MarkLogic Server

Application Builder Developer's Guide

MarkLogic 8
February, 2015

Last Revised: 8.0-1, February, 2015

Copyright © 2015 MarkLogic Corporation. All rights reserved.

Table of Contents

Application Builder Developer's Guide

1.0 Application Builder Quick Start 4
- 1.1 Overview of Application Builder 4
- 1.2 Setting Up and Starting Application Services 5
  - 1.2.1 Clean Installation 5
  - 1.2.2 Starting Application Services 5
- 1.3 Building the Oscars Sample Application 6
- 1.4 Loading the Complete Set of Oscars Data 14
- 1.5 Using the Oscars Sample Application 18
  - 1.5.1 Keyword Searching, Search Suggestions, and Parsing 18
  - 1.5.2 Browsing with Facets 20
  - 1.5.3 Search Result Page 21
  - 1.5.4 Displaying Content Details 21
- 1.6 Using Application Builder to Modify the Oscars Sample Application 22

2.0 Creating a Search Application With Application Builder 25
- 2.1 Starting Application Services 25
- 2.2 Navigating in Application Builder 25
- 2.3 Page-By-Page Walkthrough 27
  - 2.3.1 Selecting or Creating an Application 28
  - 2.3.2 Search Page 30
    - 2.3.2.1 Add/Modify Range Constraints 31
    - 2.3.2.2 Add/Modify Value Constraints 34
    - 2.3.2.3 Add/Modify Word Constraint 36
    - 2.3.2.4 Add/Modify Collection Constraint 37
    - 2.3.2.5 Modifying Search Options 38
  - 2.3.3 Assemble Page 41
  - 2.3.4 Results Page 46
  - 2.3.5 Sort Page 48
  - 2.3.6 Content Page 50
  - 2.3.7 Appearance Page 53
  - 2.3.8 Deploy Page 54

3.0 Controlling Access to Application Builder and to Generated Applications 55
- 3.1 Predefined Roles for Application Builder 55
  - 3.1.1 app-user Role 55
  - 3.1.2 app-builder Role 55
  - 3.1.3 app-builder-internal Role 55
- 3.2 Permissions on Documents 56

3.3 Modifying Roles to Meet Your Requirements 56

4.0 Extending Applications Built With Application Builder 57
4.1 Viewing the Generated Code 57
4.2 The Custom Directory 58
4.3 Customizing Applications Generated by Application Builder 59
4.3.1 Basic Design Pattern 60
4.3.2 Accessing the Code in the Custom Directory 60
4.3.3 Road Map for Application Page Objects 61
4.3.4 Customizing the Footer 63
4.3.5 Customizing the Content Display 64
4.3.6 Customizing the Results Page 65
4.3.6.1 Adding to the content.xsl Stylesheet 65
4.3.6.2 Adding More/Less Links 66
4.3.7 Adding Custom JavaScript 67
4.4 Making Further Modifications to the Application 68
4.5 Removing Modifications to an Application 68

5.0 Technical Support 69

6.0 Copyright 70
6.0 NOTICE 70

1.0 Application Builder Quick Start

Note: Application Builder is deprecated and will be removed in a future release of MarkLogic Server.

This chapter gets you started using Application Builder, a browser-based application that quickly creates fully functional search and analytics applications. These generated applications can, if needed, be customized and extended. This chapter includes the following sections:

- Overview of Application Builder
- Setting Up and Starting Application Services
- Building the Oscars Sample Application
- Loading the Complete Set of Oscars Data
- Using the Oscars Sample Application
- Using Application Builder to Modify the Oscars Sample Application

1.1 Overview of Application Builder

Using Application Builder requires no coding on your part. Its user interface is easy to use, while its search applications can have many high-end search features such as a search box with Google-style search grammar, search suggestions, faceted navigation, and results visualization widgets. It scales for huge database sizes while maintaining its speed.

The generated application uses the Search API and can be used as is or customized with your own code.You can define many aspects of an application, such as:

- Facets
- Details appearing on the search result page
- Content display control via item rendering
- Visualization widgets for search results

Typically, building an application is an iterative process. To begin, you must have a representative content set loaded in a database with any needed indexes already set up. If your content is not complete or not completely indexed, you can still generate an application and modify it as you modify your content.

In this chapter, you use Application Builder to generate a sample application based on data about the Oscar awards. For details, see “Building the Oscars Sample Application” on page 6 and “Using the Oscars Sample Application” on page 18.

1.2 Setting Up and Starting Application Services

Application Builder is bundled with MarkLogic Server Application Services. On a fresh installation of MarkLogic 8, Application Builder is preconfigured and ready to use. For an upgrade installation, your existing application data remains intact although some renaming of your Application database and App Server may occur during the installation process. This section describes the following scenarios:

- Clean Installation
- Starting Application Services

1.2.1 Clean Installation

When you install MarkLogic Server for the first time, the installation process does the following:

- Creates an HTTP App Server named `App-Services` on port 8000 for Application Services
- Creates a database named `App-Services` to store the Application Builder application documents

For a complete description of installing MarkLogic Server, see Installing MarkLogic in the *Installation Guide*.

1.2.2 Starting Application Services

To start Application Services, open a browser and go to your server's port 8000. For example, if your browser runs on the same machine as MarkLogic Server, open the following URL:

```
http://localhost:8000/appservices
```

When MarkLogic Server prompts you for a username and password, enter them for a user with either the `admin` or `app-builder` role.

For details on these roles, see "Predefined Roles for Application Builder" on page 55.

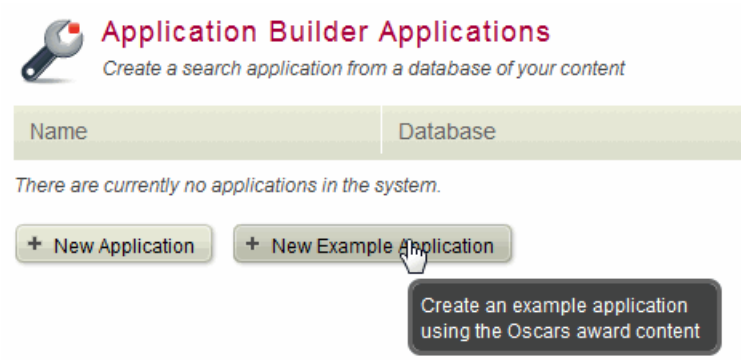
1.3 Building the Oscars Sample Application

Application Builder includes a template to build a sample application based on Oscar awards data from Wikipedia. To build this application, go through the Application Builder wizard as follows:

1. Start Application Builder by going to the following URL (If MarkLogic Server is installed on a different host or your App Server uses a different port, substitute those values):

   ```
   http://localhost:8000/appservices
   ```

2. On the Application Builder Applications screen, click New Example Application.

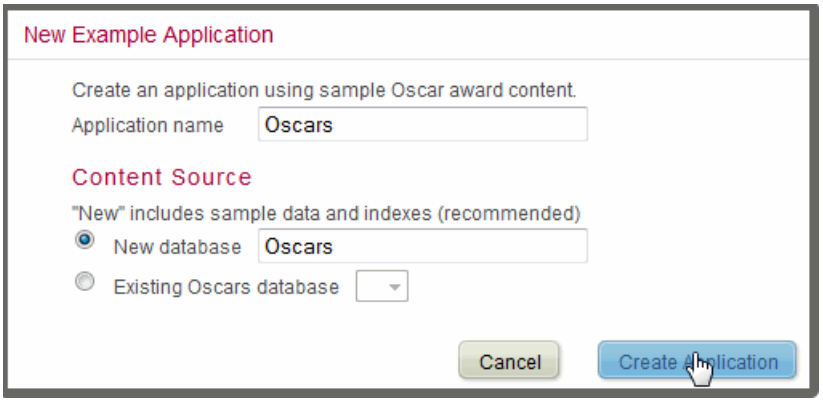


3. The New Example Application screen appears.

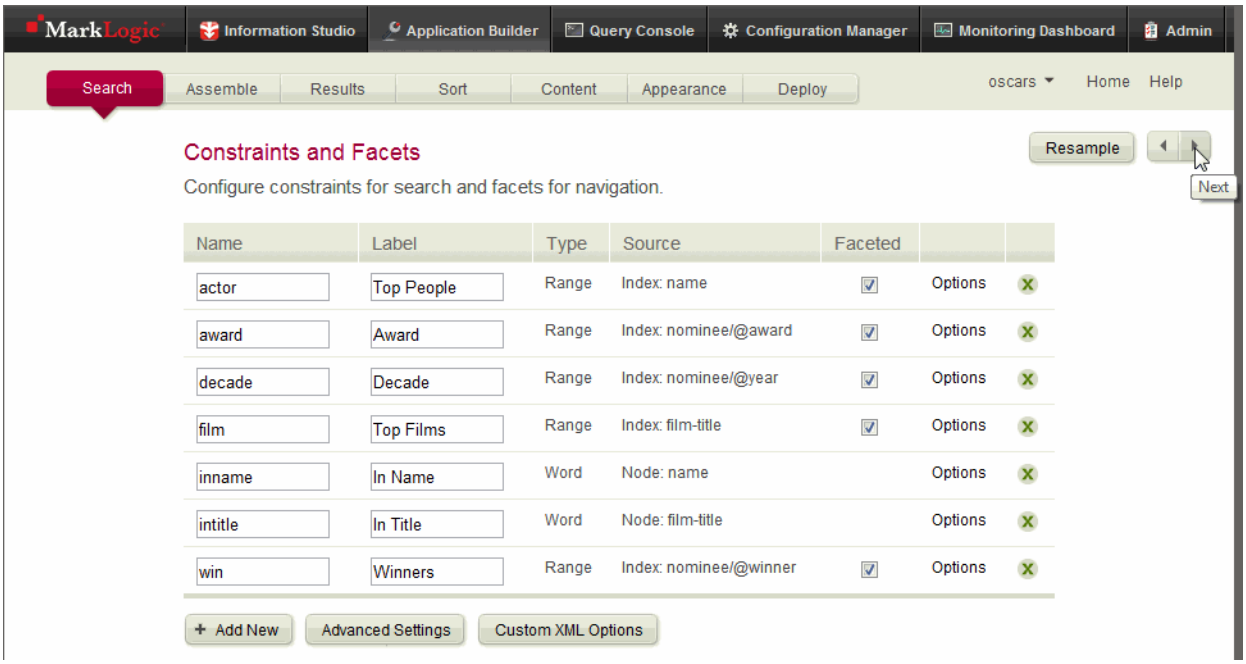
   a. Enter a name for the application, in this case `Oscars`.

   b. Select New Database and enter a database name, in this case `Oscars`.

   c. Click Create Application.



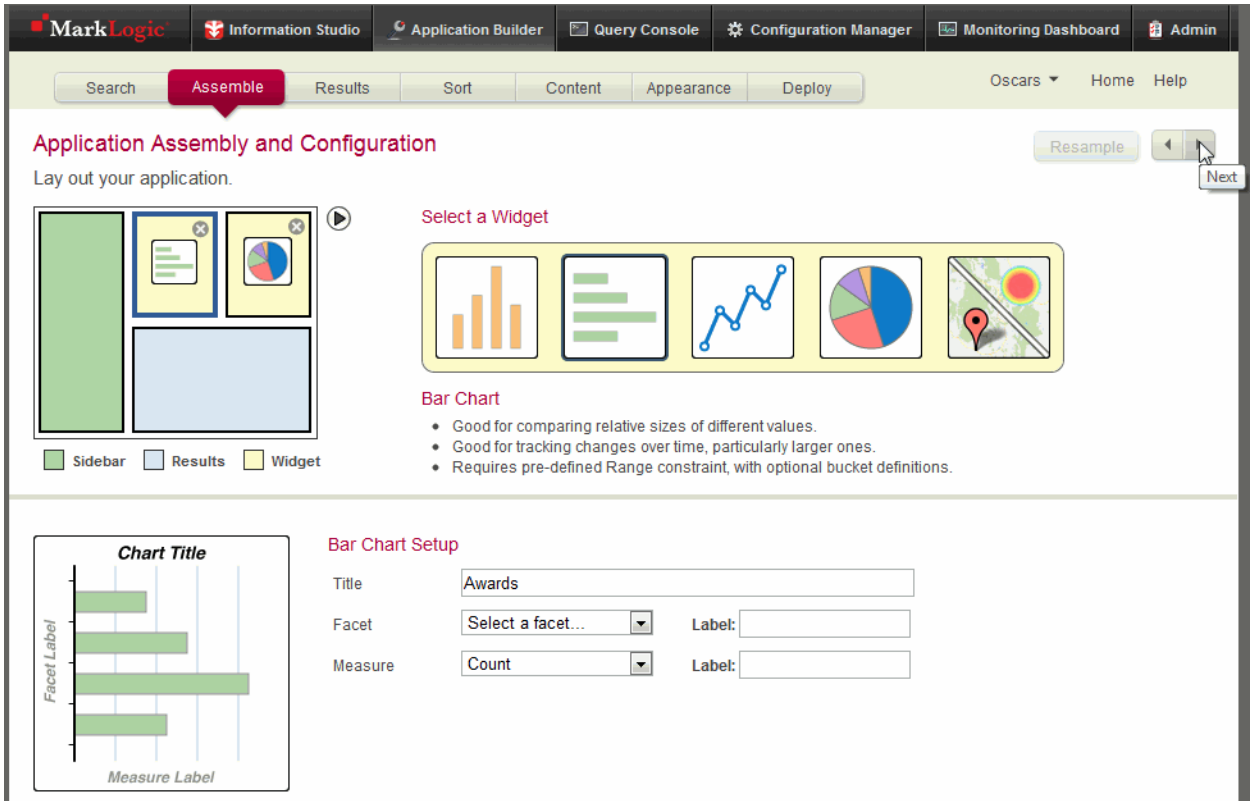
4. Application Builder creates an Oscars App Server, forest, and database and then displays the Search tab page.



a. On the Search page, you can accept the default search constraints and facets or change the settings.

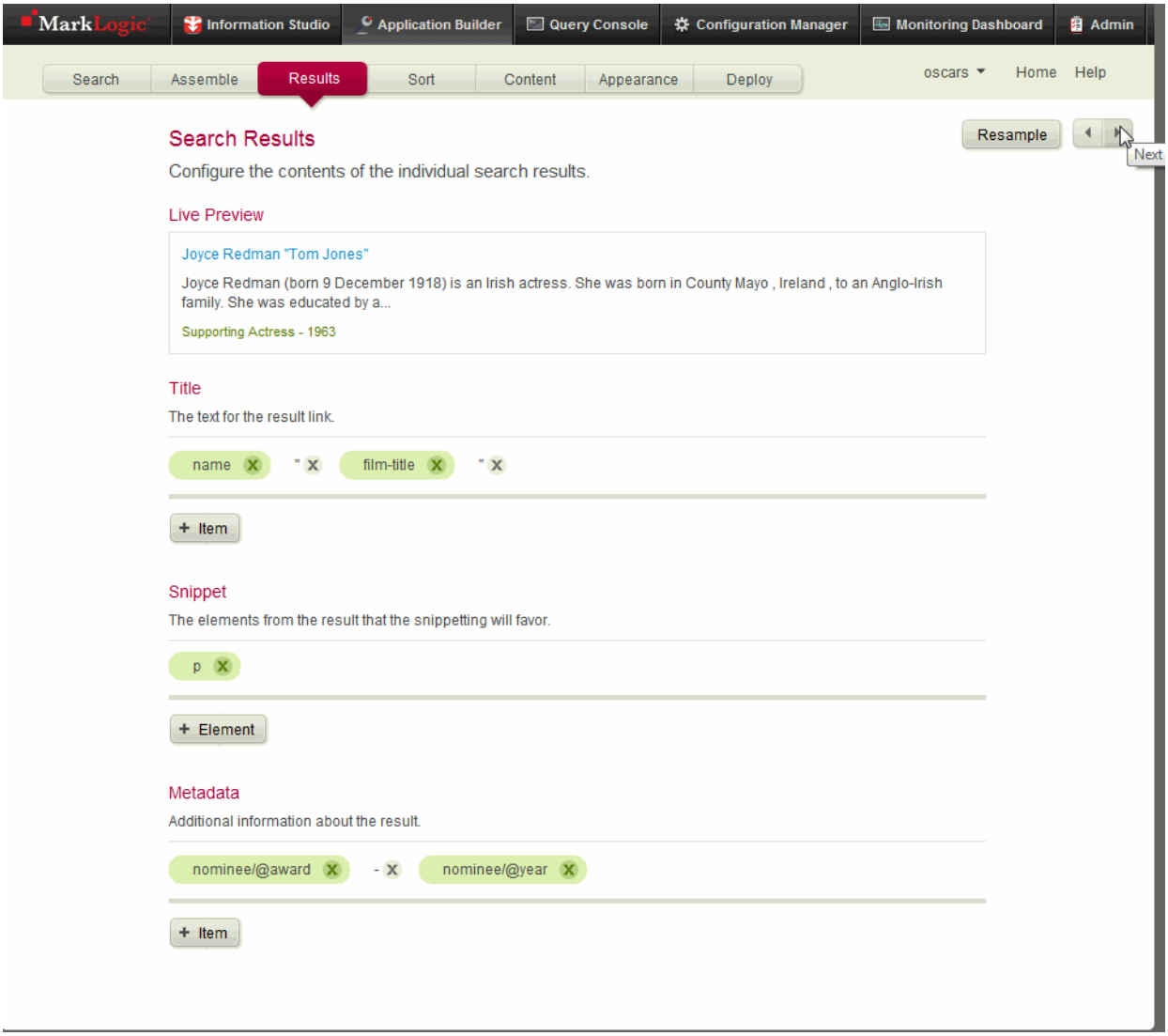
b. Click the next button at the upper right to go to the Assemble tab page.

c. On the Assemble page, you can accept the defaults widgets and layout or change the settings.



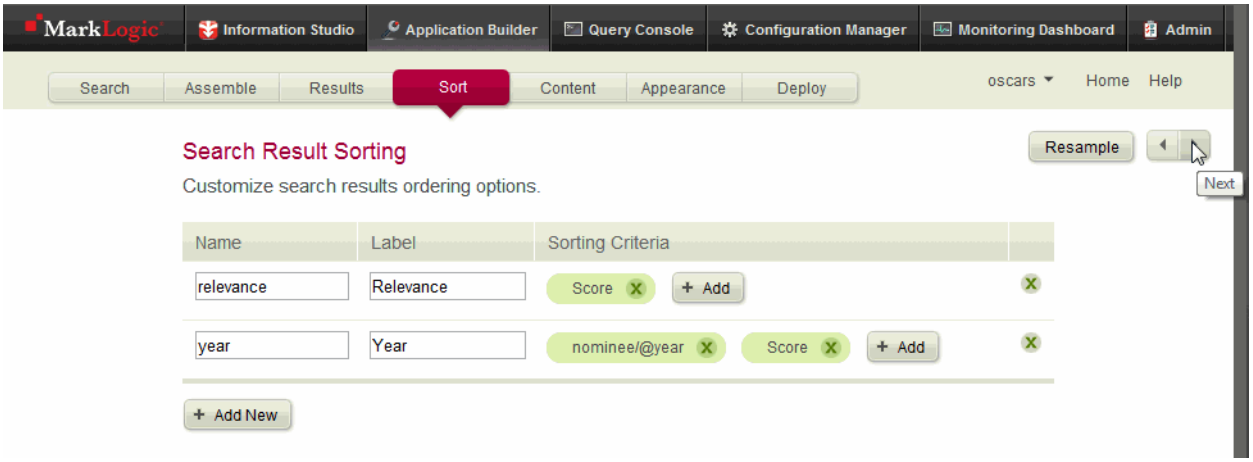
d. Click the next button at the upper right to go to the Results tab page.

e. On the Results page, you can accept the defaults for the contents of an individual search result or change the settings.



f. Click the next button to move to the Sort tab page.

