is the 'Data Binding', which is made of the Entity, Display Attributes and Conditional Expression that are used to build the expression to retrieve instances from the Domain Model.
You will see details about each of these properties in the following slides.

WEB RATIO

View Components: Properties

Data Binding

✓ Entity

✓ Display Attribute

✓ Conditional Expression

- **Key** Condition
- **Attributes** Condition
- **Relationship Role** Condition

learn@webratio.com – http://learn.webratio.com 10 2014 Copyright WebRatio

Data Binding

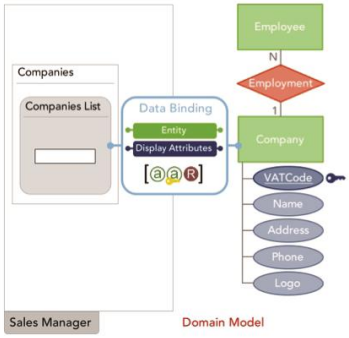
The Data Binding properties are used to define a link between the IFML model and the Domain Model.
Basically, they are used to configure the query that the component executes to retrieve data from the domain model.

WEB RATIO

View Components: Data Binding

The Data Binding is a set of properties that represent the connection between the view component and the corresponding elements of the Domain Model.

It may include a conditional expression.



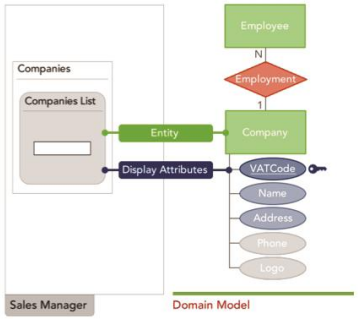
learn@webratio.com – http://learn.webratio.com 11 2014 Copyright WebRatio

WEB RATIO

View Components: Data Binding – Entity & Display Attributes

The Entity is the reference to a Domain Model object instance.

The Display Attributes are attributes of a Domain Model object instance that a view component shows to the user.



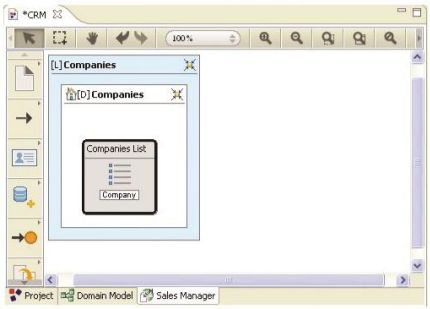
Entity & Display Attributes

Given the standard query structure you can define the 'from' and 'select' clauses using the 'Entity' and 'Display Attributes' properties.
The 'Entity' property refers to the object instance that you want to retrieve, while the 'Display Attributes' property is used to choose the data of the object instance that you want to show inside the page.

WEB RATIO

View Components: Data Binding, Entity – How to Model

1. Select the View Component.
2. Move to the Properties View.
3. Press the "Browse" button next to the "Entity" property.
4. Choose an entity.
5. Press "Ok" to confirm.

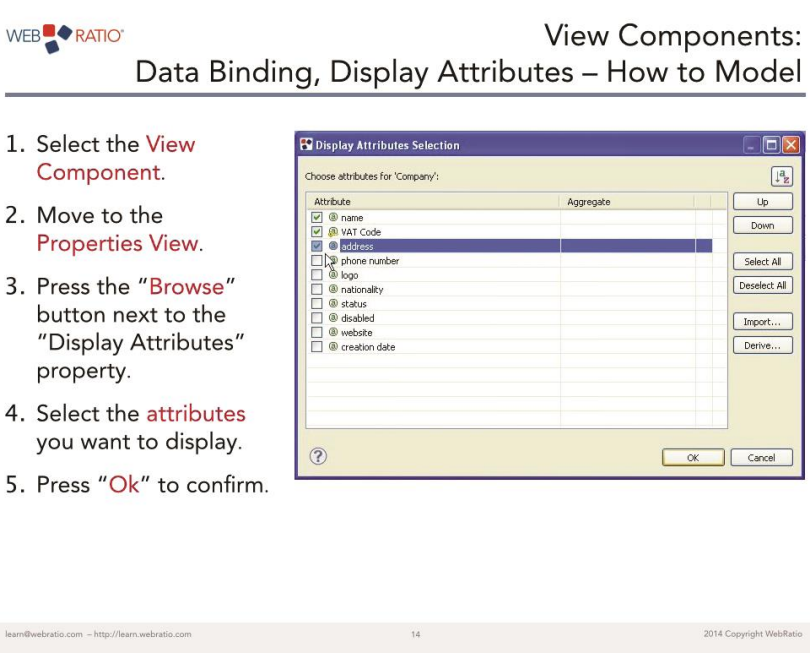


Entity - How to Model

To set the Entity property, select the View Component from the page and move to its Properties View.
You can set the 'Entity' property by pressing the 'Browse' button next to the property. You will see a dialog in which you can choose the Domain Model entity to use.
Confirm your selection by pressing the 'OK' button.

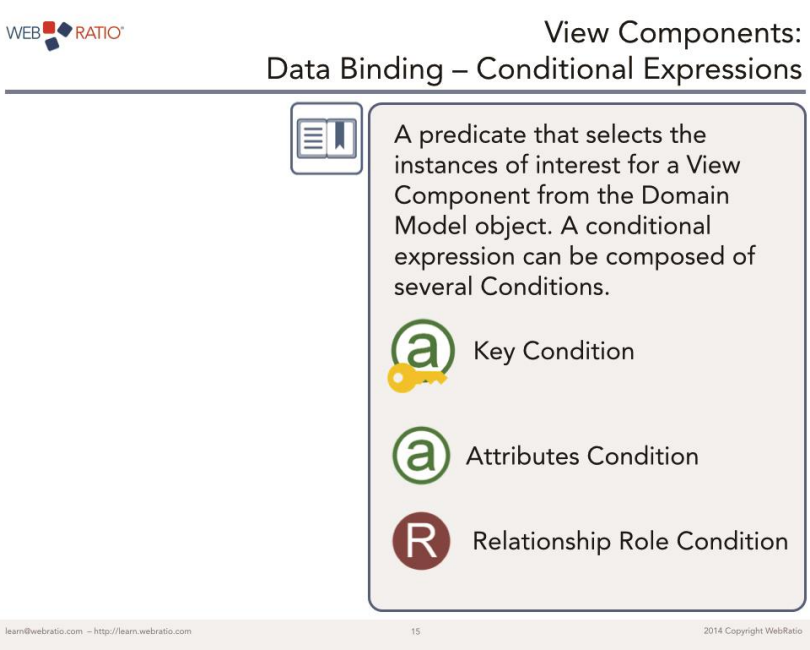
Display Attributes - How to Model

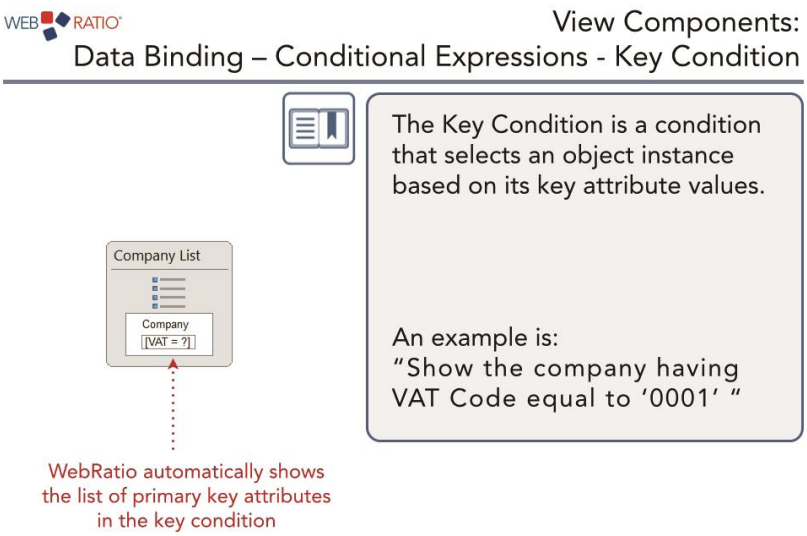
To set the list of display attributes for a View Component, select it and move to its Properties View. Press the 'Browse' button next to the 'Display Attributes' property to open the 'Display Attributes Selection' dialog. From here, check all the attributes that you want to display. You can use the shortcuts button on the right to change the order of attributes, or to select all of them in one click.
When the selection is completed, press the 'OK' button to confirm.



Conditional Expressions

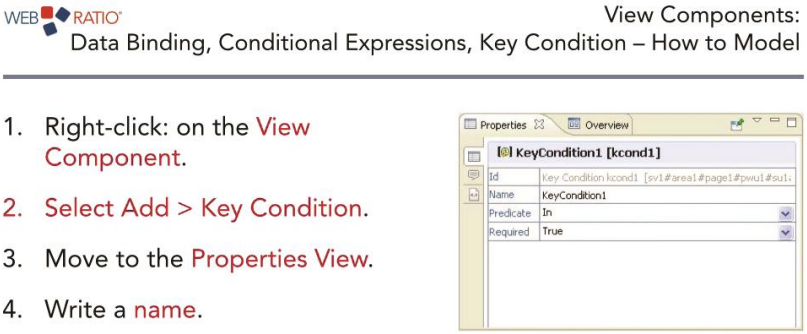
A Conditional Expression is a predicate that selects the instances of interest for a View Component from the Domain Model object. Basically, a Conditional Expression can be transposed into the 'where' clause of a query. A conditional expression can be composed of several Conditions, one for each type of information that you want to filter. The three types of available conditions are Key Condition, Attributes Condition and Relationship Role Condition.





Key Condition

The Key Condition is a condition that selects an object instance based on its key attribute values. The Key Condition looks for the Key Attributes of an entity directly from the Domain Model, and if you have more than one key attribute it will consider all of them.
An example of using the Key Condition is to show only the company that has the VAT Code equal to '0001'.
When applied, the Key Condition, like all the other types of conditions, is shown as sub-element of the view component.

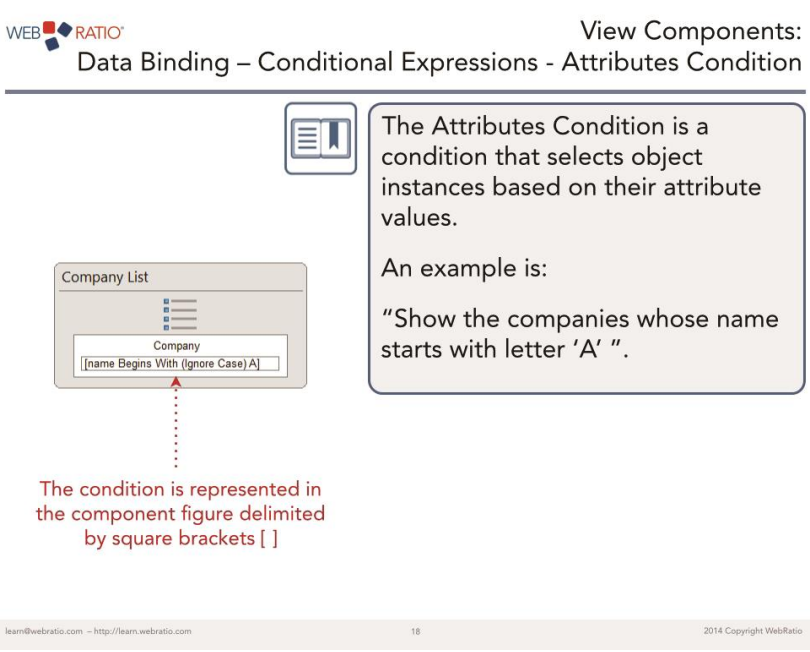


Key Condition - How to Model

To add a Key Condition to a View Component, right-click on it and select the 'Add -> Key Condition'. The new item is added as sub-element of the View Component in the Outline View.
To configure the Condition, select it from the Outline View and move to its Properties View. You can write a name for it and set the other properties, "Predicate" and "Required", as show in the slide.

Attributes Condition

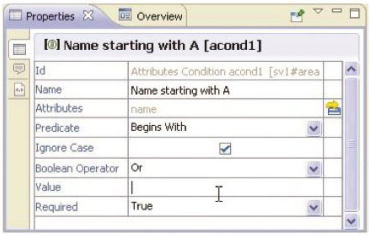
The Attributes Condition is a condition that selects object instances based on their attribute values. It refers to the attributes of the Entity of the Domain Model that can be chosen, as well as the predicates and other properties, to create the desired criteria.
An example of using the Attributes Condition is to show only the companies that have a Name starting with the letter 'A'. The Attributes Condition is shown as a sub-element of the view component and is surrounded by square brackets.



WEB RATIO

View Components:
Data Binding, Conditional Expressions, Attributes Condition – How to Model

1. Right-click: on the View Component.
2. Select Add > Attributes Condition.
3. Move to the Properties View.
4. Write a name.
5. Select the Attributes you want.
6. Choose a Predicate.
7. Write a constant value if needed.



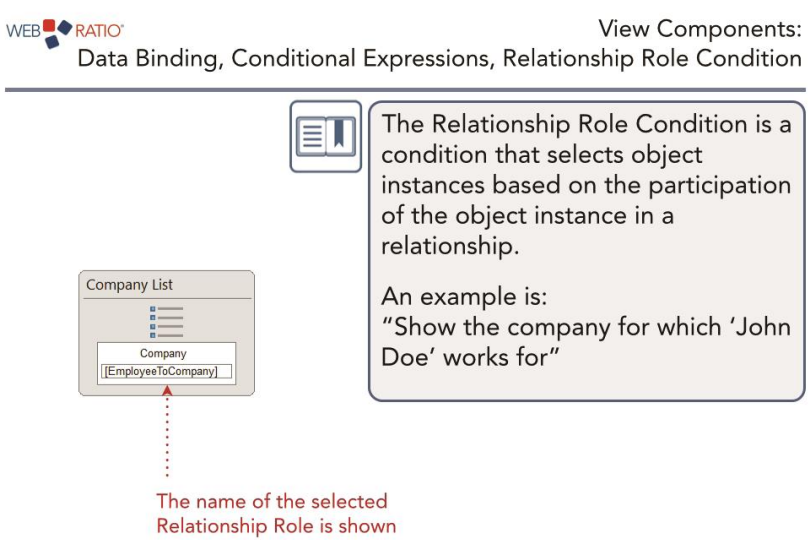
learn@webratio.com – http://learn.webratio.com 19 2014 Copyright WebRatio

Attribute Condition - How to Model

To add an Attribute Condition to a View Component, right-click on it and select the option 'Add -> Attributes Condition'. Move to its Properties View to start customizing its settings; first of all, write a name for the Attributes Condition, then select the attributes that you want to filter by selecting the desired attributes.
To do so, press the 'Browse' button of the 'Attributes' property; this will open the Attributes Selection dialog. From here you can select one or more than attributes to use for your condition. When the selection meets the requirements, press the 'OK' button to confirm.
Once you select the attributes, you can configure the other properties of the condition such as the 'Predicate', which is the logic to be checked. There are a lot of values available, and in this case your choice is 'Begins With'. If the value of this condition is constant, you can set it directly from the Properties View, in the 'Value' property.

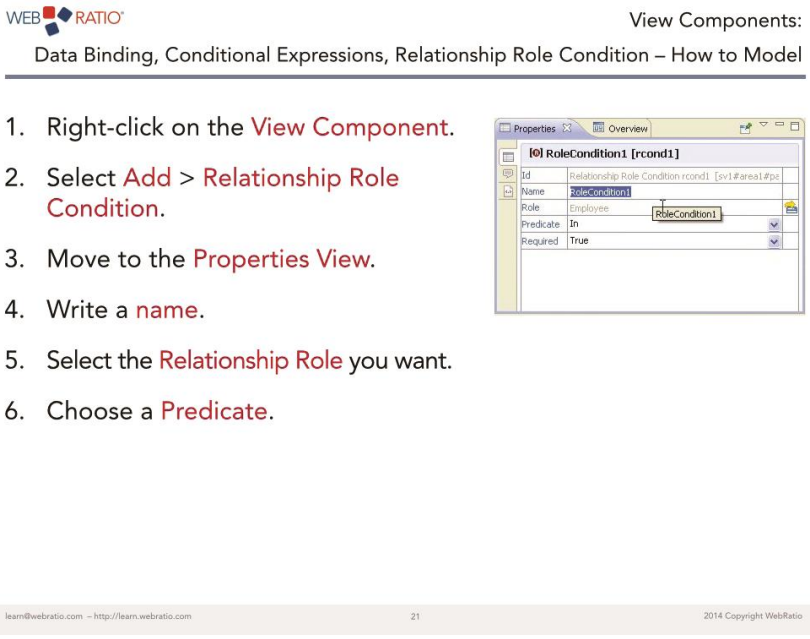
Relationship Role Condition

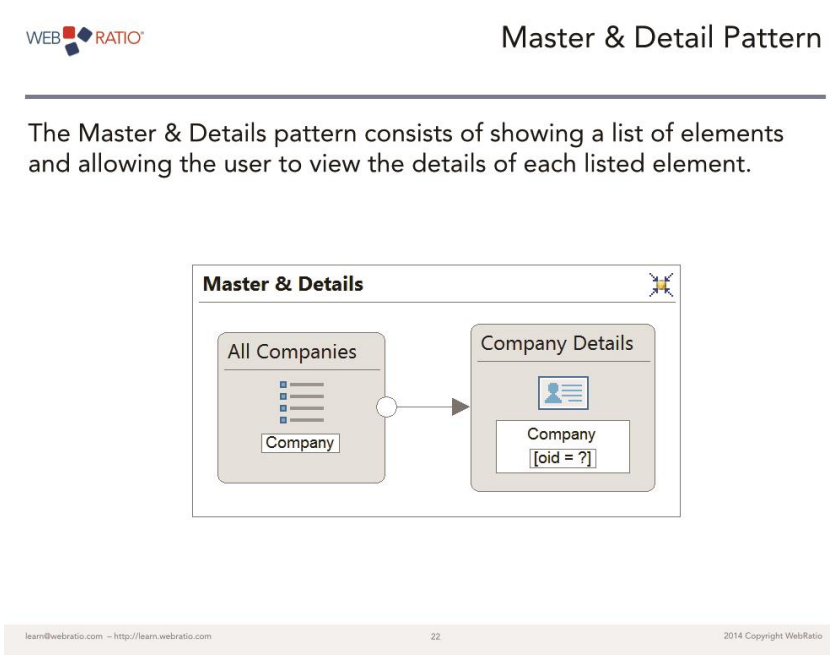
The Relationship Role Condition is a condition that selects object instances based on the participation of the object instance in a relationship. It references one foreign key for the selected Entity when looking at the Domain Model structure.
An example using a Relationship Role Condition is showing the company for which the employee 'John Doe' works. The Relationship Role Condition is shown as a sub-element of the View Component, showing the name of the selected Relationship Role.



Relationship Role Condition - How to Model

To add a Relationship Role Condition to a View Component, right-click on it and choose the option 'Add -> Relationship Role Condition'. The new item is added as a sub-node of the View Component in the Outline View. If you want to set properties for the Condition, select it from the outline and move to its Properties View and write a name for it.
The other properties, Predicate and Required, are set by default as shown in the slide.





Master & Detail Pattern

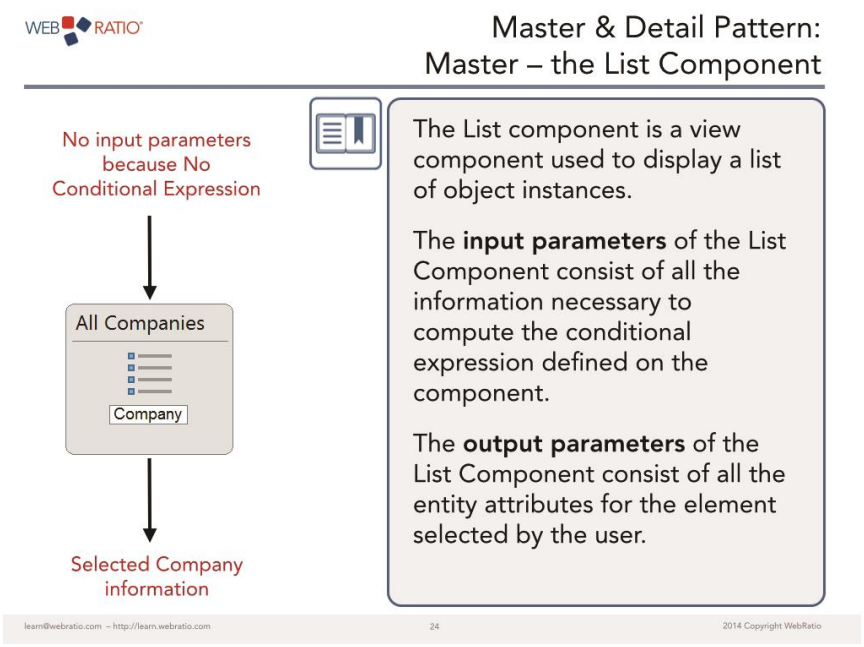
In a Web application, it is a very common pattern to show a list of objects with their details. This pattern takes the name of 'Master & Details' and consists of showing a list of elements, also called Master, and allowing the user to view the details of each listed element. As you can see, the basic pattern is two View Components inside the same page. The first component on the left, named List Component, is used to show the list of instances of the Domain Model entity, while the second one shows the Details of the element chosen by the user from the list.

The CRM Case

In the CRM Case, the Master & Detail pattern can be applied when you want to provide a page showing all companies and to let the user choose a company to see its details.
Here, you can see the correspondence between the Master & Detail pattern in WebRatio and in the generated Web application page. This is just one possible way to organize the page, showing the list of companies at the top of the page and the details below. The same WebRatio model can produce different results according to the disposition of components inside the page. Starting from this model, let's see each View Component that composes it.

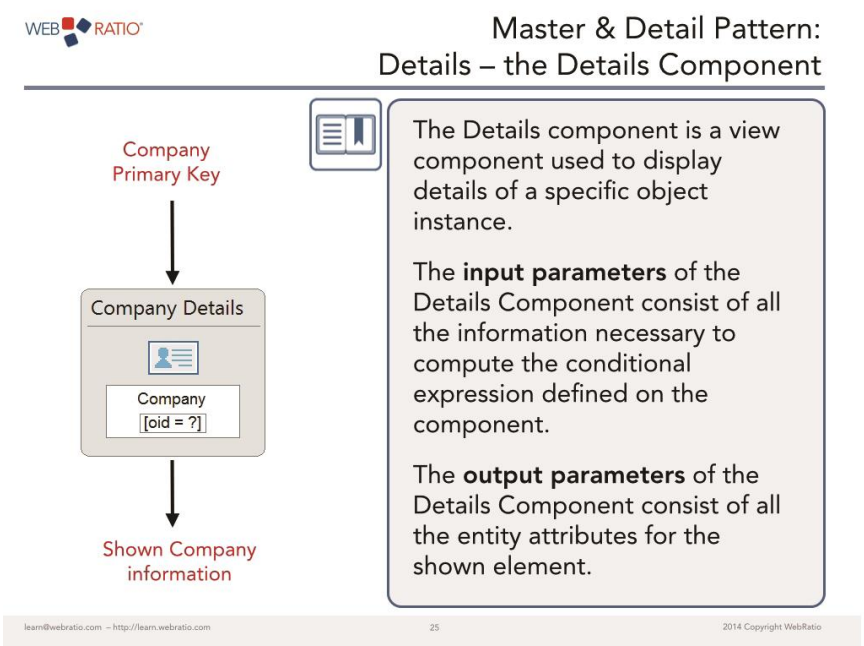
Master - the List Component

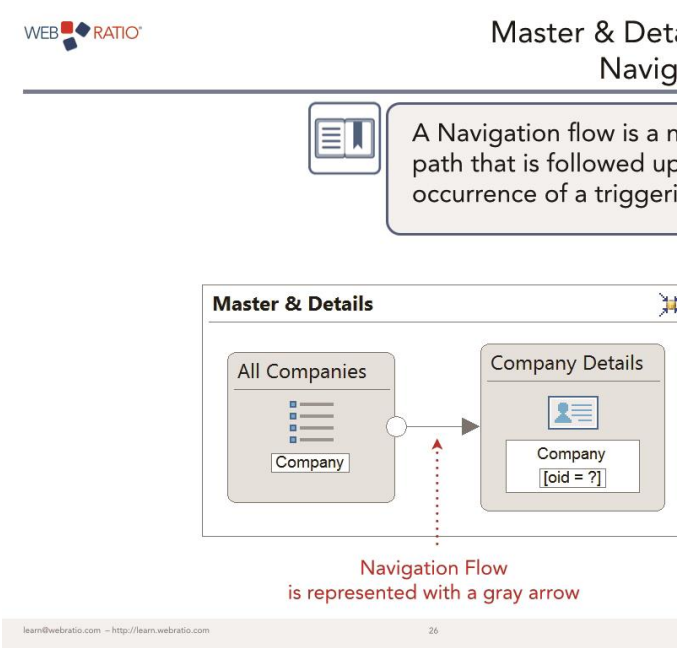
The List component is a View Component used to display a list of object instances. Since it is a View Component, you can find it in the View Component section of the toolbar.
The input parameters of the List Component consist of all the information necessary to compute the conditional expression defined on it; the List Component of the CRM Case has no conditional expression defined so there will be no input parameters.
The output parameters of the List Component consist of all the entity attributes for the element selected by the user; in the CRM Case we want to select the object instance to show its details, so you will use the company primary key.



Details - the Details Component

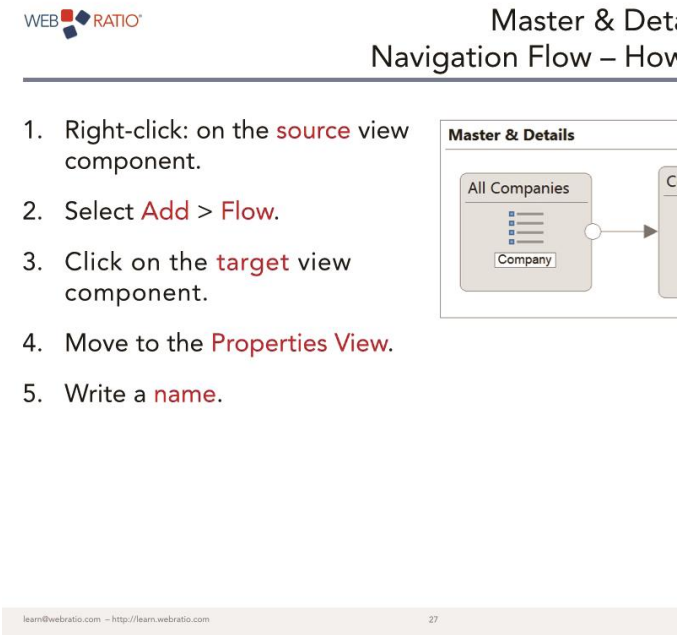
The Details component is a View Component used to display details of a specific object instance. Since it is a View Component, you can find it in the View Component section of the toolbar.
The input parameters of the Details Component consist of all the information necessary to compute the conditional expression defined on the component. The Details Component comes with a conditional expression already defined on it, since it has by default a Key Condition added. In the CRM Case, this conditional expression is used to identify the company selected from the list. The output parameters of the Details Component consist of all the entity attributes for the shown element.





Navigation Flow

A Navigation Flow is a navigation path that is followed upon occurrence of a triggering event.
An example of Navigation Flow in the CRM Case is the selection of an object from the list to see its details.
In this case, the triggering event is the selection of the user on the list of companies. It is represented with a gray arrow inside the model connecting a source and a target of navigation.
The direction of the flow is represented by the arrow.

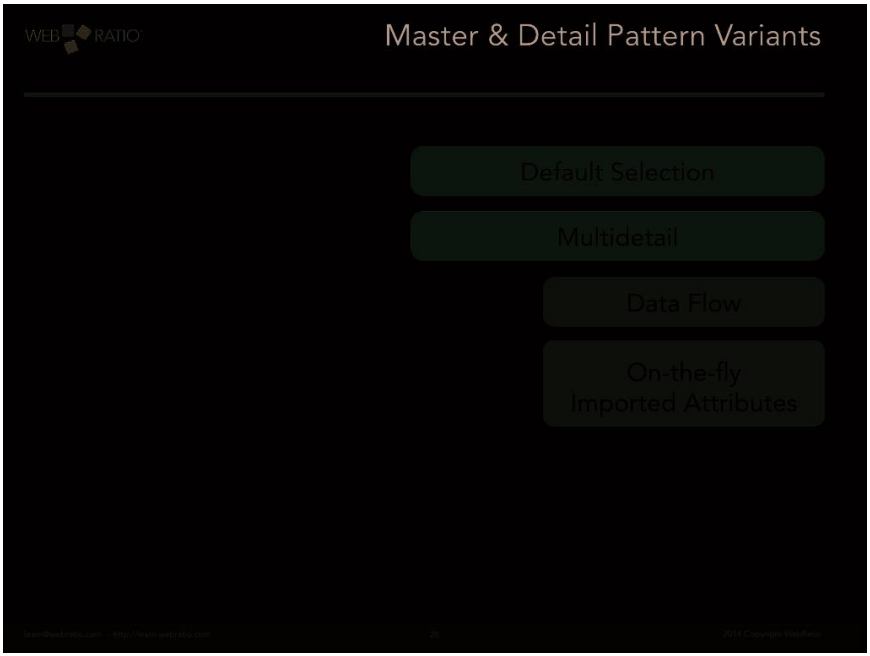


Navigation Flow - How to Model

To add a Navigation Flow, right-click on the source of the navigation and select the 'Add -> Flow' option, then define the target of the navigation by clicking on the target model object.
When the Navigation Flow is added, move to its Properties View and write a name for the Flow.

