
MarkLogic Server

Application Builder Developer's Guide

Release 4.2
October, 2010

Last Revised: 4.2-1, October, 2010

Copyright

© Copyright 2002-2012 by MarkLogic Corporation. All rights reserved worldwide.

This Material is confidential and is protected under your license agreement.

Excel and PowerPoint are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. This document is an independent publication of MarkLogic Corporation and is not affiliated with, nor has it been authorized, sponsored or otherwise approved by Microsoft Corporation.

Contains LinguistX, from Inxight Software, Inc. Copyright © 1996-2006. All rights reserved. www.inxight.com.

Antenna House OfficeHTML Copyright © 2000-2008 Antenna House, Inc. All rights reserved.

Argus Copyright ©1999-2008 Icenit Technology Ltd. All rights reserved.

Contains Rosette Linguistics Platform 6.0 from Basis Technology Corporation, Copyright © 2004-2008 Basis Technology Corporation. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>) Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved. Copyright © 1998-2001 The OpenSSL Project. All rights reserved.

Contains software derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Copyright © 1991-1992, RSA Data Security, Inc. Created 1991. All rights reserved.

Contains ICU with the following copyright and permission notice:

Copyright © 1995-2010 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Table of Contents

Application Builder Developer's Guide

Copyright	2
1.0 Application Builder Quick Start	5
1.1 Overview of Application Builder	5
1.2 Setting Up and Starting Application Services	6
1.2.1 Clean Installation	6
1.2.2 Upgrade Installation	7
1.3 Building the Oscars Sample Application	9
1.4 Using the Oscars Sample Application	12
1.4.1 Keyword Searching, Search Suggestions, and Parsing	12
1.4.2 Browsing with Facets	13
1.4.3 Search Result Page	14
1.4.4 Displaying Content Detail	14
1.5 Using Application Builder to Modify the Oscars Sample Application	14
2.0 Creating a Search Application With Application Builder	17
2.1 Starting Application Services	17
2.2 Navigating Through the User Interface	17
2.3 Page-By-Page Walkthrough	19
2.3.1 Application Builder section of the Application Services Page	20
2.3.2 Appearance Page	22
2.3.3 Search Page	23
2.3.3.1 Add/Modify Range Constraints	24
2.3.3.2 Add/Modify Value Constraints	27
2.3.3.3 Add/Modify Word Constraint	28
2.3.3.4 Add/Modify Collection Constraint	29
2.3.3.5 Modifying Search Options	30
2.3.4 Sorting Page	32
2.3.5 Results Page	33
2.3.6 Content Page	35
2.3.7 Deploy Page	38
3.0 Controlling Access to Application Builder and to Generated Applications	39
3.1 Predefined Roles for Application Builder	39
3.1.1 app-user Role	39
3.1.2 app-builder Role	39
3.1.3 app-builder-internal Role	39

3.2	Permissions on Documents	40
3.3	Adding Other Roles or Modifying the app-user Role or to Meet Your Requirements 40	
4.0	Extending Applications Built With Application Builder	41
4.1	Finding the Generated Code	41
4.2	The Custom Directory	44
4.3	Customizing Applications Generated by Application Builder	45
4.3.1	Basic Design Pattern	46
4.3.2	Accessing the Code in the Custom Directory	46
4.3.3	Road Map for Application Page Functions	47
4.3.4	Customizing the Footer	50
4.3.5	Customizing the Content Display	51
4.3.6	Customizing the Detail Page	52
4.3.7	Customizing the Generated Search Options Node	53
4.3.7.1	Adding a searchable-expression Option	53
4.3.7.2	Modifying the transform-results Option	54
4.4	Making Further Modifications to the Application	54
4.5	Removing Modifications to an Application	55
5.0	Technical Support	56

1.0 Application Builder Quick Start

This chapter gets you started using Application Builder. Application Builder is a browser-based application that allows you to create a fully functional search and analytics application very quickly. The generated application is designed to be useful as-is, and is also designed to be extremely extensible, allowing arbitrary customization to the generated application. This chapter gets you started using Application Builder and includes the following sections.

- [Overview of Application Builder](#)
- [Setting Up and Starting Application Services](#)
- [Building the Oscars Sample Application](#)
- [Using the Oscars Sample Application](#)
- [Using Application Builder to Modify the Oscars Sample Application](#)

1.1 Overview of Application Builder

Application Builder generates custom search applications based on your own content. You can define many aspects of the application, including facets, details of the search result page, and item rendering to control content display. The application that Application Builder generates is a full-featured search application with many high-end search features such as search box with a Google-style search grammar, faceted navigation, and search suggestions. Application Builder generates and deploys your customized application. The generated application is a MarkLogic Server application, so it is designed to scale to huge database sizes and still be fast.