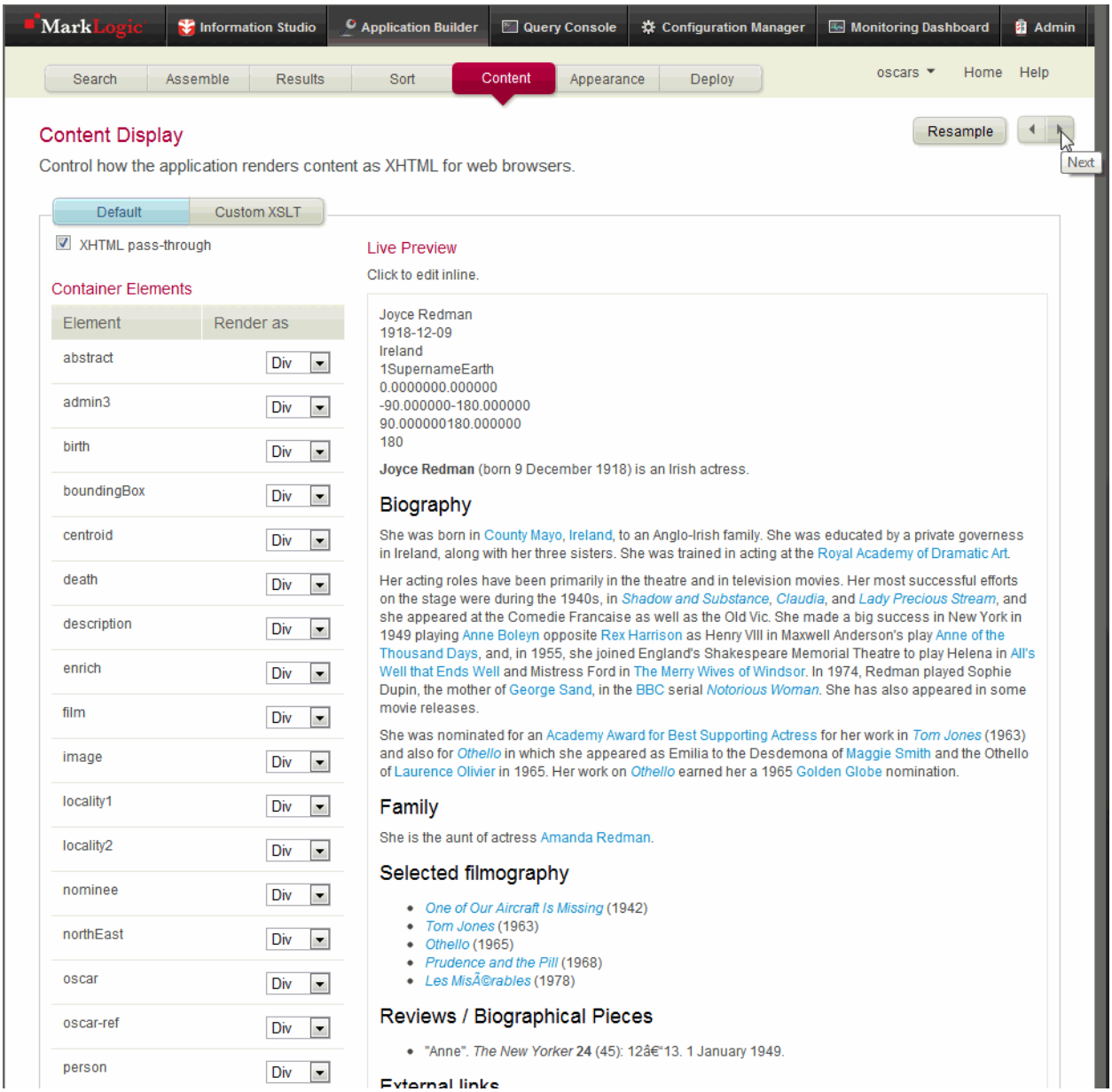
5. The Sort tab page displays.



a. On the Sort page, you can accept the default search results ordering(s) or change the settings.

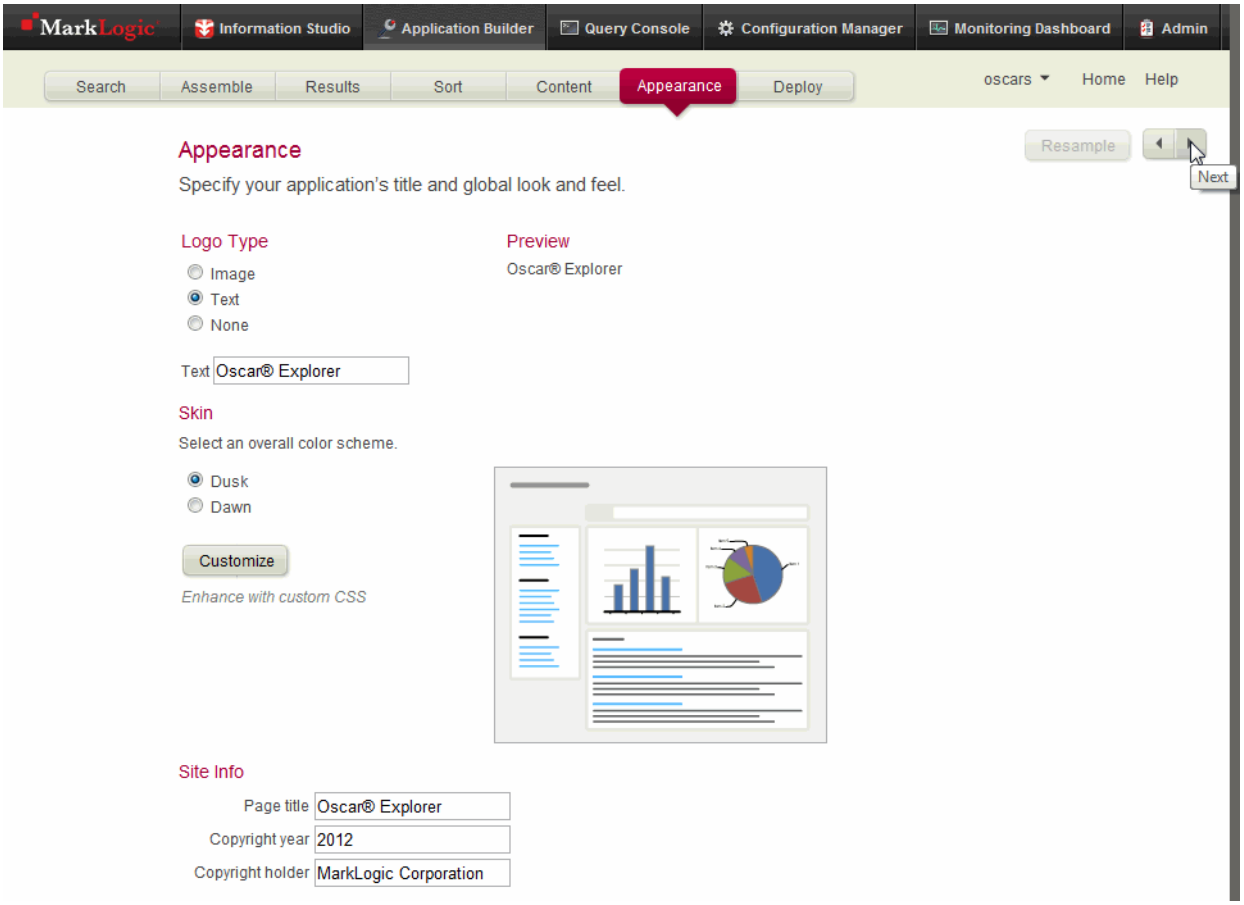
b. Click the next button to go to the Content page.

6. In the Content tab page, you can control how the application renders content as XHTML for web browsers.



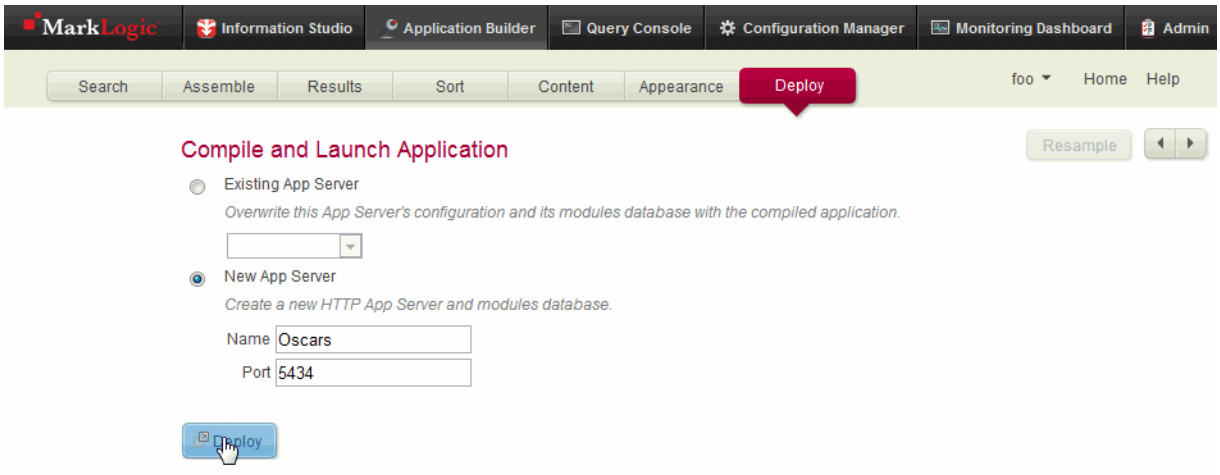
a. Click the next button to move to the Apperance tab page.

7. On the Appearance page, you can specify your application's title and overall look and feel.



a. Click the next button to go to the Deploy tab page.

8. The Deploy page appears.



a. On the Deploy page, select New App Server. (You can only select Existing App Server if an App Server is already configured for this application.) Accept the default values or provide the App Server with a name and port number.

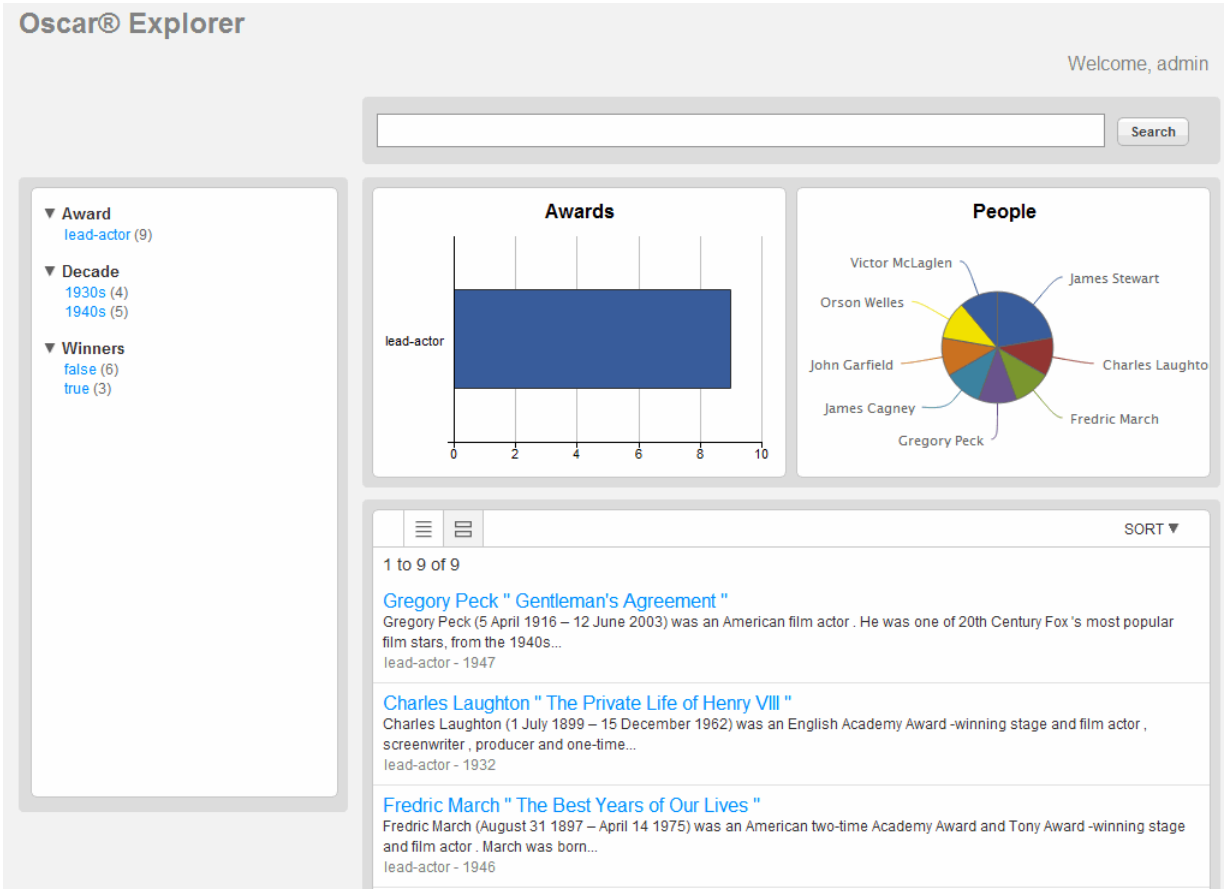
b. Click the Deploy button and confirm.

9. Application Builder creates and configures the new App Server and opens a new window where it launches the new application. This may take a short while. When Application Builder prompts you to log in, enter a username and password and click OK.

   For details about controlling access to your new application, see “Controlling Access to Application Builder and to Generated Applications” on page 55.

   Note: When you deploy an application, it appears in a new browser window using a URL whose hostname is what is stored in the MarkLogic Server configuration files (the result of an `xdmp:host-name` call). If you are running on a laptop computer that is changing networks, the hostname might not be available on your network, resulting in a 404 or similar error when the application launches (because it is trying to access a server name unavailable on your current network). Substituting `localhost` for the hostname in the URL should fix this.

10. You can test the Oscars application by entering search terms or clicking on the browse links to narrow the diplayed results.



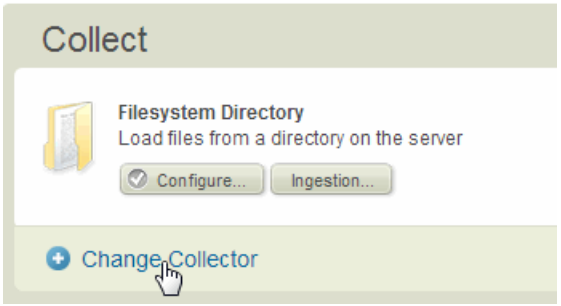
1.4 Loading the Complete Set of Oscars Data

Initally, Application Builder only loads a few sample data files for use by the Oscars application. To load the full 20 MB content set, use the following steps:

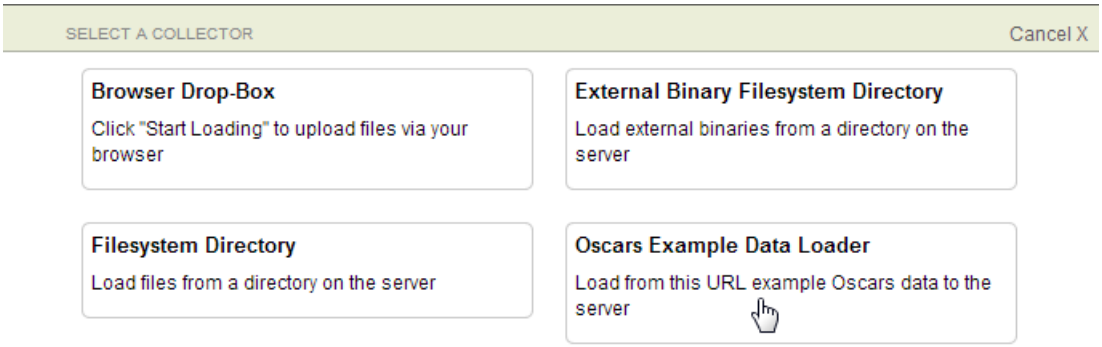
1. In the Information Studio Flows section of the Application Services page, click New Flow.

   The Flow Editor appears.

2. Click Change Collector at the bottom of the Collect section:



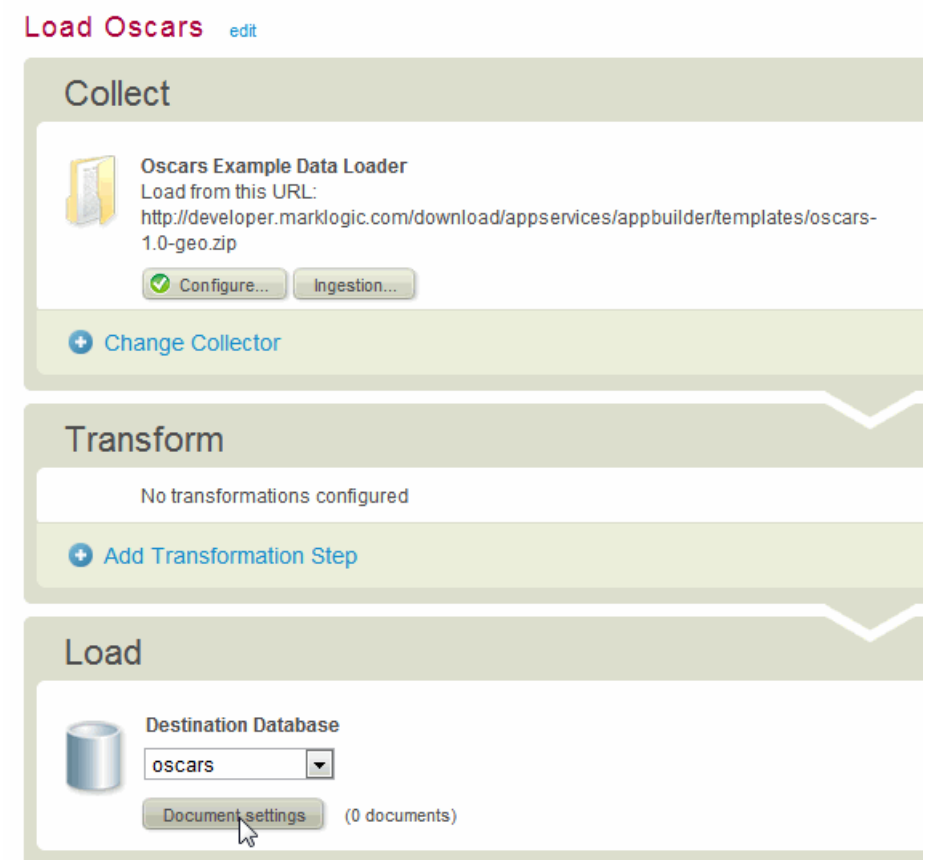
3. In the Select A Collector window, select Oscars Example Data Loader:



4. In the Configure Settings window, select Done. Do not make any changes in this window.

5. The Collect section of the Flow now shows that the Oscars Example Data Loader is configured and the URL from which it will download the data.

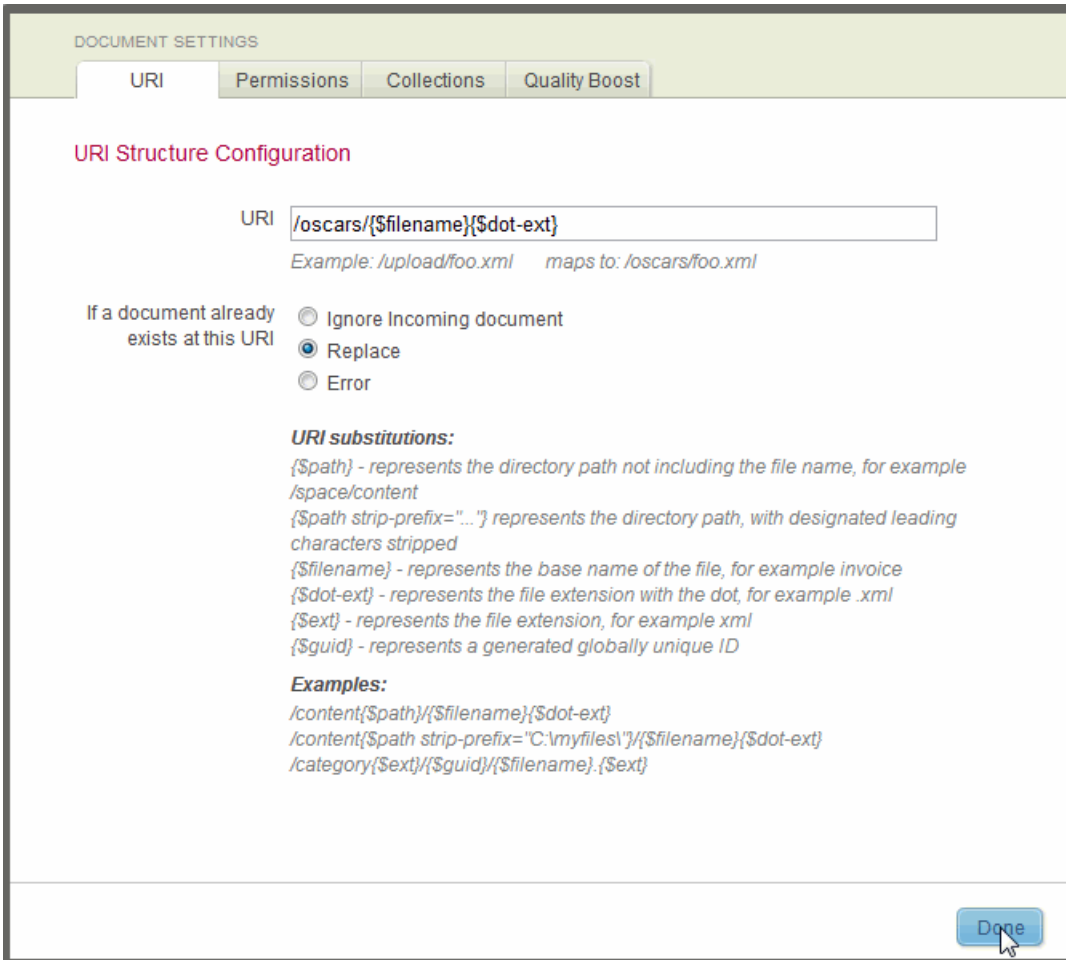
6. In the Load section, select oscars as the Destination Database and click Document Settings:



7. In the Document Settings window, change the URI Structure Configuration to:

```
/oscars/{$filename}{$dot-ext}
```

Click Done.