Master & Detail Pattern Variants

The basic implementation of the Master & Detail pattern can be easily extended and adapted to many different situations.
The two main categories of variants are the Default Selection, which automatically selects an item from the list when the page is accessed, and the Multi Detail Selection, which allows the user to see the details of more than one level of objects with only one click.



Additional Resources

Online Training

- http://www.webratio.com/learn/learningobject/view-containers
- http://www.webratio.com/learn/learningobject/view-components-data-binding
- http://www.webratio.com/learn/learningobject/master-details

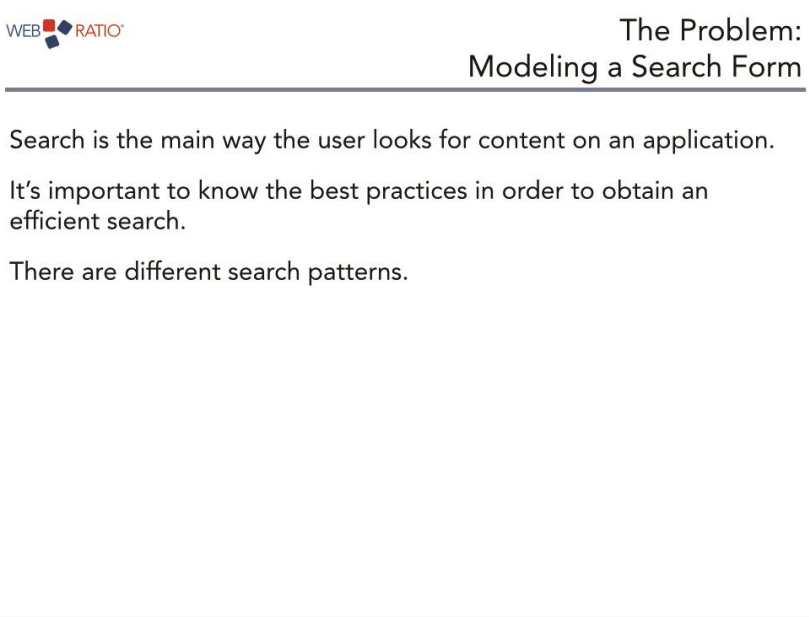
KB Articles

- http://www.webratio.com/learn/learningobject/how-to-model-the-alphabetical-index

Search pattern

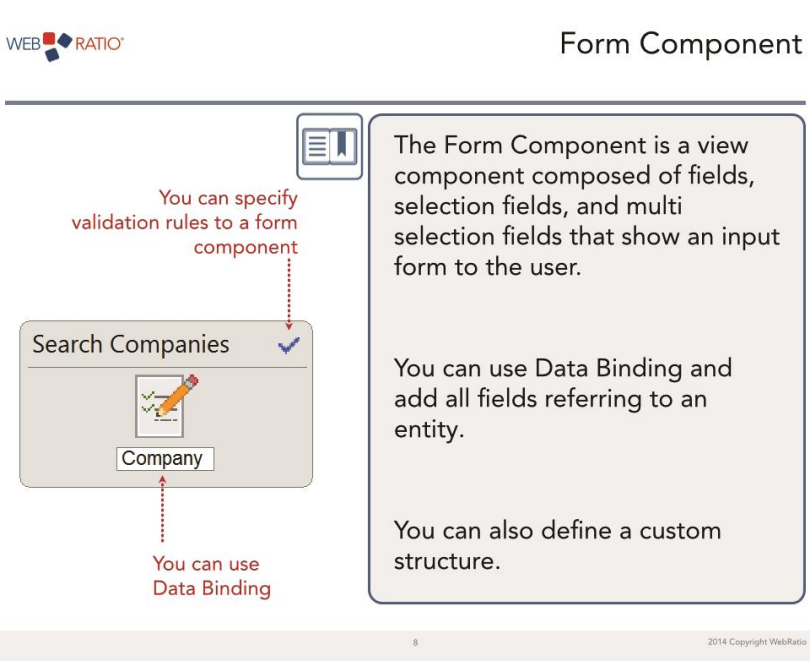
The Problem: Modeling a Search Form

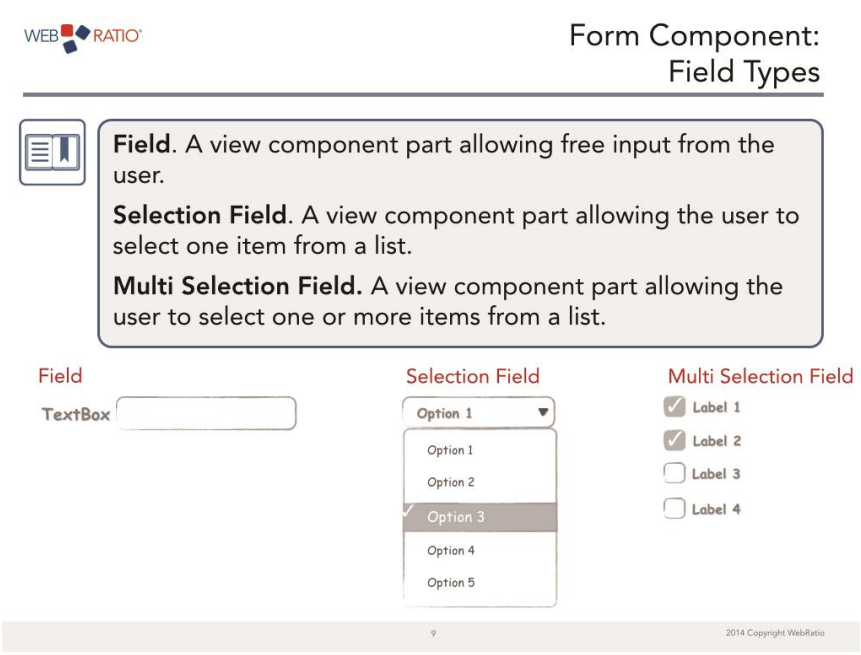
Using the search function is the main way the user looks for content on an application, so it's important to know the best practices for obtaining an efficient search.
WebRatio gives you a predefined set of View Components that can be used to model a web search, and combined to model different search patterns.



Form Component

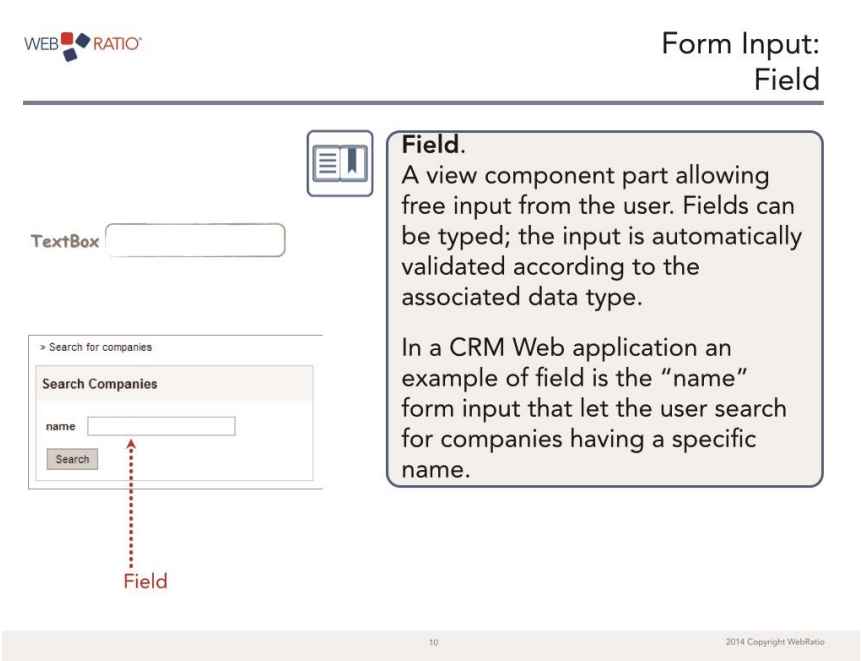
The basic View Component for the search pattern is the Form Component.
It is composed of fields, selection fields and multi-selection fields and is used to build and show an input form to the user.
You can create the Form Component, manually designing its structure. Or the Form Component can be constructed automatically starting from a Domain Model entity and using Data Binding information, so that each field references the entity attributes.





Form Component: Field Types

A Form Component is composed of sub elements called Fields. Three types of fields are available in WebRatio, and each is used for different purposes and has a different inner structure.
The Field is a View Component part that allows free input from the user.
An example of Field is the text input for the 'username' of a user.
The Selection Field is a View Component part that allows the user to select one item from a list. An example is the choice of the Company's country.
The Multi Selection Field is a View Component part that allows the user to select one or more items from a list. An example is the list of employees enrolled for an event.

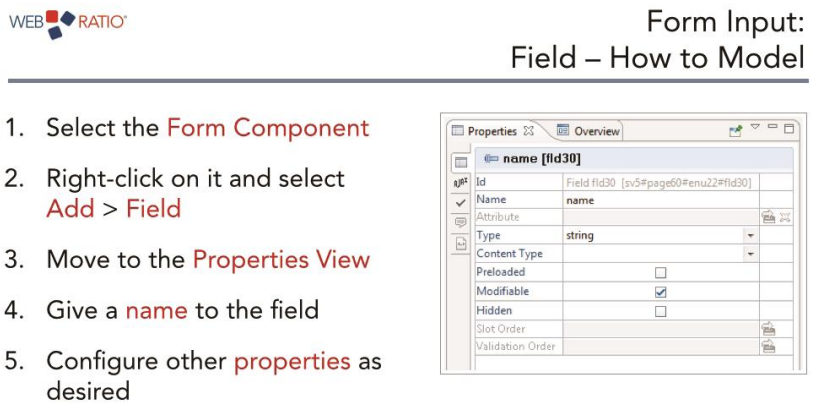


Field

A Field is a View Component that allows free input from the user.
The input accepted by a Field depends on its data type.
If a field is used to store the 'username' it will be of type string.
Each Field is automatically validated according to the associated data type; this means that if you define a Field to contain Integer data, you cannot enter alphabetical characters and, if you try to do it, on submit the Field is validated and a validation error is shown.
In a CRM Web application an example of search field is the 'name' form input that lets the user search for companies having a specific name.

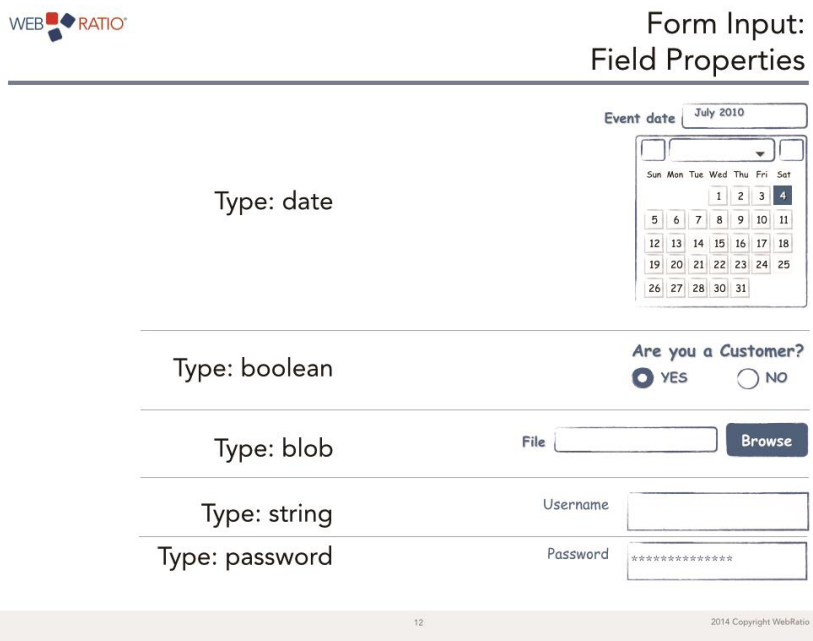
Field - How to Model

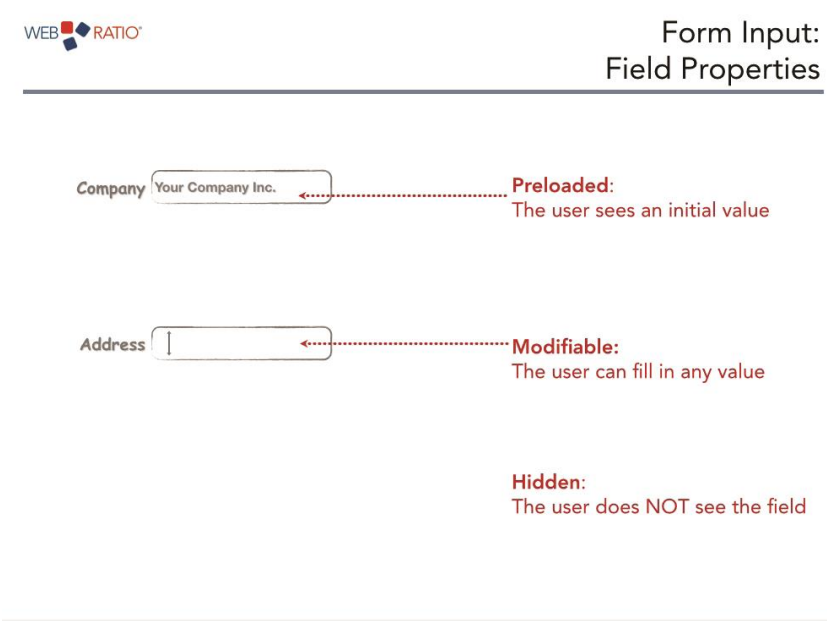
To model a Field, start selecting a Form Component; right-click on it and select the option 'Add -> Field'.
Move to the Properties View of the field and use the 'Name' property to give the field a name, and then configure the other properties as desired.



Field Properties

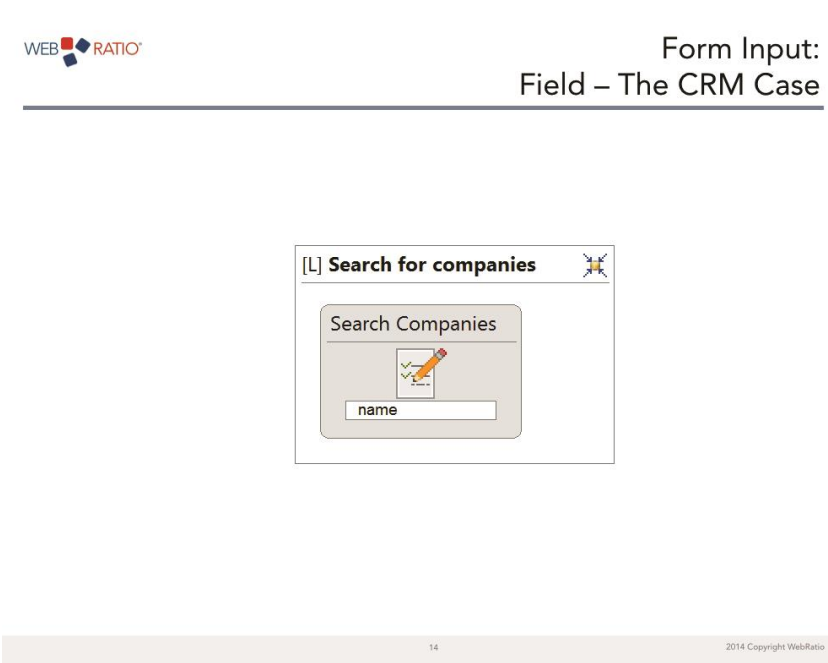
You can use these set of properties to configure a Field. Taking apart the 'Name', which is just the label to show for the Field inside a page, the next one is the 'Type'.
The 'Type' not only decides the management of the Field content from a validation point of view, but also defines how the Field is rendered inside the page.
As you can see, according to the choice made for the 'Type' property a different visualization is provided; for 'date' fields a calendar, for 'Boolean' fields a radio button, for 'blob' fields a button for file upload, for 'string' an input text, and for 'password' an input text with the content obscured.





Field Properties

The other specific properties of a Field are related to the user interaction.
The 'Preloaded' property is a Boolean flag used to specify if the user can see an initial value inside that field.
The 'Modifiable' property is a Boolean flag used to specify if the user can fill in a value.
The 'Hidden' property is a Boolean flag used to specify whether or not the user can see the field.

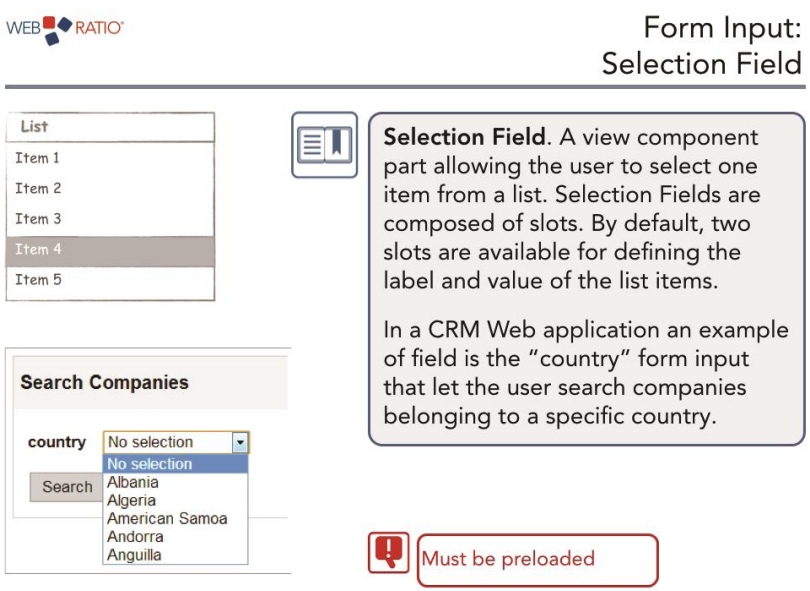


Field - The CRM Case

Let's apply these concepts to the CRM case study to create a Form for searching companies. You need to have an accessible Page and put a Form Component inside it.
Suppose that you want to search companies by name, so you will add a field of type 'string' to the form and rename it 'name'.
You can show all the parts of the Form Component from the 'Appearance' tab of the Form Properties View; this will help you to understand the Form structure by directly looking at the model.

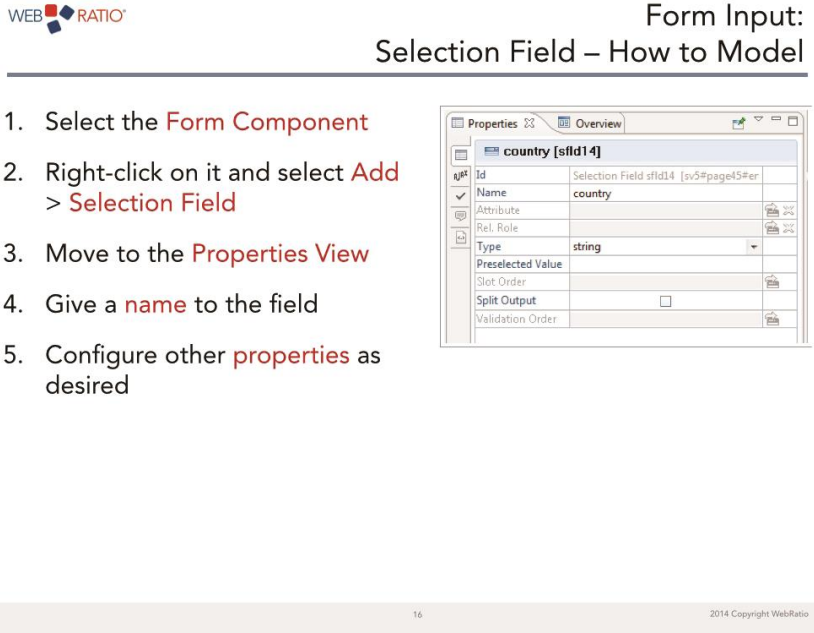
Selection Field

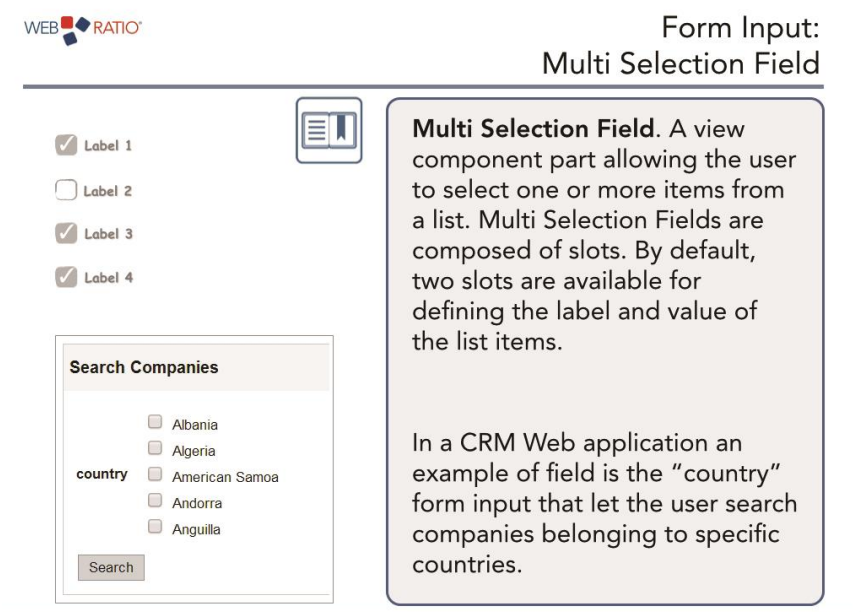
A Selection Field is a View Component part that allows the user to select one item from a list. Each Selection Field is composed of slots and, by default, there are two slots available in a Selection Field; one is to define the label and the other is for the value of the list items. Both labels and values must be preloaded.
In a CRM Web application, an example of selection field is the 'country' form input that lets the user search companies belonging to a specific country.



Selection Field - How to Model

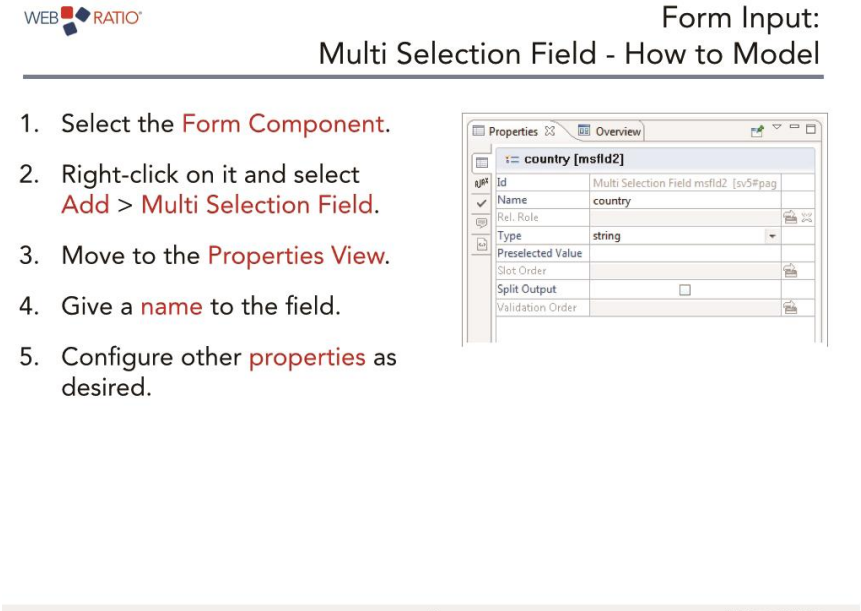
To add a Selection Field, select a Form Component and right-click on it and choose the 'Add -> Selection Field' option. Move to the properties view and change the 'Name' property to set a new name for the Field. Configure all the other properties as desired.





Multi Selection Field

A Multi Selection Field is a View Component part that allows the user to select one or more items from a list. As with Selection Fields, Multi Selection Fields are also composed of slots. By default, there are two slots available in a Multi Selection Field; one is to define the label and the other is for the value of the list items. Both labels and values must be preloaded.
In a CRM Web application, an example of multi selection field is the 'country' form input that lets the user search companies belonging to specific countries.

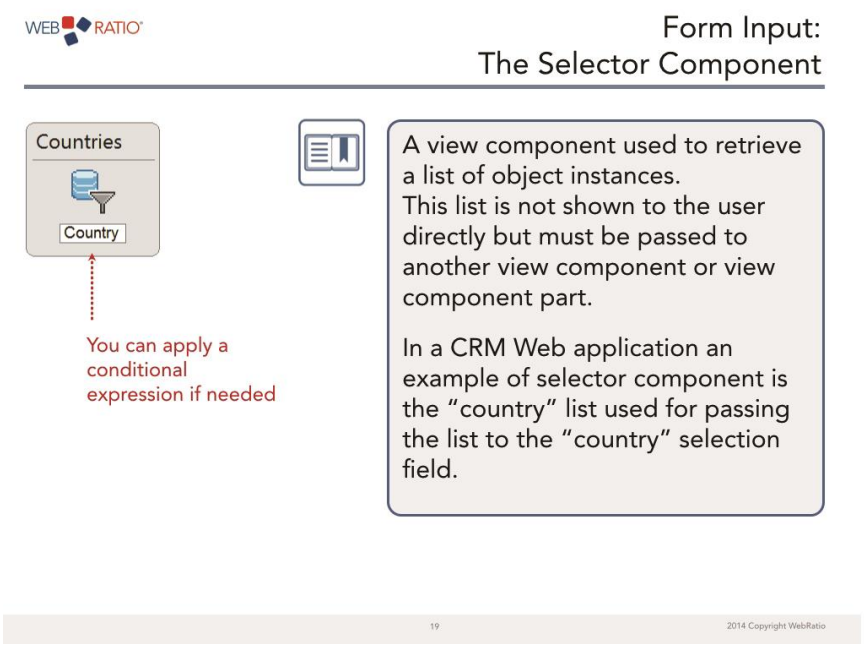


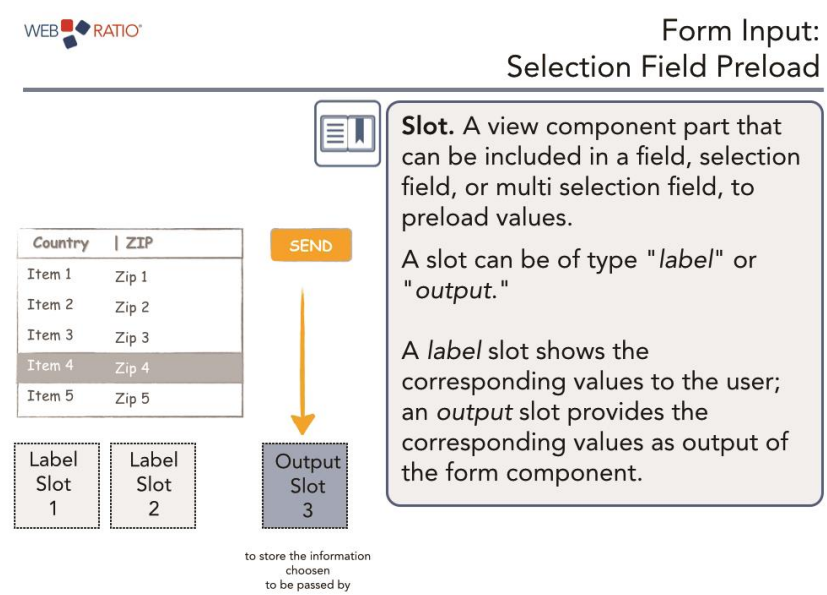
Multi Selection Field - How to Model

To add a Multi Selection Field, select a Form Component and right-click on it, and choose the 'Add -> Multi Selection Field' option. Move to the properties view and change the 'Name' property to set a new name for the Field. Configure all the other properties as desired.

The Selector Component

When modeling pages with forms, data retrieval is a key aspect to take care of.
To retrieve information from the Domain Model without showing it in the page you have to use a Selector Component. The Selector Component is a View Component used to retrieve a list of object instances depending on the Data Binding properties. The list of data retrieved is not shown directly to the user and can be passed as input to another View Component or to a View Component part.
If you have a form with Selection Fields or Multi Selection Fields, one way to preload the content of their label and value slots is to use a Selector Component.
In a CRM Web application, an example of using the Selector Component is to retrieve the 'country' list used to preload the 'country' selection field.



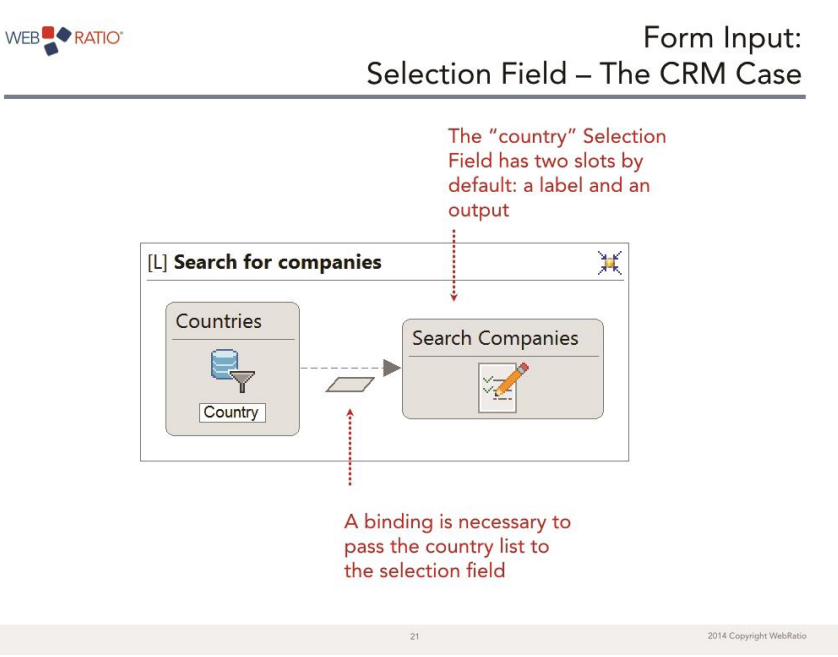


Selection Field Preload

If you want to customize the structure of a Selection Field instead of using a Selector Component, you can use the Slots.
A Slot is a View Component part that can be included in a field, selection field or multi selection field to preload values. Slots can be configured to be 'label' or 'output'.
The Slot of type 'label' is used to manage the values shown to the user; a Slot of type 'output' is used to manage the output value corresponding to each label shown to the user used by the web application.
For instance, let's suppose that you want to let the user choose a country from a Selection Field. Each country is shown inside the option list with the country name and the related zip code. This means that there will be two Slots used in 'label' mode to show data to the user. Since the user also needs to make a selection from this list, an additional 'output' Slot must be provided, which contains the information chosen to be passed as output.