No coding is required to create an application with Application Builder. The user interface is simple, yet allows you to build some very complex components such as facets. The generated application is an XQuery application that uses the Search API. It is designed to either be used as-generated with no additional coding required. It also has extensive hooks so that you can customize it with your own XQuery code, if you so choose.

Building an application is typically an iterative process. A representative set of your content must be loaded in a database, and it should have any needed indexes set up. If your content is not complete or not completely indexed, that is OK, you can still use Application Builder to generate an application and modify the Application Builder application as you modify your content.

You can use Application Builder to generate a sample application based on data about the Oscar awards. For details on the sample application, including step-by-step instructions on creating it with Application Builder, see “Building the Oscars Sample Application” on page 9 and “Using the Oscars Sample Application” on page 12.

1.2 Setting Up and Starting Application Services

Application Builder is bundled with MarkLogic Server Application Services. On a fresh installation of 4.2, Application Builder is pre-configured and ready to use. On an upgrade installation, you have to create an App Server and a database before you can use it. This section describes the steps needed in each case to start Application Builder, and includes the following parts:

- [Clean Installation](#)
- [Upgrade Installation](#)

1.2.1 Clean Installation

When you install MarkLogic Server for the first time, the installation process creates an HTTP App Server, named `App-Services`, on port 8002 for Application Services, as well as a database named `App-Services` to store the Application Builder application documents. For details on installing MarkLogic Server, see [Installing MarkLogic Server](#) in the *Installation Guide*.

Note: If you are upgrading a server with an earlier version of Application Builder, the `App-Builder` database will be renamed to `App-Services`, but all of your application data will remain intact.

To start Application Services, open a browser and go to port 8002 of your server. For example, if your browser is running on the same machine as MarkLogic Server, open the following URL in a browser:

```
http://localhost:8002
```

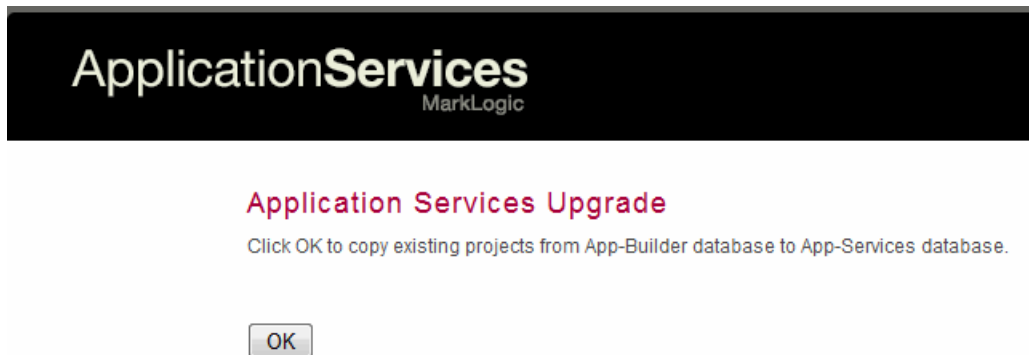
When it prompts you for a username and password, enter user credentials for a user that has either the `admin` role or the `app-builder` role. For details on the roles that Application Builder uses, see “Predefined Roles for Application Builder” on page 39.

1.2.2 Upgrade Installation

If you install MarkLogic Server as an upgrade installation, and if the previous installation did not have Application Builder configured (for example, if you are upgrading from 4.0 or 4.1), then the installation does not create the forest, database, and App Server that Application Builder requires.

The upgrade procedure from either 4.0 or 4.1 to 4.2 is similar:

- If you are upgrading from either 4.0 to 4.2, then the upgrade installation will create a forest and database, named `App-Services`.
- If you are upgrading from 4.1 to 4.2, and if the previous installation had Application Builder configured, then the upgrade installation will rename the `App-Builder` App Server to `App-Services` and will create a forest and database, named `App-Services`. Click OK when prompted to move all of the data previously in your `App-Builder` database into the `App-Services` database. You can then delete the `App-Builder` database and forest.



- If you are upgrading from 4.1 to 4.2, and if the previous installation did not have Application Builder configured, then the upgrade installation will not create a forest and database, named `App-Services`.