8. Click Start Loading to load the content into the database.

The data downloads automatically over your Internet connection, while a spinning icon appears until the load is complete. When done, you will see different count values and additional facet values.

Note: While downloading the sample page, do not navigate away from it or close the browser window until the spinning icon disappears and the page reloads, otherwise the download might be interrupted.

1.5 Using the Oscars Sample Application

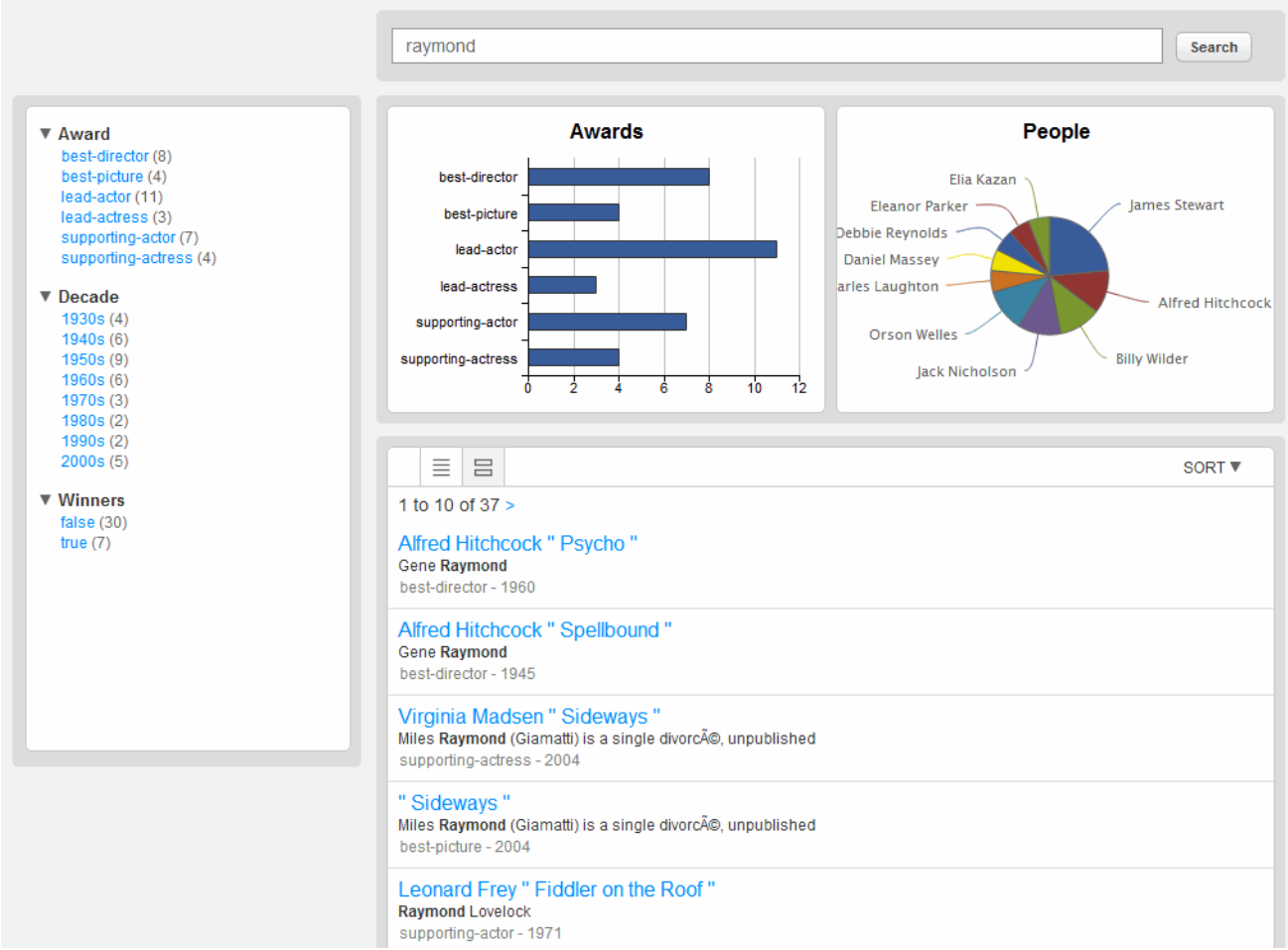
The Oscars sample application enables you to search, browse, and display articles about Oscar award winners from the last nine decades. It is uses the Search API's standard features, including query text parsing, faceted navigation, snippetting, and many more. For details about the Search API, see Search API: Understanding and Using in the *Search Developer's Guide*.

While you can learn about the application by playing with it, this section highlights some of its main features, including:

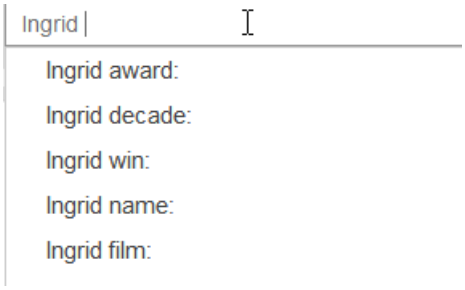
- Keyword Searching, Search Suggestions, and Parsing
- Browsing with Facets
- Search Result Page
- Displaying Content Details

1.5.1 Keyword Searching, Search Suggestions, and Parsing

You can enter keywords into the search box and press return to search the database. For example, a search for `raymond` shows snippets for the first 10 of 37 results.



As you type search terms, the application suggests your possible full search. For example, as you type `Ingrid Bergman`, you might see something like the following:



Because the application uses the Search API, you can use standard search grammar such as combining multiple terms with AND semantics, treating double-quoted phrases as phrases, and so on. For information about the search grammar for the Search API, see Search API: Understanding and Using in the *Search Developer's Guide*.

You can also search using constraints. For example, the following query text finds everything about the actor Dustin Hoffman:

```
actor:"Dustin Hoffman"
```

This is not a standard full-text search, but is a constraint showing all the documents matching where an particular value in the source XML, `<actor>`, has the content `Dustin Hoffman`. You can combine the constraint with other terms to further narrow the results:

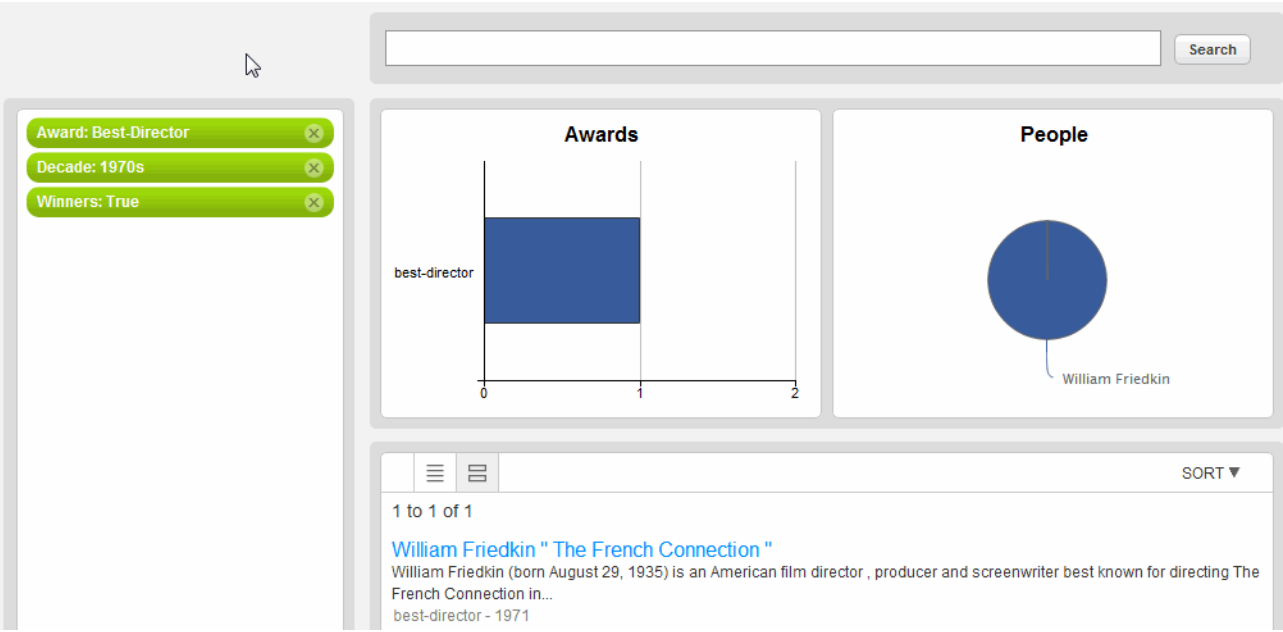
```
buck actor:"Dustin Hoffman"
```

When you click on any links in the user interface, notice that the query text in the search box shows the current query.

1.5.2 Browsing with Facets

The left side of the application shows facets used to browse through the content. When you click on a facet, it narrows the overall search results to those results in the facet's category, while keeping the existing categories or search terms active.

For example, if you first click on the Award:Best Director browse link, then on the Decade:1970s link, and then on the Winners:True link, your results are all of the 1970s winners of the Best Director award.



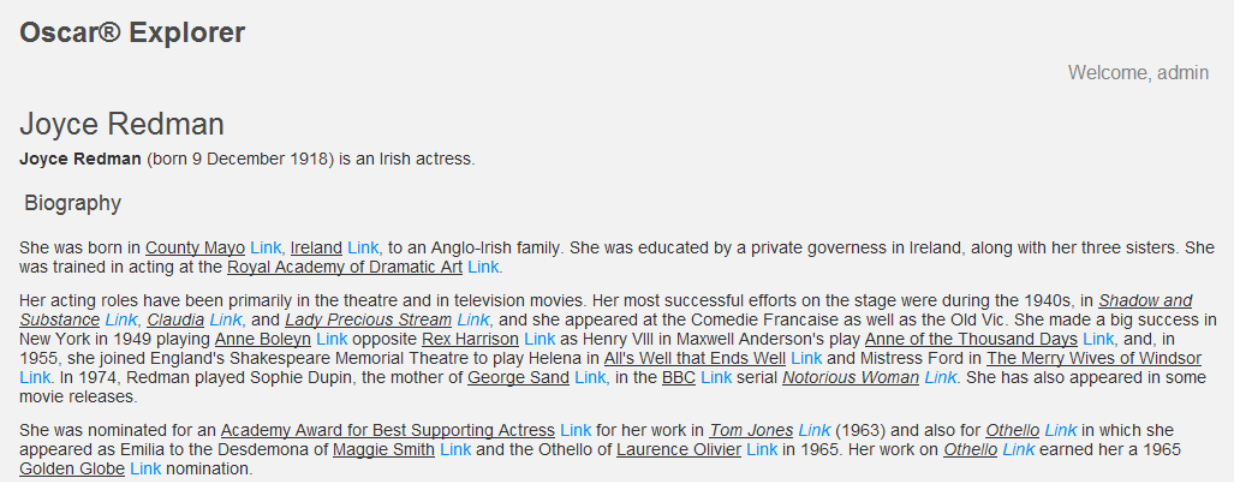
Each of the browse facets has a count of how many of its results match your current query.

1.5.3 Search Result Page

The search result page shows a link with a text summary of the content, highlighted snippets of the content matching your search, and other information about the search match. Clicking the result link takes you to the content details.

1.5.4 Displaying Content Details

The content details page includes the complete content for the search result. The rendering is based on the configuration specified on Application Builder's Content Display page. The page's style is based on the skin you chose and on any custom CSS entered on the Appearance page.



1.6 Using Application Builder to Modify the Oscars Sample Application

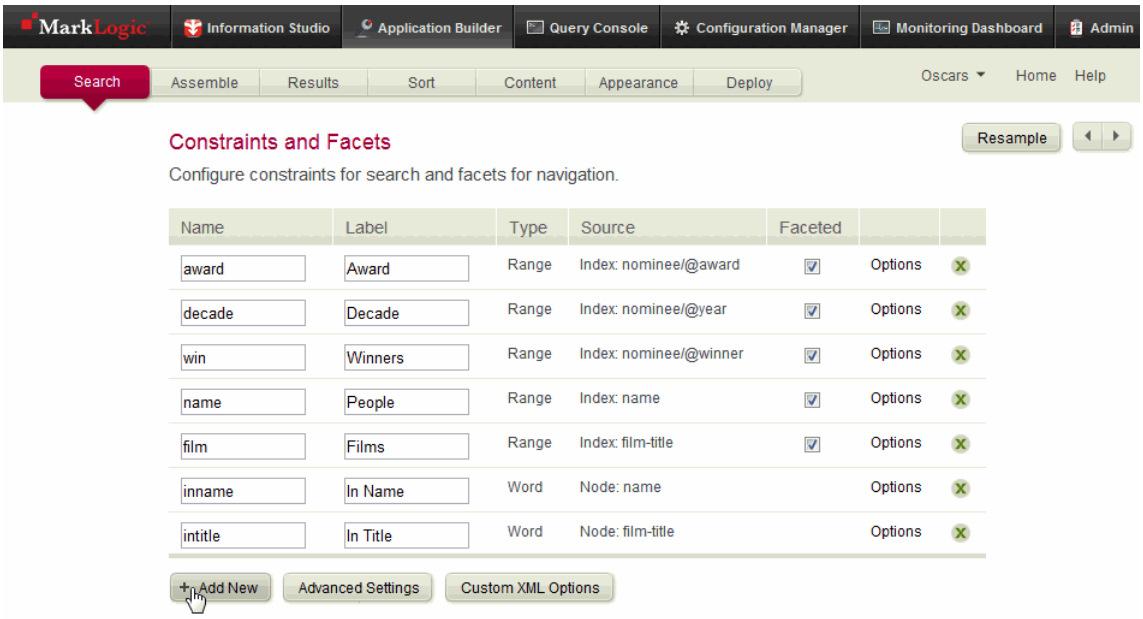
This section describes how to add a year facet to the Oscars application. With the year facet, you can first drill down on results with the decade facet, then drill down further using the year facet. The year facet uses the same index as the decade facet.

To create a year facet, do the following:

1. Start Application Builder (for example, open `http://localhost:8000/appservices` in a browser).

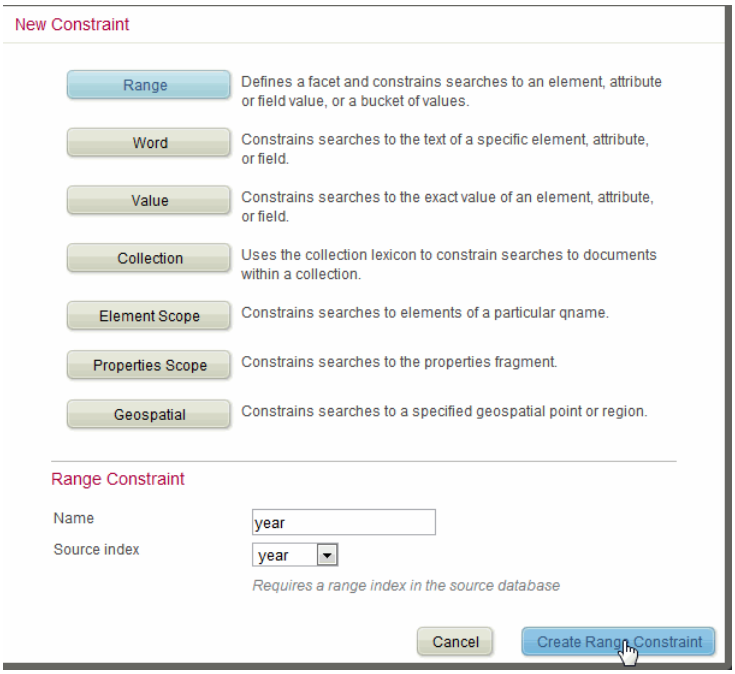
2. On the Application Builder Applications page, click the application name that you used for your Oscars sample application (in this case, `Oscars`).

3. Click the Search tab. At the bottom of the Search page, click Add New:



4. In the New Constraint dialog, Click Range.

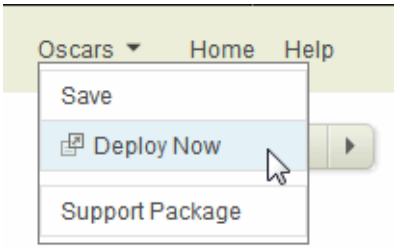
a. In the New Constraint dialog, enter `year` for the Name and select `year` for the Source Index.



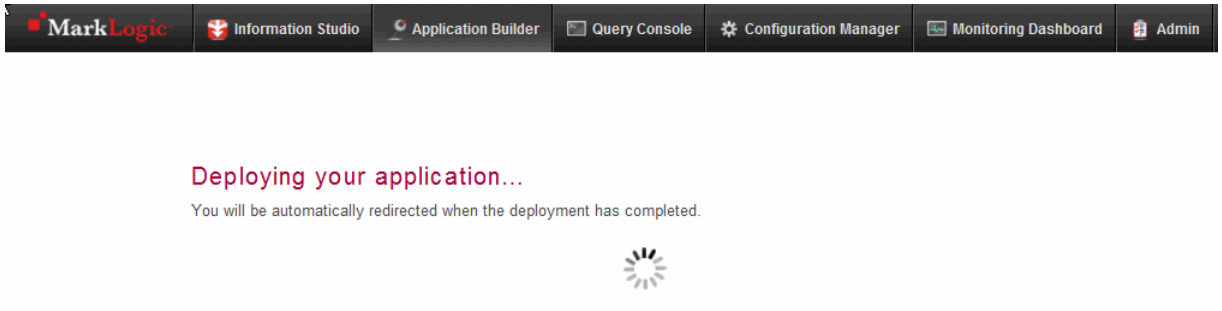
b. Click Create Range Constraint. Application Builder creates the constraint.



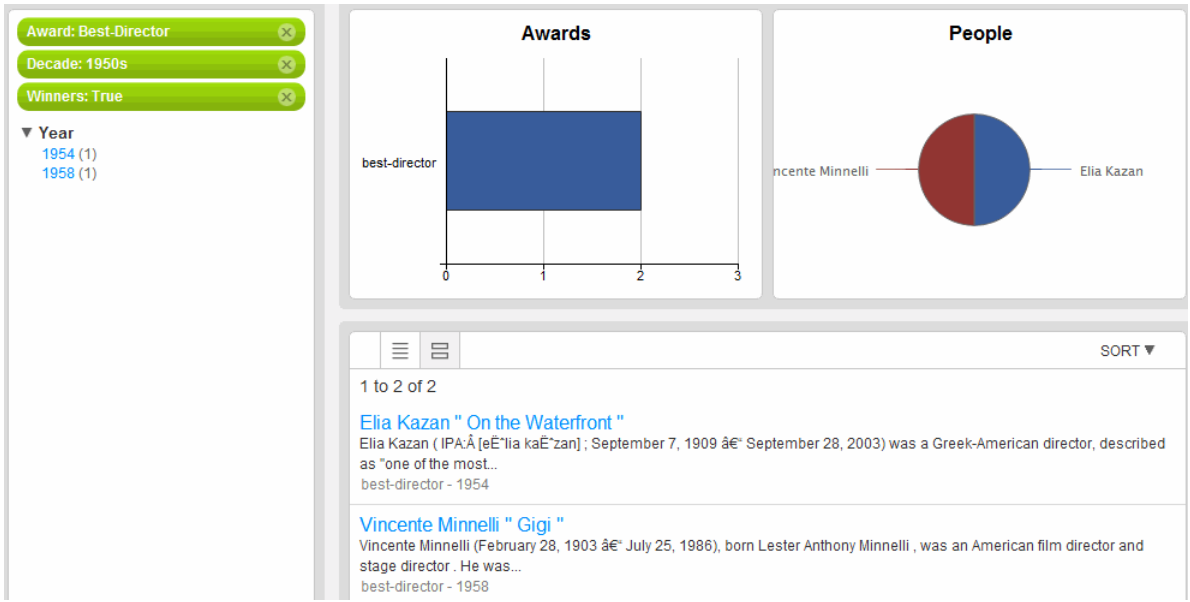
5. In the application name menu, click Deploy Now from the Oscars pull-down menu:



6. Application Builder compiles and deploys the new application code to your App Server's modules database. During deployment, the following appears in a new window:



When Application Builder is done, the newly modified application replaces the status page, including the new year facet. Test the facet by doing a search, selecting a decade, and then selecting a year to find the results for a single year from that decade.