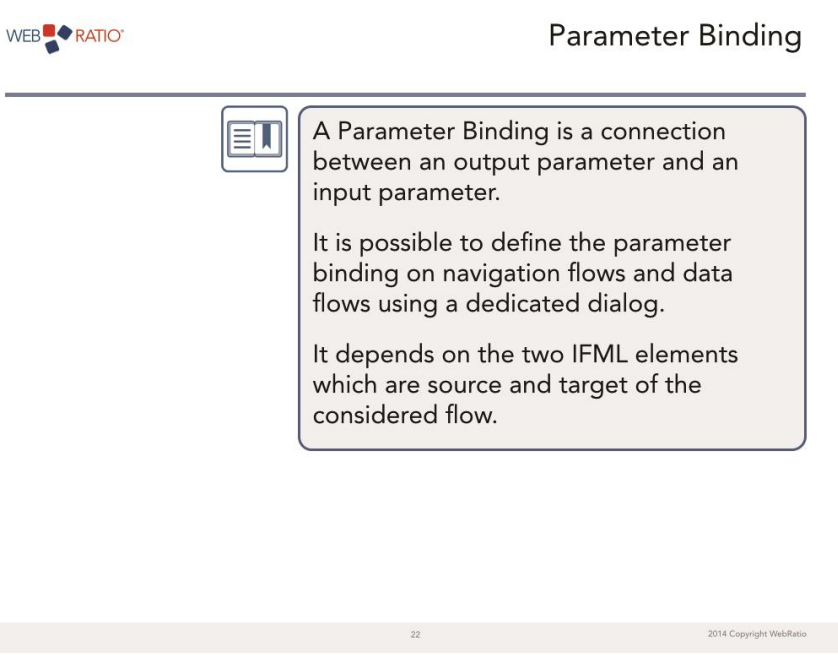
Selection Field - The CRM Case

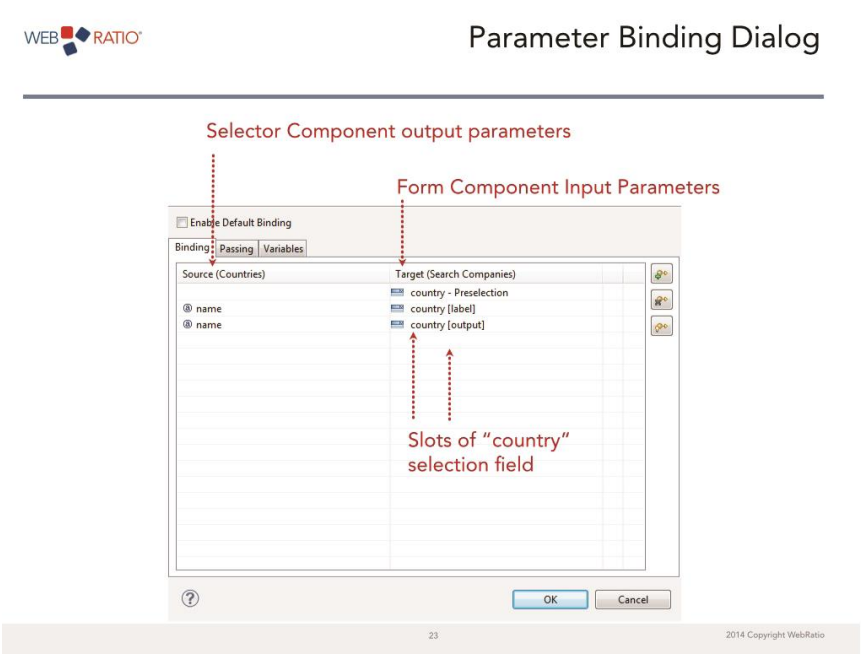
Let's see how to make a page that lets the user search companies by their country.
The Form Component has a Selection Field called 'country', with the two default slots, a label and an output. In this situation, the information used to build the Selection Field comes from the Domain Model, so you need to retrieve data using a Selector, and pass the list of retrieved countries to the Selection Field slots.
To pass data from one component to another you can use a data flow and define its 'binding'.



Parameter Binding

A Parameter Binding is a connection between an output parameter and an input parameter that can define all the types of flow available in WebRatio.
It is used to assign the value of a parameter of the flow's IFML target with one parameter of the flow's IFML source. It is defined using a particular dialog called 'Parameters Binding' dialog that can be opened by double-clicking on the flow.



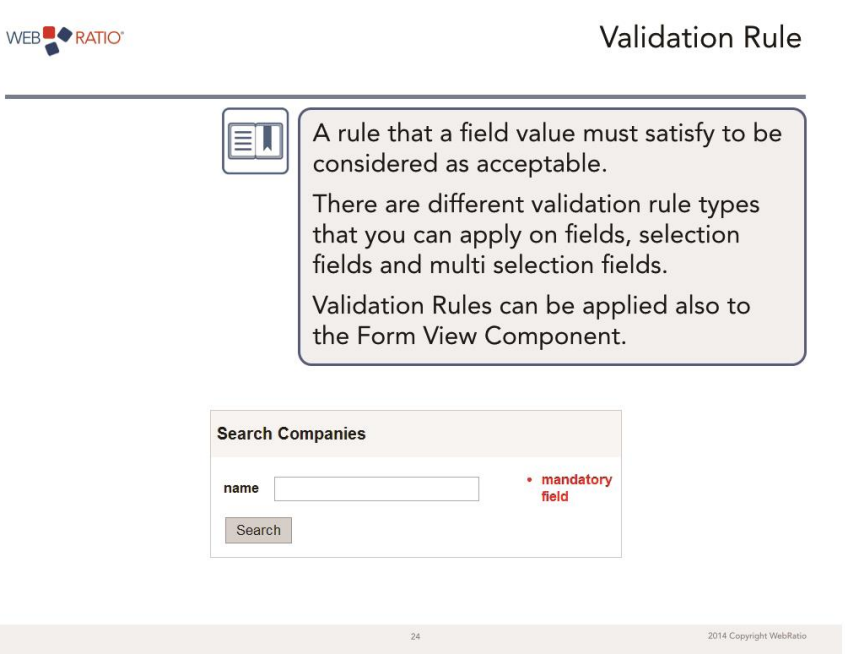


Parameter Binding Dialog

The 'Parameter Binding' is where you can configure the parameter binding on a flow. With reference to the example of the IFML page to search companies by country, this is the parameter-binding dialog of the data flow.
As you can see, this dialog is split into two columns. The one on the left collects all the Source's output parameters; in this case all the values retrieved by the Selector Component. On the right you have all the target's exposed input parameters; in this case all the input parameters accepted by the Form Component, hence the Slots of the 'country' Selection Field.
You can assign parameters by double-clicking on the left side of each row that contains an input parameter on the right; a dropdown is shown with all the possible values, so all the output parameters exposed by the source of the flow.

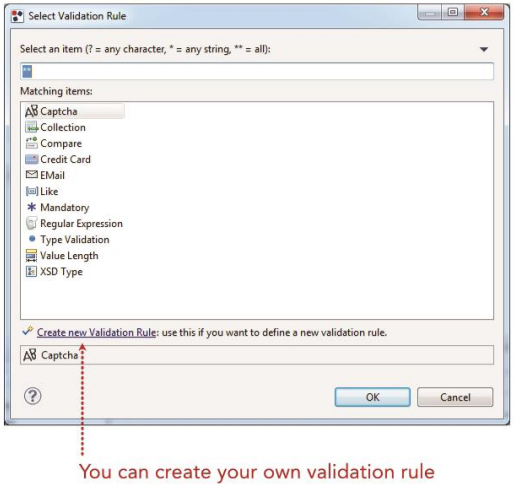
Validation Rule

Another key concept for the Form Component and its View Component parts is the content validation through the usage of Validation Rules. A Validation Rule is a rule that the content of a field must satisfy to be considered as acceptable. An example is the 'Mandatory' Validation Rule that states that a field is mandatory, so it cannot be empty. If a field with a 'Mandatory' rule is left empty and the user submits a form, an error message is shown near the field as feedback for the failed validation. Since the content of fields can be widely different there are many validation rules that can be applied; WebRatio offers a predefined set of rules and also allows you to create new ones. Validation Rules can be applied to any type of Field and can also be applied to an entire Form Component.



WEB RATIO

Validation Rule Types



25 2014 Copyright WebRatio

Validation Rule Types

This is the dialog for the Validation Rule selection, and you can see in the list the default Validation Rules available in WebRatio.
This dialog offers the possibility to start the process for creating a new Validation Rule, which is a custom View Component Part.

WEB RATIO

How to Add a Validation Rule

Select the Form Component.

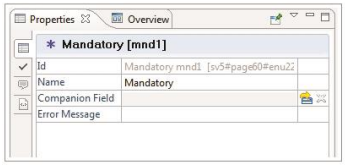
Move to the Outline View.

Select the field you want.

Right-click on the field and choose Add > Validation Rule.

Choose the Validation Rule type you want.

Press "OK" to confirm.



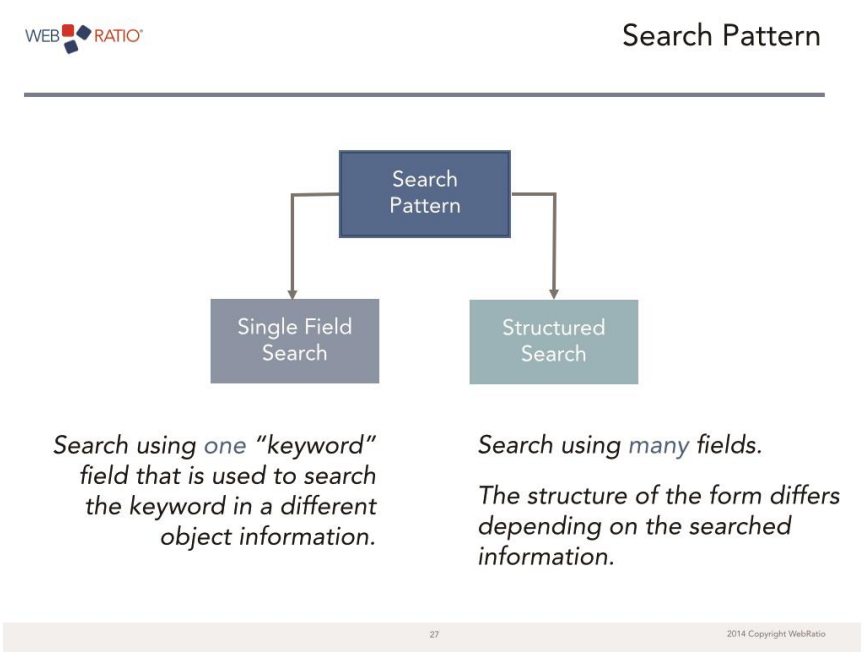
26 2014 Copyright WebRatio

How to Add a Validation Rule

To add a Validation Rule to a Field, select the Form Component to which the Field belongs, and move to the Outline View.
From the hierarchical view select the desired field; right-click on it and choose the 'Add -> Validation Rule' to open the 'Select Validation Rule' dialog.
Choose the validation rule that you are interested in and press the 'OK' button to confirm.

Search Pattern

When it comes to developing any kind of search in your Web application, remember that there are a lot of possible solutions that can be clustered into two categories.
You can easily state that there are two ways to implement the Search Pattern, the Single Field Search and the Structured Search.
Single Field Search is the Search made using one 'keyword' field that filters more than one property of the object instances to look up.
Structured Search is the Search made using many fields. The structure of the Form and the number of fields is strictly dependent on the structure of the information to look up.

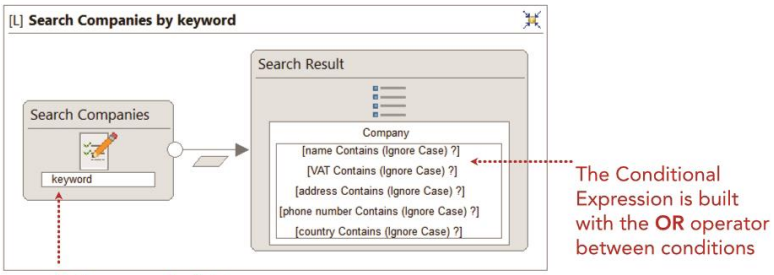


WEB RATIO

Single Field Search:
The CRM Case – Model

Search companies by keyword.

The companies are retrieved by looking for the keyword in the company name, address, VAT and so on.



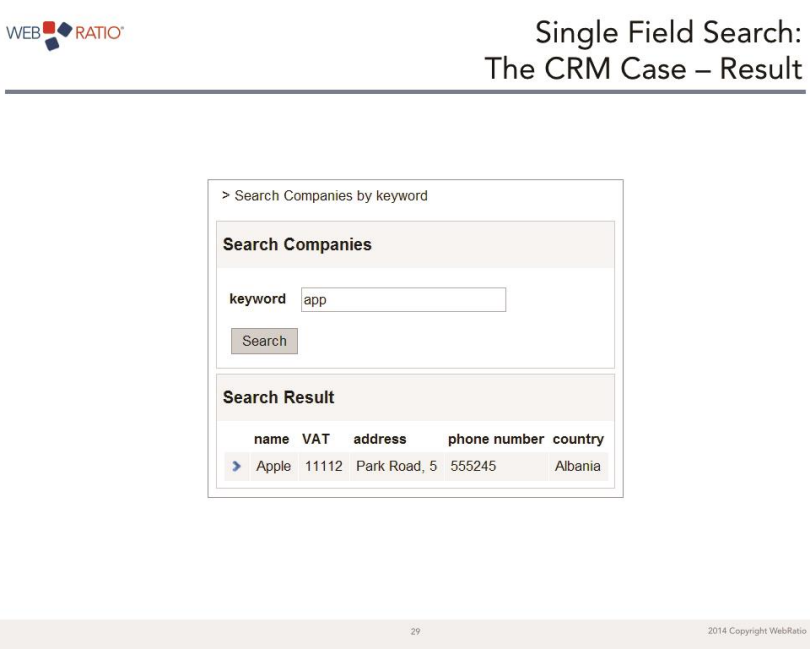
28

The CRM Case - Model

Let's see a practical example of Single Field Search for the CRM Web application.
Suppose we have a page in which, using a keyword, the user can filter at the same time the values of the name, address, VAT, phone number and country attributes.
The page contains a Form Component with a unique field, the keyword field, and a list of companies with a Conditional Expression composed of as many attribute conditions as the attributes to filter.
By default, the Conditional Expression links all the attribute conditions with the 'AND' logical operator.
To perform the query on all the columns, the logical operator must be switched to 'OR' since the keyword value must be used as source for all the columns at the same time.
The normal navigation flow between the Form Component and the List provides to each attribute condition the same input value, which is the keyword field.

The CRM Case - Result

This is how the page looks in the generated Web application.
The page shows only one field and the list of results filtered on each column with the value of the keyword.

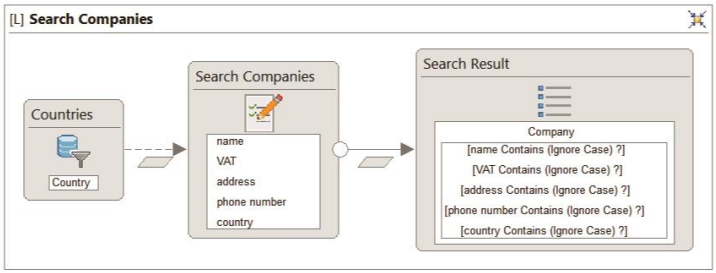


WEB RATIO

Structured Search:
The CRM Case – Model

Search companies.

The search is composed of different fields and each field value is used for searching in a specific company property.

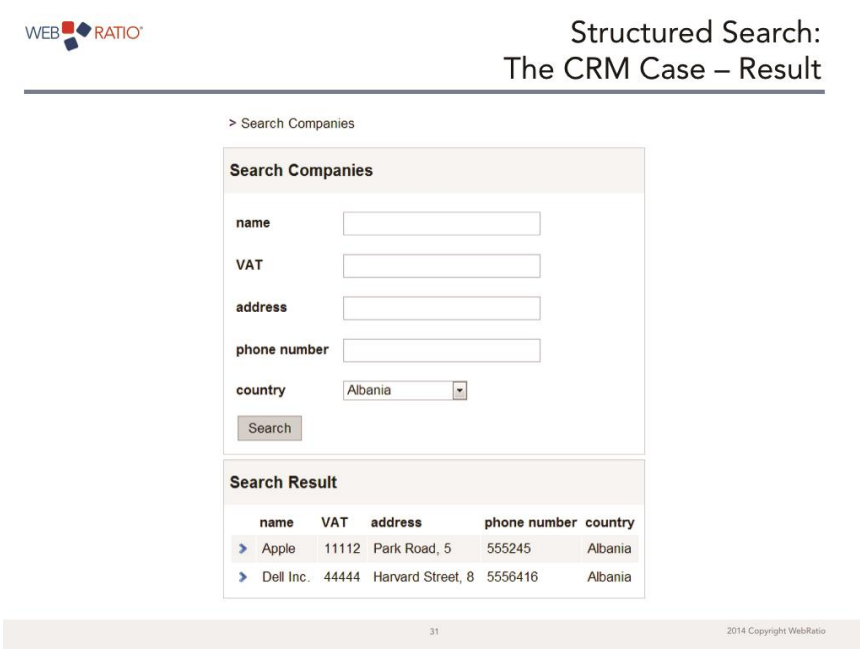


The CRM Case - Model

Let's see a practical example of structured search for the CRM Web application. Suppose we have a page that gives different possibilities to the user for filtering companies. Also, the attributes used to filter are the same as for the keyword search, but the Form shows the user a different field for each attribute.
The name, VAT, address and phone number attributes are represented inside the Form with Fields, while the country attribute is shown inside the Form as a Selection Field.
Since the country list is database information, it is retrieved using a Selector component that passes the list to the Selection Field for labels and values. The normal navigation flow that passes the data to the Conditional Expression of the list associates each attribute condition with a different field of the Form.

The CRM Case - Result

The generated page of the Web application shows a group of fields belonging to the Search Companies form: four fields and one selection field.
Here it will be possible for the user to insert different values, for example in the name field and the address field.



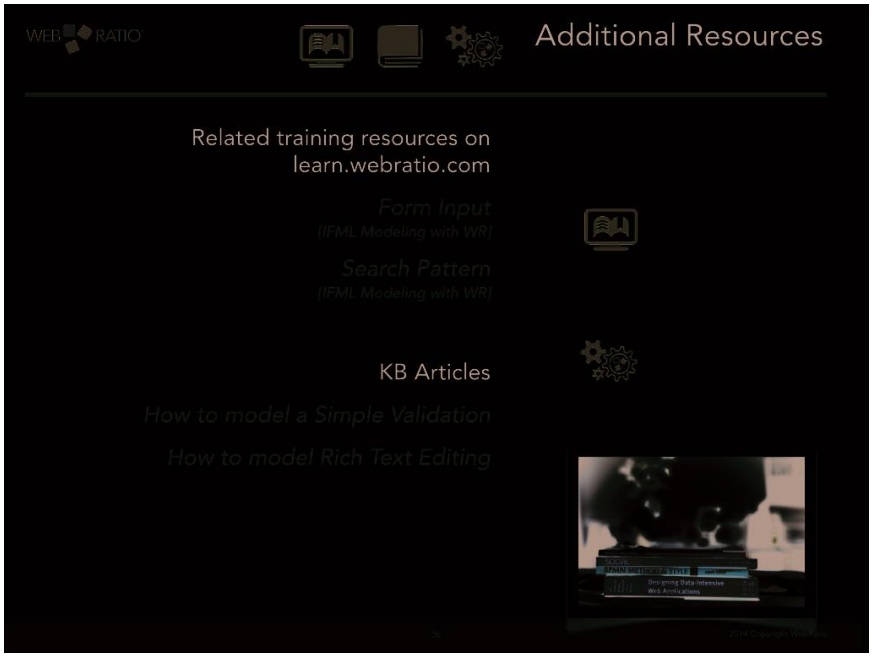
Additional Resources

Online Training

- http://www.webratio.com/learn/learningobject/form-inputs
- http://www.webratio.com/learn/learningobject/search-pattern

KB Articles

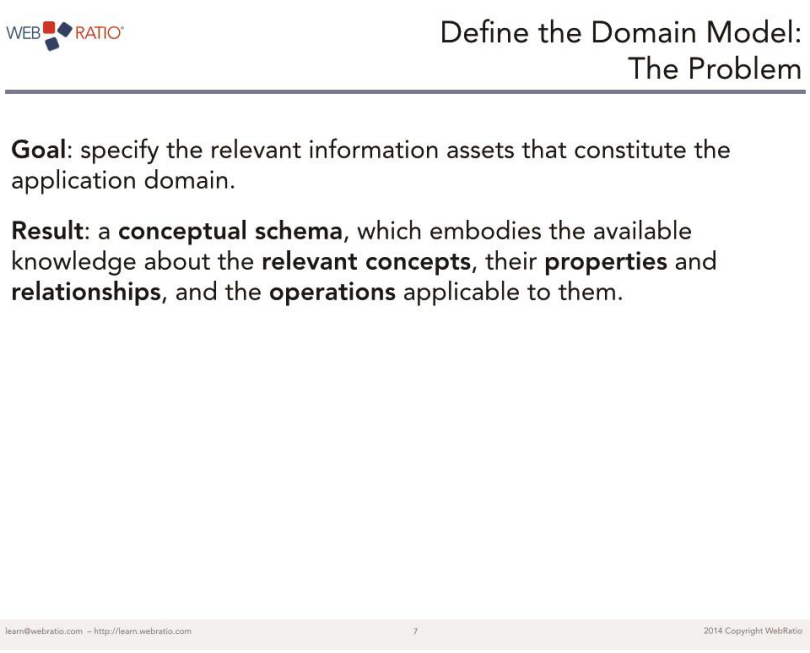
- http://www.webratio.com/learn/learningobject/how-to-model-a-simple-validation
- http://www.webratio.com/learn/learningobject/how-to-model-rich-text-editing



Domain Model

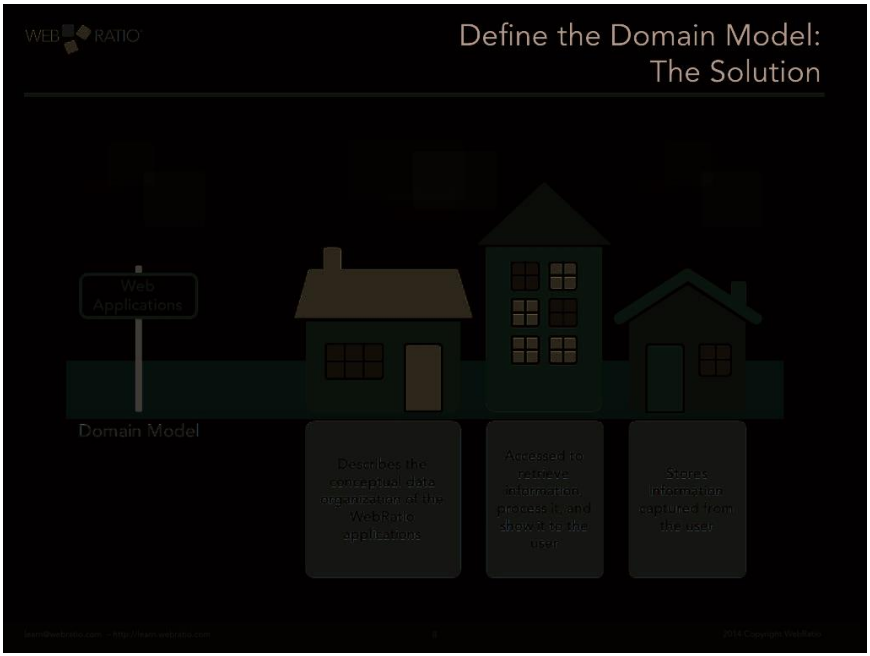
Define the Domain Model: The Problem

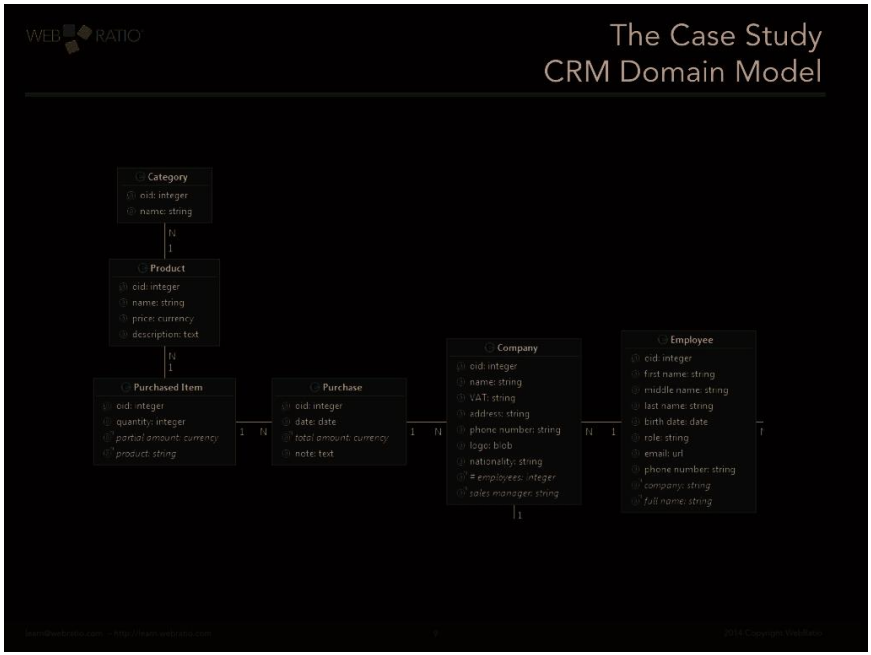
When designing a Web application, the data structure is the first object to define in a Web Project since you need to specify the relevant information assets that constitute the application domain. The result is a conceptual schema that embodies the available knowledge about the relevant concepts, their properties and relationships, and the operations applicable to them.



Define the Domain Model: The Solution

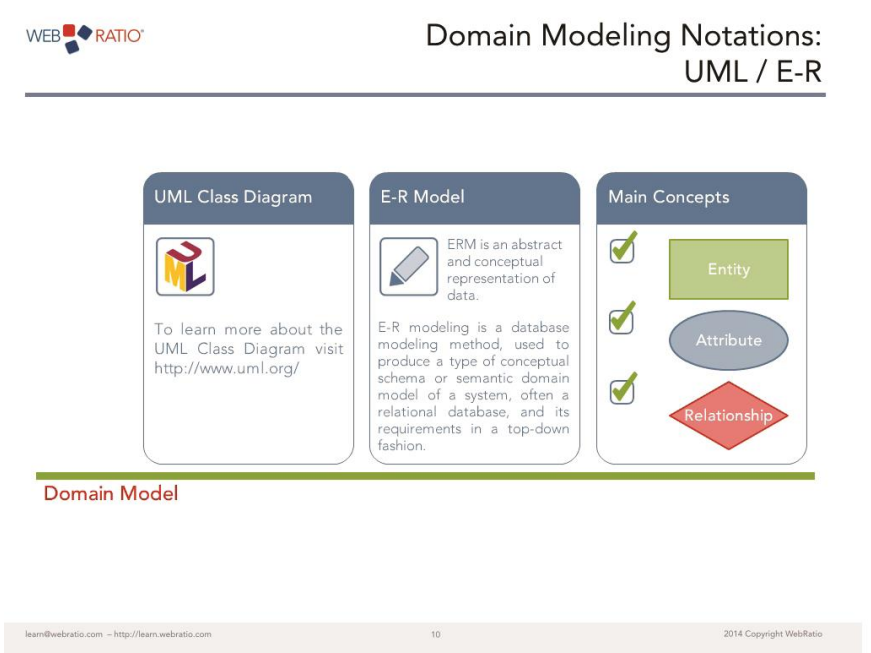
This schema is specified in a dedicated section of the project called Domain Model. WebRatio generates data-centric Web applications.
The Domain Model view is created by default in each Web Project that you create and cannot be removed. The Domain Model of a Web application is like the foundation of a house; it is the underlying structure shared by all the pieces of the web application.
The Domain Model describes the conceptual data organization of WebRatio applications and is accessed to retrieve information, process it, and show it to the user. The Domain Model also stores information captured from the user, through the usage of forms.





The Case Study: CRM Domain Model

An example of a Domain Model is the one of the CRM Web application case studies.
This screen is just a piece of the entire Domain Model and focuses on the main elements.
As you can see, the structure can be easily understood even without knowing the domain-modeling notation.