If the upgrade created an `App-Services` forest and database, you must create an HTTP App Server, with the following settings (all other setting can keep their default values):

Field	Setting
Name	App-Services
Root	Apps/
Port	8002
Database	App-Services
Error Handler	/error.xqy

The following is what the configuration for the App-Services server should look like in the Admin Interface. For more information about administration of MarkLogic Server, see the *Administrator's Guide*.

The screenshot displays the 'HTTP Server: App-Services' configuration window in the MarkLogic Admin Interface. The window has a title bar with 'ok' and 'cancel' buttons. Below the title bar, there is a description: 'http server -- A HTTP server specification.' and 'delete' and 'disable' buttons. The configuration fields are as follows:

Field	Value	Description
server name	App-Services	The server name.
root	Apps/	The root document directory pathname.
port*	8002	The server socket bind internet port number.
modules	(file system)	The database that contains application modules.
database	App-Services	The database name.
error handler	/error.xqy	The script that handles 400 and 500 errors for this server.

1.3 Building the Oscars Sample Application

Application Builder allows you to configure and generate applications, and it includes a template to build a sample application based on Oscar awards data from Wikipedia. Building the Oscars sample application is a very simple process of running through the Application Builder wizard.

Note: If you plan to deploy the full data set for the Oscars sample application on a 32-bit Microsoft Windows XP system, you should have at least 3GB of system memory.

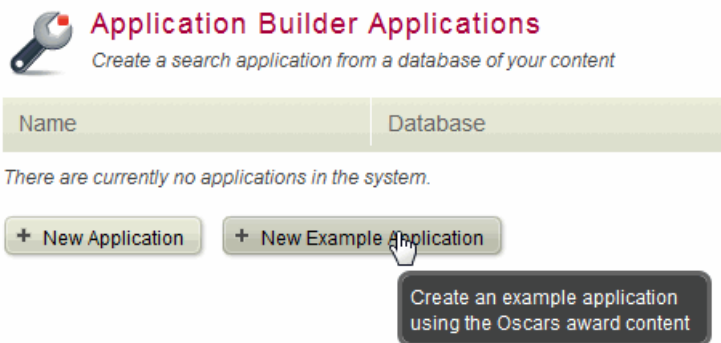
To build the Oscars sample application, perform the following:

1. Start Application Builder in a browser by going to the following URL:

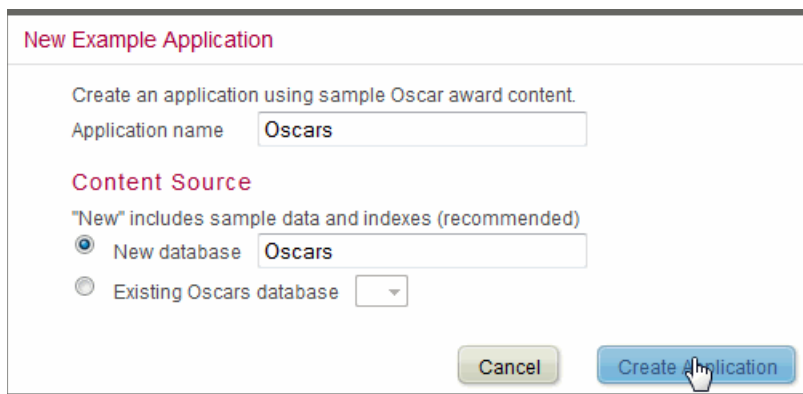
`http://localhost:8002`

If MarkLogic Server is installed on a different host or your App Server is on a different port, substitute the appropriate values for the ones above.

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



3. Enter a name for the application (Oscars) and select New database. Enter a name for the database (Oscars) and click Create Application.



- Application Builder creates an Oscars App Server, forest, and database.

The screenshot shows the 'Appearance' page of the Application Builder interface. The header includes the 'ApplicationBuilder' logo and 'MarkLogic Application Services'. A navigation bar contains tabs: 'Appearance' (highlighted), 'Search', 'Sorting', 'Results', 'Content', and 'Deploy'. On the right, there are links for 'Oscars', 'Home', and 'Help'. The main content area is titled 'Appearance' and includes the instruction 'Specify your application's title and global look and feel.' Below this, there are two sections: 'Logo Type' with radio buttons for 'Image', 'Text' (selected), and 'None'; and 'Preview' showing 'Oscar® Explorer'. A text input field labeled 'Text' contains the value 'Oscar® Explorer'.