Note: The year facet in this example is available at all times, whether or not you have clicked on the decade facet. To only display the year facet after selecting a decade, you need to add additional display logic. The Oscars example application is not set up to do hierarchical logic, but you can modify it to do so. For details on modifying the application, see "Extending Applications Built With Application Builder" on page 57.

2.0 Creating a Search Application With Application Builder

This chapter describes how to use Application Builder to create a search application.

- Starting Application Services
- Navigating in Application Builder
- Page-By-Page Walkthrough

2.1 Starting Application Services

Application Builder is bundled as part of the Application Services suite of applications. To start Application Services, open the following URL in a browser window:

```
http://localhost:8000/appservices
```

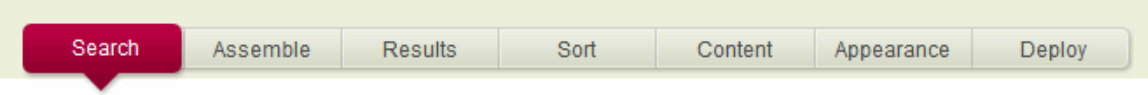
If your instance of MarkLogic Server is running on a different host, or if Application Services is configured on a different port, substitute those values for host and port.

To use Application Builder, your login account must have the `app-builder` role. To use Information Studio, it must have the `infostudio-user` role. The `admin` role has access to both applications.

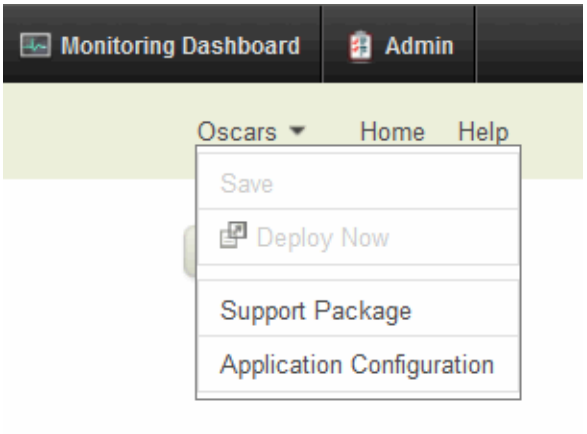
To begin, log in as a user with the `app-builder` or `admin` role. For details about the `app-builder` role, see “Predefined Roles for Application Builder” on page 55.

2.2 Navigating in Application Builder

Application Builder’s user interface is a straightforward tabbed interface, where each tab configures different application functionality.



Click any tab to go to that page, and click the next and back buttons to navigate to the adjacent screens. The following table describes the Application Builder's navigation elements that appear on most of its pages.

Navigation Element	Description
Tabs	Clicking a tab displays that page in Application Builder. Changing tabs automatically saves the application's state.
Application name menu	If you click on the name of the application towards the upper right corner of the screen, a drop-down options menu appears: 
	Save saves the application to the database.
	Deploy Now immediately deploys the application. It is onlyt available after the first deployment of the application.
	Support Package generates a zip file of the application (including all application code) in case you need to contact MarkLogic Support. The zip file also includes a small document sample from your content database.
	Application Configuration displays the current application's XML in a new window. This includes the Search API options element, which is helpful if you are generating Search API code.
Home	The Home button takes you back to the Application Services page.
Help	The Help button displays the help content for the current page.

Navigation Element	Description
Next/Back	Use these buttons to navigate to the next or previous page. 
Resample	When available, the Resample button lets you sample random documents in the content database to build constraints, search results, and render content. **Note:** Resampling replaces many of the current settings on the page, so certain customizations you have made are lost after resampling.

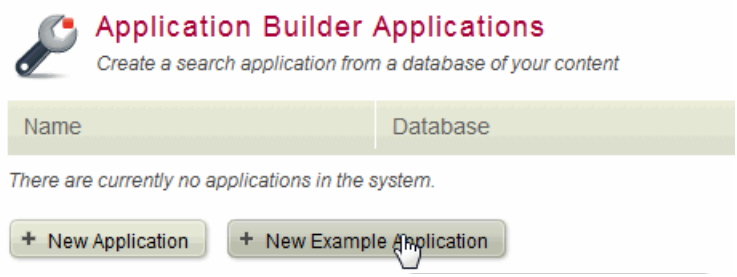
2.3 Page-By-Page Walkthrough

This section describes each page of Application Builder, and includes the following parts:

- Selecting or Creating an Application
- Search Page
- Assemble Page
- Results Page
- Sort Page
- Content Page
- Appearance Page
- Deploy Page

2.3.1 Selecting or Creating an Application

When you start Application Services or click the Home link in the upper right part of other Application Builder pages, the Application Services page opens with the Application Builder Applications section at the top.



The Application Builder Applications section lists all applications in the `App-Services` database and lets you create new applications, new sample applications (such as the Oscars sample application), or modify existing applications. An application stores all of the information needed to generate itself, including configured constraints, results page settings, deployment options, etc.

When you create an application, you specify the database to use with it. If you are creating a new Oscars example application, you can create a new database for it. You must load a representative sample of content and set up any indexes for the database outside of and before you use Application Builder. Application Builder looks at the indexes configured in your database and uses them to help you configure the application. It is not a problem if your database structure is still evolving, because you can modify the application as needed.by clicking the application name on the Application Services page.

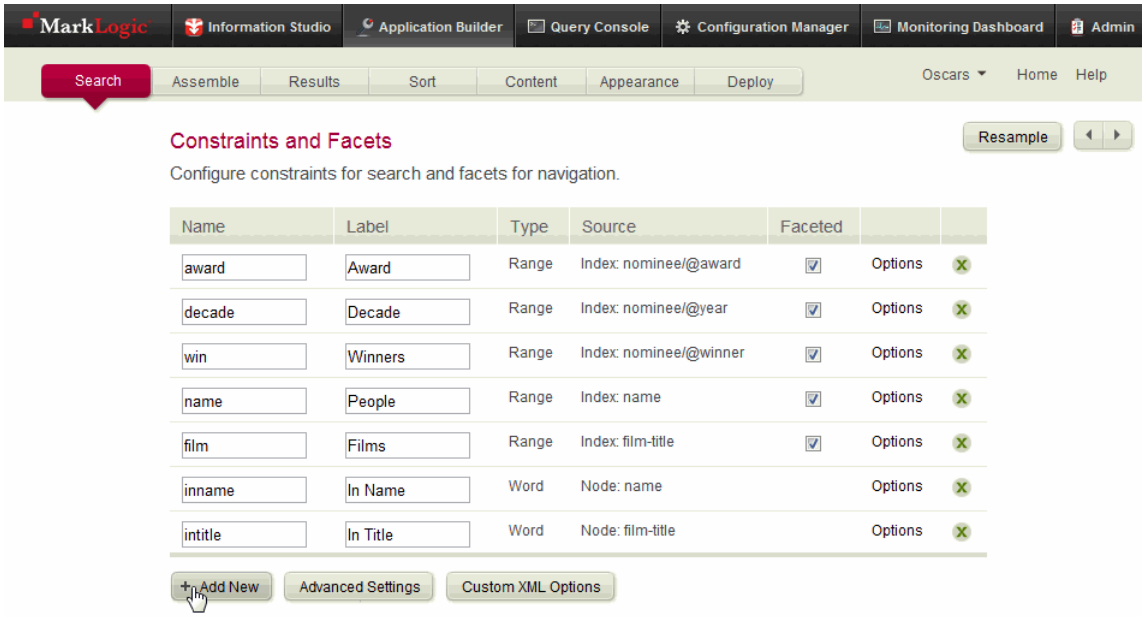
> **Note:** Application Builder assumes there is no database fragmentation. If your database is fragmented, you might need to modify the generated application for it to work properly. For details on modifying the generated application, see “Extending Applications Built With Application Builder” on page 57.

The following table lists the actions on the Application Builder Applications section of the Application Services page:

Action	Description
New Application	Creates a new application. After clicking the New Application button, enter the application name and choose an existing database for it to use.
New Example Application	Creates a new application designed to use the Oscars sample data. After clicking the New Sample Application button, enter the name of the application and choose a database to use for the Oscars content. Choosing a new database is the normal way to create the Oscars sample application. If you choose an existing database, it should have the settings used for the Oscars database. For more details on building the Oscars Sample, see “Building the Oscars Sample Application” on page 6.
Edit	Open an existing application to modify settings and/or deploy the application.
Delete	Permanently delete an application from the `App-Services` database. Deletes the application, but not the database or App Server used by the application.

2.3.2 Search Page

The Search page is where you configure constraints and facets. The facets enable you to drill down on your result set, narrowing the search to a given category.



You can configure existing constraints or add new constraints. You can use Range and Collection constraints to create facets, and all constraints can be used as query text with the constraint name and value. For example, the following query text in the generated Oscars sample application returns all awards from the 1980s:

```
decade:1980s
```

The following sections describe several kinds of constraints you can add or modify on the Search page, as well as modifying other application search options.

- Add/Modify Range Constraints
- Add/Modify Value Constraints
- Add/Modify Word Constraint
- Add/Modify Collection Constraint
- Modifying Search Options

2.3.2.1 Add/Modify Range Constraints

A range constraint uses range indexes to support queries and create facets.

To add a new constraint, click Add New.

To edit an existing constraint, click the Options link on its row.

Range constraints can be faceted and used in queries with text like the following:

```
decade:1980s
```

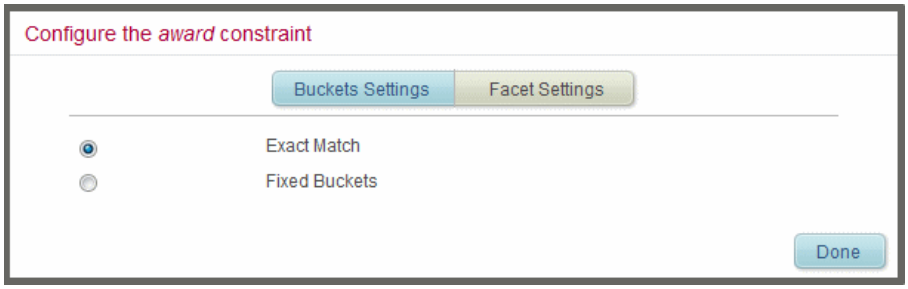
You can create a range constraint on an element, attribute, or field. There are three types of Range constraints:

- Exact Match
- Fixed Buckets
- Relative Buckets (only available for date ranges)

You can modify existing buckets by editing the fields in the table listing the buckets.

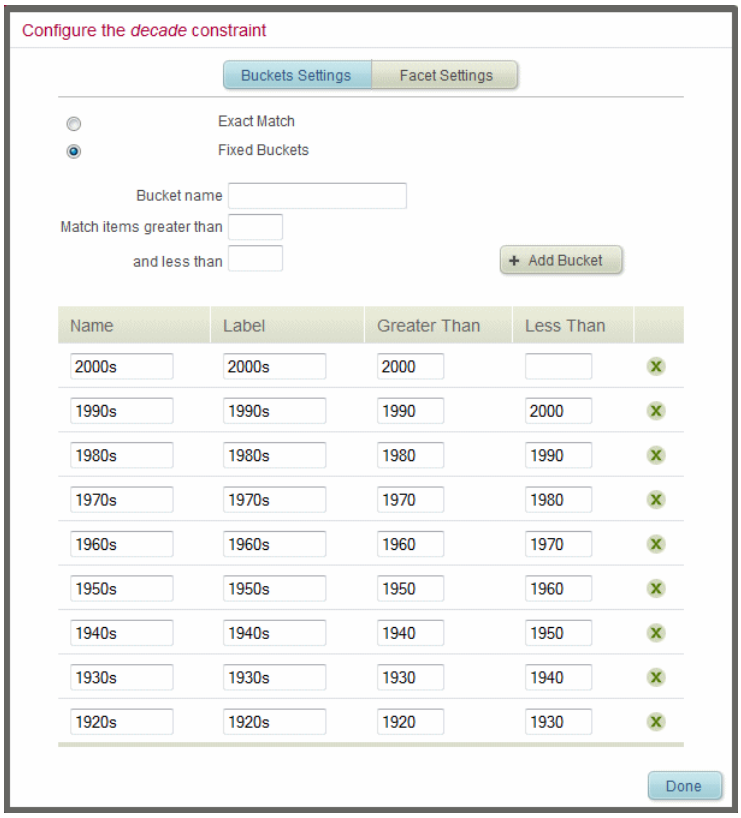
Exact Match Range Constraints

Exact match Range constraints match on each individual value of the specified Range index. These are useful if you want every value to have significance in constraints and facets.



Fixed Buckets Range Constraints

Fixed buckets constraints let you specify labels corresponding to ranges of values. These are useful when you want to specify ranges of values for queries and facets, such as decades on time-based ranges.



Relative Buckets Range Constraints

Relative buckets constraints are used with date ranges (elements or attributes of type `xs:date` or `xs:dateTime`). You can see the relative buckets if you have a range index set up for one of the date type elements in your database.

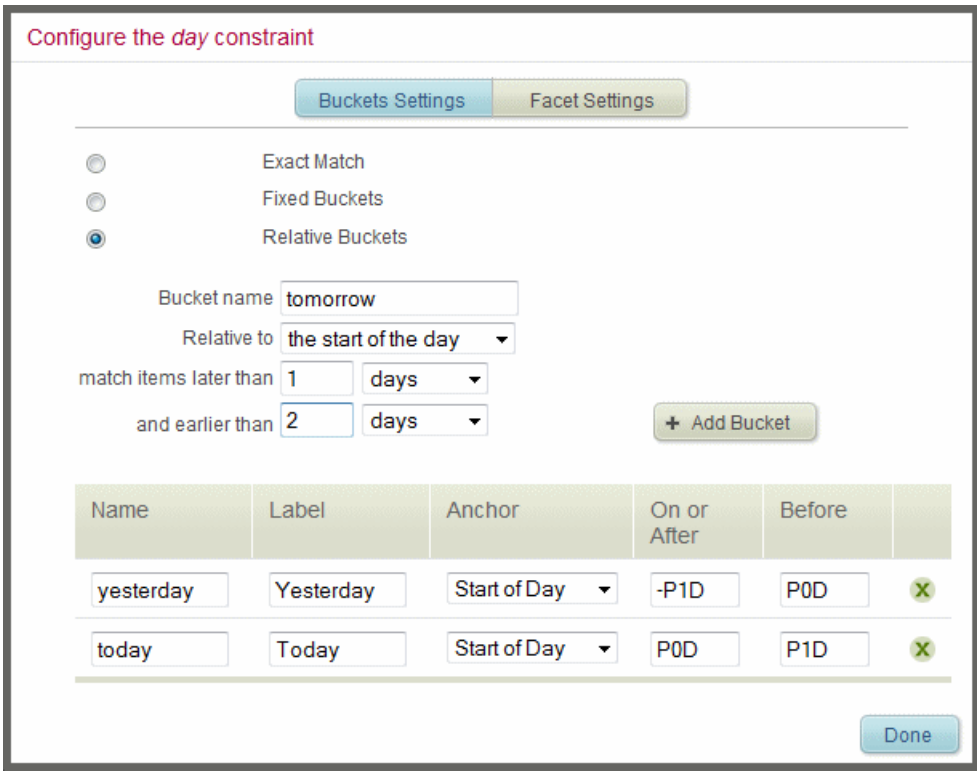
To create a new bucket that represents a range of dates, fill in the form items and click Add Bucket. The bucket table adds a new row with the range you specified.

The Anchor column fields in the Relative Buckets Configuration dialog specify what part of the `date` or `dateTime` value to start counting where the bucket boundaries are.

The On, After, and Before fields in the relative buckets table must be represented as durations. Durations specify time periods, and can have arithmetic done on them. For information about the durations syntax, see http://en.wikipedia.org/wiki/ISO_8601#Durations or the appropriate section of the XML Schema specification (http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#isoformats). For example, a duration of `P0D` means zero days from the specified value, a duration of `P1D` means one day after the specified value, and a duration of `-P1D` means one day before the specified value. Similarly, a duration of `P1Y` means one year later, a duration of `-P3M` means three months earlier, and a duration of `P10H` means ten hours later.

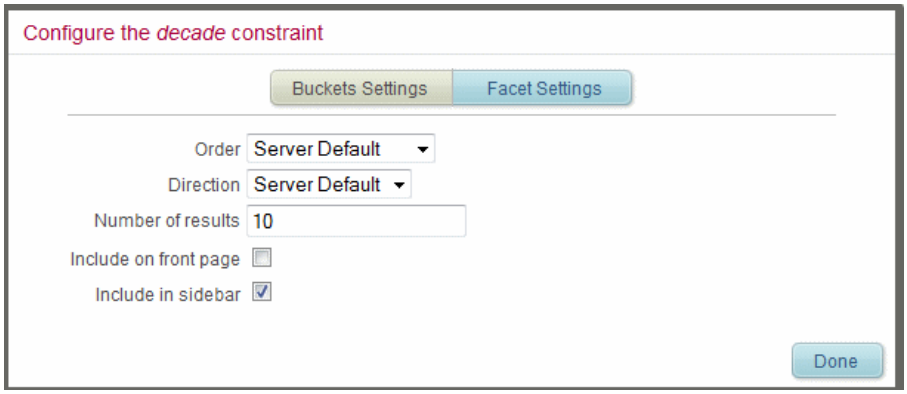
To see the Relative Buckets Configuration dialog using the Oscar content, create an element range index on `wiki:date` of type `date`. (You need to use the admin interface or Information Studio to create the index.) For more details on element range indexes see Range Indexes and Lexicons in the *Administrator's Guide*.

The following example shows a Relative Buckets constraint dialog:



Facet Settings

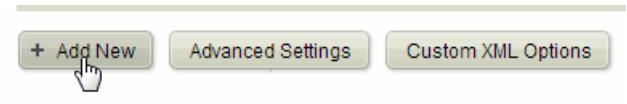
Each faceted range constraint, also has Facet Settings that control things such as the number of facets that appear in the results.



2.3.2.2 Add/Modify Value Constraints

A value constraint uses element, attribute, or field values to create a constraint.

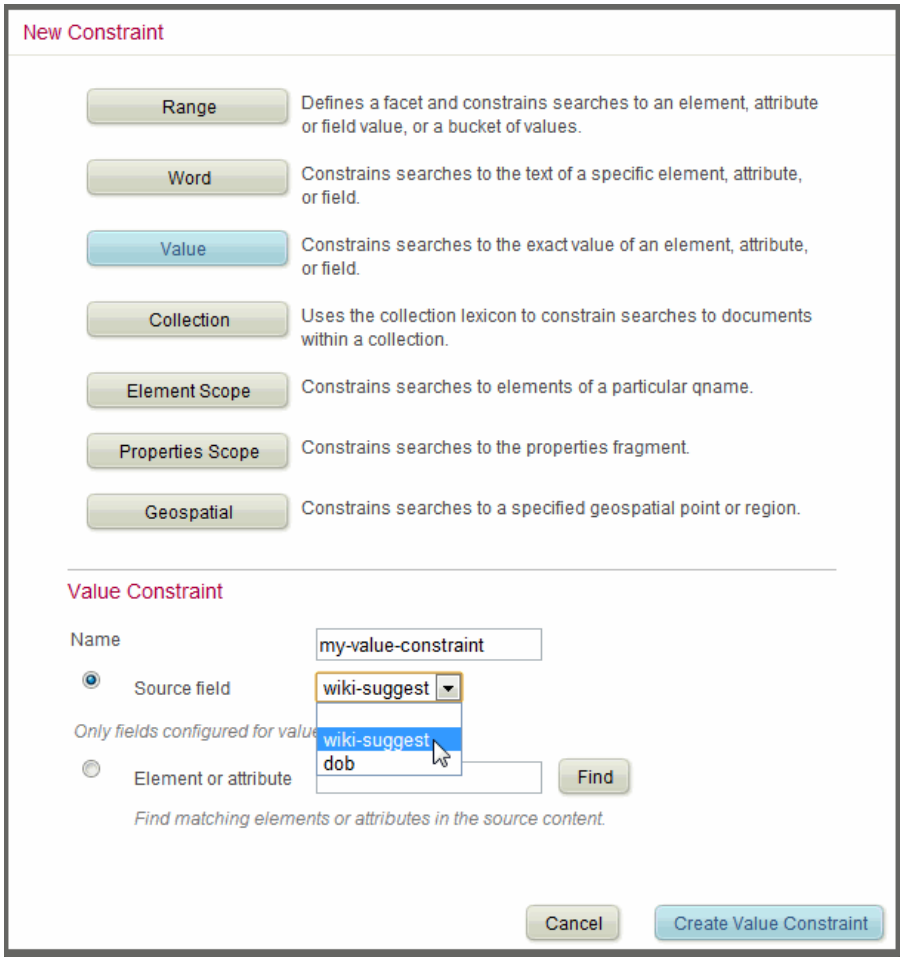
To add a new constraint, click Add New:



To edit an existing constraint, click the Options link on the existing constaint's row.