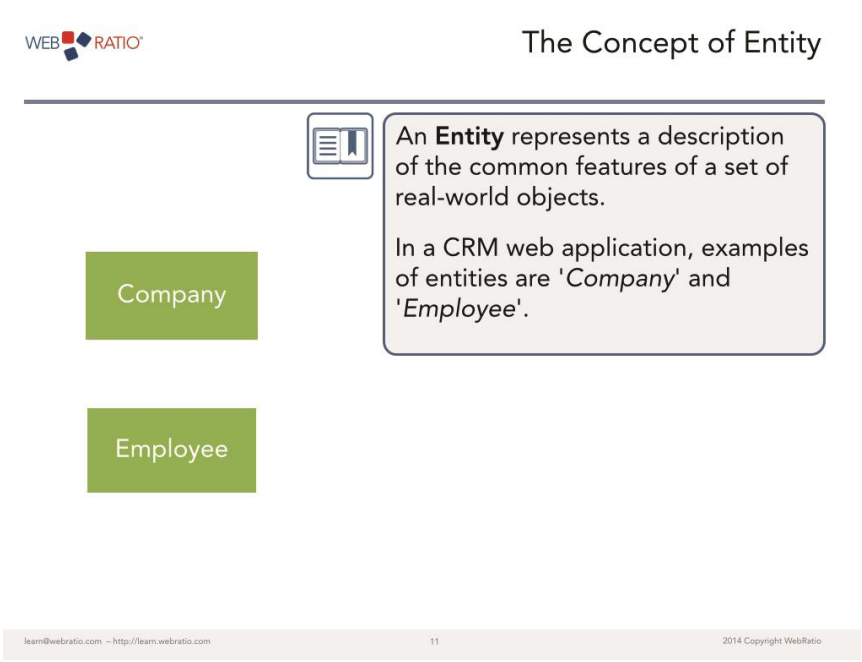


Domain Modeling Notations: UML / E-R

The Domain Model is compatible both with the Entity-Relationship model and UML class diagrams.
Entity-Relationship modeling, also known as ERM, is a database modeling method used to produce a type of conceptual schema of a system, often a relational database and its requirements, in a top-down fashion.
ERM is based on three main concepts: entity, attribute and relationship.
WebRatio uses the E-R notation for specifying the Domain Model.
Let's review each concept in detail.

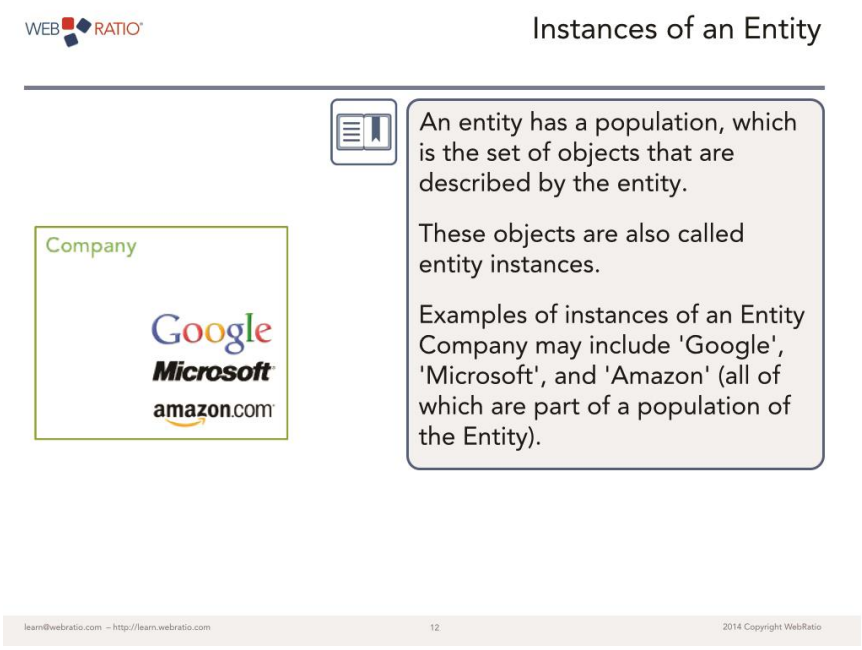
The Concept of Entity

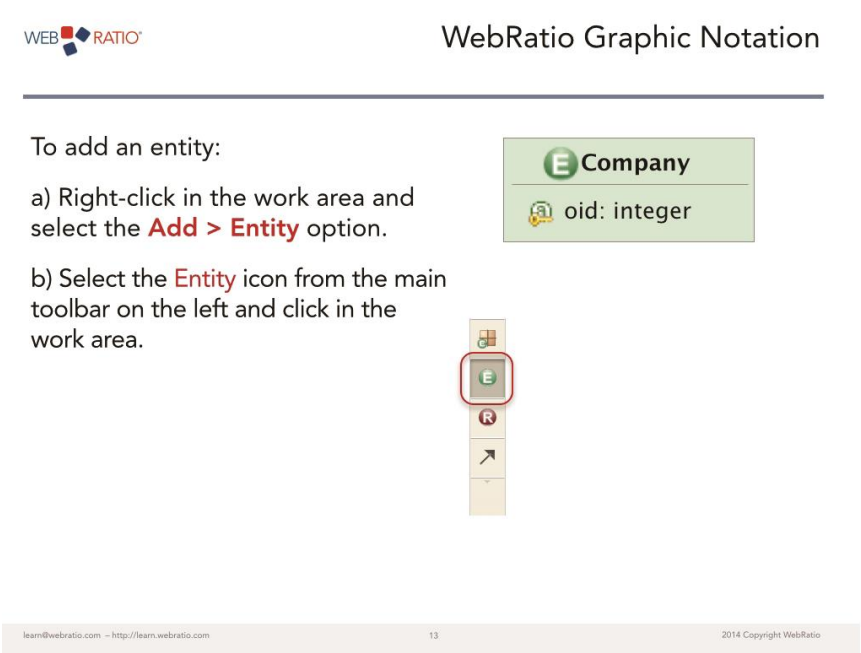
An Entity represents a description of the common features of a set of real-world objects. In a CRM web application, examples of entities are ‘Company’ and ‘Employee’. An Entity has a specific name that helps you understand the entity’s purpose.



Instances of an Entity

An Entity has a population, which is the set of objects that are described by the Entity. These objects are also called entity instances. Examples of instances of the Company Entity may include items such as ‘Google’, ‘Microsoft’ and ‘Amazon’.





WebRatio Graphic Notation

Inside the Domain Model, Entities are denoted by means of rectangles with the entity name at the top.

To add an Entity to the Domain Model, right-click in the work area and select the 'Add -> Entity' option. Alternatively, you can select the Entity icon from the main toolbar on the left and click on the work area.

When you add an Entity to the Domain Model you also get another bit of information, the 'oid' element that represents the entity's primary key.

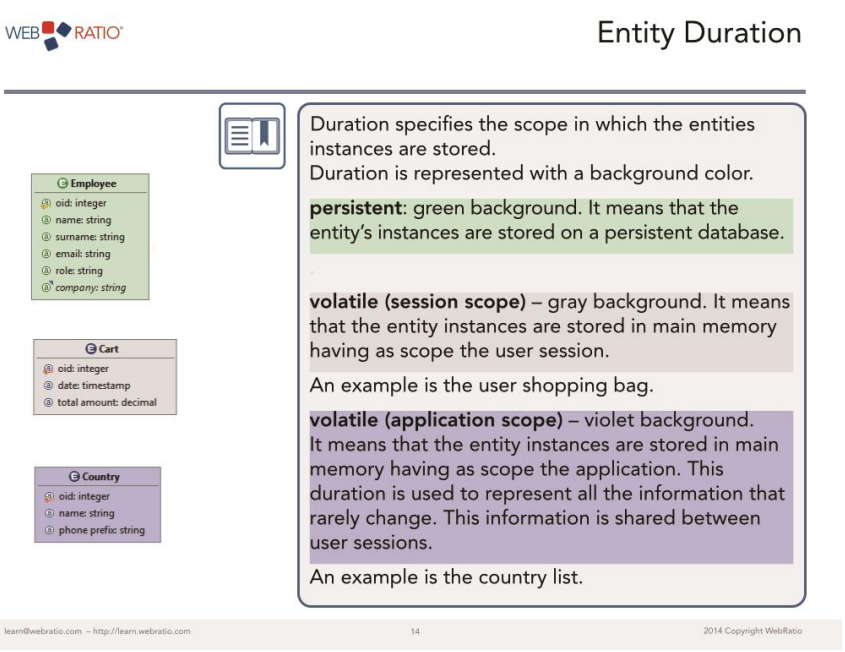
Entity Duration

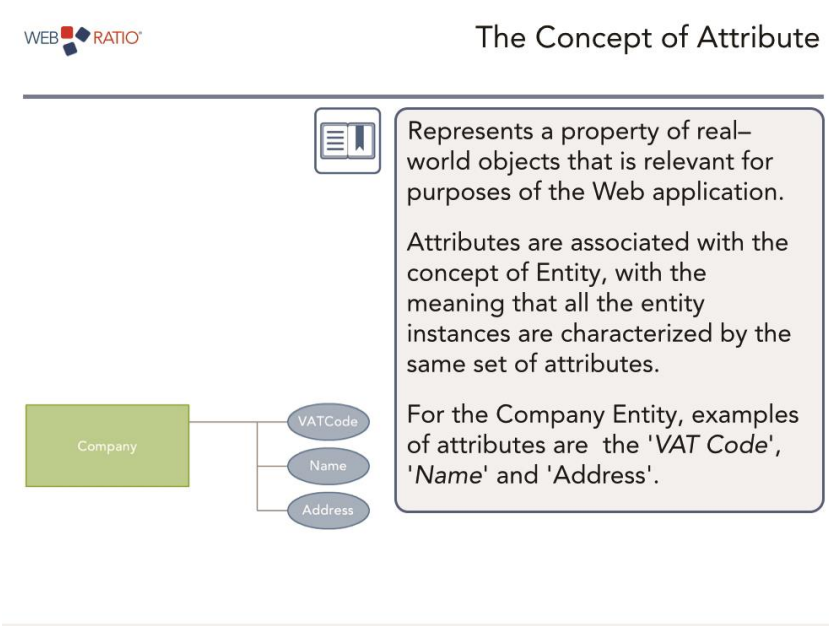
Additional information you can set on an Entity is the Duration, which is the main property of an Entity. It specifies the scope in which the entity's instances are stored. Duration is graphically represented with a background color that changes according to the value of this property. There are three available options for this property. The default option is Persistent, which means that the entity stores information on a persistent database. The background color of the graphic representation is green. An example of a persistent entity in a CRM Web application is the 'Company' entity.

Volatile Session Scope means that the entity instances are stored in the Web application's memory with scope related to the user session. The background color of the graphic representation is gray. An example of Session Scope Volatile entity is the user's 'Shopping Bag'.

Volatile Application Scope means that the entity instances are stored in the Web Application's memory having the application as scope. This duration is used to represent all the information that rarely changes.

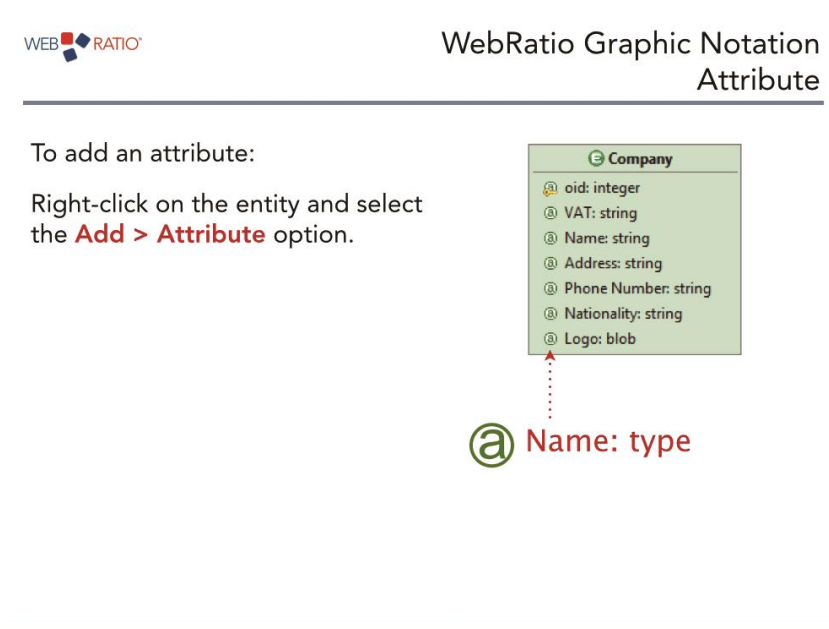
The background color of the graphic representation is violet. An example of Volatile Application Scope is the 'country' entity listing all the existing countries.





The Concept of Attribute

An Attribute represents a property of real-world objects that is relevant for purposes of the Web application.
Attributes are strictly associated with the concept of Entity, with the meaning that all the entity instances are characterized by the same set of attributes. Examples of attributes for the Company Entity are the 'VAT Code', the 'Name' and 'Address'.

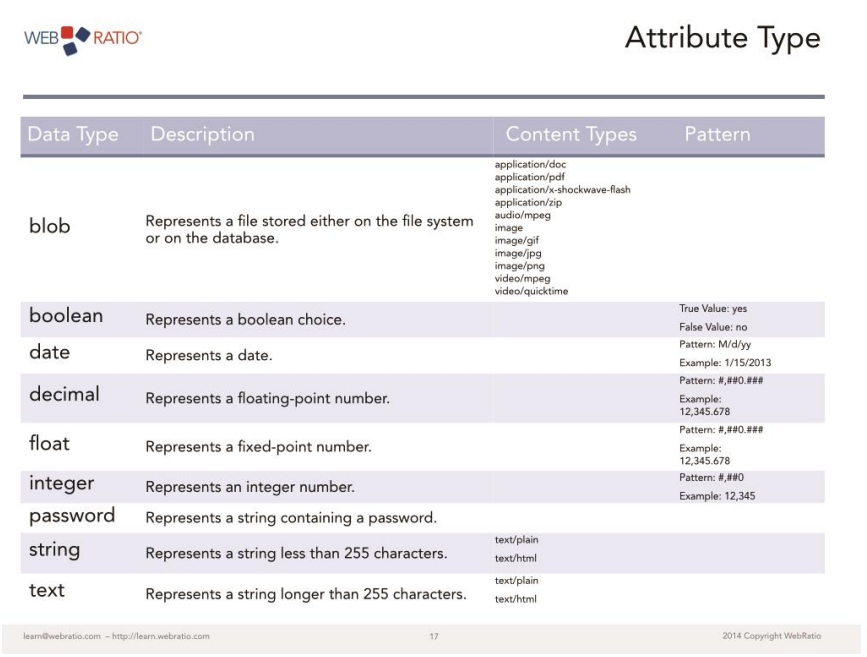


WebRatio Graphic Notation: Attribute

In WebRatio, attributes are shown as sub-elements of an entity.
Each attribute has an icon, a name and a type.
To add an attribute to an entity, right-click on the entity and select the Add -> Attribute option.
You can add several attributes at the same time. Right-click on the Entity and choose the 'Edit Entity' option.

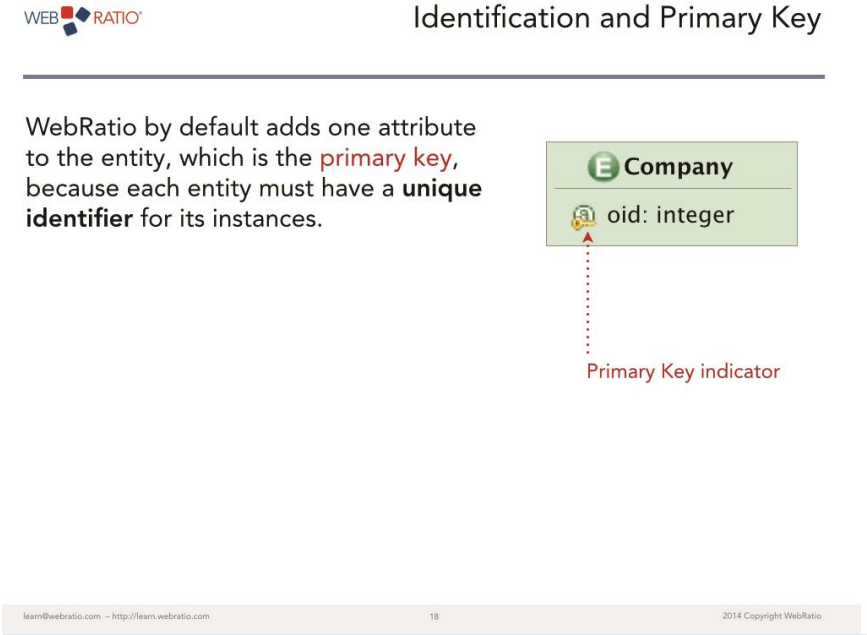
Attribute Type

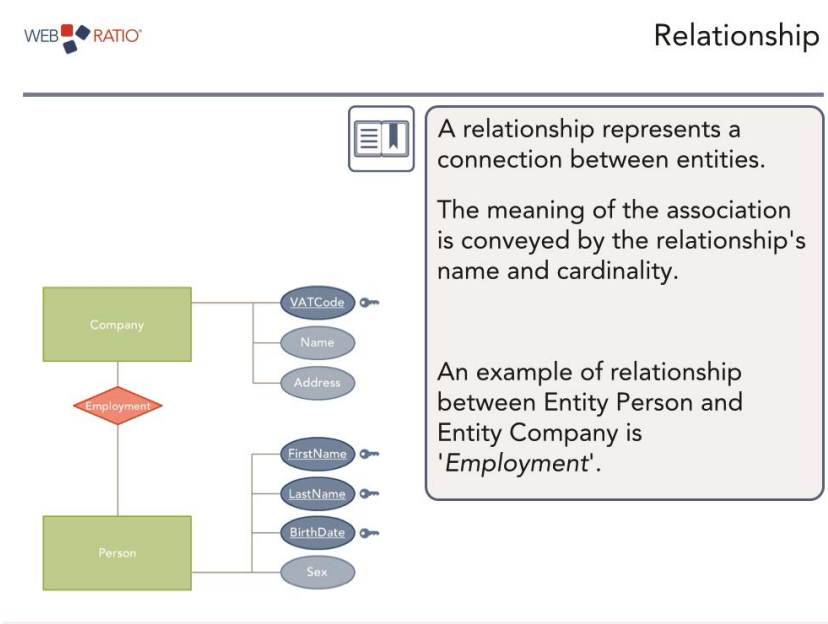
For each entity's attribute, it is necessary to specify a name and a data type. An attribute's name must be unique inside the same entity. The data type is the kind of information contained in the attribute. WebRatio is able to map each generic data type to the specific database type when necessary.
Available data types include primitive types such as string, integer, and so on. Some data types also have a Content Type to restrict the kind of information contained in the attribute. Regarding the type of data stored, some data types also have a Pattern used to format their values, such as the 'Date' type.



Identification and Primary Key

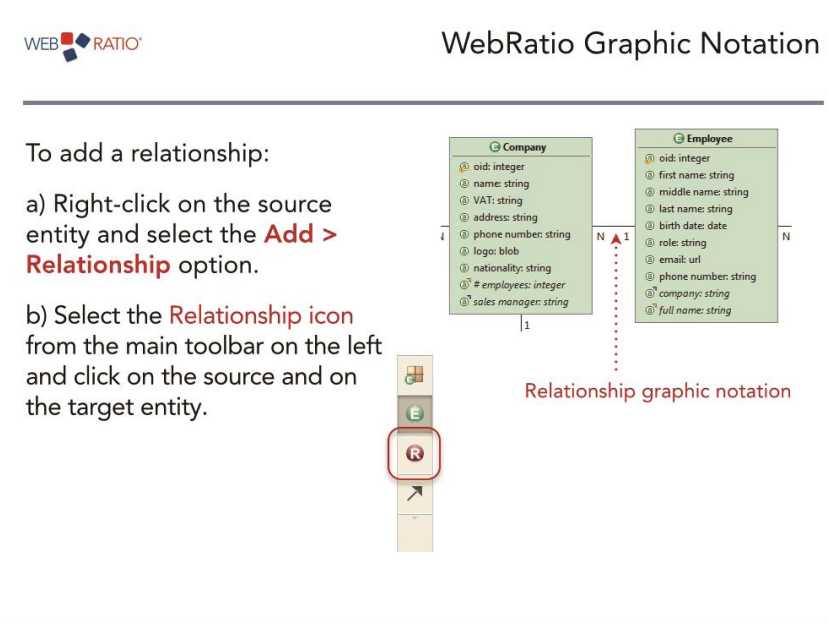
By default, WebRatio adds one attribute to the entity, which is the primary key of the Entity; this happens since each entity must have a unique identifier for its instances. The added attribute is automatically of the integer type integer and comes with name 'oid'. This does not mean that all the identifiers will be always the same since you can change the values of these properties using the attribute properties view. You can also change the primary key by deleting the default one and creating your own.





Relationship

A Relationship represents the connection between two entities of the Domain Model.The meaning of the association is conveyed by the relationship's name and cardinality. An example of a relationship between Entity Person and Entity Company is 'Employment'.

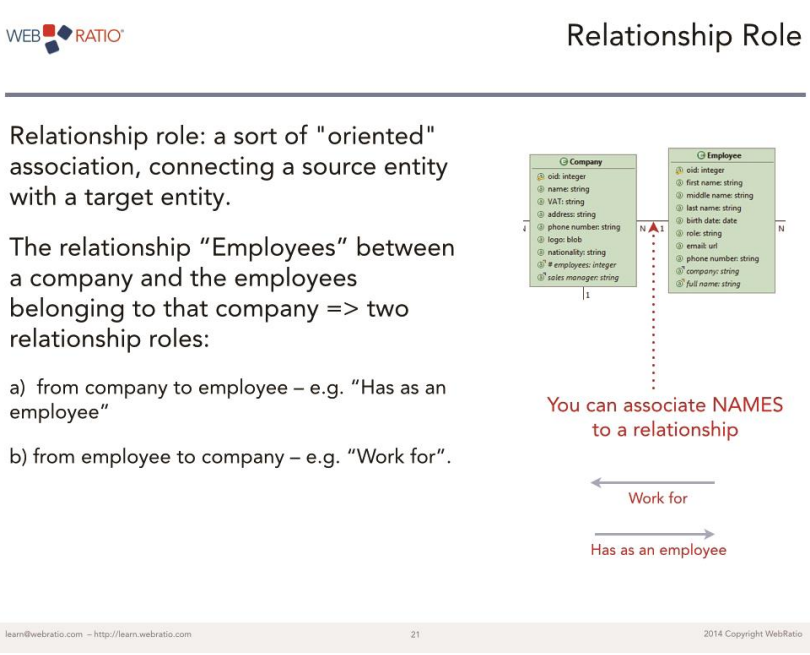


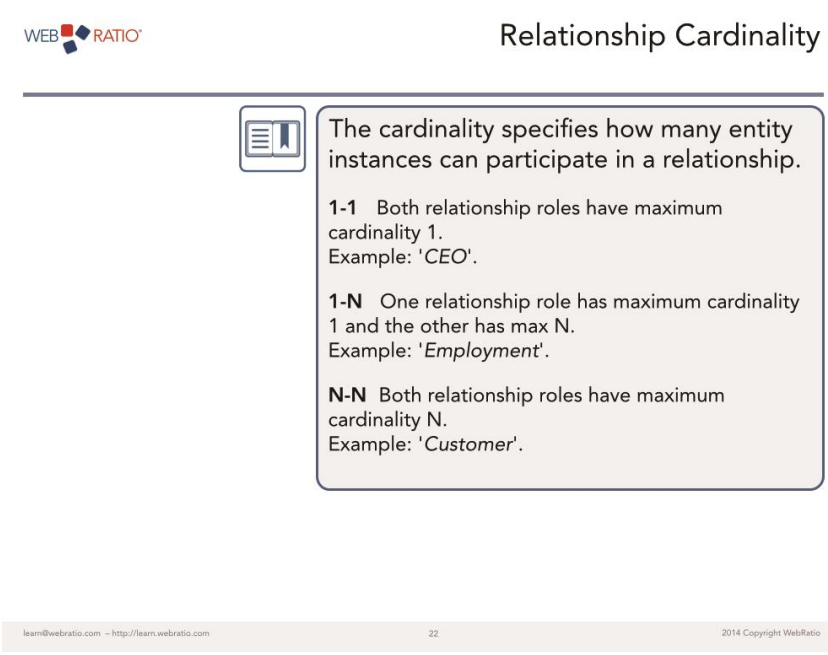
WebRatio Graphic Notation

In WebRatio, Relationship are represented with solid lines connecting entities. To add a Relationship, right-click on the source entity and select the Add -> Relationship option, or select the Relationship icon from the main toolbar on the left and click on the source, and then on the target entity. The name that is automatically assigned to the Relationship strictly depends on the order of clicks made.
By default, WebRatio concatenates the names of the source entity and the target entity, using an underscore as separator. In the example of the relationship connecting Company and Person, the name is 'Company_Person'.

Relationship Role

Each Relationship is characterized by two Relationship Roles, each one explaining the function that one of the participating entities plays in the relationship. The two available roles are respectively called ‘Direct’ and ‘Inverse Role’, and correspond to the order of clicks for source and target entities.
A Relationship Role is a sort of oriented association, which connects a source entity with a target entity. For the CRM Web Application, examples of Relationship Roles are the two associations connecting Employee and Company; one role is ‘Works for’ to identify the function of the Employee related to the Company, the other is ‘Employees’, which expresses the relationship from the Company perspective.





Relationship Cardinality

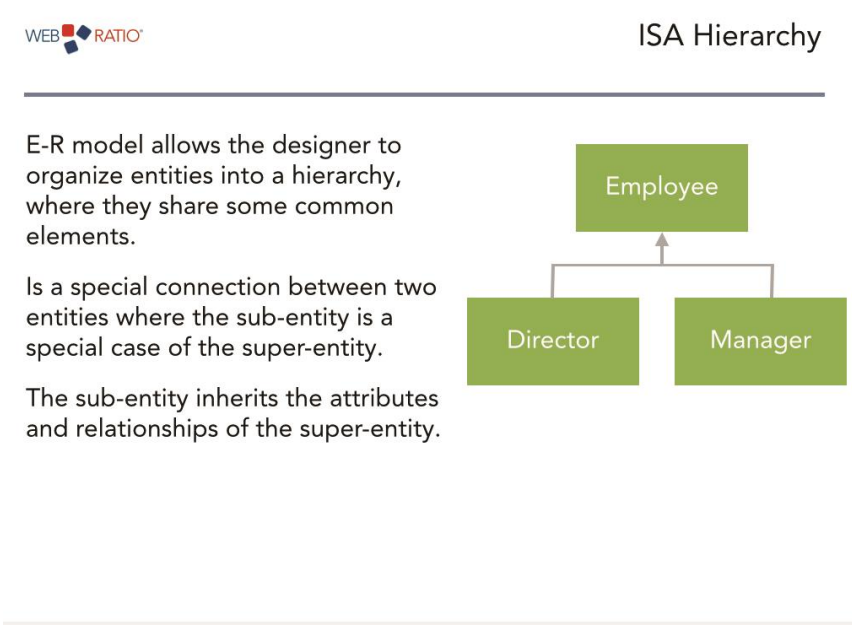
The relationship cardinality specifies how many entity instances can participate in the relationship. It can be defined directly on relationship roles with the maximum cardinality constraints, which denote the maximum number of objects involved in each role. The available values are one (1) and many (N). Starting from the maximum cardinality constraints, you get relationship cardinality, which is a combination of the possible values. One-to-one (1-1) relationship exists when both roles have maximum cardinality of one (1). An example is the relationship connecting a Company and its CEO.

One-to-many (1-N) relationship is when one role has one (1) and the other role has (N) as maximum cardinality. An example is the relationship connecting a Company and its Employees.

Many-to-many (N-N) relationships exists when both roles have maximum cardinality many (N). An example is the relationship connecting Company and its Customers. The minimum cardinality is not set since it is '0' by default.

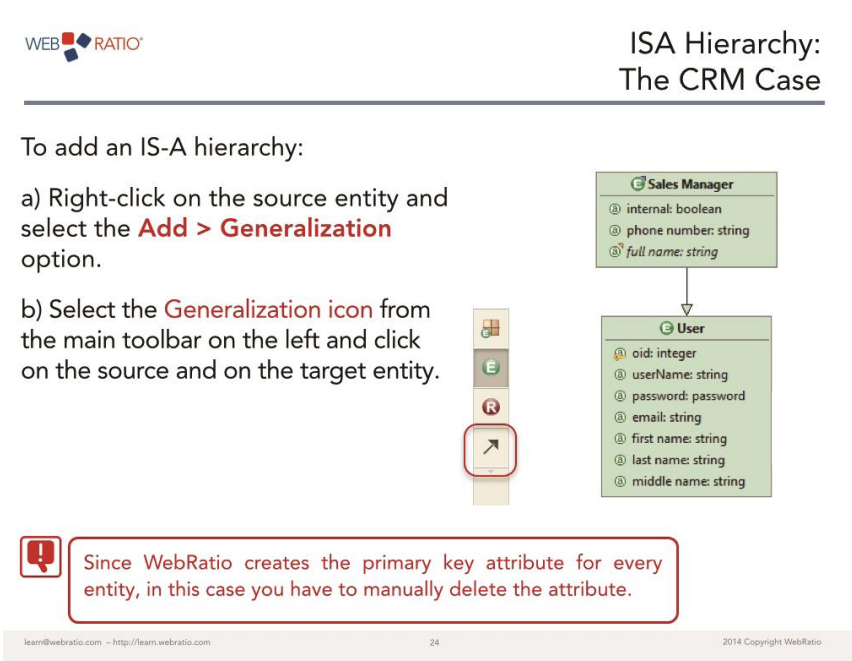
ISA Hierarchy

ERM allows the developer to organize entities into a hierarchy to share some common properties. This basic generalization hierarchy, also called ISA Hierarchy, has one super-entity and one or more sub-entities, which share all the attributes and relationships of the master entity.
An example of ISA Hierarchy can be defining the Director and the Manager as special types of Employees. An Entity defined with the ISA constraint can also have specific attributes defined.



ISA Hierarchy: The CRM Case

To add an ISA Hierarchy, you can right-click on the source entity, select the Add -> Generalization option, and then click on the target entity.
Alternatively you can select the Generalization icon from the toolbar, click on the source and then on the target entity.
In the CRM Case Study, the 'Sales Manager' Entity is considered as a special type of user. Since WebRatio automatically creates the primary key attribute for every entity, in this case you have to manually delete the key attribute. The reason is that since each Sales Manager is a User, there is no need to have a duplicate primary key for both objects.
The unique identifier must stay on the super-entity, which is 'User'.



WEB RATIO

Schema Normalization

Normalization: apply a set of rules to E-R schema for minimum redundancy in the data and a model that can be easily translated in a physical database.

E-R concepts affected by normalization:

- ✓ Multivalued Attributes
- ✓ Structured Attributes
- ✓ Relationship with Attributes
- ✓ N-ary Relationships

learn@webratio.com – http://learn.webratio.com 26 2014 Copyright WebRatio

Schema Normalization

Standard ERM includes some concepts that are not minimal and cannot be modeled inside the Domain Model. You need to 'reduce' the complexity of the E-R diagram using the three main concepts of ERM, entity, attribute and relationship. This process of simplification is called 'Normalization', which means applying a set of rules to ER schema to achieve minimum redundancy in the data and a model that can be easily translated in a physical database.
There are four concepts of the standard ERM affected by normalization; multivalued attributes, structured attributes, relationship with attributes and n-ary relationships.

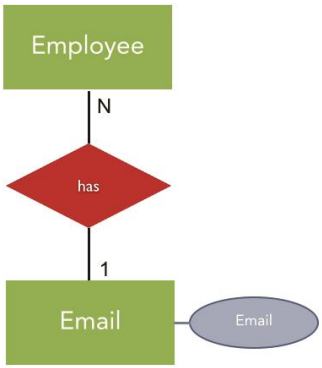
WEB RATIO

Multivalued Attributes

Attributes that have a set of values for the same entity instance.

An email can be a good example of this attribute. Employees can have several email addresses.

<u>How to normalize:</u>
promote attributes connected with a 1-N relationship with the main entity to new entities.



The NEW Entity is called a 'weak entity' because its instances do not exist.

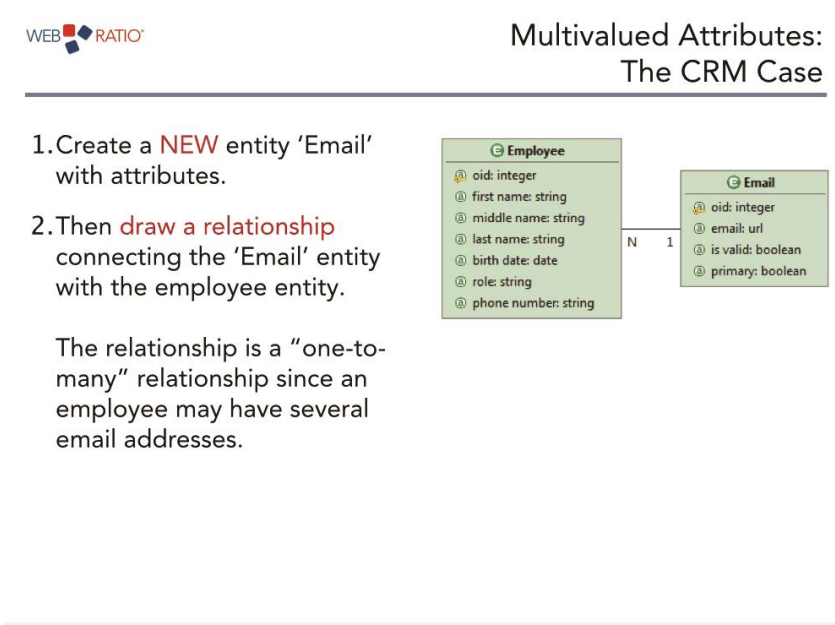
learn@webratio.com – http://learn.webratio.com 27 2014 Copyright WebRatio

Multivalued Attributes

Multivalued attributes are used to assign a set of values to the same entity instance. The email address of an employee is a good example of multivalued attribute, since each employee is supposed to have several email addresses.
The procedure to normalize this type of construct is to transform the multivalued attribute into an entity connected through a 1-N relationship to the main entity. In the email example, the email attribute becomes a separated entity. The email entity takes the name of 'weak entity' because its instances cannot exist without the main entity instance, the employee. This means that if you delete an 'Employee' you should also remove all the 'Email' addresses since they cannot exist without the 'strong entity' instance.

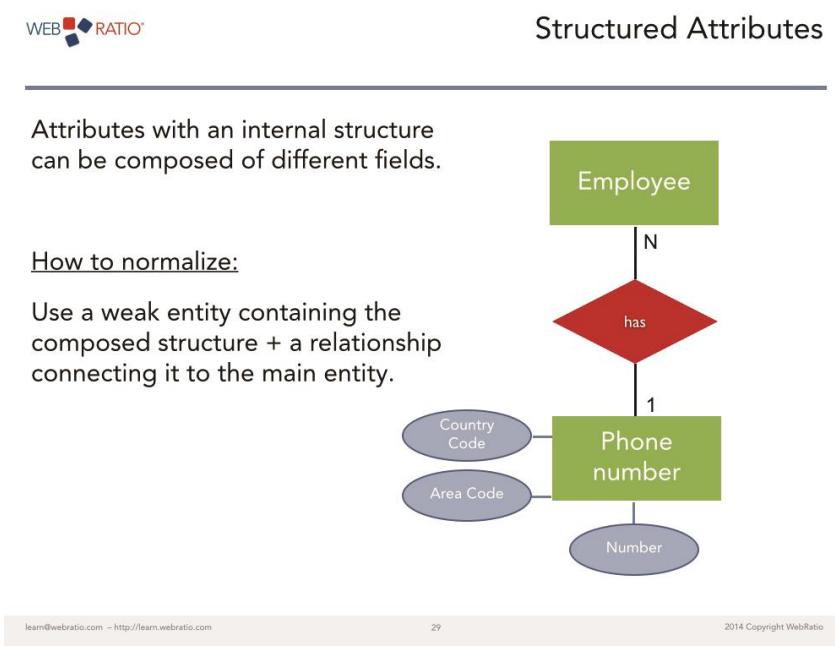
The CRM Case

Since each employee has one or more emails, you can create a new entity 'Email' with the 'email' attribute that contains the real address.
Draw the relationship connecting the 'Email' entity with the 'Employee' entity, adding a one-to-many relationship.
Then you can add other attributes to the 'Email' entity, such as Boolean flags, to understand if the address is 'valid' or is the 'primary' email for the Employee.



Structured Attributes

Structured attributes, also known as composed attributes, are attributes with an internal structure composed of different information.
The Phone Number of an Employee is an example of structured attribute since it is made of different parts, such as the country code, the exchange code, the internal code and the number itself.
The procedure to normalize this type of attribute is to use a separated entity containing the composed structure, and a relationship connecting it to the main entity.

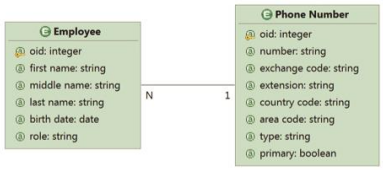


WEB RATIO

Structured Attributes: The CRM Case

The phone number can be split into several parts (e.g. country code, exchange code, number...).

If you want to handle each item of information separately then create a separate entity for the phone number.



The NEW entity "Phone Number" is called 'weak entity' since it can NOT exist without the associated "Employee" objects.
For Example, you should delete the phone number of John Doe (+99 991) if you delete 'John Doe' employee.

learn@webratio.com – http://learn.webratio.com 30 2014 Copyright WebRatio

Structured Attributes: The CRM Case

In the CRM Web Application, it can be useful to separate each part composing the phone number, so we can add a new entity 'Phone Number', with attributes like 'country code', 'exchange code' and number.
Connect this entity with the Employee entity using a one-to-many relationship since each Employee can have different phone numbers.
Like for Multivalued attributes, this new entity is a weak entity so if you delete an instance of the Employee entity you should also delete its phone numbers.

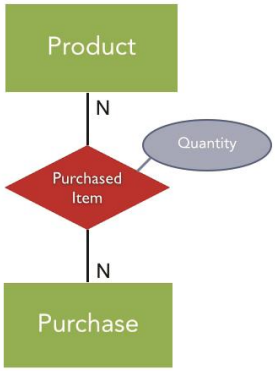
WEB RATIO

Relationship with Attributes

Is a description of a property that refers to a pair of objects.

Relationships need attributes to be completely defined.

How to normalize:
In the Domain Model, relationship can not have attributes, therefore promote the relationship into an entity + create 1:N relationships.



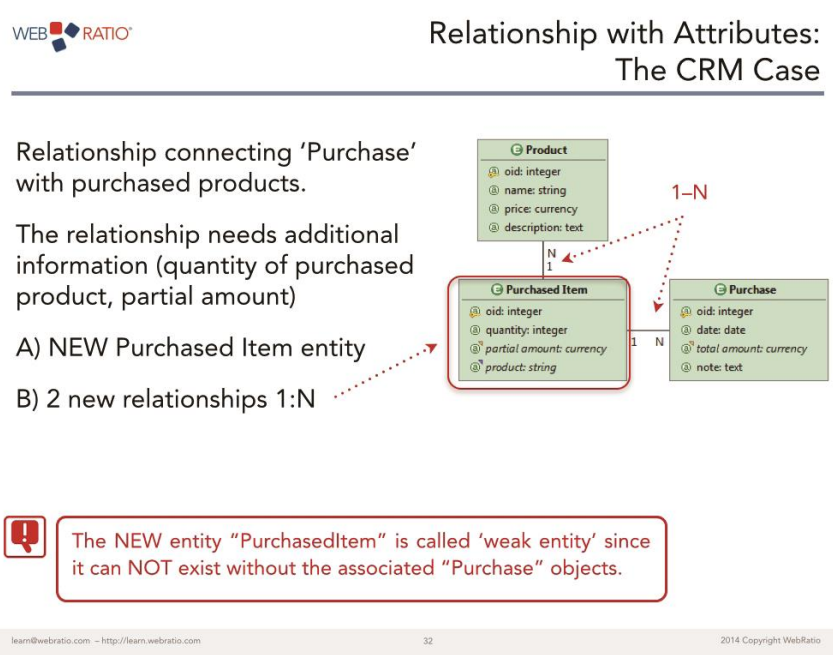
learn@webratio.com – http://learn.webratio.com 31 2014 Copyright WebRatio

Relationship with Attributes

A relationship with attributes is used to describe properties that refer to a pair of objects. It is used when a relationship needs attributes to be completely defined. An example is the relationship between Product and Purchase to describe a Purchased Item. Without the 'quantity' property this relationship does not make sense.
In a standard ERM application, relationships can have attributes; in the Domain Model, this is not supported. To normalize this kind of relationship, you need to promote the relationship into an entity so that the attribute can be easily added as a property, and connect both the entities with the new one using one-to-many (1:N) relationships.

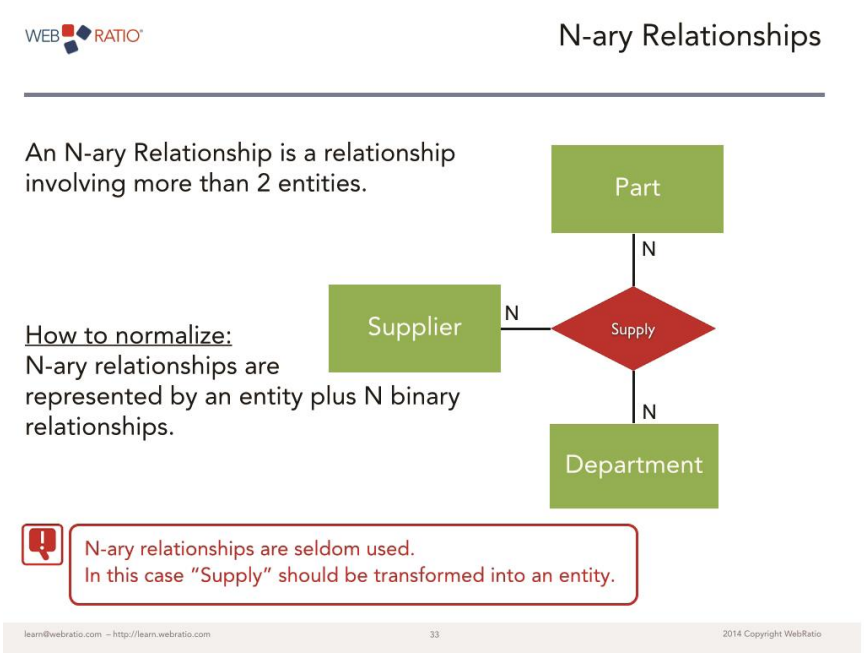
Relationship with Attributes: The CRM Case

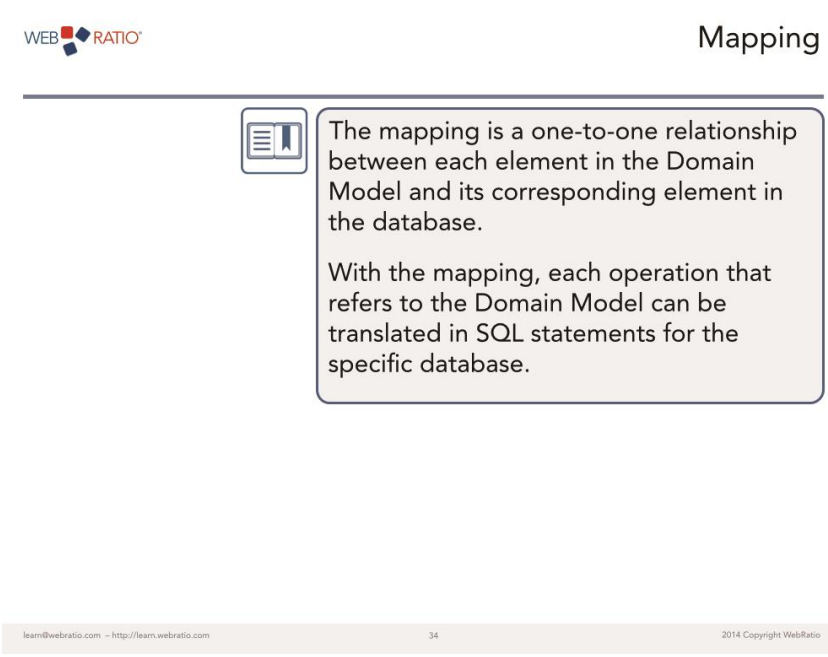
Let's use the same example of the ERM inside the Domain Model of the CRM Case Study.
At the beginning, you have the 'Product' and the 'Purchase' entity connected with a many-to-many (N-N) relationship. To specify the purchased quantity of a product, we create a new entity 'Purchased Item' with the 'quantity' attribute. The 'Product' and 'Purchase' entities are both associated to the new entity with new one-to-many (1-N) relationships.
As you can see, using the two new relationships the cardinality of the two ends is still many-to-many (N-N). The new entity 'Purchased Item' is a weak entity since it cannot exist without the associated 'Purchase' object.



N-ary Relationships

ERM allows creating N-ary Relationships as relationships involving more than two entities.
An example of an N-ary Relationship is the 'Supply' relationship connecting the entity's 'Part', 'Supplier' and 'Department'.
Since in WebRatio's Domain Model relationships can only be binary, the solution is to promote the N-ary Relationship into an entity, and then connect all the other entities with binary relationships. Since this kind of relationship is not very common, there are no examples in this guide.





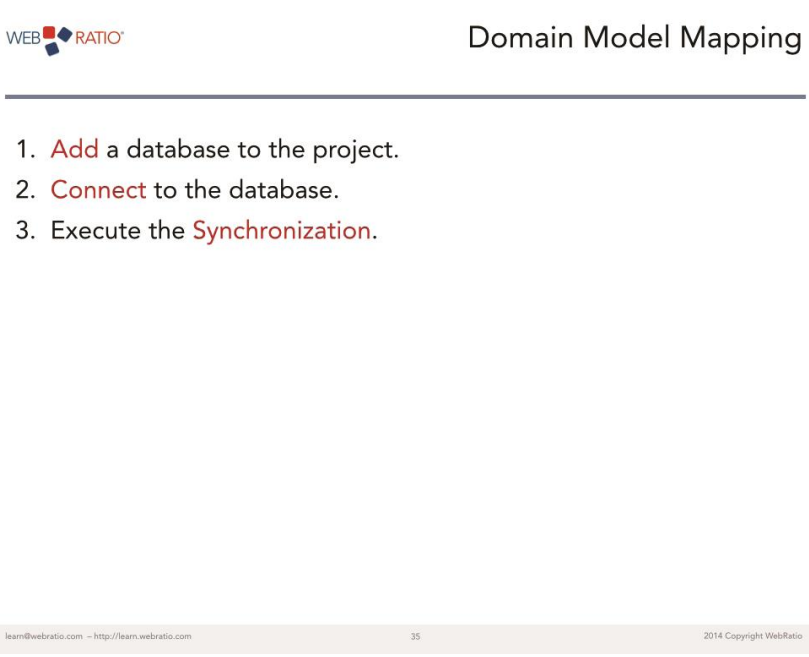
Mapping

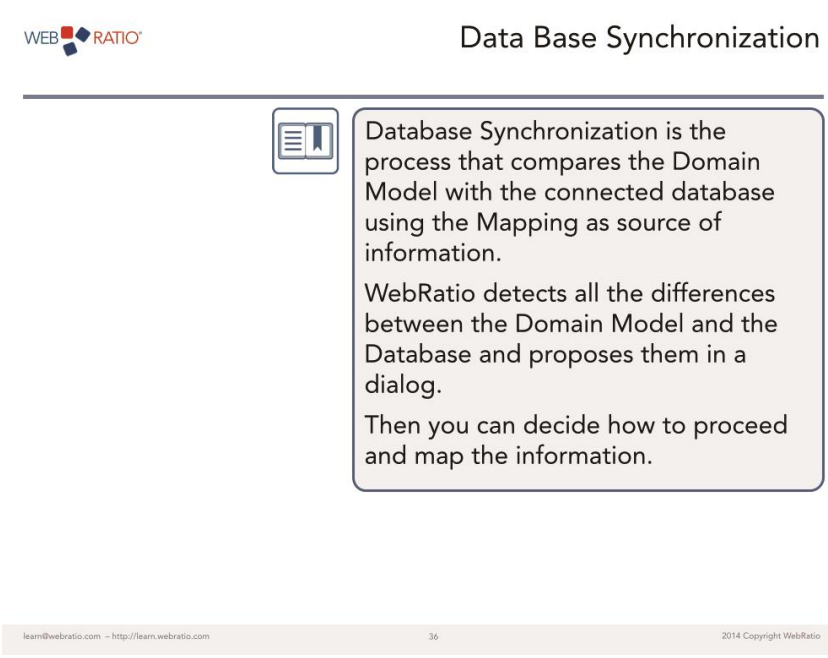
Given that the Domain Model is just the abstract representation of data, it can be transposed into physical objects obtaining the mapping. In fact, once the Domain Model has been defined, it can be connected to a database, if necessary.

The database connection allows you to define the mapping that can be defined as a one-to-one relationship between each element in the Domain Model and its corresponding element in the Database. The mapping translates each operation associated with Domain Model items into a SQL statement for the specific Database type.

Domain Model Mapping

To effectively map the Domain Model you need to go through three steps; add a database, connect to itm and map the structure into it.
To add a database, right-click in the work area and choose the Add -> Database option, or right-click on the Domain Model node in the Outline View and choose the same command.
Once you add the Database node you can set its connection properties from the property view. The connection properties for the Database include the 'Type' of Database, and in WebRatio you can use almost every type of database. The other connection properties are the 'URL', the 'Username' and 'Password' to access the Database.
When the property's configuration ends, test the connection to the database using the Refresh option. The Refresh option can be successful, or can result in a failure. The result, either a failure or success, is shown as a sub-node of the Database node in the outline view.
Failures are usually related to connection issues due to wrong property settings; the failure reason is shown in an 'Error' node. If the refresh is successful, you are able to see the metadata-node with all the meta-information of the Database.
The last step is to transform the Domain Model into database objects using the process called 'Synchronization'.





Database Synchronization

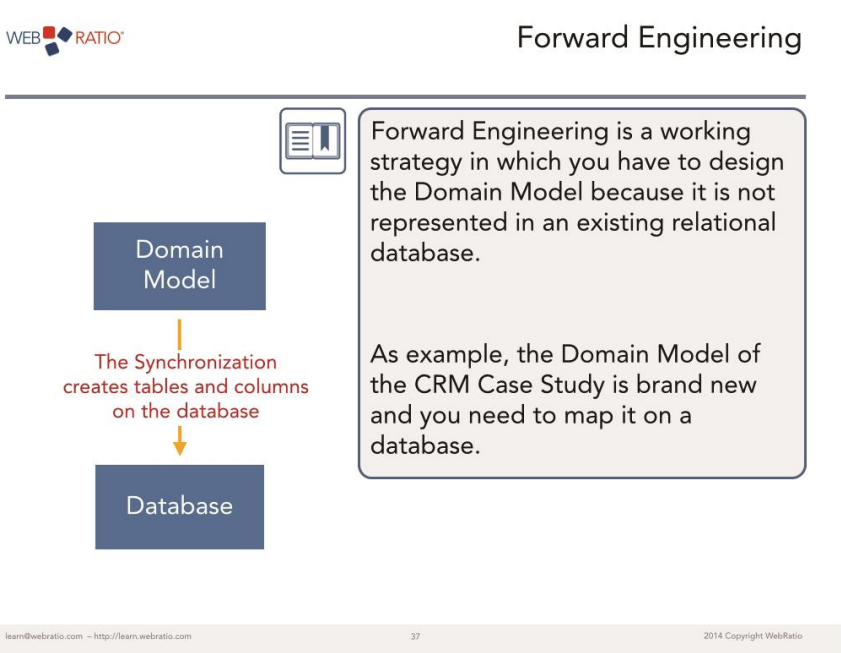
Database Synchronization is the process that creates the mapping. This process compares the Domain Model and the connected database content using the Mapping as source of information, placing all the found differences inside a dedicated dialog.

Since looking for differences is a two-way process, you see all the items defined on the Domain Model that are not referenced to the Domain Model, and vice-versa. This means that once you have the dialog with all the differences you can decide how to proceed and eventually map the information, working from the Domain Model to the Database (Forward Engineering) or from the Database to the Domain Model (Reverse Engineering).

To start the Synchronization process, right-click on the work area or on the Database node, and choose the Synchronize option.

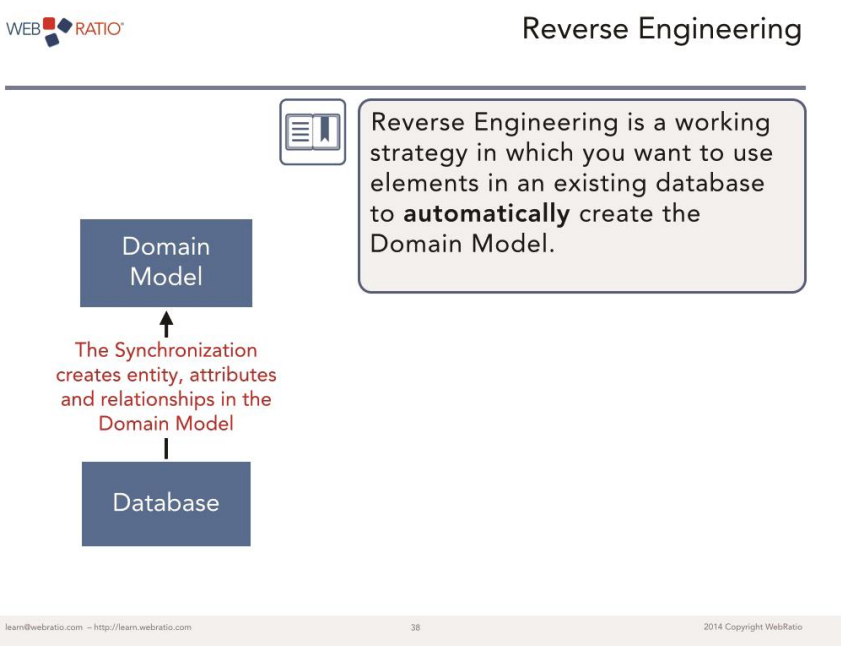
Forward Engineering

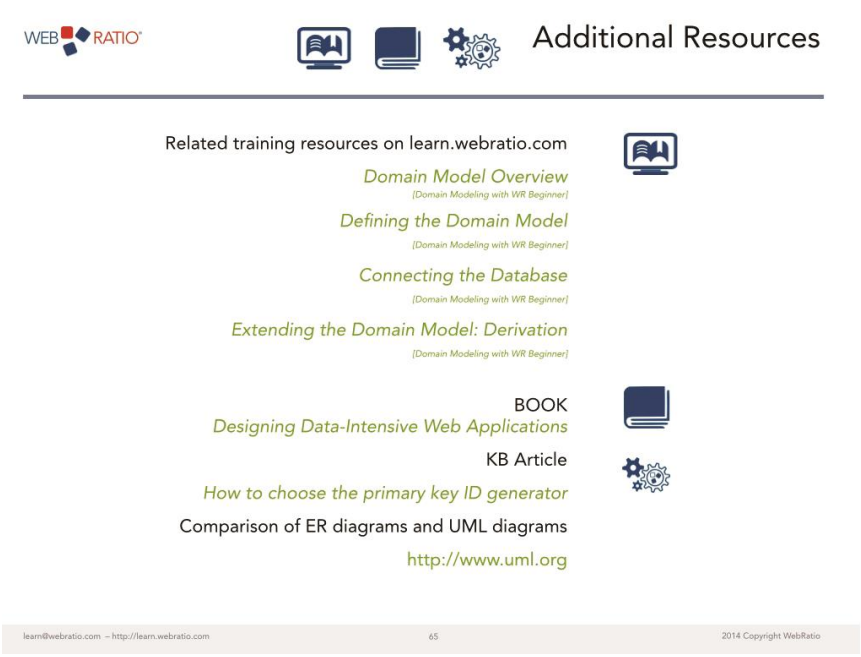
Forward Engineering is the strategy used when you are developing a Web application and the database does not exist, so it must be created from scratch. The strategy expects you to design the Domain Model and all its items: entities, attributes and relationships. When the Domain Model is completed, you can run the synchronization process that will create tables and columns on the database.
For example, the Domain Model of the CRM Web Application is new and you need to map it on a database.



Reverse Engineering

Reverse Engineering is the strategy used when you want to develop a Web application that works on an existing Database, and you want to create a structure that reflects tables and columns into Domain Model items.
This time the synchronization creates entities, attributes and relationships in the Domain Model.
For example, consider importing an already existing table with its data inside the Domain Model of the CRM Web application.





Domain Model Generated Code

Hibernate is the framework used to establish a communication between the logical data structure, which is the Domain Model, and the physical structure represented by the database.
The generated Web application uses Hibernate as the database persistence layer, so the generation process must produce a set of files that can be used by Hibernate.
The generation process uses the mapping information of the Domain Model to create a set of files that can be used by Hibernate to manage the entities.
For each Domain Model entity, the generation process produces an XML Hibernate mapping file and a Java Class that can be found inside the WEB-INF / classes / com / webratio / webapp folder of Tomcat's webapp.