On the New Constraint page, enter a name for the constraint and enter either `select a field` from the Source Field drop-down list or enter an element or attribute name to match on. When you click Find for elements or attributes, Application Builder searches through sample documents in your database to find wildcard-matching elements or attributes.

Note: To configure a field value constraint, the field must be configured and you must have Field Value Searches enabled, either for the database or for the individual field. For information on configuring fields, see Fields Database Settings in the *Administrator's Guide*.



Value constraints differ from range constraints in these ways:

- You do not need a range index for value constraints, but you do need range indexes for range constraints.
- You cannot create facets from value constraints, but you can facet range constraints.
- Value constraints are text queries, range constraints are type-aware matches.

Value constraints match when the text you search for matches the element or attribute value. For example, the following value-constraint query text matches the element `<author>Raymond Carver</author>`, assuming it is case-insensitive (which is the default).

```
author:"raymond carver"
```

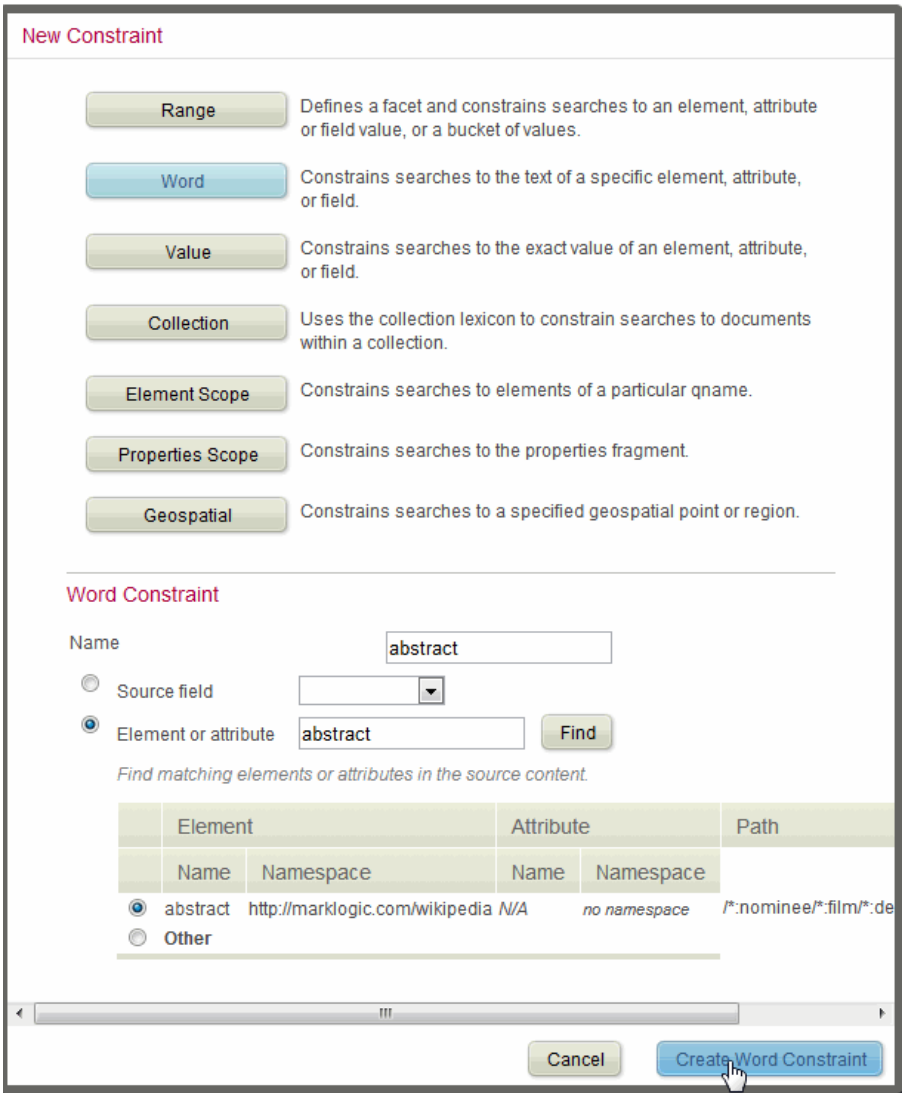
2.3.2.3 Add/Modify Word Constraint

A word constraint uses elements, attributes, or fields to find words, matching documents with the specified word. For example, if you have created a word constraint on the ABSTRACT element, the following search finds documents with the word hello in an ABSTRACT element:

```
abstract:hello
```

To add a new constraint, click Add New.

To edit an existing constraint, click the Options link on the existing constraint's row.



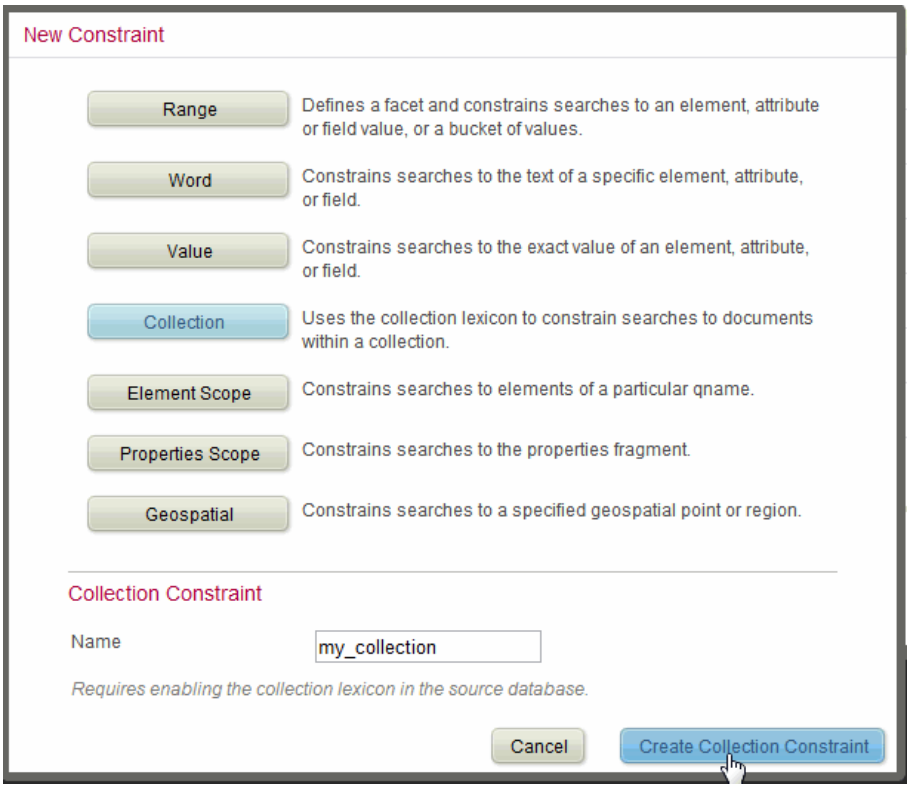
Word constraints cannot be faceted and do not require special indexes. If you specify an element, attribute, or field that does not exist, queries using the constraint return no results.

2.3.2.4 Add/Modify Collection Constraint

A collection constraint constrains a search to items in the specified collection and enables facets on the collection.

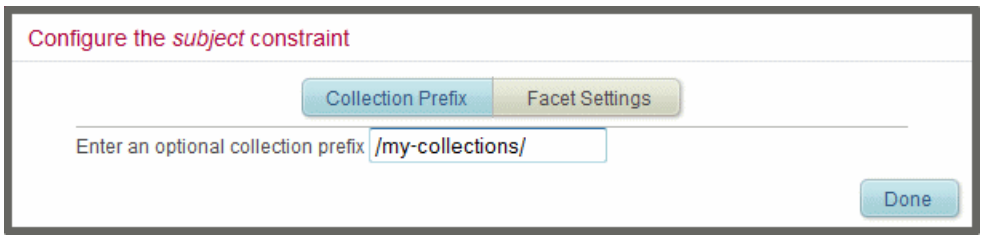
To add a new constraint, click Add New.

To edit an existing constraint, click the Options link on that constraint's row.



A Collection constraint requires the collection lexicon is enabled in the database. You can also create facets with a collection constraint.

You can optionally specify a prefix to concatenate with the collection constraint value. To create a collection prefix, click the Options link for the constraint and set a prefix in the constraint dialog as shown in the following screen capture.



For example, assume all documents in your database have collection URIs begining with the string `/my-collections/` similar to the following:

```
/my-collections/math
/my-collections/economics
/my-collections/zoology
```

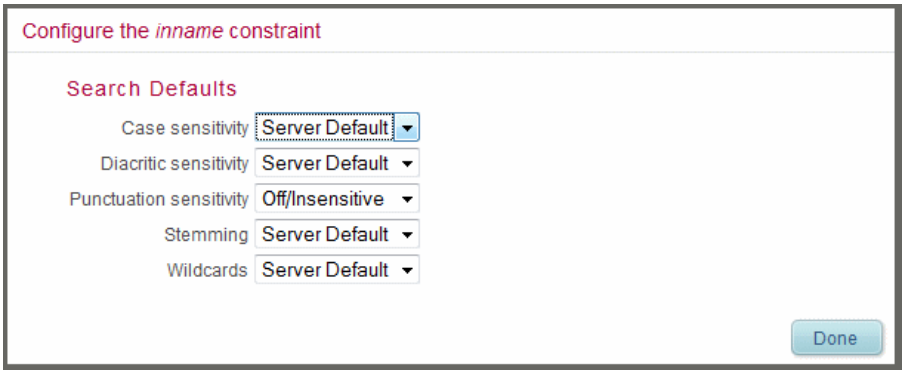
In this case, the following query text examples match documents in the corresponding collections:

```
subject:math
subject:economics
subject:zoology
```

In the constraint dialog, you can click Facet Settings to create facets on the collections.

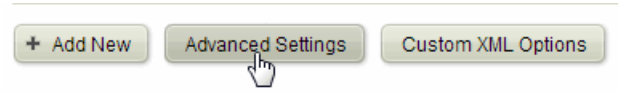
2.3.2.5 Modifying Search Options

For word and value constraints, you can modify the search options (case sensitive/insensitive, diacritic sensitive/insensitive, and so on) by clicking the constraint's Options link. Most options are set to the MarkLogic Server defaults as shown in the following screen capture.

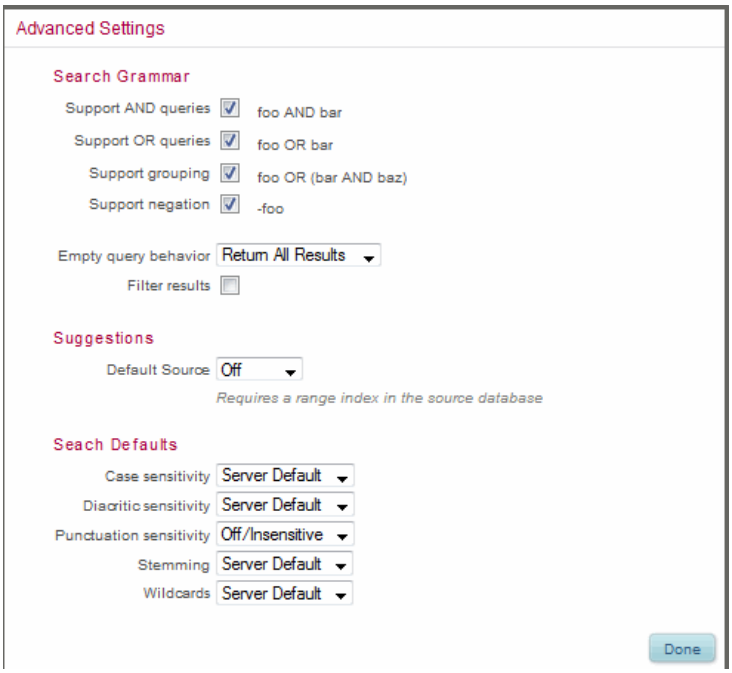


For details on the MarkLogic Server defaults for query behavior, see the documentation for the individual `cts:query` constructors in the *XQuery Built-In and Module API Documentation*.

Also, clicking the Search page's Advanced settings button displays a dialog that allows you to modify the search grammar and search options for everything besides the constraints in your application (some constraints let you set their search options independently). By default, many options are set to the Search API and MarkLogic Server defaults.

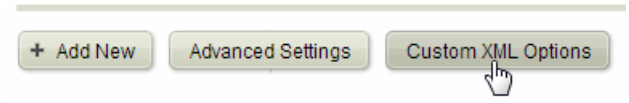


The advanced settings dialog looks similar to the following screen capture:

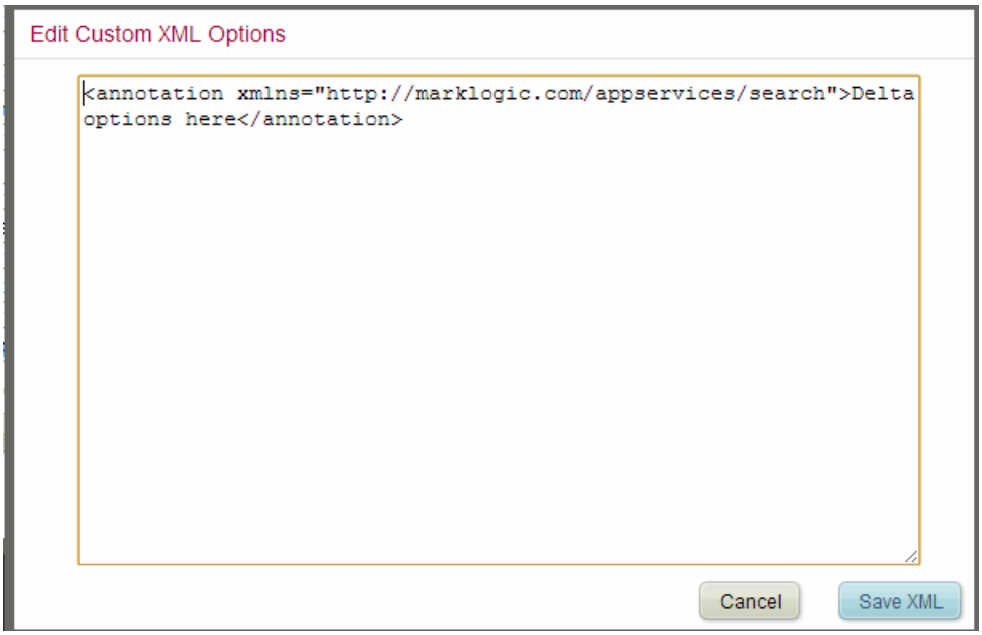


In the Advanced Settings dialog, select the Filter Results check box to specify whether searches should be run unfiltered or filtered. Filtered searches are more accurate in some cases. Unfiltered searches are always faster.

Clicking the Custom XML Options button allows you to append existing search options at deploy time:



A dialog box appears in which you can include additional search options that will be appended to the search options node when the application is deployed.



For details on the various search options in MarkLogic Server, see the *Search Developer's Guide*. Some examples of additional search options are listed in the table below.

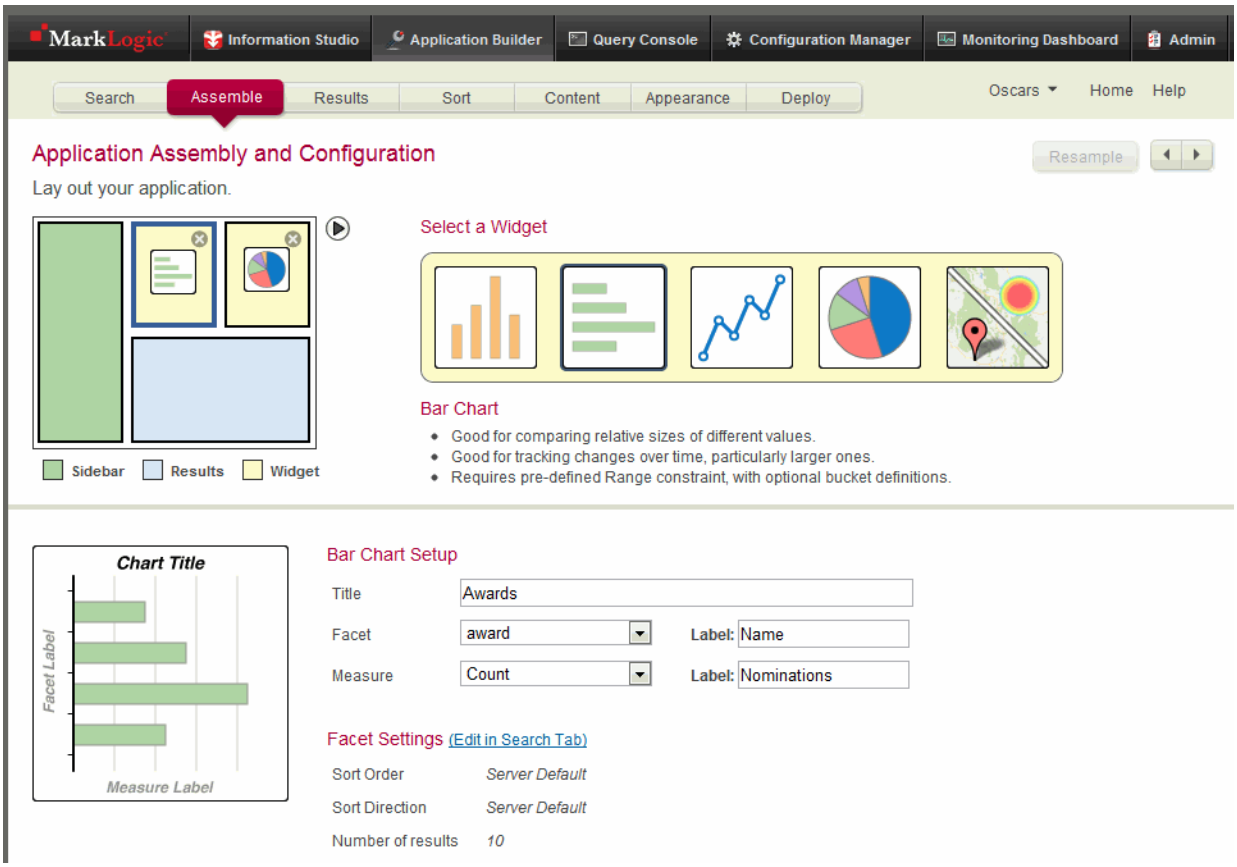
Option	Description
`<search:additional-query>`	Add an additional query (for example, a query on a particular directory or collection) across all queries.
`<search:debug>`	Set to `true` to enable additional debug information in the search response. This would be used during development by way of a tool like firebug.
`<search:annotation>`	Add an annotation to the options. This will not have any effect on the generated application, but a downstream application might access the `v1/config/query/all` endpoint and do something based on the presence or absence of this annotation.

Option	Description

`<search:constraint name="name">`
`  <search:custom facet="false">`
`  .....`
`  </search:custom>`
`</search:constraint>`
	Add a custom contraint search option, as described in Creating a Custom Constraint in the *Search Developer's Guide*.
	
	Implement the parser using either the structured query or multi-format parse function interface. Be sure to install the XQuery module that implements the custom constraint, as described in Installing or Updating an Asset in the *REST Application Developer's Guide*.

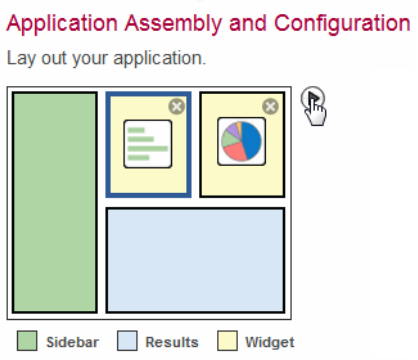
2.3.3 Assemble Page

Use the Assemble page to select and configure any widgets for your application. Widgets are described in the Data Visualization Widgets chapter in the *Search Developer's Guide*.

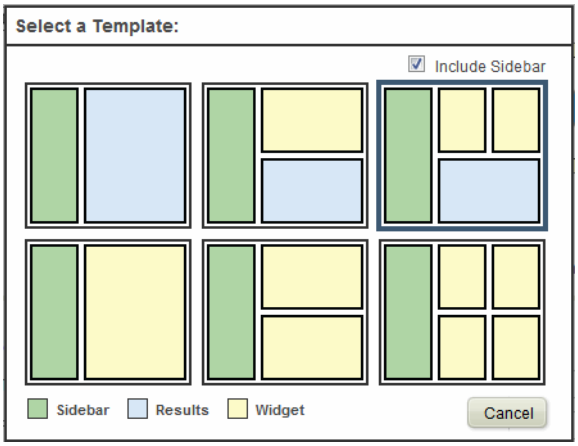


To configure the layout of the widgets on your application page, do the following:

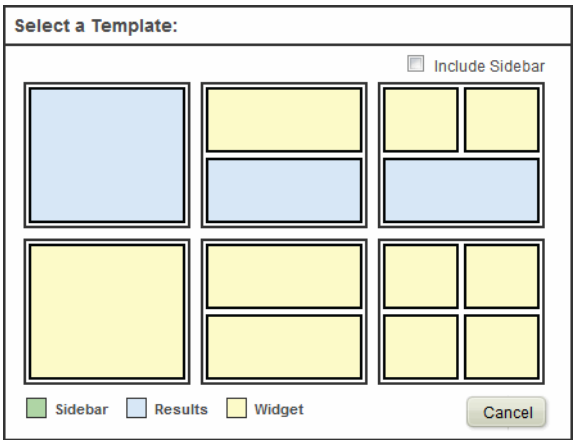
1. Click on the circled pointer at the right of the “Layout your application” section in the top left of the page.



2. A “Select a Template” popup window appears, giving you layout choices for your applications results page. By default, the “Include Sidebar” button is checked. Unchecking it shows you the second popup window below, which gives you layout choices which do *not* include the sidebar at the left of each choice. Note that the layouts are color-coded. The sidebar is green, the text search results area is blue, and visualization widgets are yellow. While there can be only at most one sidebar or text results area, there can be up to six visualization widgets in a single results page layout. Click one of the layouts to select it for your application’s search results page. You can go back and change the layout at any time while you are in Application Builder.



.



3. When you select a layout, the menu closes and your chosen layout is displayed within the box. On the Assemble page, the "Layout your application" image has changed to show your selected layout. The image below shows a selected layout of a sidebar, text area, and two visualization widgets, which have defaulted to a bar chart and pie chart.



.

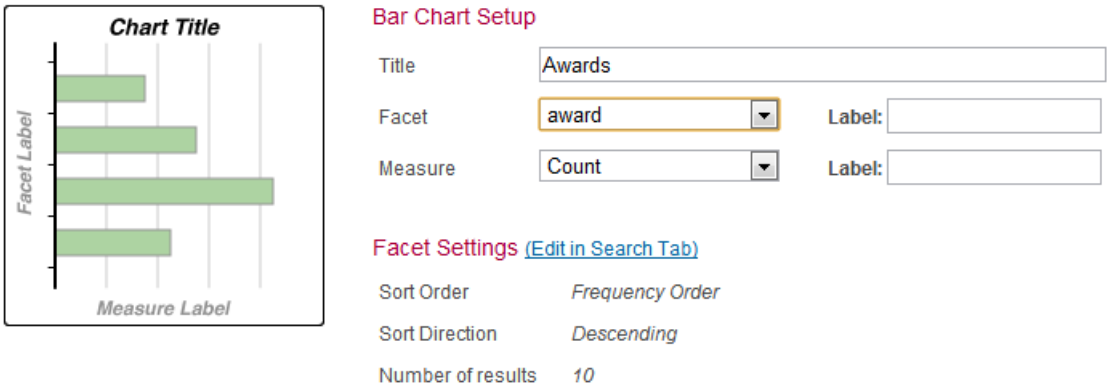
4. To select or change the type of a visualization widget in your layout, first click on the widget you want to change in the "Layout your application" image. In the image above, the leftmost widget has been selected, as indicated by both its outline being blue, rather than black, and thicker than the outline of the other, non-selected, widget. To make the widget "blank" (i.e. no type has been selected for it), click the circled x in the widget's upper right corner.

5. Next, click on one of the five widget types shown in the "Select a Widget" image. Your choices are, in order, Column Chart, Bar Chart, Line Chart, Pie Chart, and Map, as shown in the image below. When you click on one of the widgets, the clicked on widget in your layout image changes to show that widget's icon. Also, the selected widget's name appears under the selection area, along with suggestions of when it is good or bad to use that widget type and any search-specific requirements in order to use that widget type



.

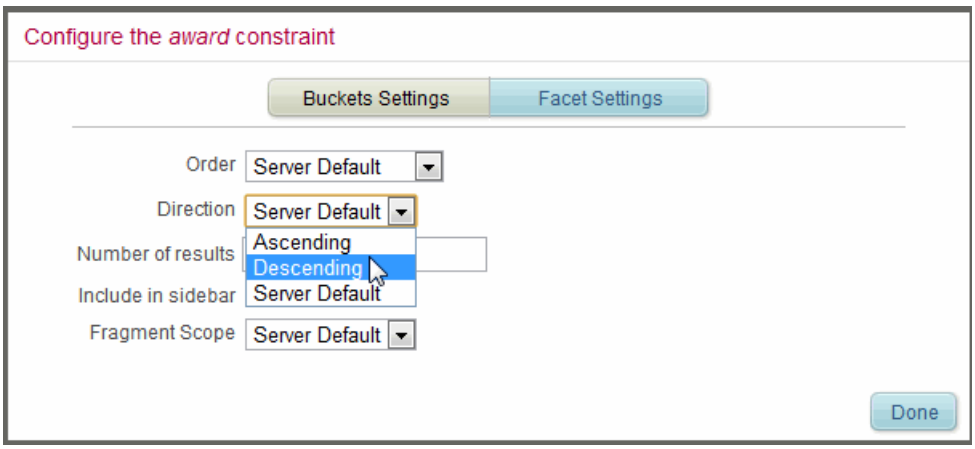
6. Each widget must be associated with a facet, which is set at the bottom of the Assemble page. For example, to have the bar chart represent the number of awards on the data set, set the facet to 'award' and the measure to 'Count':



7. The Facet Settings portion of the Assemble page defines the Sort Order and Sort Direction. These settings can be changed by clicking Edit in Search Tab and clicking Options for the constraint.

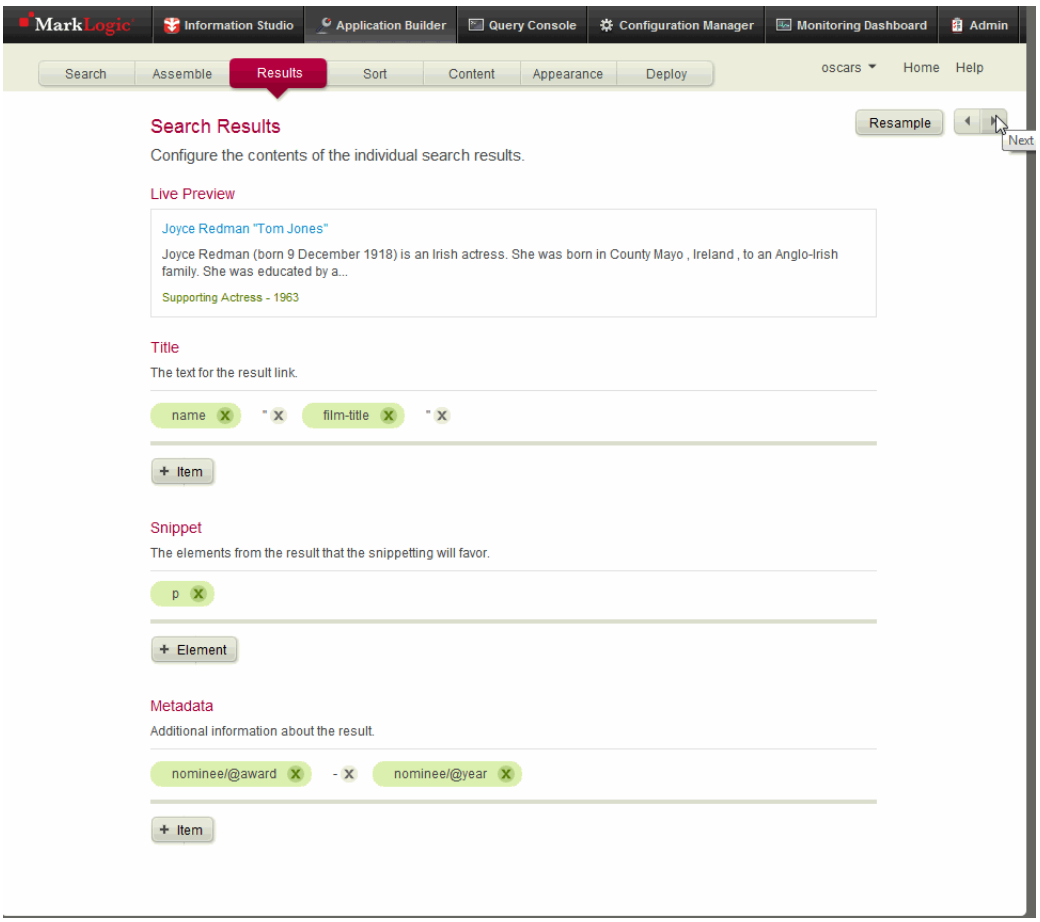


8. In the Configure constraint pop-up window, select Facet Settings. Here you can change the sort order and direction of the data in the widget, as well as the maximum number of results, whether the results are displayed in the sidebar, and the fragment scope.



2.3.4 Results Page

Use the Results page to configure individual search results, including the title, the snippet, and any metadata you want shown.

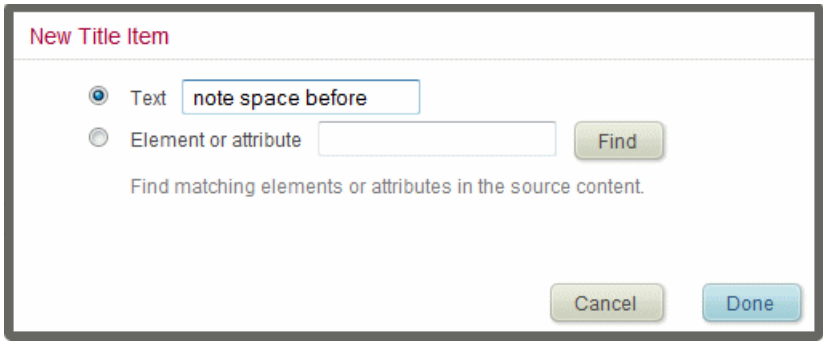


As you make changes, they appear in the preview area. However, not all changes appear, depending on which pieces of content Application Builder has sampled. For example, if you change a preferred element, depending on the content in the sample documents, there might not be a match in those elements and the snippet shown in the preview would therefore not change. To best test the snippetting, do it on the deployed application with different searches and with a more robust sampling of content in your database.

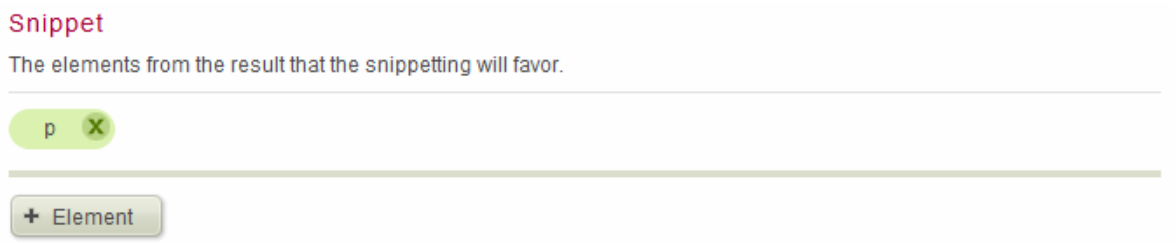
The title is a link to the result content. You can add parts of the content from elements or attributes and literal text, and can concatenate them together in any order. When you add a new item, it displays to the right of any existing items. To delete an item, click its corresponding X button.



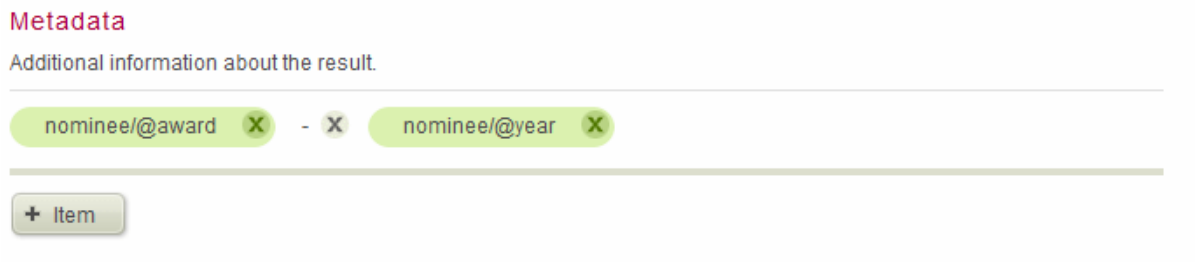
When you add literal text, add a space at the beginning if you want one to appear there.



You can configure snippets by adding one or more elements that the snippetting algorithm will favor. It uses these to try and choose the best parts of the document to show in the snippets, based on the elements you specified as preferred.



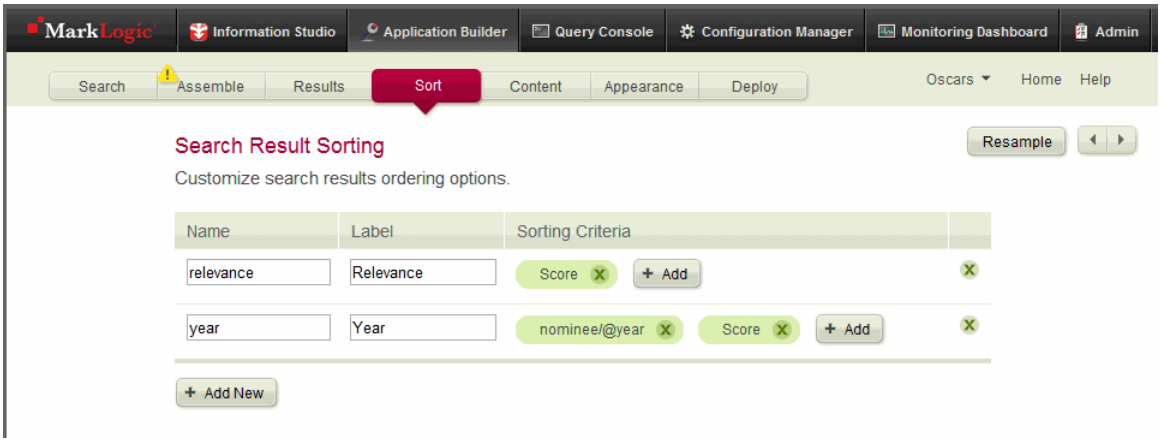
The metadata displays below the snippet, and you configure it the same way as the title.



2.3.5 Sort Page

Use the sorting page to create different sorting operators for your application. By default, searches sort in relevance order (ordered by score).

The sorting page lets you configure other operators for different sorting orders. Each line in the sort table represents a sort operator, and the green rounded buttons represent a key on which to sort.

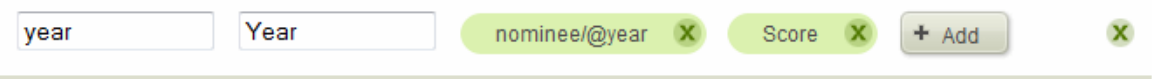


The Sorting page lets you do the following actions:

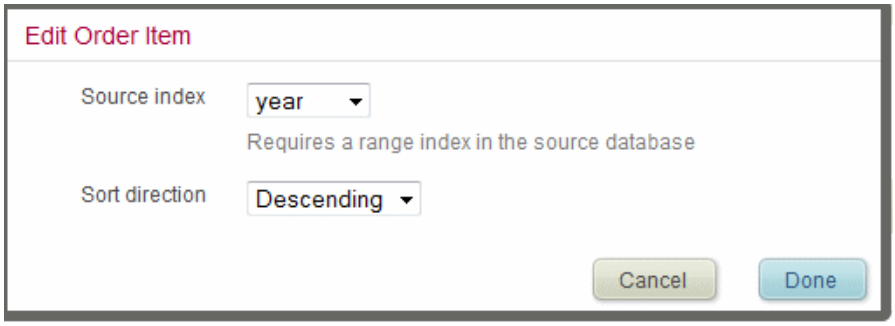
- Specify your results order, by using the name of the sorting criteria with a sort operator. For example, the following specifies sorting by year:

  ```
  sort:year
  ```

- Create different sort orders, by creating range indexes on your desired sort keys. To add a new sort key to an existing sort operator, click Add. The new field is placed after existing sort keys.
- Remove a key by clicking the X in the key's labeled button.
- Remove the sort operator, by clicking the X at the right end of the row.

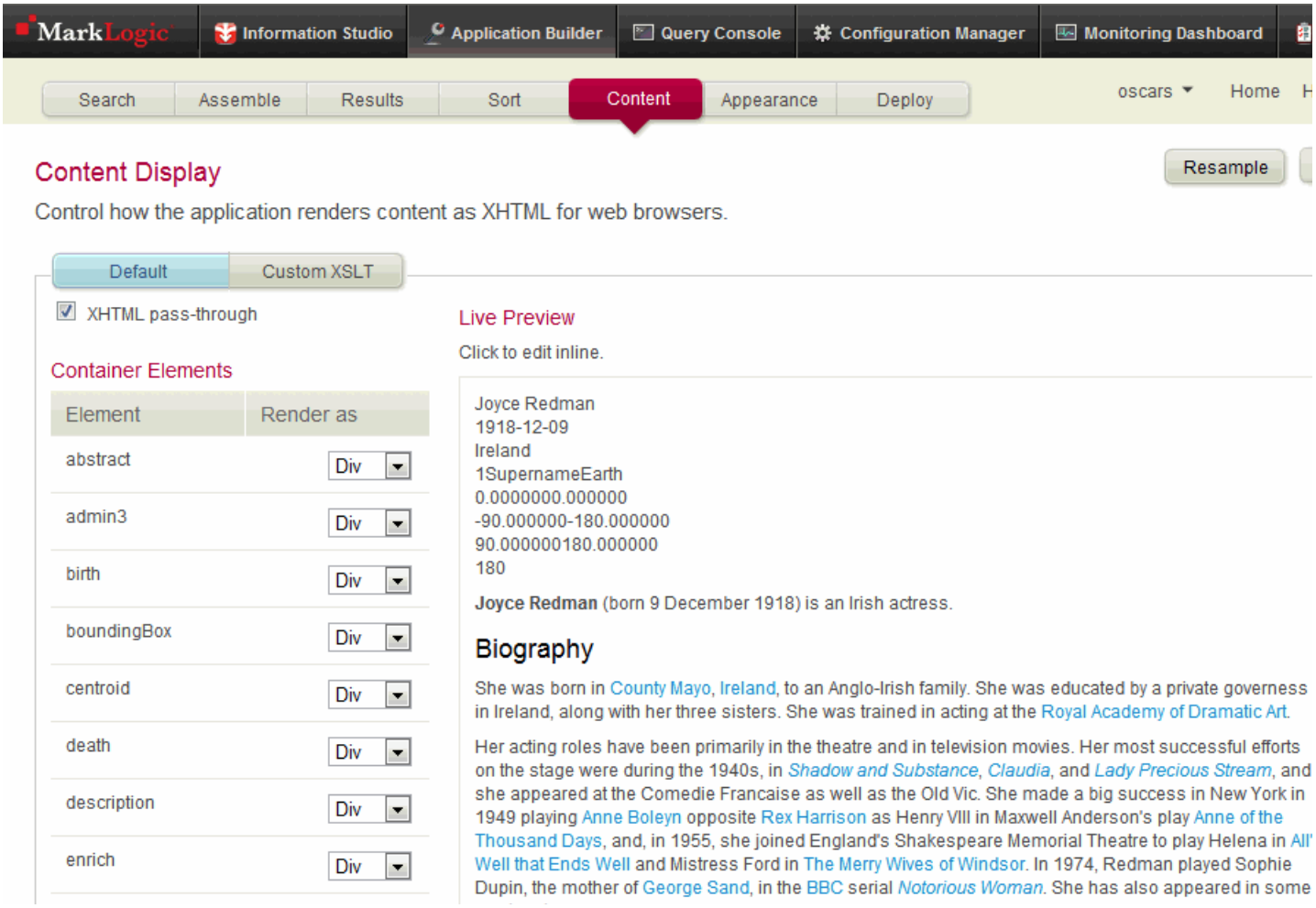


- Specify a custom sort order, add new keys or delete existing keys.
- Edit the source index or sort order of an existing key by clicking the key.

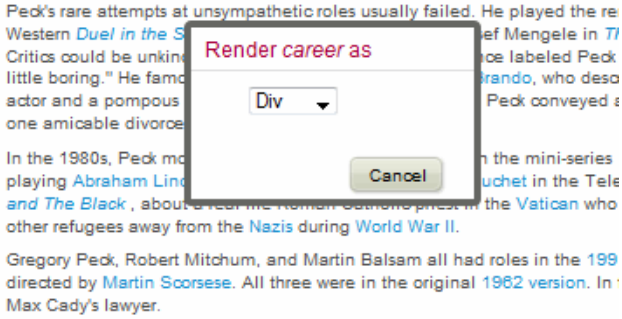


2.3.6 Content Page

The content page lets you control how the search content is rendered to XHTML in the generated application. Application Builder looks at the content and divides it up into *container elements* (elements that have children) and *simple elements* (elements with no children). You can choose None (no rendering), Div, Span, or Para for each container element, and None (no rendering), Div, Span, Header1, Header2, Para, Strong, and Emphasis for each simple element.