Additional Resources

Online Training

- http://www.webratio.com/learn/learningobject/domain-model-overview
- http://www.webratio.com/learn/learningobject/defining-the-domain-model
- http://www.webratio.com/learn/learningobject/connecting-the-database
- http://www.webratio.com/learn/learningobject/extending-the-domain-model-derivation

Book

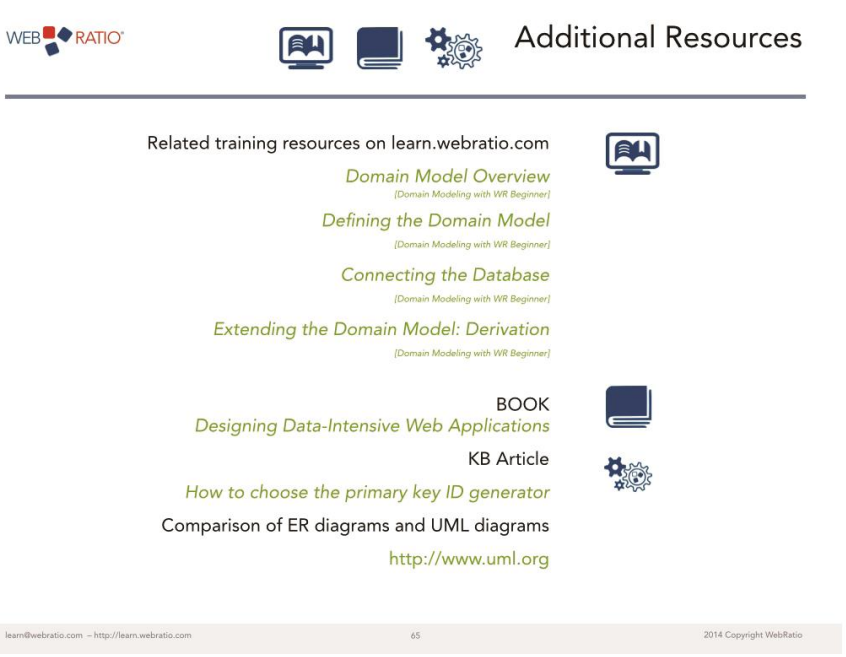
- http://www.webratio.com/learn/learningobject/designing-dataintensive-web-applications-the-morgan-kaufmann-series-in-data-management-systems

KB Article

- http://www.webratio.com/learn/learningobject/Choosing-the-primary-key-ID-Generator

Comparison of ER diagrams and UML diagrams

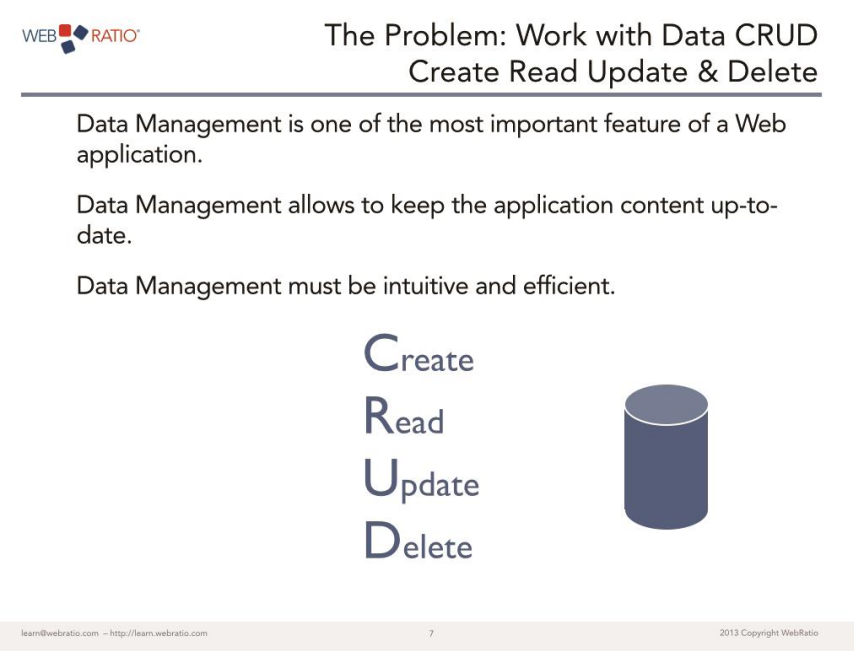
- http://www.uml.org



CRUD

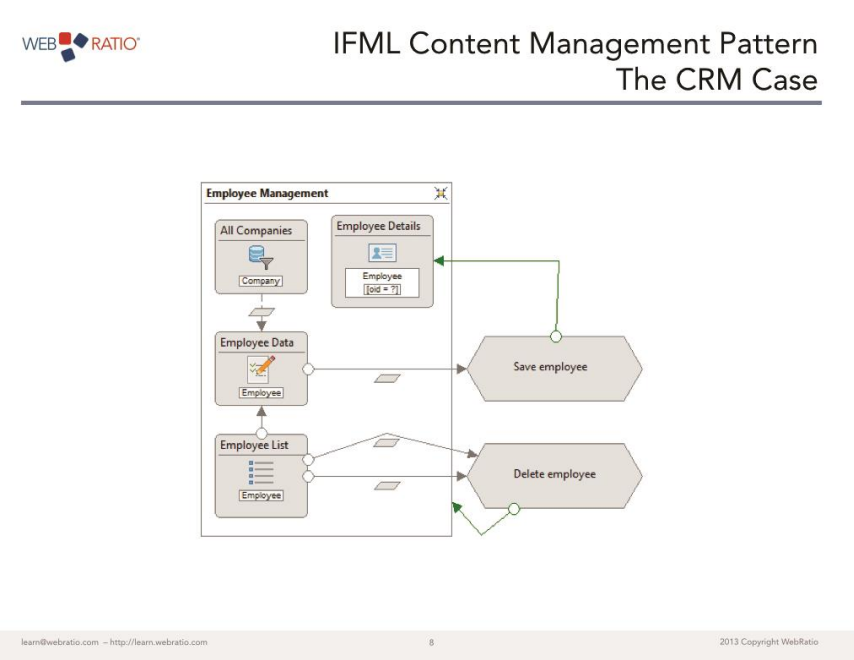
The Problem: Work with Data CRUD

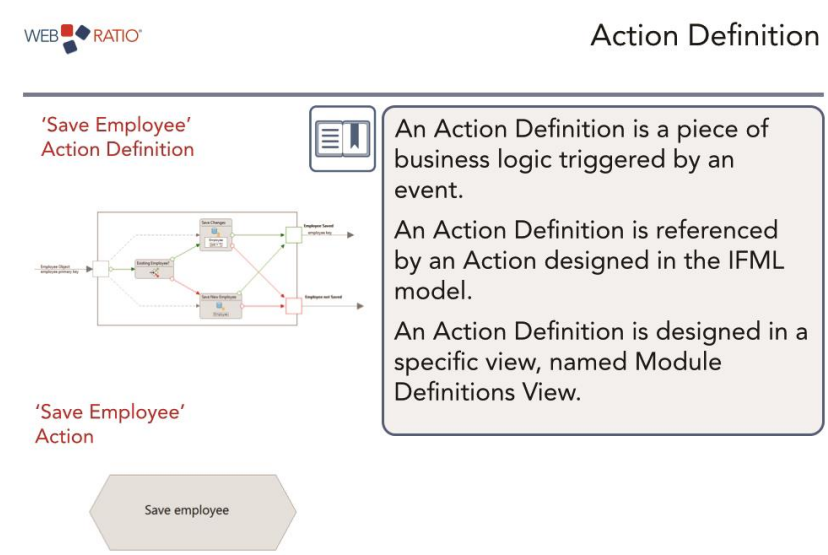
When developing a Web application, allowing the user to manage the content is an important consideration.
Data Management allows the user to handle the content of the application and, at the same time, keeps the content to be up to date.
Since Data Management involves the user, it must have an intuitive interface and it must be efficient.
Data Management in Web applications is handled with the CRUD pattern.
Each letter represents a basic operation that you can do with data; C stands for Create, R for Read, U for Update, and D for Delete.
You already know how to Read data from the Domain Model; data is read with any View Component that shows content inside a page.



The CRM Case

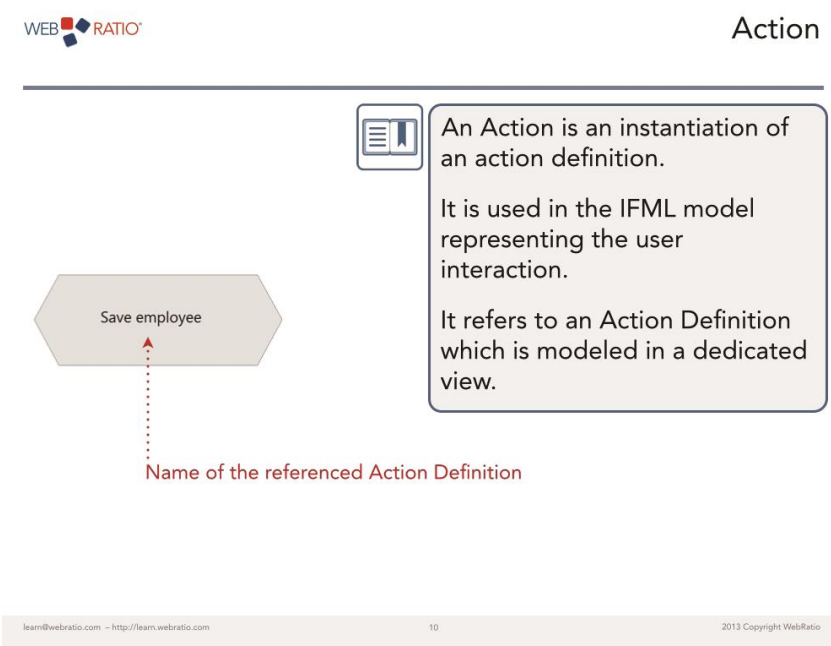
This is the basic example of the IFML model structure for managing object instances.
The page is made up of a Form Component that allows the user to manage the object instance data. There is a Selector Component that loads the Company list inside a Selection Field of the Form Component.
The Read part is achieved using a Details Component and a List Component. The Details shows the last object managed, while the List allows the user to select an object to update or delete.
The Create and Update parts are made using the 'Save object' Action, while the Delete operation is achieved with the 'Delete object' Action.





Action Definition

The Data management pattern introduces a new IFML concept, the Action Definition.
An Action Definition is a piece of business logic triggered by an event, that you can reference anywhere in the IFML model using an Action.
Action Definitions are designed in a specific model view called Module Definitions View.
Inside a Module Definitions View, Action Definitions are represented by gray rectangles.

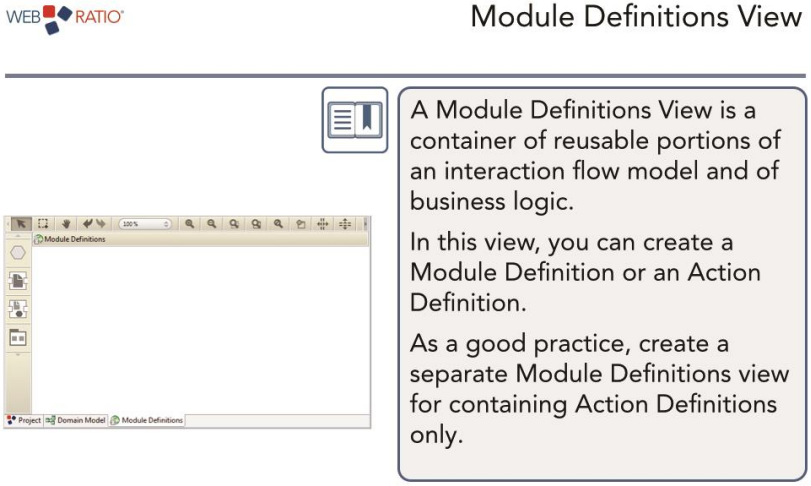


Action

Once you design an Action Definition, you can use an Action to create a reference to that definition.
Actions can be placed outside pages or inside other Actions Definitions, and are represented by gray hexagons.

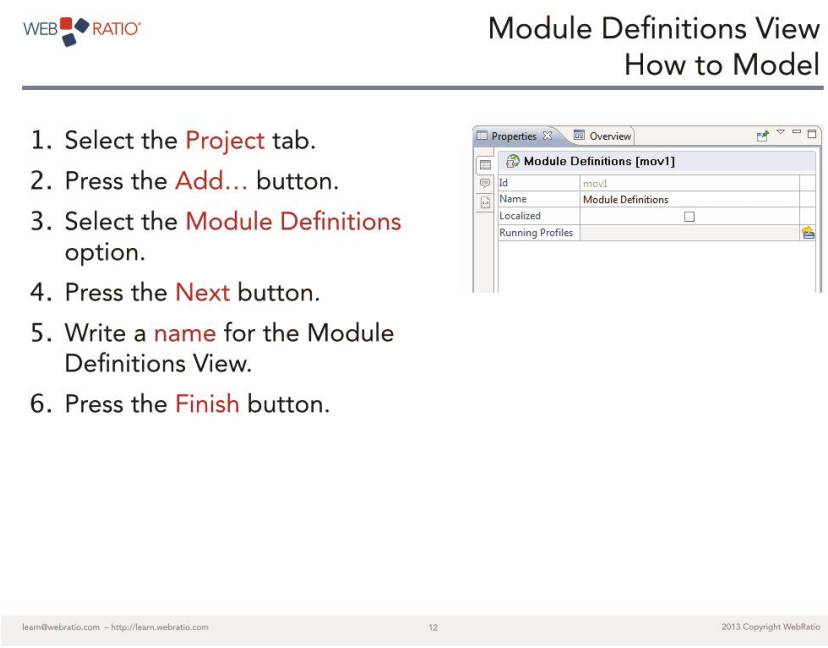
Module Definitions View

As mentioned previously, Action Definitions must be designed inside a Module Definition View.
A Module Definition View is a container of reusable portions of an interaction flow model and action definition. It can be added to the project from the Project tab.
This View is made to contain every type of reusable model so you can design both Action Definitions and Module Definitions. Since Actions Definitions and Module Definition have different meanings and roles in the Web Project structure, it is a suggested practice to split the Module Definitions View, and create one view for Actions Definitions and another one for Modules Definitions.



How to Model

Let's see how to add a Module Definition View to the Project structure.
Select the Project tab and press the 'Add…' button to open the 'Add View' dialog. Select the 'Module Definitions' option and press the 'Next' button, and write a name for the Module Definitions View.
Since we are creating the Module Definitions View for the Actions, let's call it 'Action Definitions'. Press the 'Finish' button to confirm your choice.

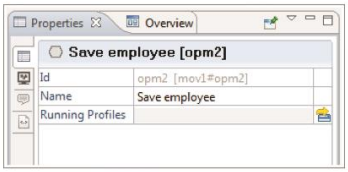


WEB RATIO

How to Create an Action Definition

To add an Action Definition:

1. Move to the Module Definitions View.
2. Right click in the Work Area.
3. Select the Add > Action Definition option.
4. Move to the Properties View.
5. Write a name for the Action Definition.
6. Double click on the Action Definition.
7. Start modeling the business logic.



learn@webratio.com – http://learn.webratio.com 13 2013 Copyright WebRatio

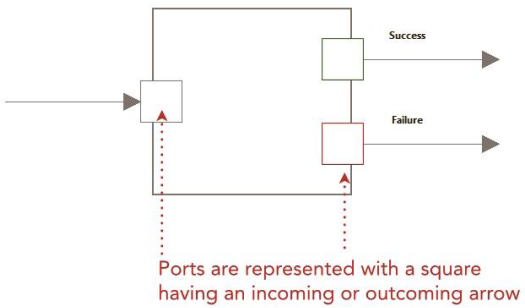
How to Create an Action Definition

To add an Action Definition to your Project you have to move to a Module Definition View, right-click in the Work Area, and select the 'Add -> Action Definition' option. You will see a gray rectangle added to the work area; select it and Move to its properties view. From here, you can set the Action Definition name.
To open the work area of Action Definition, double-click on it and then start modeling it.

WEB RATIO

Structure of an Action Definition

The structure of an Action Definition is a white box that has a unique entry point, called Input Port and multiple exit points, called OK Ports and KO Ports.



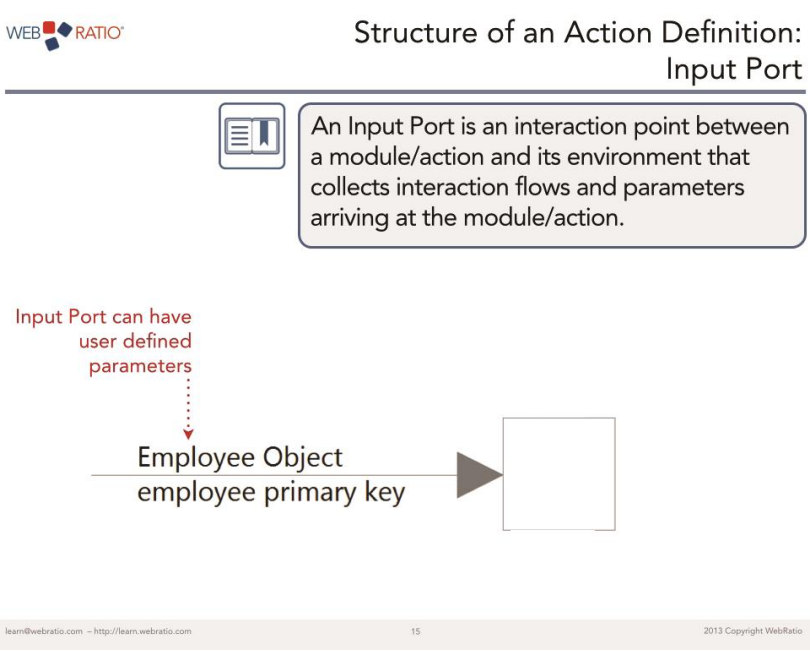
learn@webratio.com – http://learn.webratio.com 14 2013 Copyright WebRatio

Structure of an Action Definition

All Action Definitions share the same skeleton. The skeleton is the structure, represented by a white box, used as the boundary of the business logic to be defined.
The structure has a unique entry point, called the Input Port, and multiple exit points called OK Ports and KO Ports. Ports are represented by white squares that have an incoming or outgoing flow to indicate the flow direction. Usually the number of exit points starts at two: one OK Port and one KO Port. This is due to the fact that an Action Definition can end with a Success or with a Failure.\
You can easily customize the structure of an Action Definition adding or removing exit points. The number of OK and KO Port can vary depending on the desired result that you want to obtain.

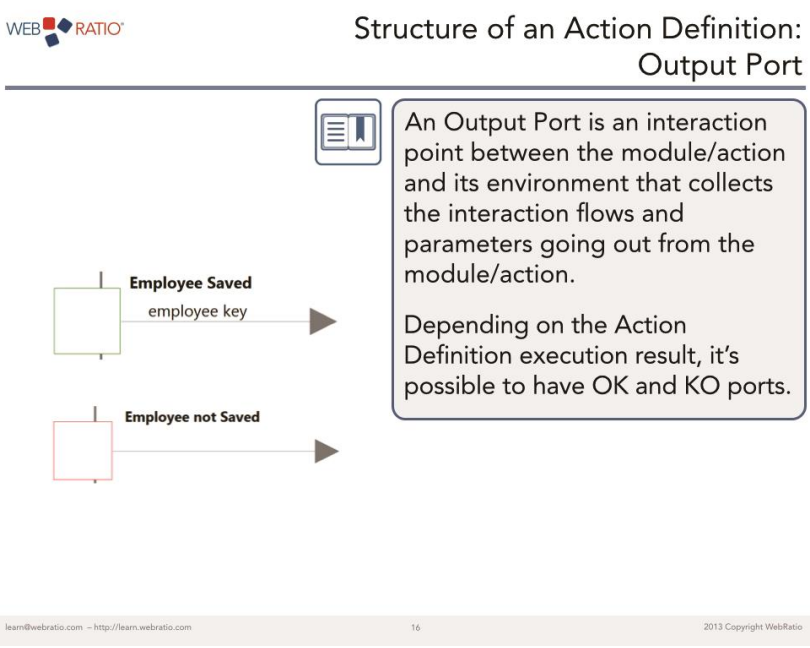
Input Port

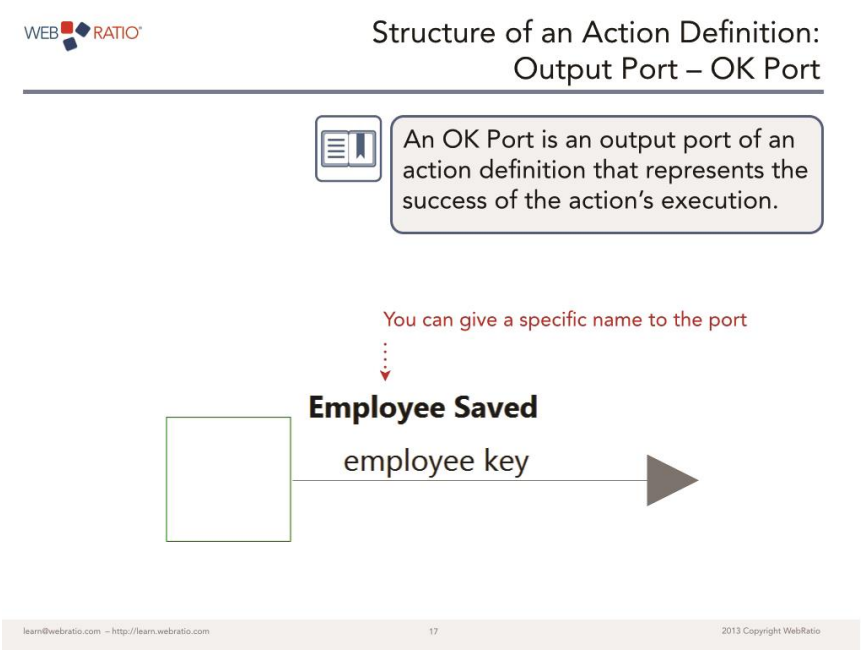
An Input Port is an interaction point, between a Module or an Action and its environment, that collects interaction flows and parameters arriving at the Module or at the Action.
Each Action Definition must have an Input Port, which is mandatory since the business logic to be executed must have a starting point.
On Input Ports, you can define parameters to be received from the Action Definition that you can use at any point in the business logic.



Output Port

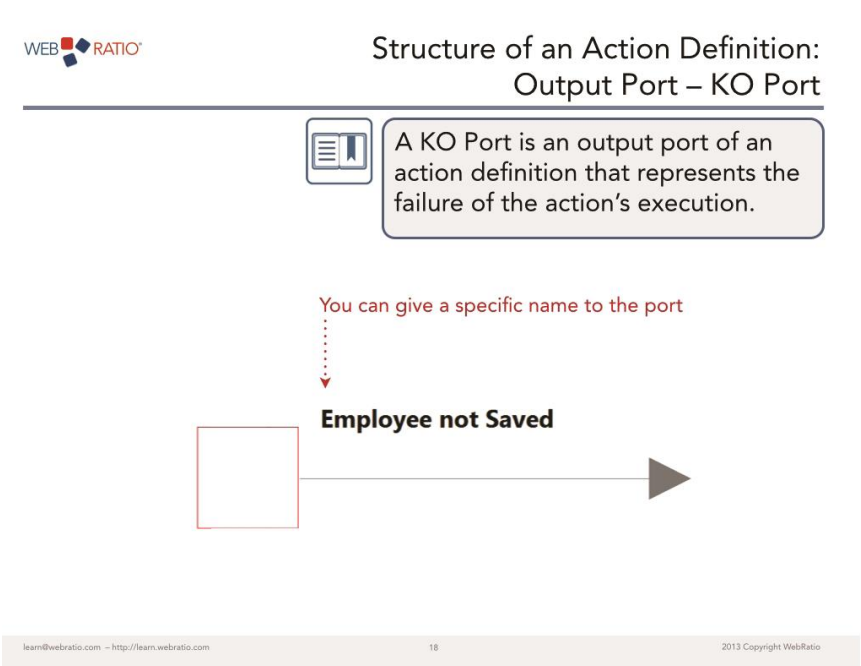
An Output port is an interaction point, between a Module or an Action and its environment, that collects the interaction flows and parameters going out from the Module or Action.
Depending on the business logic results, you may have different configurations for Output ports. For instance, you can have multiple OK and KO ports, each with a different meaning.
As for Input Ports, on Output Ports you can define parameters; parameters are returned by the Action Definition and can be used outside the Action.





Output Port . OK Port

An OK Port is an output port of an action definition that represents the success of the action's execution.
It is represented as a white square with a green border, and since it is a specific type of Output Port it can have associated OK Port Parameters. There can be many OK Ports inside an Action Definition, so you can distinguish them by giving each OK Port a different name.

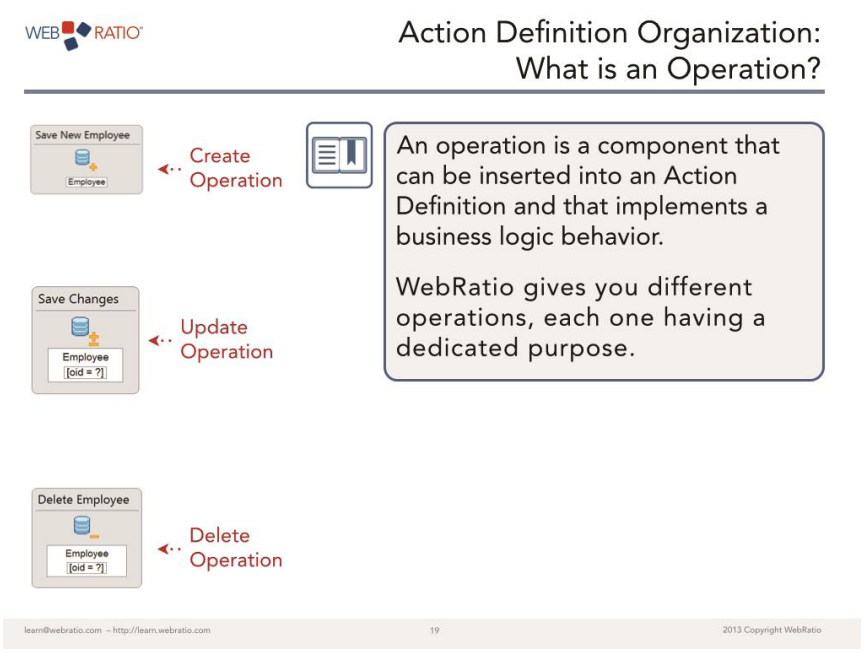


Output Port - KO Port

A KO Port is an output port of an action definition that represents the failure of the action's execution.
It is represented as a white square with a red border, and since it is a specific type of Output Port it can have associated KO Port Parameter. There can be many KO Ports inside an Action Definition, so you can distinguish them by giving each KO Port a different name.

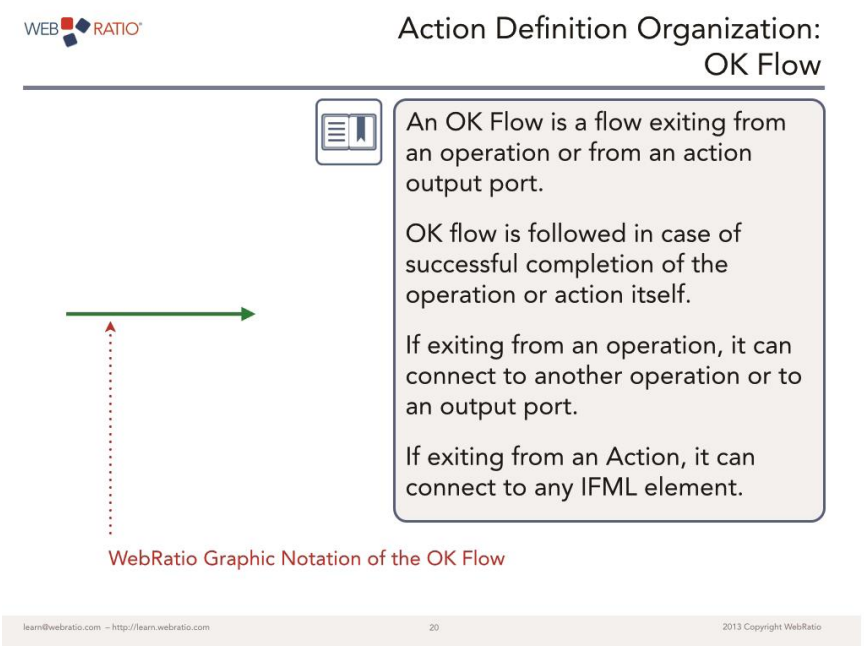
Action Definition Organization: What is an Operation?

The business logic in WebRatio is modeled inside the Action Definitions and uses a different set of components with respect to site views. This set of components is called an operation. An operation is a component that can be inserted into an Action Definition, and that implements a business logic behavior.
WebRatio offers different operations, each one having a dedicated purpose.
The three most important operations for Data Management are the Create Operation, the Update Operation and the Delete Operation.
Each operation placed inside an Action Definition needs to be activated in order to be executed. For this purpose, WebRatio has two special types of flows: the OK Flow and the KO Flow.

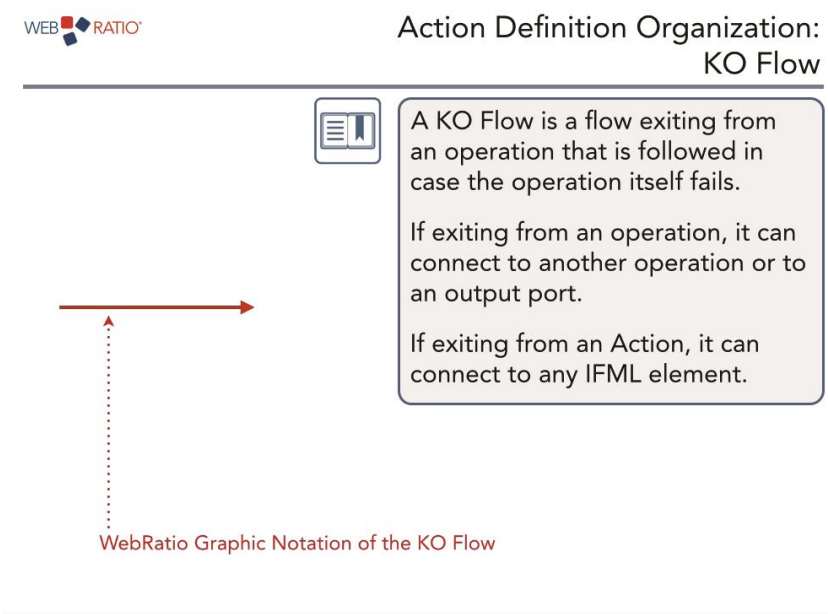


OK Flow

An OK Flow is a flow exiting from an operation or from an action output port. It is followed in case of successful completion of the operation or the action itself. If exiting from an operation, the OK Flow can connect to another operation or to an output port. If exiting from an action, it can lead to any IFML element.
An OK Flow is represented by a solid green arrow.