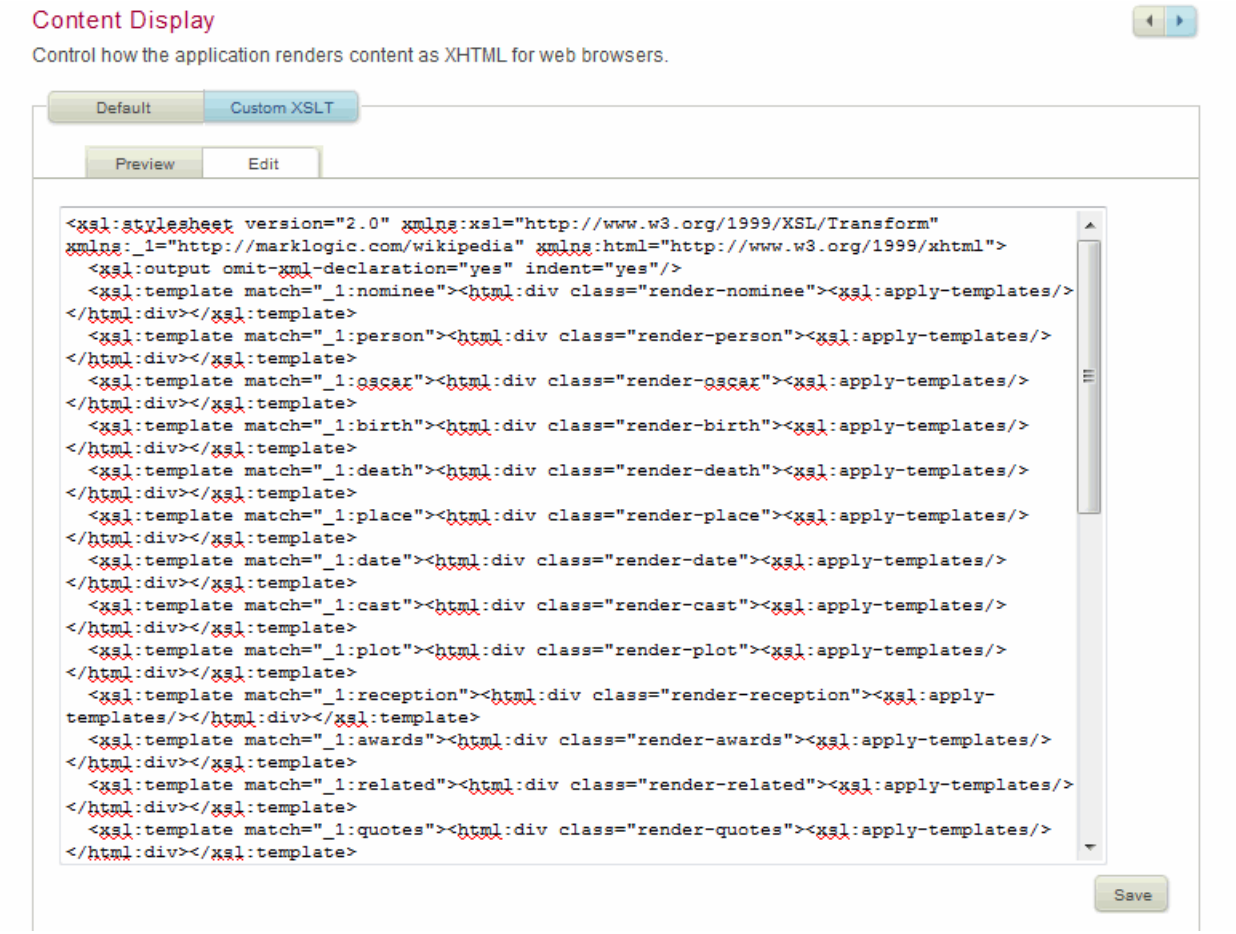


The rendering choices appear in the live preview window. If you click on a part of the preview, a menu appears letting you change the rendering for that element.



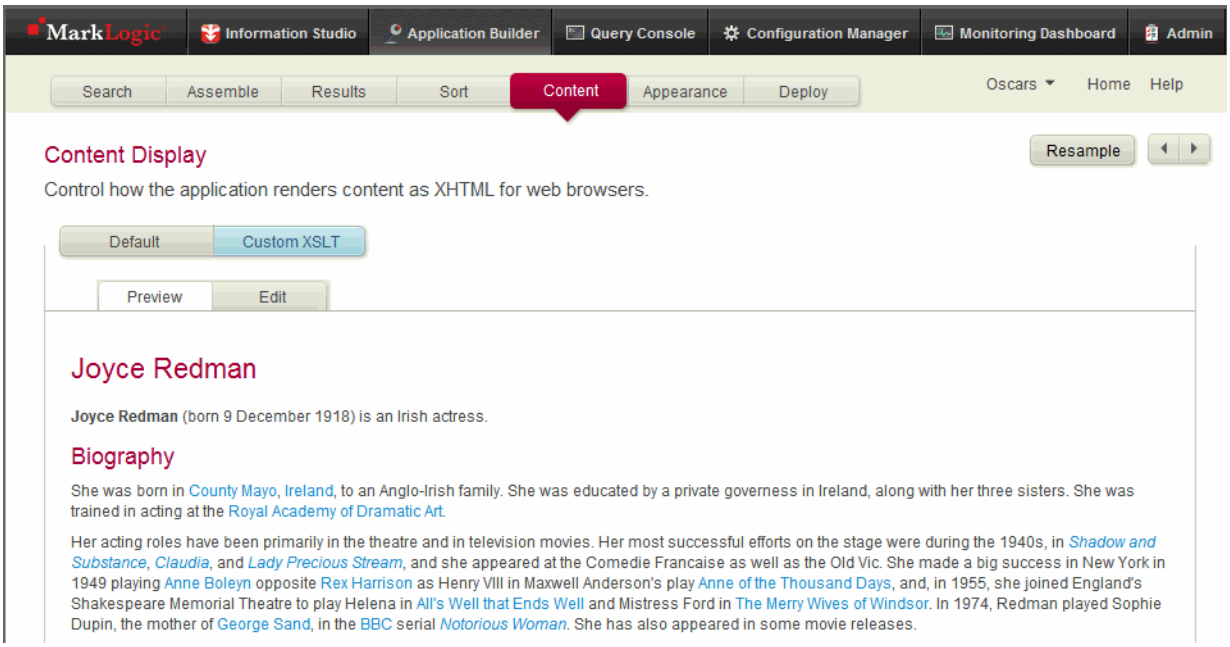
If the XHTML pass-through checkbox is selected, then any content in the XHTML namespace (`http://www.w3.org/1999/xhtml`) is passed through without change to the displayed application.

To display the XSLT stylesheet used to render the content page, select the Custom XSLT tab, then click Edit. Here, you can make more detailed modifications to the XSLT.



Click Preview to preview your changes to the XSLT stylesheet. When you are satisfied with the results, click Edit to return to the XSL stylesheet and click Save.

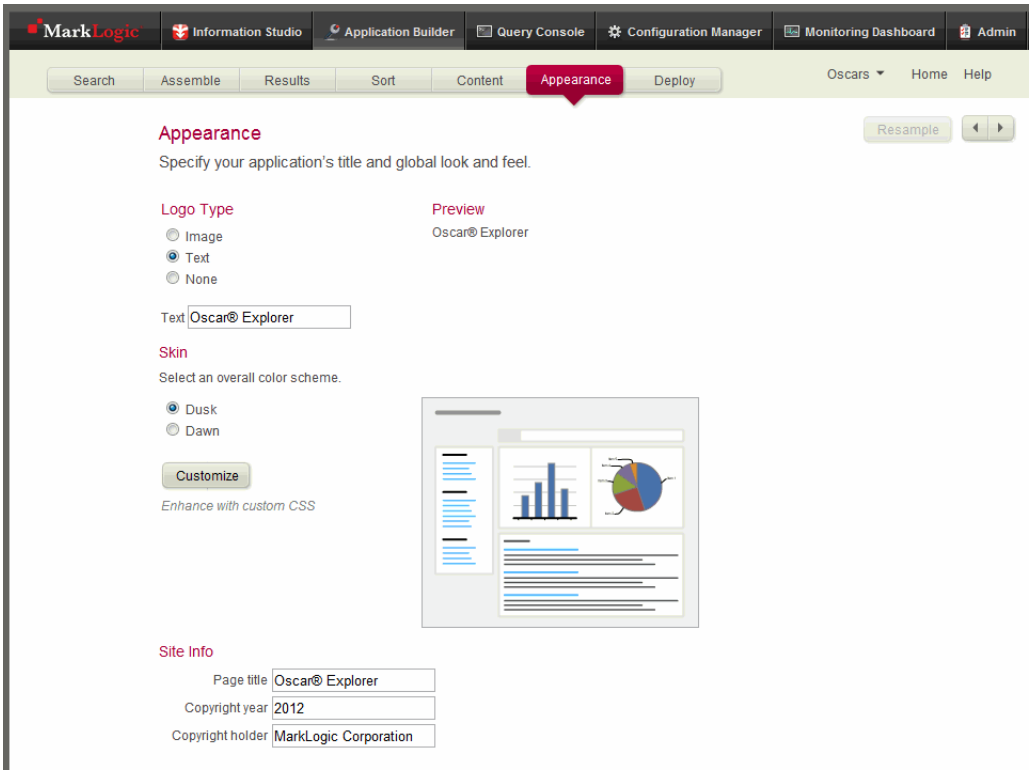
Note: If you navigate away from the Custom XSLT pages without saving your changes, they are lost.



To further customize the rendering, you can customize the application after it is built by using the XSLT files in the `/application/custom` directory as described in “Extending Applications Built With Application Builder” on page 57.

2.3.7 Appearance Page

The Appearance page is where you configure your application's look-and-feel. For example, you can specify a title, add a logo, and add custom CSS code. All options have default values, which you can use if they fit your application.



The following table lists the actions on the Appearance page:

Action	Description
Logo Type	Select an image and a URL for the image, or enter text in place of the logo. This becomes the title or logo of your application's main page.
Skin	Select one of the available CSS skins to define the application's look and feel. Click the Customize button to enter additional CSS code.
Site Info	Metadata about the site. The page title appears in the top bar of the browser window for the application (the `html/head/title` element), and copyright information appears in the footer.

2.3.8 Deploy Page

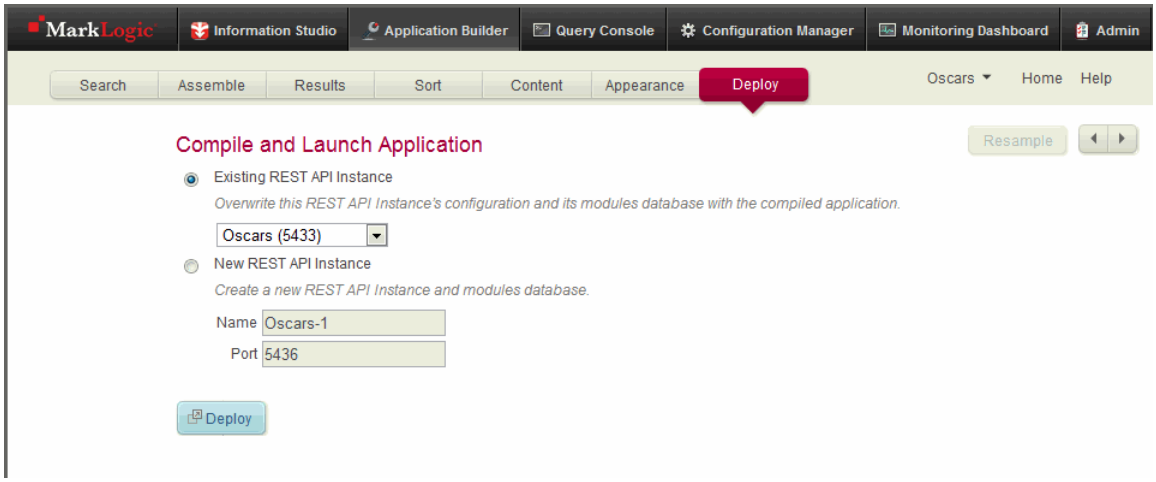
Uset the deploy page to select or configure the App Server where you want to deploy your generated application. When you click the Deploy button, Application Builder performs the following actions:

- Compiles the application based on the application settings.
- Deploys the compiled application code to the modules database of the specified App Server.
- Launches the newly compiled application.

If you use an existing App Server, it must be configured to use a modules database with a root of `/`; Application Builder does not deploy code to an App Server with any other root or with its root on the filesystem.

> **Note:** Each application created by Application Builder must make use of a separate modules database and REST API instance.

If you create a new App Server, Application Builder also creates a modules database for the new App Server, and configures the database with a root of `/`.



> **Note:** When you deploy an application, the newly generated application opens in a new browser window using a URL with the hostname that is stored in the MarkLogic Server configuration files (the result of an `xdmp:host-name` call). If you are running on a laptop computer that is changing networks, it is possible that the hostname is not available on your network, resulting in a 404 or similar error when the application launches (because it is trying to access a server name that is not available on your current network). In these cases, substituting `localhost` for the hostname in the URL should enable the application to launch.

3.0 Controlling Access to Application Builder and to Generated Applications

Application Builder lets you configure and generate search applications without writing code. Both Application Builder and its applications control access by using the MarkLogic Server security model. This chapter describes the security roles needed to run Application Builder and its generated applications, and includes the following sections:

- Predefined Roles for Application Builder
- Permissions on Documents
- Modifying Roles to Meet Your Requirements

3.1 Predefined Roles for Application Builder

Application Builder uses the following three predefined roles:

- app-user Role
- app-builder Role
- app-builder-internal Role

For details about the MarkLogic Server security model and about configuring users and roles, see *Understanding and Using Security Guide* and Security Administration in the *Administrator's Guide*.

3.1.1 app-user Role

The `app-user` role is a minimally privileged role needed to run any generated application. You must grant this role to all users allowed to run a generated application.

3.1.2 app-builder Role

Assign the `app-builder` role to users allowed to run Application Builder and generate applications with Application Builder. Application Builder performs many administrative tasks on MarkLogic Server (for example, creating databases and App Servers), and the `app-builder` role has the necessary privileges for those tasks. However, `app-builder` privileges are minimized to the needed functions and to amped functions, and users with this role can create these resources on MarkLogic Server. Therefore, only trusted users (users who are assumed to be non-hostile, appropriately trained, and follow proper administrative procedures) should be granted the `app-builder` role.

3.1.3 app-builder-internal Role

The `app-builder-internal` role is used by Application Builder to amp certain functions that Application Builder performs. You should not explicitly grant this role to any user; it is only for internal use by Application Builder.

3.2 Permissions on Documents

To let users search generated applications, all documents in that database must have read permission for a role the users have. If you want to permit all users to search everything in the database, you can add a permission to every document for the `app-user` role with the `read` capability. You can give different users different content access by having document permissions based on the level of access control you want to maintain.

For the Oscar sample application, Application Builder adds a `read` permission for the `app-user` role and an `update` permission for the `app-builder` role to all documents in the generated Oscars database.

To add a `read` permission for the `app-user` role and an `update` permission for the `app-builder` role to a document, perform an update to the document:

```
xdmp:document-add-permissions("/example.xml",
  (xdmp:permission("app-user", "read"),
   xdmp:permission("app-builder", "update")))
```

For your applications, you should define a security policy and add the appropriate permissions to the documents in your database to implement it.

For more details on permissions, see Document Permissions in the *Understanding and Using Security Guide*.

3.3 Modifying Roles to Meet Your Requirements

For generated applications, the `app-user` role provides sufficient privileges for any user to run them. However, if you modify the application to perform other actions, you might need to provide additional user privileges.

There are two techniques for providing additional user privileges:

- Add another role with the privileges and assign that role to the application user.
- Modify the existing `app-user` role.

The first one is a safer technique, because it leaves the `app-user` role intact and does not affect any future Applicatoin Builder constructes applications.

In some cases, if you are comfortable with the security implications, it might make sense to modify the `app-user` role by adding other permissions to it.

Warning If you modify the `app-user` role, use caution. All applications generated by Application Builder use the `app-user` role.

4.0 Extending Applications Built With Application Builder

Application Builder generates search applications by using defined templates. This chapter describes how to extend and modify the generated applications and contains the following sections:

- Viewing the Generated Code
- The Custom Directory
- Customizing Applications Generated by Application Builder
- Making Further Modifications to the Application
- Removing Modifications to an Application

4.1 Viewing the Generated Code

When you deploy an application, Application Builder deploys the application's code to an App Server's modules database. The App Server is specified on Application Builder's Deploy page. To view the code, look in the modules database in the `/application/` directory. Use any method you care to look at documents in a database, such as setting up a WebDAV server to the modules database. The entire application is there, including all images, libraries, and other components.

The easiest way to display the application page in your browser and use the 'View Page Source' feature to view the HTML source and navigate to the stylesheet and JavaScript files.

Warning Do not update files in any of these directories. Otherwise, the next time you make changes to your application in App Builder, changes you made to these modules will be overwritten. Instead, modify the files described in "The Custom Directory" on page 58.

The basic files and directories in the `/` directory are shown below:

File	Description
`index.html`	File that defines the default HTML page for the application. This is what loads in the browser when the user loads the application.
`application/app.xsl`	File that defines the XSLT used to render all pages in the application, except result pages. .
`application/app-content.xsl`	File that defines the XSLT stylesheet used to render the content page described in "Content Page" on page 50.
`application/app.css`	File that defines the CSS specified in the Customize field on the Appearance page.

File	Description
`application/app.js`	File that is the instantiation of the widget library for all the widgets on the page.
`application/skin.css`	File that defines the CSS specific to the chosen skin (dawn/dusk)
`application/skin.js`	File that applies a highcharts global theme object to the page, according to the selected skin.
`application/constraint`	Directory containing the XQuery extension needed for map widgets.
`application/css`	Directory containing the CSS definitions that define the layout for the application theme.
`application/custom`	Directory containing all the files that can be used to extend and modify the application.
`application/images`	Directory containing images used by the application.
`application/lib`	Directory containing the library JavaScript files for the widgets and visualizations.
`application/skins`	Directory containing the Dusk and Dawn skins defined in the Appearance page.

4.2 The Custom Directory

Use the files in the `/application/custom` directory to override the generated application files and preserve your changes when the application is redeployed by Application Builder.

The files in the `/application/custom` directory are shown below:

File	Description
`app.xsl`	Contains the XSLT used to render all pages in the application, except result pages. This file overrides the generated `/index.html` file. For an example, see “Customizing the Footer” on page 63.
`content.xsl`	Contains the XSLT used to render result pages. This file overrides the XSLT stylesheet used to render the content page described in “Content Page” on page 50. For an example, see “Customizing the Results Page” on page 65.

File	Description
`rewrite.xml`	Contains URIs for additional content. This file extends the `MarkLogic/Modules/MarkLogic/rest-api/endpoints/config.xqy` file. The URI of the new content must begin with `/content/`.
`app.js`	JavaScripts added to this file extend (but do not override) the JavaScript generated by Application Builder. For an example, see "Adding Custom JavaScript" on page 67.
`app.css`	Styles added to this file override and extend the styles generated by Application Builder. For an example, see "Customizing the Content Display" on page 64.
`app-config.js`	JavaScripts added to this file extend the configuration of your application and widgets. For an example, see "Adding Custom JavaScript" on page 67.

4.3 Customizing Applications Generated by Application Builder

This section describes how to customize an Application Builder generated application using Application Builder, and contains the following parts:

- Basic Design Pattern
- Accessing the Code in the Custom Directory
- Road Map for Application Page Objects
- Customizing the Footer
- Customizing the Content Display
- Customizing the Results Page
- Adding Custom JavaScript

Note: To modify the Application Builder generated applications, make your changes to the files located in the `/application/custom` directory, as described in "The Custom Directory" on page 58.

4.3.1 Basic Design Pattern

Applications generated from Application Builder are designed to be extensible. The basic design pattern to modify a generated application is as follows:

- Display the application page in your browser.
- Use the browser 'Display Page Source' feature to view the page source.
- Locate the stylesheets, JavaScripts, or HTML to be modified in the page source.
- Copy the code from the page source to the respective file in the custom directory. For example, if you want to change a style definition in the `application/skin.css` file, copy the definition to the `custom/app.css` file.
- Modify the code in the custom file to override the code in the generated file.
- Refresh the application page in your browser to see the results.

4.3.2 Accessing the Code in the Custom Directory

To modify a generated application, you must gain access to the files in the `/application/custom` directory in the application's modules database. One common way is to create a WebDAV App Server to the modules database, and then use it to access and modify the code. Many code development tools support WebDAV for editing. For details on WebDAV Servers, see WebDAV Servers in the *Administrator's Guide*.

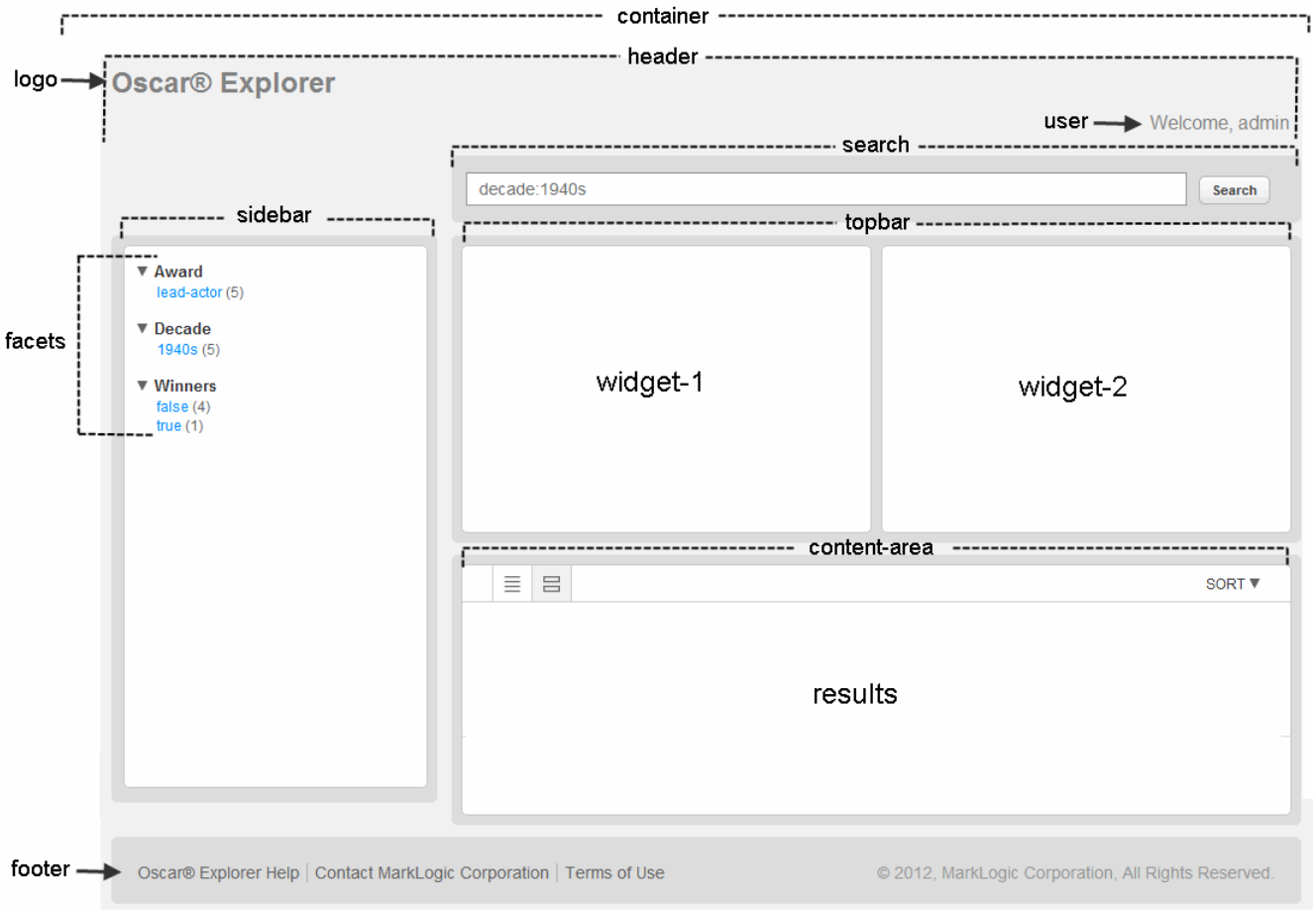
Note: If you are using oXygenXML, access the modules database through WebDAV.

Warning Log in as a user with permissions for `app-builder`. Do not use WebDav logged in as the `admin` user, as this will create files readable only by `admin`.

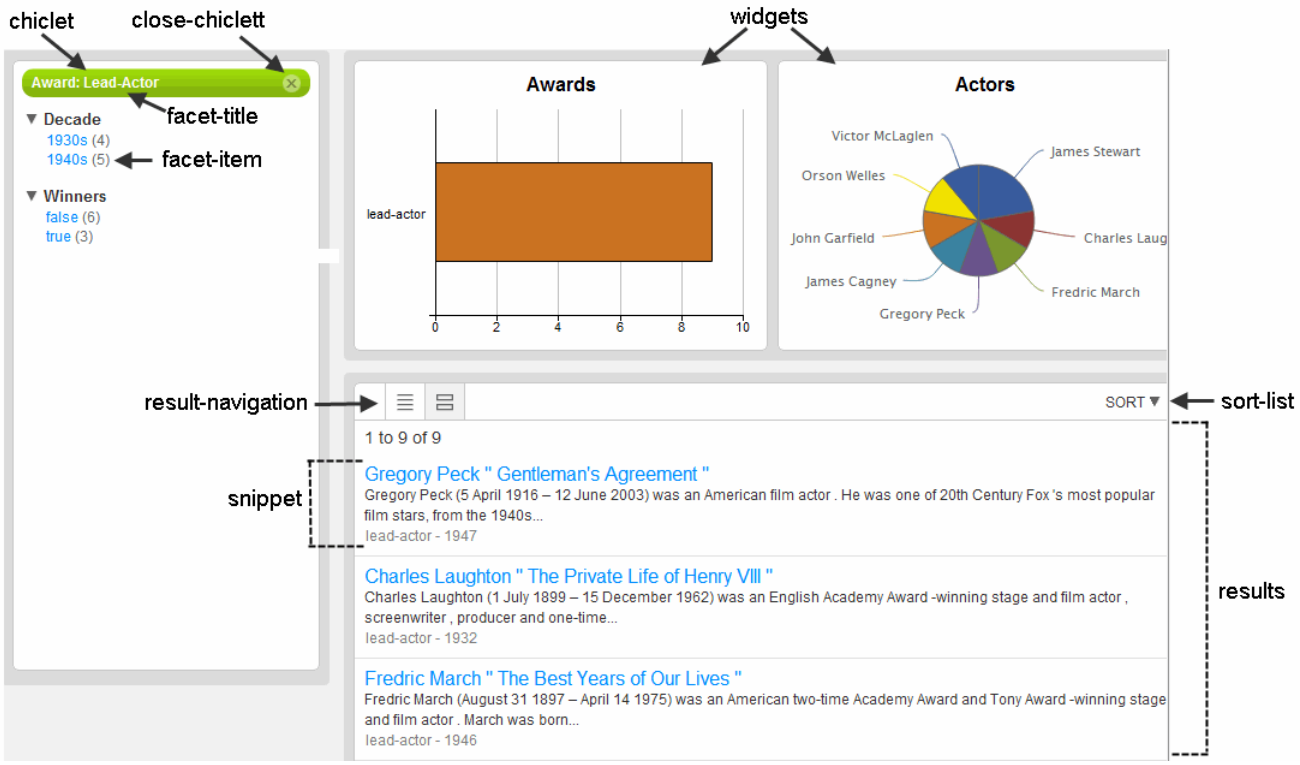
4.3.3 Road Map for Application Page Objects

This section shows the basic objects that control the specific sections of the application pages. All of these can be modified in the `custom` files.

The objects below are common to the search and navigation page.



The content and sidebar portions of the search page are controlled by the following objects:



4.3.4 Customizing the Footer

To modify the footer in your generated application, find the `footer` definition in the `/index.html` file:

```
<div id="footer" class="footer">
  <p>
    <span class="copyright">
      © 2012, MarkLogic Corporation, All Rights Reserved.
    </span>
    <a href="/content/help">Oscar® Explorer Help</a>
    <span class="pipe"></span>
    <a href="/content/contact">Contact MarkLogic Corporation</a>
    <span class="pipe"></span>
    <a href="/content/terms">Terms of Use</a>
  </p>
</div>
```

For example, to change the copyright portion of the footer from that generated by Application Builder:

```
© 2012, MarkLogic Corporation, All Rights Reserved.
```

to:

```
© 2012, My Company, All Rights Reserved.
```

Add the following `xsl:template` element to the `/custom/app.xsl` file:

```
<xsl:template match="*:span[@class eq 'copyright']">
  <span class="copyright">
    © 2012, My Company, All Rights Reserved.
  </span>
</xsl:template>
```

When you refresh your application page, it now uses the new footer, with the change persisting after redeploying the application in Application Builder.

4.3.5 Customizing the Content Display

To modify the content display in your generated application, locate the code in the `application/app.css` or `application/skin.css` file. For example, the following define the body background and the chiclet:

```
body {
  color: rgb(50,50,50);
  font-family: "Lucida Grande", Helvetica, Arial, sans-serif;
  background: blue;
}

#sidebar .chiclet {
  color: white;
  text-transform:capitalize;
  background: url('images/facet_green.png') left top repeat-x;
}
```

To change the body background to blue, the snippet text to green, and the chiclet text to yellow, add the following to the `custom/app.css` file:

```
body {
  color: green;
  font-family: "Lucida Grande", Helvetica, Arial, sans-serif;
  background: blue;
}

#sidebar .chiclet {
  color: yellow;
  text-transform:capitalize;
  background: url('../images/facet_green.png') left top repeat-x;
}
```

Note: The relative pathname to the chiclet background URL must be modified to point to the `images` directory from the `custom` directory.

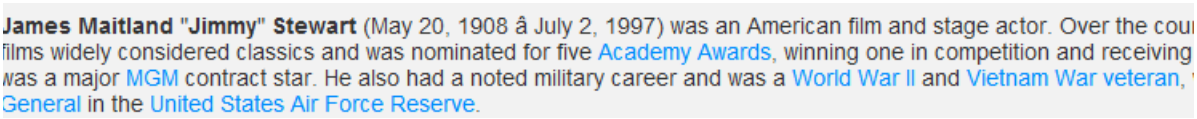
4.3.6 Customizing the Results Page

There are several ways to modify the results page, including the following:

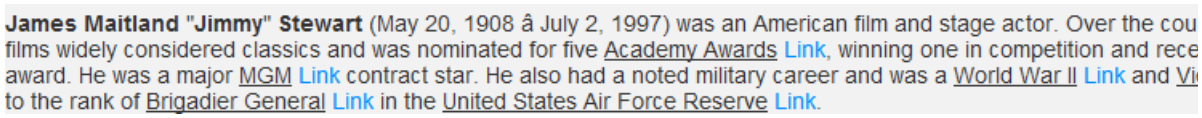
- Adding to the content.xsl Stylesheet
- Adding More/Less Links

4.3.6.1 Adding to the content.xsl Stylesheet

You can modify the look of the results page by adding XSLT to the `content.xsl` file. This overrides the XSLT stylesheet generated by Application Builder. For example, you want to change the links in the Oscars application to the Wikipedia pages from this:



to look like this:

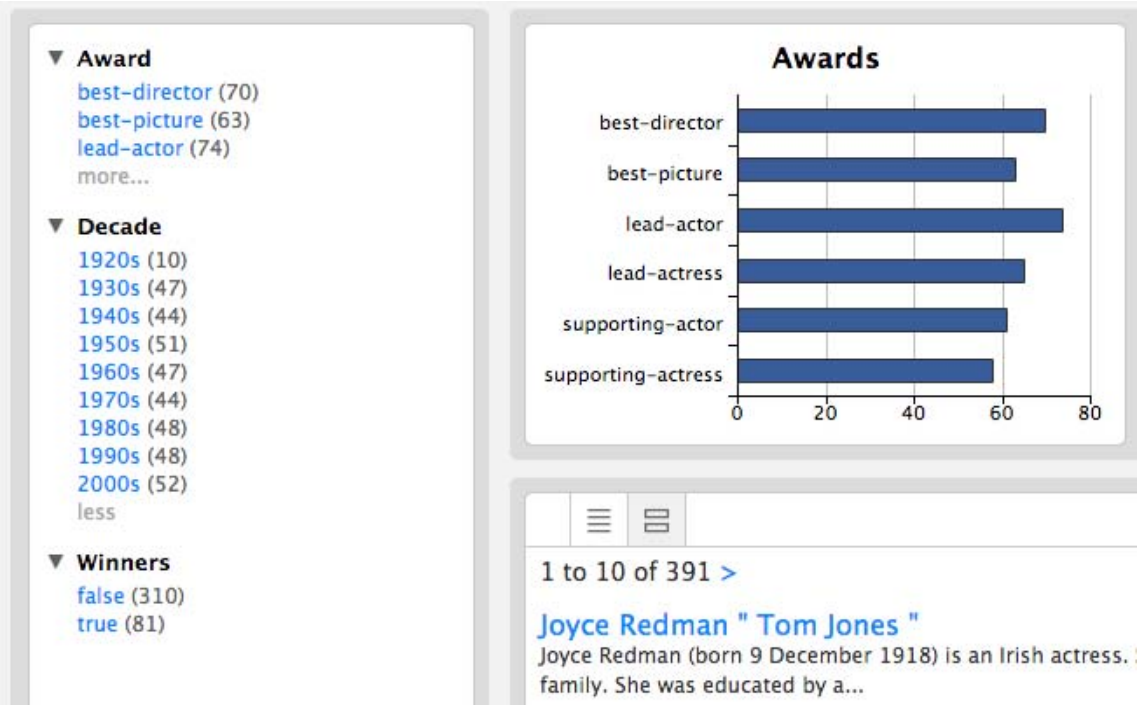


You can add the following template to the `content.xsl` file:

```
<xsl:template match="xhtml:a">
  <xsl:choose>
    <xsl:when test="starts-with(@href, 'http://example.com/')">
    </xsl:when>
    <xsl:otherwise>
      <span style="text-decoration: underline;">
        <xsl:apply-templates/>
      </span>
      <a href="{@href}" title="External link"> Link</a>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

4.3.6.2 Adding More/Less Links

You can add more/less links to make your facets show more or less content, as in the following screenshot.



To add these links, specify a `limit` property in the `sidebar_config` variable `custom/app-config.js` file. The limit property goes in the JSON `sidebar_config` content as in the following code snippet:

```
var sidebar_config=
{"limit": 3,
 "facets": ...
...
}
```

When enabled, a user clicks the “more...” link to show more results, and the “less” link to show the number of results set in the `limit` parameter. Note that you can still control the total number of facet items shown in the facet lists in the Facet Settings located under the Application Builder Search page.

4.3.7 Adding Custom JavaScript

The following are simple examples of modifying the behavior of existing JavaScript and adding new JavaScript to an application.

For example, you want to disable to widget-2 widget (In the sample Oscars applicaiton, this is the pie chart widget, by default) from refreshing the snippets displayed in the results panel. To do this, you can add the following JavaScript code to the `custom/app.js` file to disable the widget's selection behavior from updating other charts / results:

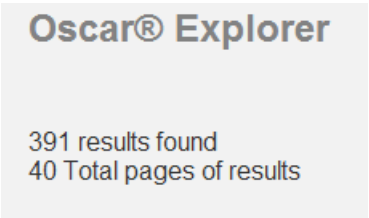
```
$('#widget-2').unbind('selection');
```

An example of adding new JavaScript is as follows. The first line writes a new empty widget container into the top of the page. It then creates a widget on that container which renders the total number of results along with the total number of pages of results:

```
$('#content').before('<div id="totalCount" class="widget"></div>');

var widget = ML.createWidget($('#totalCount'), function (data) {
   $('#totalCount').html(data.total + " results found<br />" +
   Math.ceil(data.total / 10) + " Total pages of results");
});
```

When you refresh the search page in your browser, the following should appear in the upper-left portion of the page:



You can modify the widgets and other objects created by means of the JavaScript generated by Application Builder by editing the `custom/app-config.js` file. The `custom/app-config.js` file contains a number of commented examples that make use of the `.extend` utility to merge a new configuration into an existing configuration.

For example, one of the examples shows how to change the title of the pie chart widget:

```
// example of changing title for pie chart
//$.extend( pie_config_2, {title: "People Title"})
```

To enable this modification, simply remove the comment:

```
$.extend( pie_config_2, {title: "People Title"})
```

And observe the results in your deployed application.

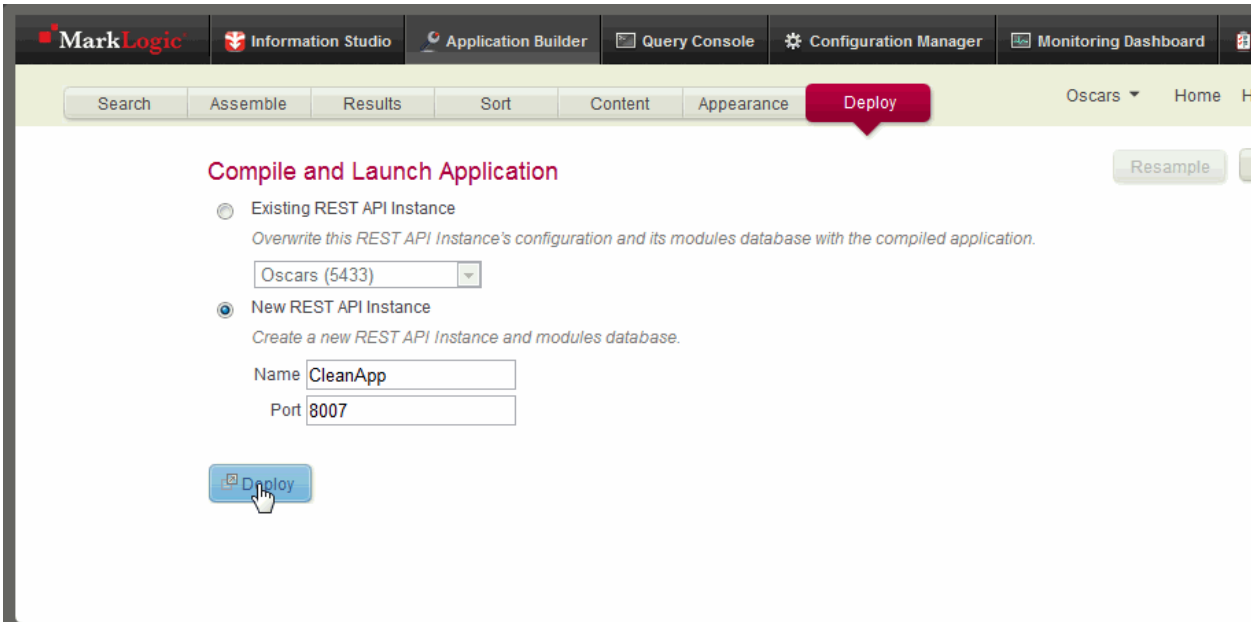
4.4 Making Further Modifications to the Application

As described in “Viewing the Generated Code” on page 57, the generated application consists of HTML, XSLT, JavaScript, and CSS code that you can modify in any way you want. The example modifications shown in this chapter are simply an introduction that you can use as a starting point for making more substantial modifications.

4.5 Removing Modifications to an Application

You can remove all of the modifications to an application in the `/application/custom` directory by redeploying the application to another App Server.

For example, to redeploy an existing application to new server CleanApp, navigate to the Deploy page, select New App Server, enter the new name and port, and click the Deploy button:



5.0 Technical Support

MarkLogic provides technical support according to the terms detailed in your Software License Agreement or End User License Agreement.

We invite you to visit our support website at http://help.marklogic.com to access information on known and fixed issues, knowledge base articles, and more. For licensed customers with an active maintenance contract, see the Support Handbook for instructions on registering support contacts and on working with the MarkLogic Technical Support team.

Complete product documentation, the latest product release downloads, and other useful information is available for all developers at http://developer.marklogic.com. For general questions, join the general discussion mailing list, open to all MarkLogic developers.

6.0 Copyright

MarkLogic Server 8.0 and supporting products.

NOTICE

Copyright © 2018 MarkLogic Corporation.

This technology is protected by one or more U.S. Patents 7,127,469, 7,171,404, 7,756,858, 7,962,474, 8,935,267, 8,892,599 and 9,092,507.

All MarkLogic software products are protected by United States and international copyright, patent and other intellectual property laws, and incorporate certain third party libraries and components which are subject to the attributions, terms, conditions and disclaimers found at http://docs.marklogic.com/guide/copyright/legal.

MarkLogic and the MarkLogic logo are trademarks or registered trademarks of MarkLogic Corporation in the United States and other countries. All other trademarks are property of their respective owners.

For all copyright notices, including third-party copyright notices, see the Combined Product Notices for your version of MarkLogic.