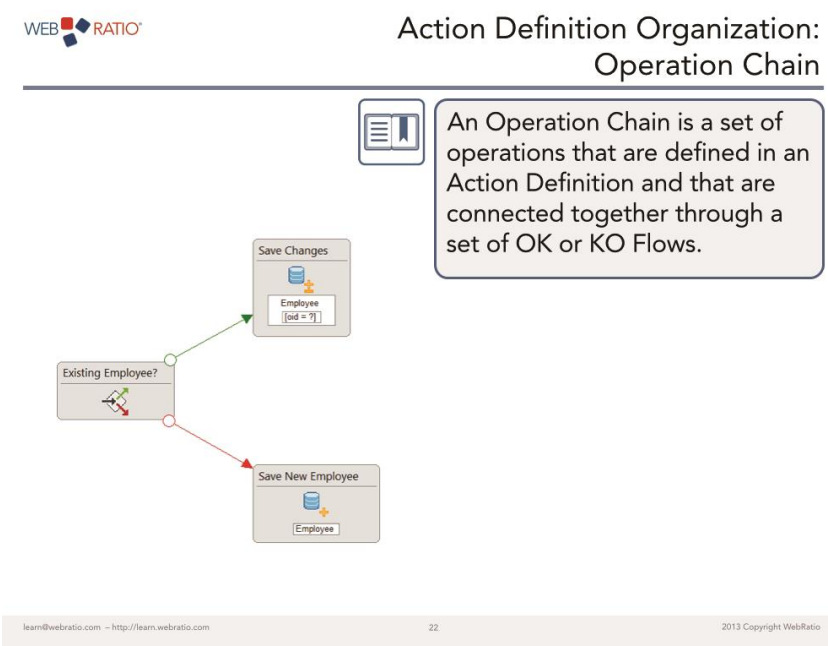


WebRatio Graphic Notation of the OK Flow



KO Flow

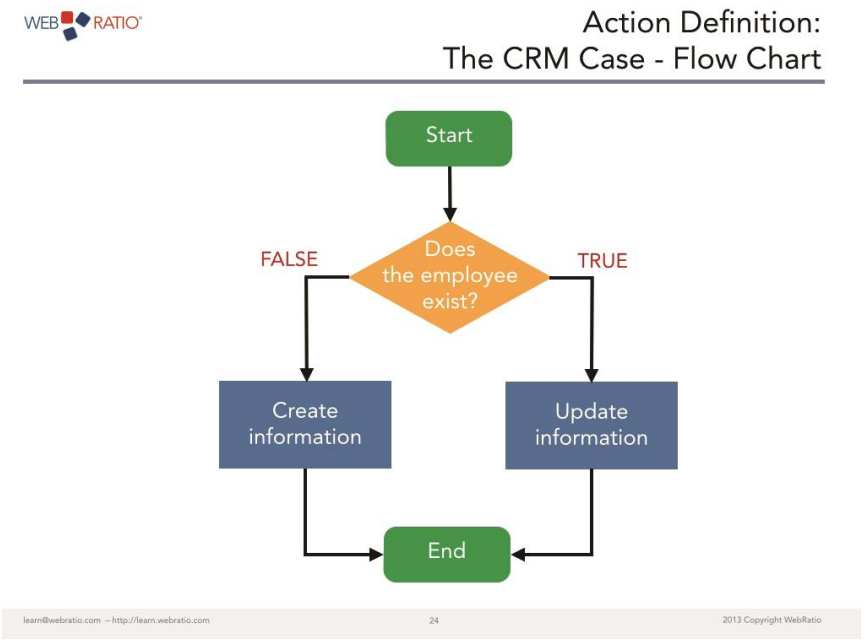
A KO Flow is a flow exiting from an operation that is followed in case the operation itself fails. If exiting from an operation, it can lead to another operation or to an output port. If exiting from an action, it can lead to any IFML element.
A KO Flow is represented by a solid red arrow.



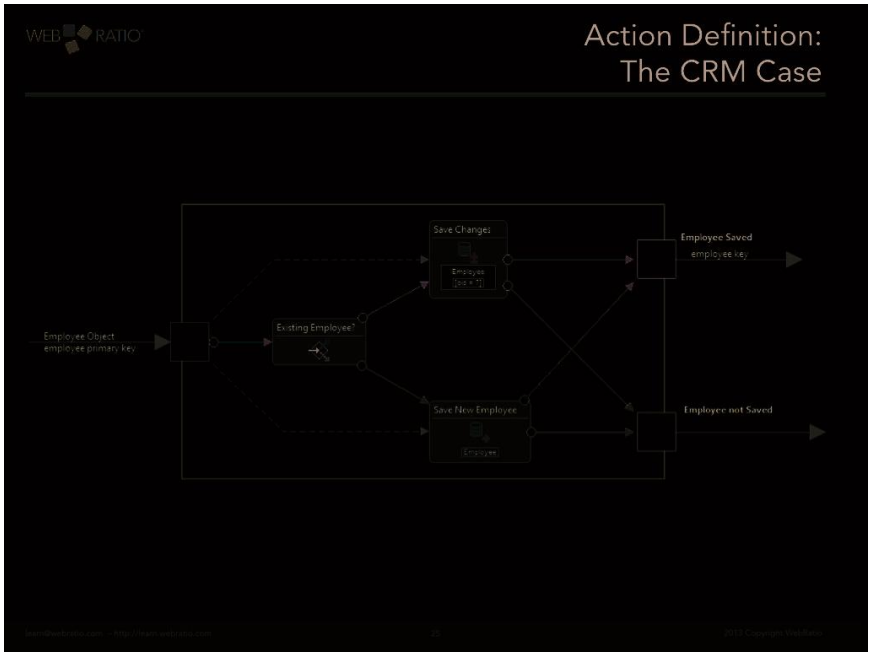
Operation Chain

Once you have Operations, OK Flows, and KO Flows, you can start to create Operation Chains.
An Operation Chain is a set of operations defined in an Action Definition and connected together through OK or KO Flows.
Usually, an Operation Chain is meant to be a sequence of operations, but according to the logic to be evaluated it may lead to a tree of operations with mutually exclusive branches.

The CRM Case - Flow Chart



This is the structure of the 'Save Employee' Action Definition; let's see each piece in detail.
The skeleton is made of one Input Port, one OK Port and one KO Port.
The Input Port has two parameters; the Employee Primary Key and the Employee Object.
The Employee Object contains all the properties of the employee shown inside the Form that can be inserted or edited.
The OK Port, renamed 'Employee Saved', handles the successful execution of the Action Definition's business logic; the KO Port, renamed 'Employee not Saved', instead, handles the failure of the Action Definition's business logic.
The business logic definition starts with the Is Not Null operation activated by an OK Flow starting from the Input Port, which brings to the Is Not Null the primary key of the employee in order to understand whether to create or update the object instance.
The Is Not Null creates two branches. The OK Flow models the case in which the primary key contains a value, so the object has to be updated; the KO Flow handles the case in which you have to create a new object.
The Update Operation is activated by the OK Flow and receives from the Input Port the Employee Object and the Employee Primary Key through a data flow. As you can see, the Update Operations needs the Primary Key to identify the object to be updated and needs the Employee Object to know the data to save.
The Update Operation, named 'Save Changes', can end successfully and so the OK Port is reached with an OK Flow passing the Primary Key of the updated object; if the Update Operation fails, the KO Port is reached with a KO Flow.
The Create Operation is activated by the KO Flow starting from the Is Not Null operation.
The Create Operation named 'Save New Employee' receives only the Employee Object from the Input Port through a Data Flow. It can end successfully following the OK Flow that leads to the OK Port returning the Primary Key of the brand new created object; otherwise, if the operation fails the KO Port is activated with a KO Flow.

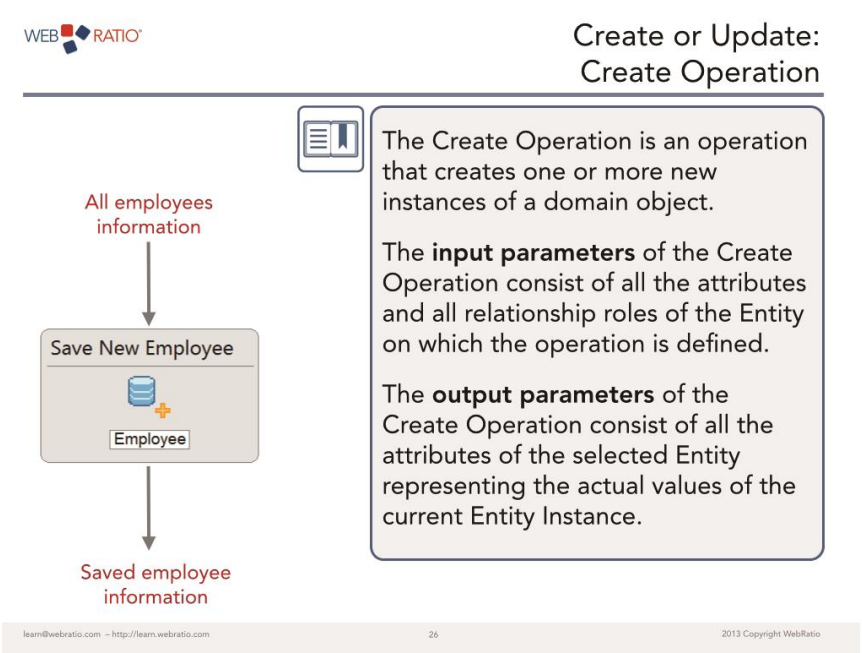


Action Definition: The CRM Case

The flow chart for the 'Save Employee' Action Definition summarizes the flow of the business logic.

As you can see, the first operation made is the check on the existence of the employee instance, and then in one situation the object is created while in the other one it is updated.

It is a suggested practice to create a flow chart before modeling actions definitions in order to organize the operation chains.



Create Operation

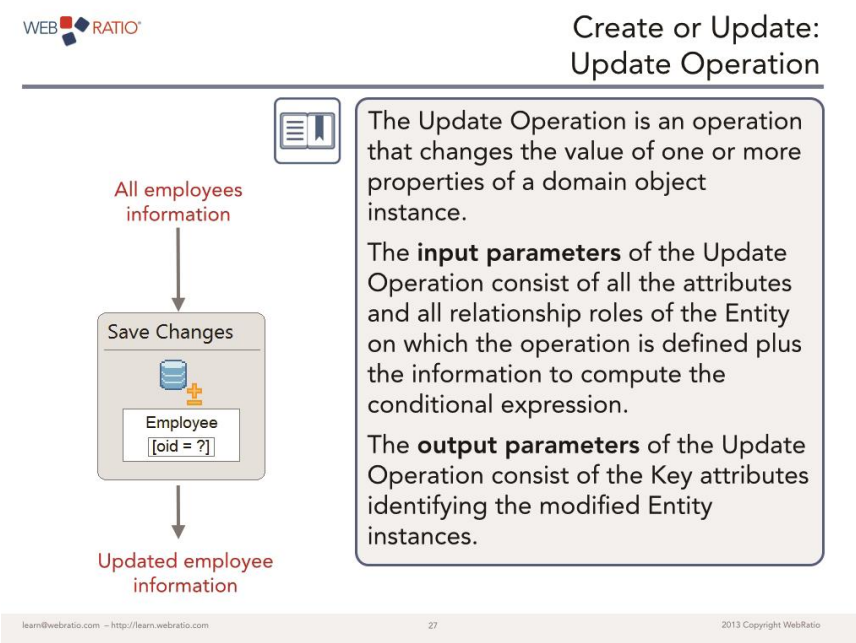
The Create Operation is an operation that creates one or more new instances of a domain object.

The input parameters of the Create Operation consist of all the attributes and relationship roles of the Entity on which the operation is defined.

The output parameters of the Create Operation consist of all attributes of the selected Entity that represent the actual values of the current Entity instance.

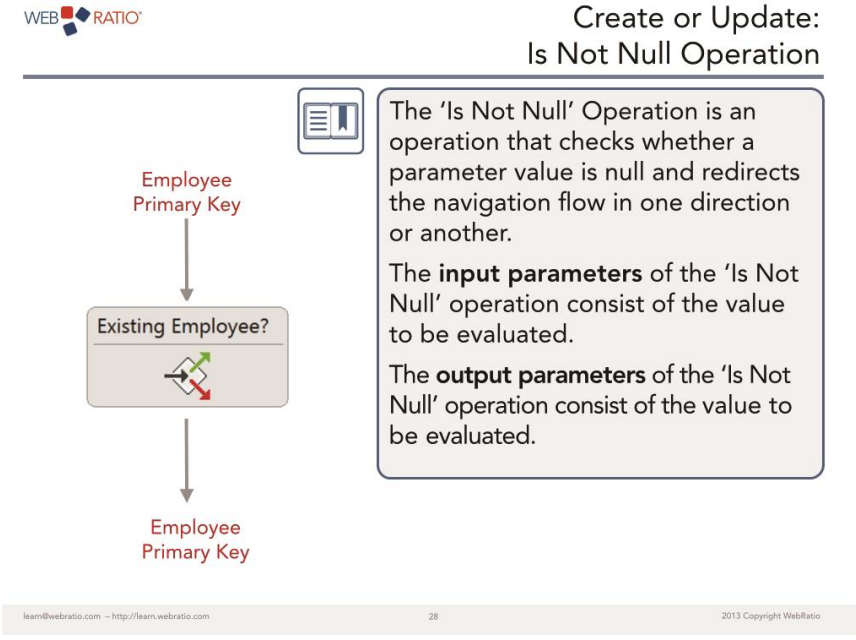
Update Operation

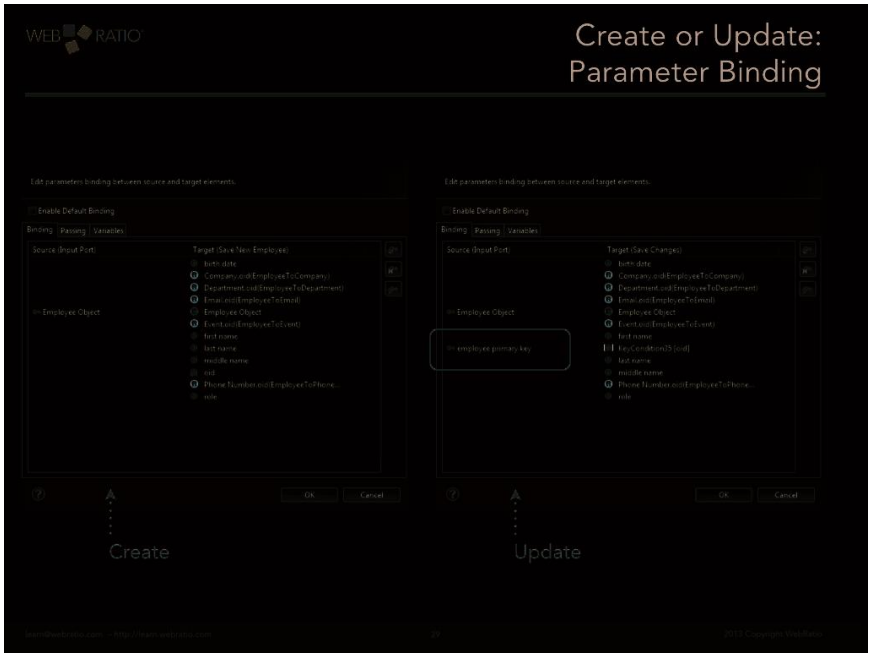
The Update Operation is an operation that changes the value of one or more properties of a domain object instance.
The input parameters of the Update Operation consist of all the attributes and relationship roles of the Entity on which the operation is defined, plus the information to compute the conditional expression.
By default the conditional expression contains a Key Condition since the operation expects, as input, the primary keys of the object to be updated, but the conditional expression can be changed as you desire adding conditions to the operation.
The output parameters of the Update Operation consist of the Key attributes identifying the modified Entity instances.



Is Not Null Operation

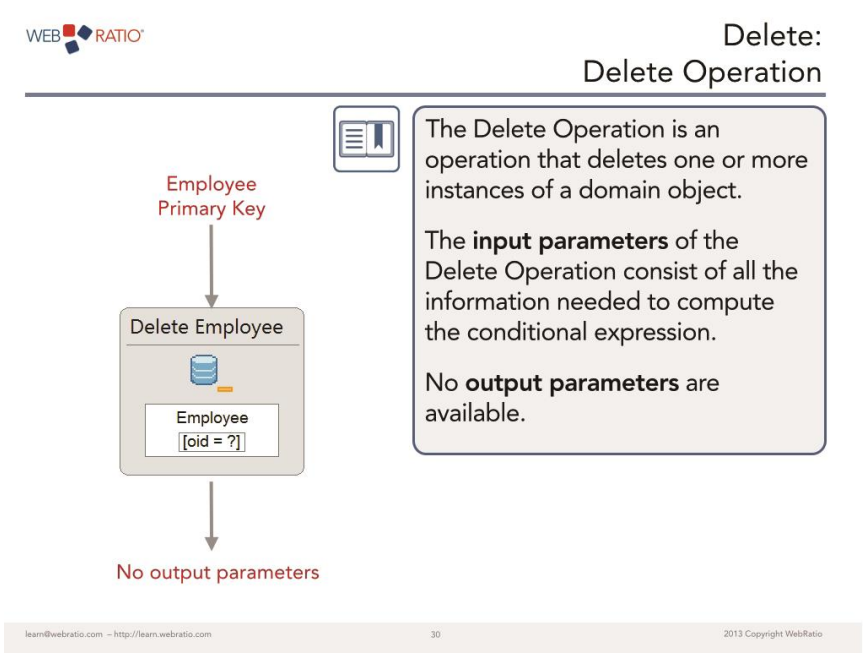
The Is Not Null Operation is a control flow operation that checks whether a parameter value is null and redirects the navigation flow in one direction or another.
The input parameters of the 'Is Not Null' operation consist of the value to be evaluated.
The output parameters of the 'Is Not Null' operation consist of the same value used for the evaluation.
In this case the Is Not Null Operation is used to check whether the incoming value for the Primary Key of the Employee is null or not; if the value is null it means that the Employee does not exist, so it will be created, otherwise it will be updated.





Parameter Binding

This slide shows the parameter binding configuration on the Data Flows that start from the Input Port and lead to the Create or Update operation.
The binding configuration for Create, as you have already seen in the CRM Case overview, is made up only of the Employee Object, while the Update also includes the Employee Primary Key.

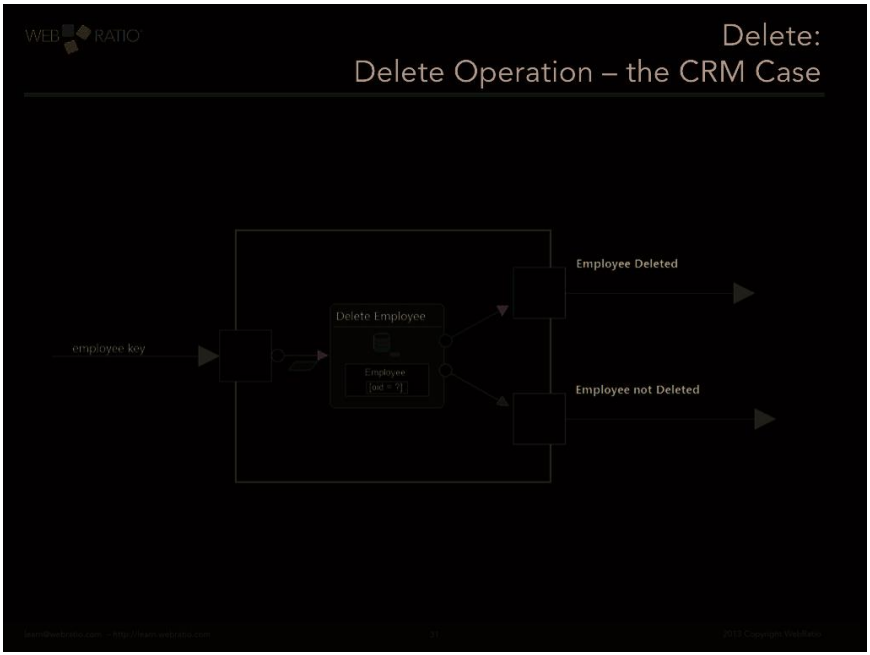


Delete Operation

The Delete Operation is an operation that deletes one or more instances of a domain model object.
The input parameters of the Delete Operation consist of all the information needed to compute the conditional expression. By default, the Delete Operation has a predefined conditional expression that is made up of a Key Condition, since the operation needs to identify the instance to be deleted.
As for the Update Operation, the conditional expression can be changed by adding or removing conditions.
From the Delete Operation there are no available output parameters.

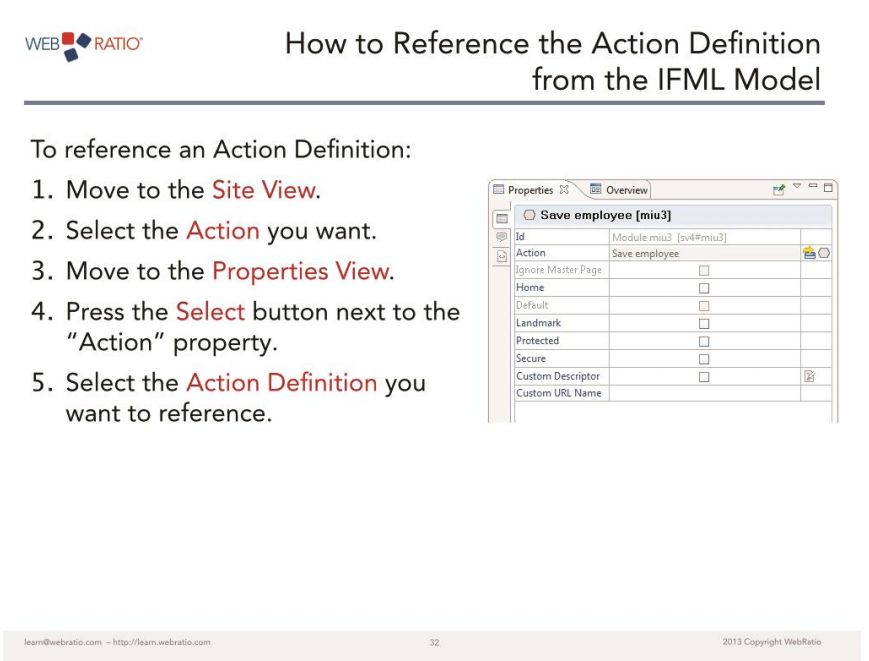
Delete Operation - The CRM Case

This is the structure for the Delete Employee Action Definition. The skeleton is made up of one Input Port, one OK Port and one KO Port. The Input Port receives one parameter from the external environment, which is the primary key of the employee to be deleted.
It directly activates the Delete Operation that can end successfully or not; you have an OK Flow that leads to an OK Port in case of success (the OK Port is renamed 'Employee Deleted'). Otherwise you have a KO Flow that leads to the KO Port in case of failure.



How to Reference the Action Definition from the IFML Model

Once you design an Action Definition, you can use it anywhere in your model through the Action concept.
Let's see how to reference an Action Definition from a Site View.
Move to a Site View of your project and select the Action from the container's section of the toolbar and place it into the work area.
Select it and move to its properties view.
To refer an existing action press the 'Select' button next to the 'Action' property. A dialog called 'Action Definition Selection' is shown and, from here, you can pick the desired Action Definition and then press 'OK' to confirm your choice.



Additional Resources

Online Training

- http://www.webratio.com/learn/learningobject/master-details
- http://www.webratio.com/learn/learningobject/action-definitions-operations
- http://www.webratio.com/learn/learningobject/looping-branching
- http://www.webratio.com/learn/learningobject/data-management-ifml-model
- http://www.webratio.com/learn/learningobject/data-management-action-definition

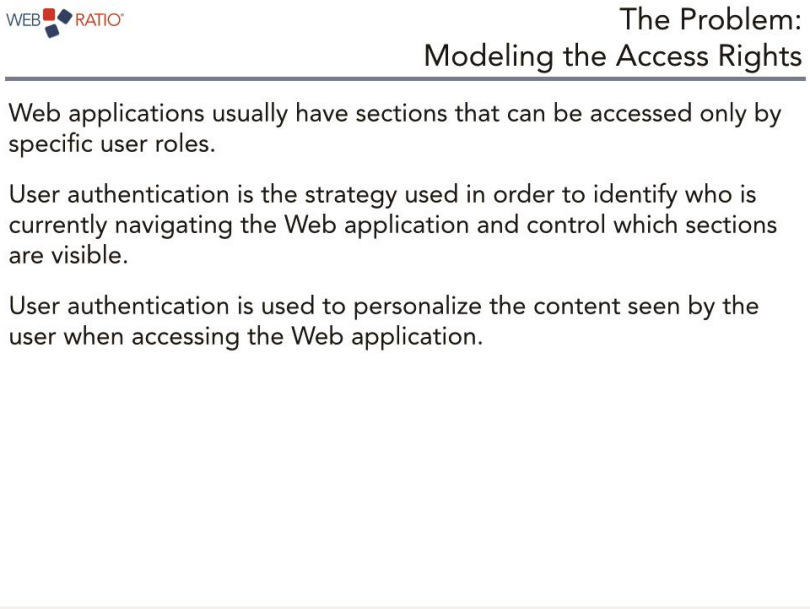
KB Articles

- http://www.webratio.com/learn/learningobject/how-to-model-a-simple-validation
- http://www.webratio.com/learn/learningobject/how-to-model-user-and-group-management

Access Rights

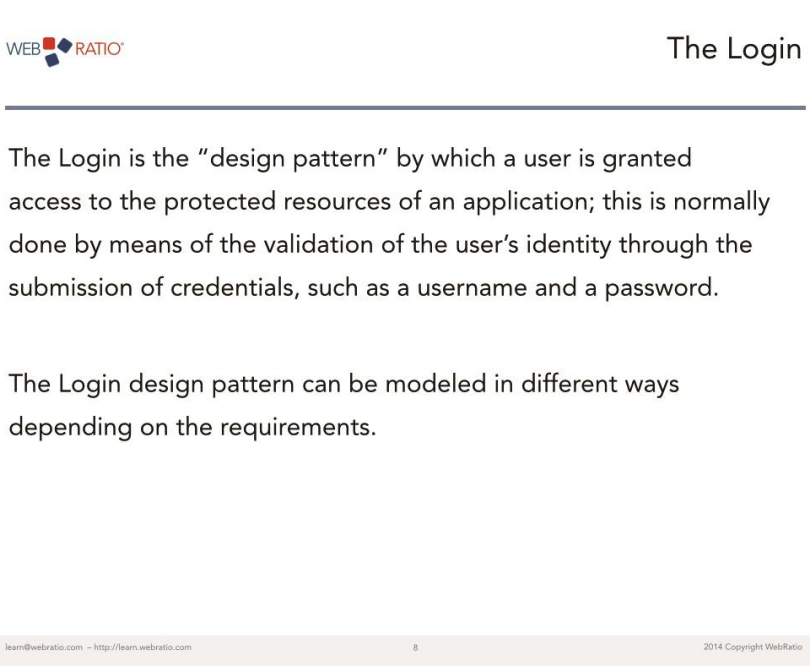
The Problem: Modeling the Access Rights

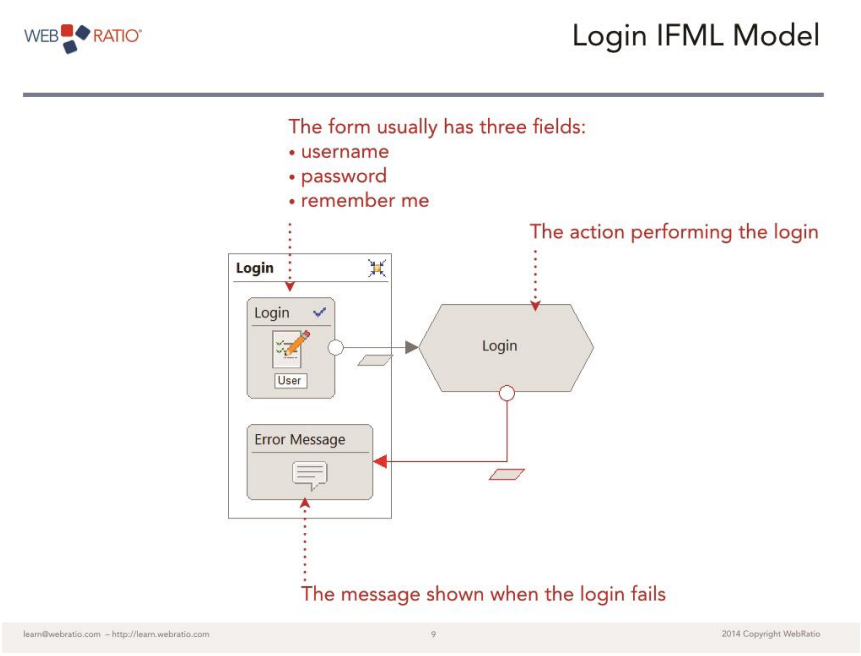
Web applications have users and usually there are sections that can be accessed only by users with specific roles.
To identify users, the mechanism used is called 'User authentication'. It identifies who is currently navigating the Web application and it can be used to personalize the content seen by the user when accessing the Web application.



The Login

The Login is the 'design pattern' used to grant a user access to protected resources and sections of an application; Login is usually done by means of the validation of the user's identity through the submission of credentials, such as username and password.
The Login design pattern is very flexible, and depending on the requirements, can be modeled in different ways.



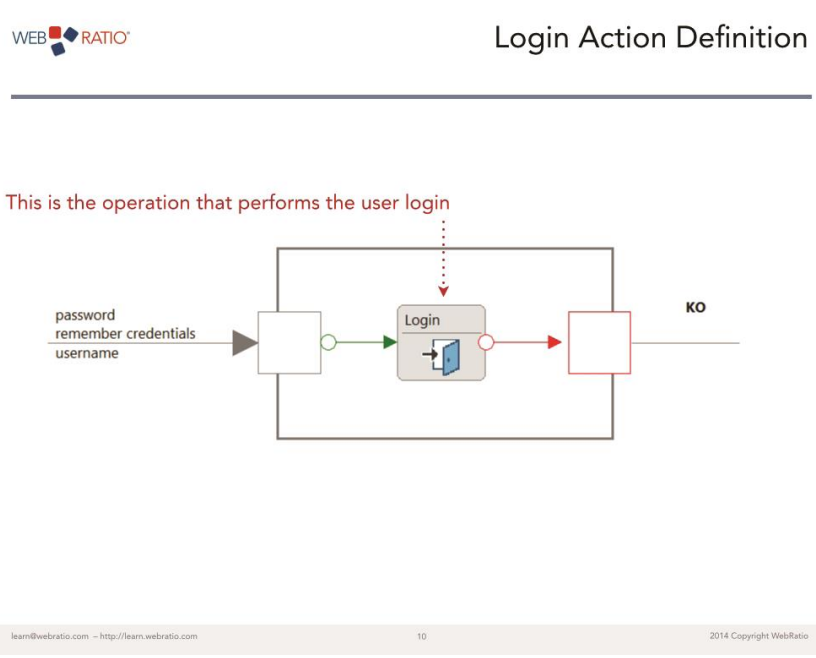


Login IFML Model

This slide shows the basic IFML Login pattern.
The model is composed of a page containing a form and an action that performs the required business logic.
The main View Component of the Page is a Form that usually has three fields to be filled in by the user; username, password, and remember me. The fields 'username' and 'password' are the user credentials, while the 'remember me' field, which is not mandatory, is used to preserve the user session.
Inside the same Page you also always have a Message View Component, used to show a feedback message if the Login fails.
Outside the page you have an Action that references the 'Login' Action Definition; it receives the user credentials from the Form and processes the login request. If the Login is successful, the user is redirected to the assigned resource; otherwise, the user is redirected to the login page where an error message is shown.

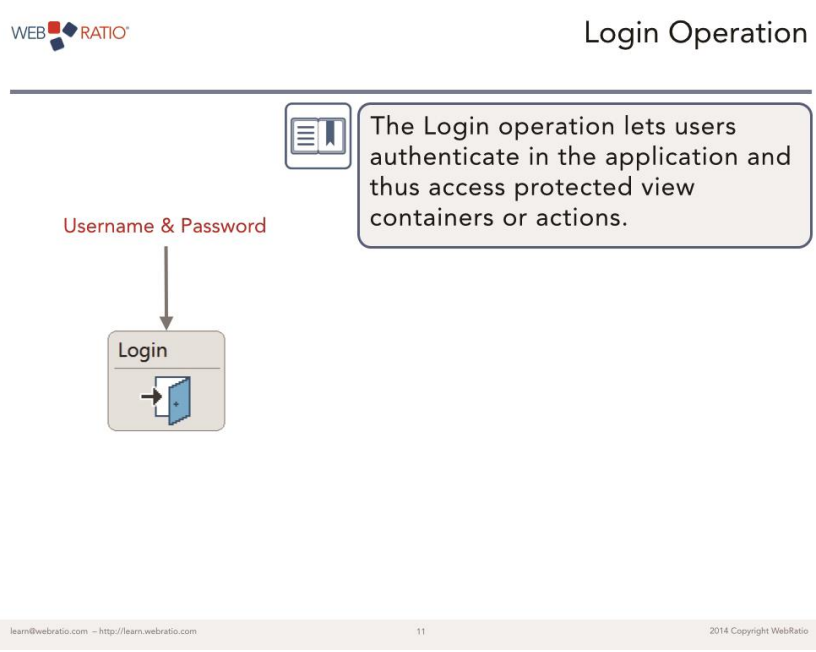
Login Action Definition

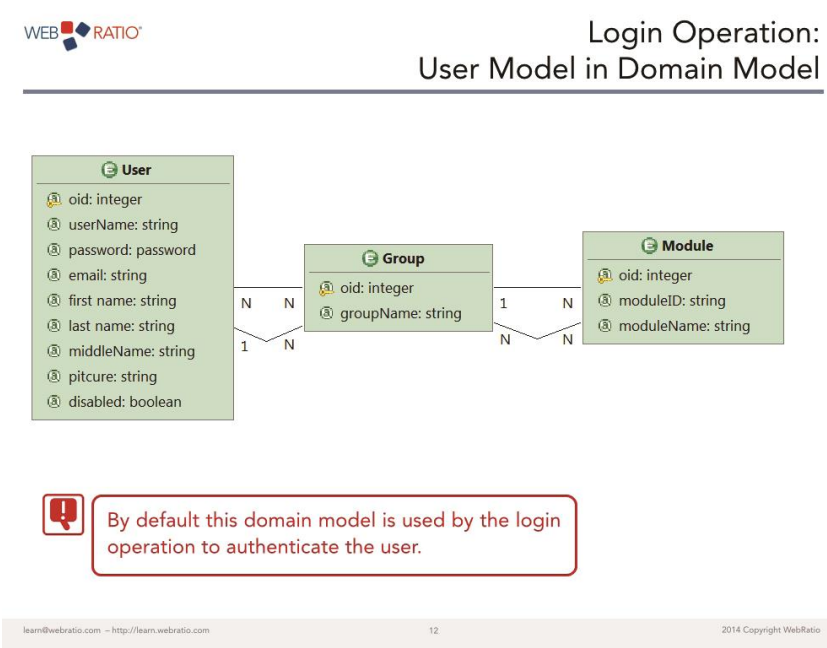
The skeleton of the Login Action Definition includes an Input Port, a KO Port, and the Login Operation that performs the user authentication.
The Input Port receives the user credentials for the login, and for this purpose has three Input Port Parameters defined, one for each field of the Form.
The Login Operation receives the parameters and is activated with an OK Flow.
If the Login succeeds, the redirect is done implicitly so there is no need to model an outgoing OK Flow. A KO Flow leads to the KO Port in case of failure.



Login Operation

One of the default operations available in WebRatio is the Login Operation, which you can find inside the toolbar section of the session components.
This operation lets users authenticate in the application and thus access protected sections; it considers the available input parameters and uses a specific portion of the Domain Model, called 'User Model', as the source of data to validate the access request.





User Model in Domain Model

The User Model is the portion of the Domain Model considered by the Login Operation to check the user credentials.

It is composed of the User, Group and Module entities and the relationships between them.

These three entities are automatically added to the Domain Model when you create a new Web Project and, since they are fundamental for user authentication, they cannot be deleted.

The User and Group entities are connected with a one-to-many (1-N) and a many-to-many (N-N) relationship.

The one-to-many, also called 'default group' is used to identify the default group for each user, while the other relationship identifies all the other groups to which the user belongs.

The same structure of relationships can be found between the Group and Module entities; the one-to-many (1-N) identifies the default module for each group, while the many-to-many (N-N) contains all the modules related to a user.

Authentication Algorithm Flow Chart

The structure of the Domain Model is browsed by the Login Operation with a special algorithm named Authentication Algorithm. The logic of the algorithm is shown in the slide and let's now see how it works.

The algorithm starts searching for a User with the provided combination of username and password; if the user is not found, the algorithm ends by throwing an error.

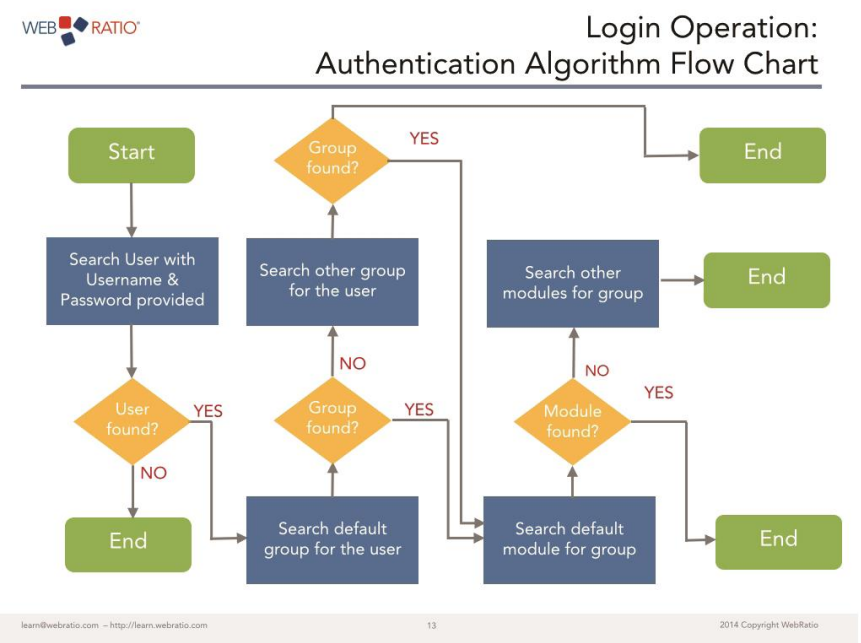
If the user is found, the algorithm starts to search the default group for the user, browsing the one-to-many (1-N) relationships between the user and group entity; if the default group exists, the algorithm looks for the associated default module. If the default group is not found, then the algorithm searches for other groups associated with the current user with many-to-many relationships (N-N).

If there are no groups at all associated with the current user, the algorithm ends by taking the user back to the login page. But if a group is found, the algorithm starts to search for the associated modules.

The iteration process for modules is similar to the one for groups; first it looks for a default module and, if it does not exist, it searches for modules in the many-to-many relationship with groups.

If there are no modules at all associated with the current user, the algorithm ends by taking the user to the Login page.

If the user exists and is associated both to a group and a module, the Login operation is able to build the redirect link to the correct protected resource.



WEB RATIO

Implicit Login

When the user tries to access directly a protected resource without having the access rights, WebRatio automatically shows a login page, which is not modeled.

learn@webratio.com – http://learn.webratio.com 14 2014 Copyright WebRatio

Implicit Login

In WebRatio, there is an implicit mechanism that prevents the access of external users to protected resources. This mechanism is called 'Implicit Login' and it consists of showing a default login page, not modeled, to the user. This page is included in the Style Project applied to the Web application.
At this point, the user is asked to insert credentials. If the user has rights to access to that protected page, then the Login mechanism redirects the user to the desired page; otherwise the user is bounced back to the implicit login page.

WEB RATIO

The Logout

The Logout is the design pattern by which a user terminates his session on a Web application.

After the Logout is performed the user must log in another time to start a new session.

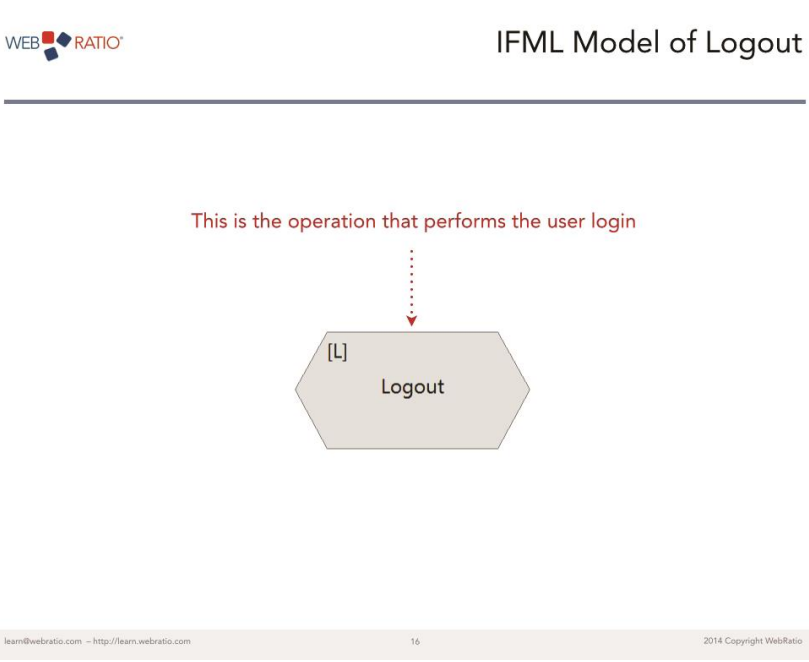
learn@webratio.com – http://learn.webratio.com 15 2014 Copyright WebRatio

The Logout

The Logout is the design pattern by which a user terminates a session on a Web application. After the Logout is performed, the user must log in again to start a new session.

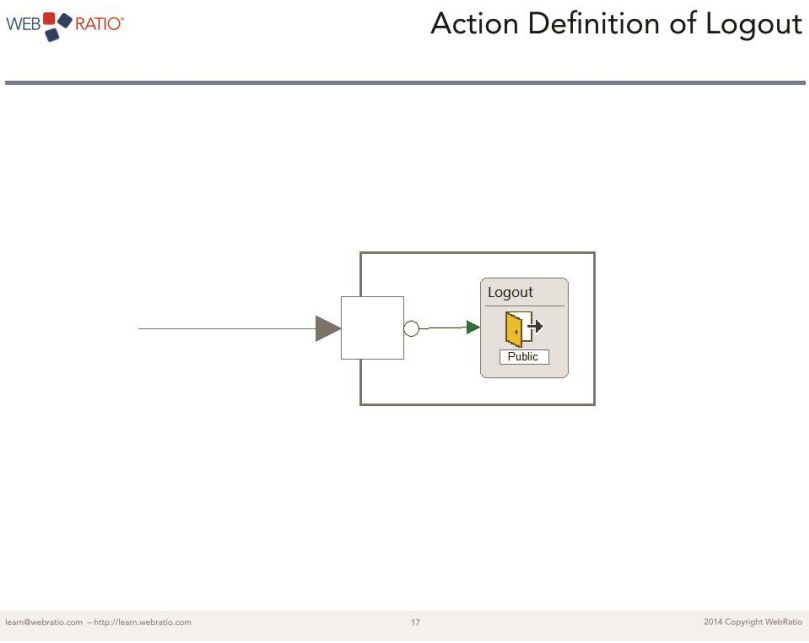
IFML Model of Logout

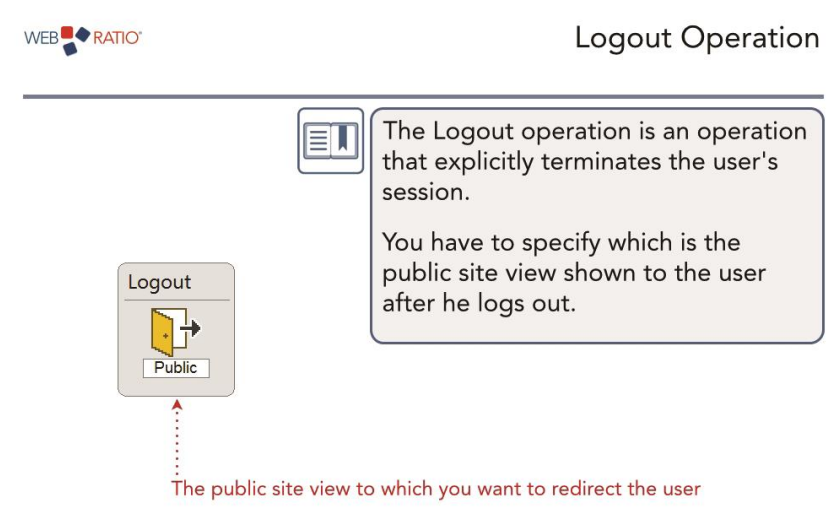
Since the Logout does not require any user input and only destroys the current session, the IFML model is only composed of an Action that references the 'Logout' Action Definition.
The Action must be activated by the user, and the easiest way is to put the 'Landmark' flag on it, so that the Logout function will be available from any point of the site view through the main menu.



Action Definition of Logout

The Logout Action Definition skeleton is made only of an Input Port with no Output Ports.
The Input Port with an OK Flow directly activates the operation used to perform the Logout, which is named Logout Operation.





Logout Operation

The Logout Operation explicitly terminates the user's session. Like the Login Operation, it is a Session Operation that can be found in the dedicated toolbar section.
To use the Logout Operation you need to specify which public site view to redirect the user to after the logout. If you have only one public site view it is automatically set on the operation, otherwise the selection must be manually done from the operation's properties view.

Context Paramenters

Once you have the user authentication, another important aspect to cover is the storage of common data along with the session. In WebRatio this is done through the usage of 'Context Parameters'.

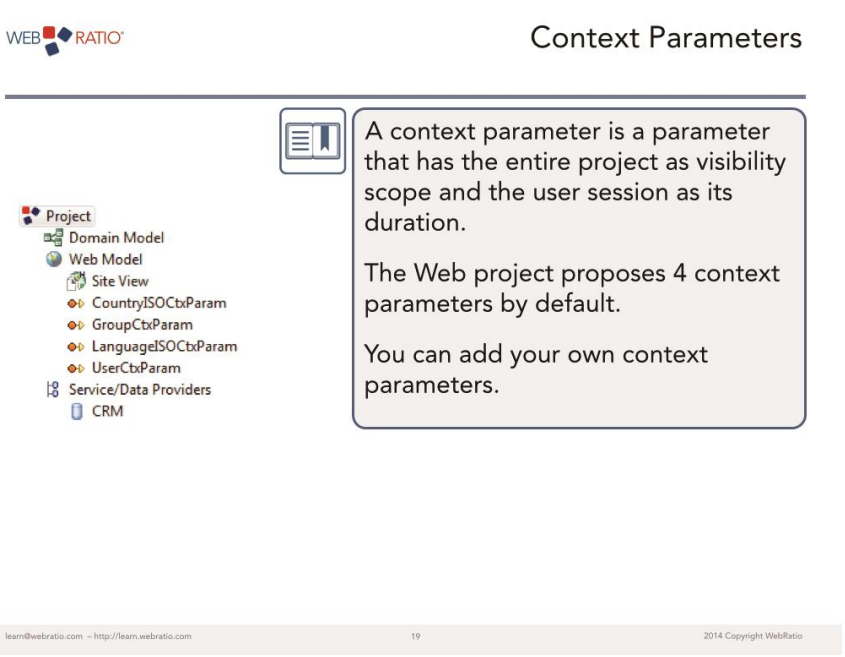
A Context Parameter is a parameter that has the entire project as visibility scope and the user session as its duration.

Every time that you create a new Web Project automatically WebRatio adds four context parameters. The four default context parameters are: UserCtxParam, GroupCtxParam, LanguageISOCtxParam and CountryISOCtxParam.

The first two are related to the user session and respectively store the primary key of the user and the primary key of its group. The value of these parameters is set as soon as the user logs into the application.

The other two parameters are related to the localization of the application for the current user and will be explained later.

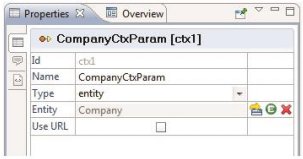
If you need extra Context Parameters you can add your own.



WEB RATIO

Context Parameters: How to Model

1. Select the Project tab.
2. Move to the 'Outline View'.
3. Right-click on the project node and select Add > Context Parameter.
4. Move to the Properties View.
5. Give a name to the context parameter.
6. Select the type.
7. If you choose "entity" type then press the 'Select' button next to the Entity property and choose the desired entity.



How to Model

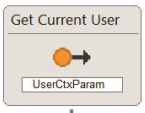
Let's see how to add a new Context Parameter. Select the Project tab of your Web Project and move to its 'Outline View'. From the project outline view select the Project node and right-click on it, and select the 'Add -> Context Parameter' option to create a new one.
Move to its properties view to customize it and give a name to the 'Context Parameter'. Set the type of parameter; if your selection is 'entity' then press the 'Select' button next to the 'Entity' property and choose the desired entity.

WEB RATIO

Get Component

A component that can be used as view component or operation.

It retrieves the values of one or more context parameters and produces them in output (as output parameters of the component).



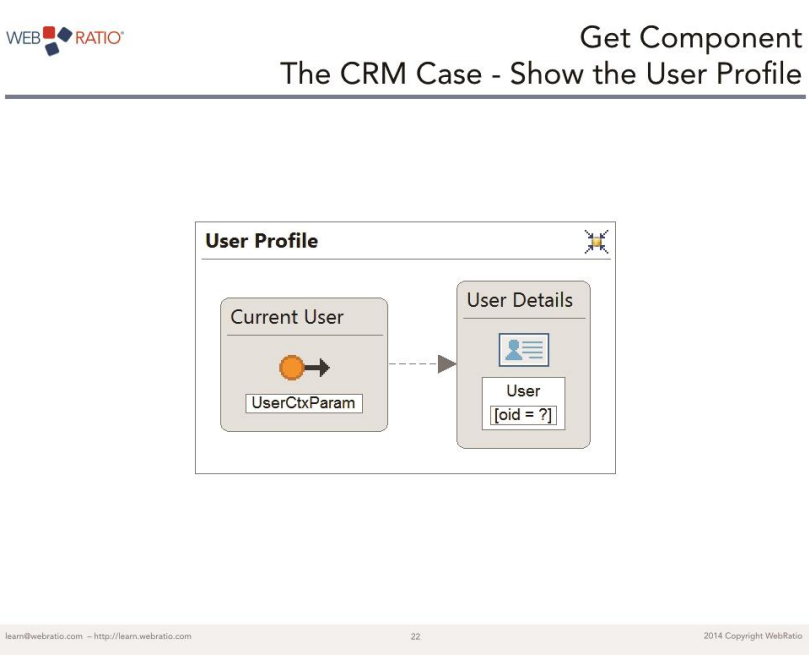
Value contained in the context parameters listed

Get Component

There are three session components that can be used to manage and retrieve context parameters, they are the Get, Set and Reset components.
The Get is a component that can be used as view component or operation, so it works both inside a page and inside an action definition.
It retrieves the values of one or more context parameters and produces them in output as output parameters of the component.
It is a Session Component, so it is inside the Session Components session of the toolbar.

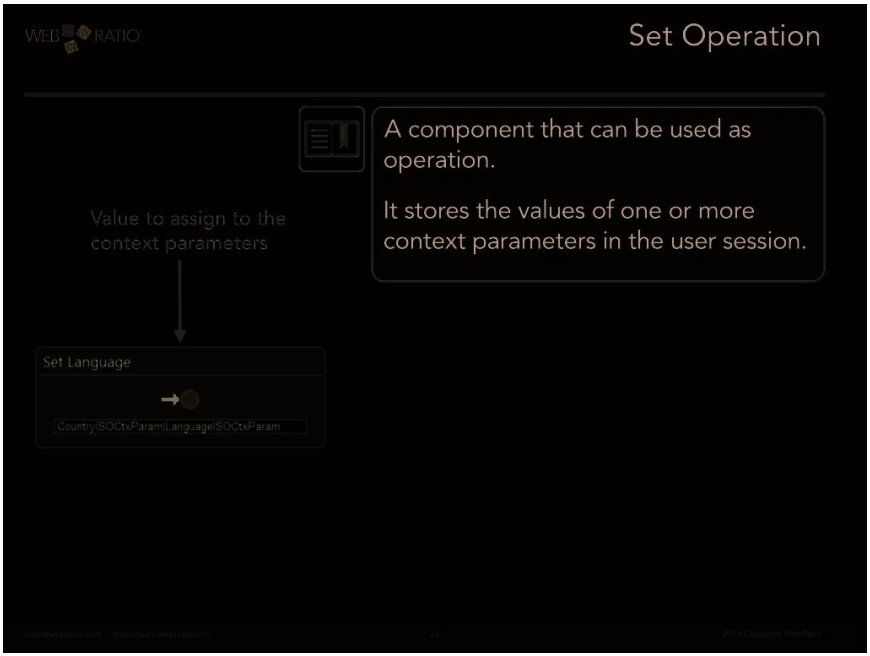
The CRM Case - Show the User Profile

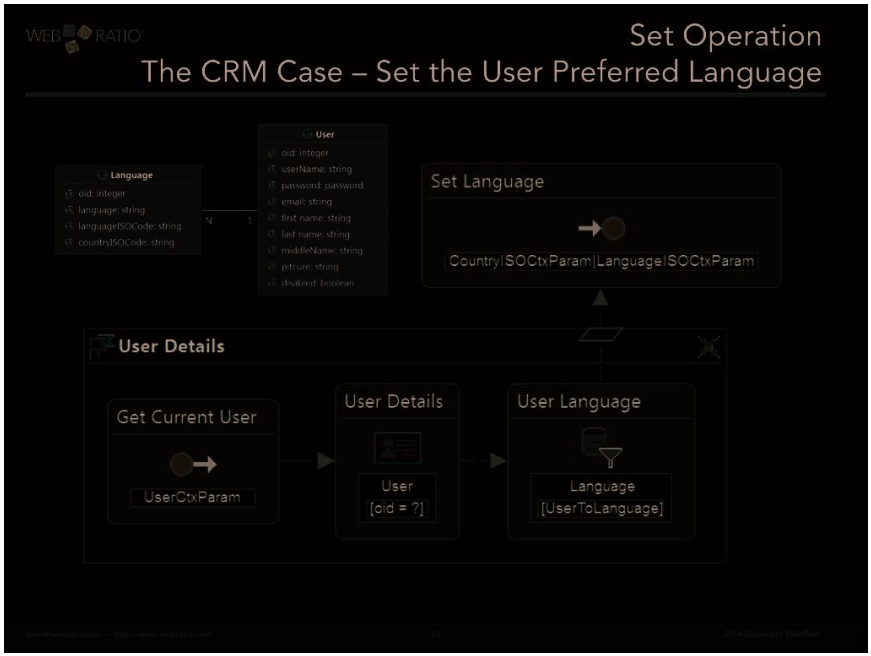
This is a basic example of usage of the Get Component, a page showing the user profile information.
Here, the Get Component is used to retrieve from the session the value of the currently logged-in user from the UserCtxParam parameter. The primary key of the user is then passed as primary key to a Details Component based on the User entity.



Set Component

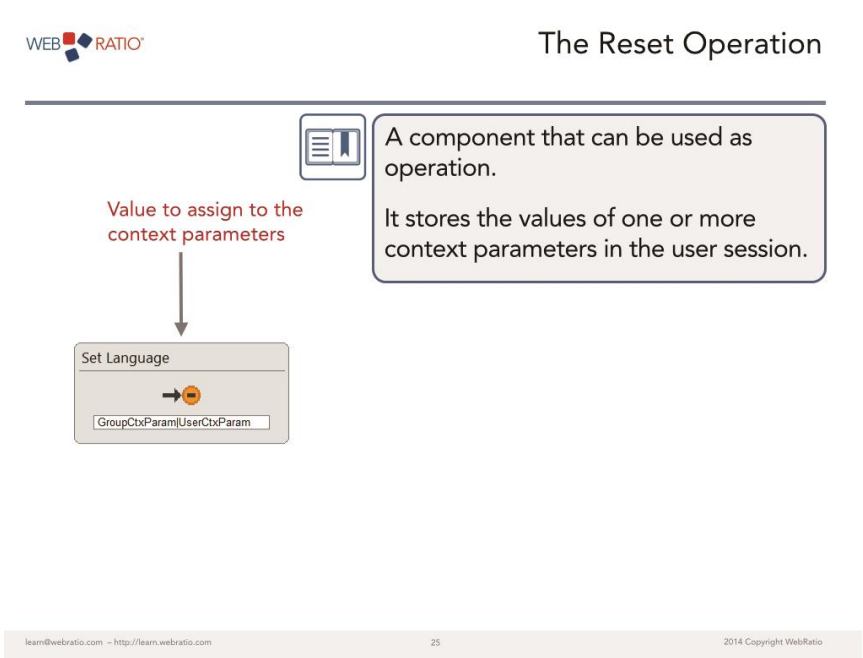
The Set Operation is an operation used to store the values of one or more context parameters in the user session. The value for each parameter is passed as input parameter to the operation.
It is a Session Component, so it is inside the Session Components session of the toolbar.





The CRM Case - Set the User Preferred Language

This is a basic usage example of the Set Component, a page to set the user default language as the predefined language for the user session.
As you can see, the User entity is connected with the Language entity with a one-to-many (1-N) relationship. The details page has a selector that retrieves the user language given the user primary key.
The ISO Codes for the language and country are then stored in the user session with the Set Component.

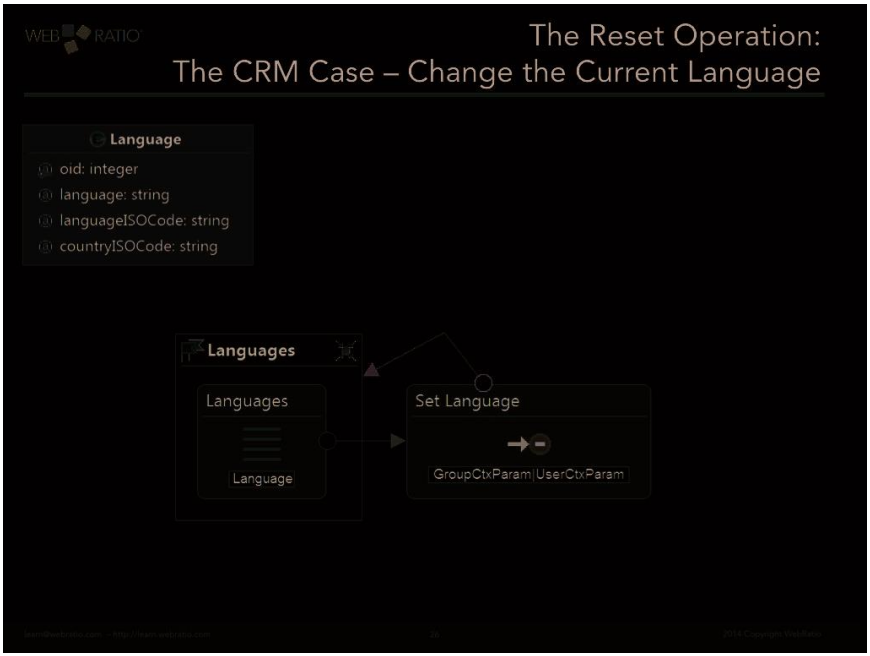


The Reset Operation

The Reset operation is an operation used to store the values of one or more context parameters in the user session. It is used to change an existing value to a new one and can be triggered by the user interaction. It is a Session Component, so it is inside the Session Components session of the toolbar.

The CRM Case - Change the Current Language

An example of usage for the Reset Operation changes the current language used for the application.
This example requires a Domain Model entity, called 'Language', used to store all the available languages.
The IFML model has a page with a list of available languages that the user can select. Each selection activates the Reset Operation that resets the language value for the currently logged in user. The selected language is passed with a normal navigation flow that is outgoing from the list and activates the Reset Operation.



Additional Resources

Online Training

- http://www.webratio.com/learn/learningobject/access-control-personalization-ifml-model
- http://www.webratio.com/learn/learningobject/access-control-personalization-action-definition

KB Articles

- http://www.webratio.com/learn/learningobject/how-to-model-password-recovery