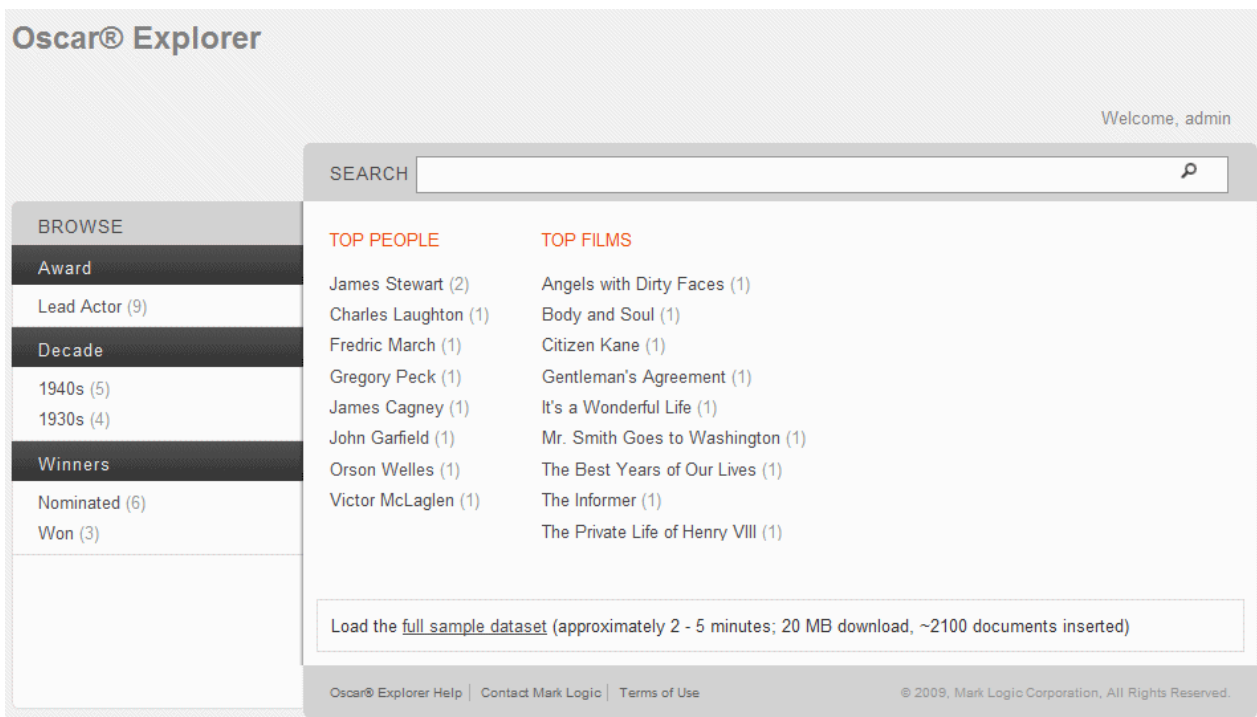
- On the Appearance page, you can change any of the settings you want, or just accept the defaults.
- Click the Deploy tab. The Compile and Launch Application page appears.

The screenshot shows the 'Compile and Launch Application' page. The header is the same as the previous page. The navigation bar now has the 'Deploy' tab highlighted. The main content area is titled 'Compile and Launch Application' and includes the instruction 'Specify the App Server on which to deploy your configured application.' There are two radio button options: 'Existing App Server' (with a dropdown menu) and 'New App Server' (selected). Below 'New App Server', there are input fields for 'Name' (containing 'Oscars') and 'Port' (containing '8003'). A 'Deploy' button is located below these fields. At the bottom, there are 'Back', 'Resample', and 'Next' buttons.

- On the Compile and Launch Application page, select the New App Server button. Accept the default values or provide the App Server with a name and port number. You can only select existing App Server if there is already an App Server configured for this application.
- Click the Deploy button and confirm. Application Builder opens a new window, creates and configures the new App Server, and launches the newly created application in the new window. This may take a short while.
- When the new application is complete, it prompts you to log in. Enter a username and password and click OK. For details about controlling access to the built application, see “Controlling Access to Application Builder and to Generated Applications” on page 39.

Note: When you deploy an application, it opens the newly generated application in a new browser window using a URL with the hostname that is stored in the MarkLogic Server configuration files (the result of an `xamp:host-name` call). If you are running on a laptop computer that is changing networks, it is possible that the hostname will 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 that is not available on your current network). In these cases, substituting `localhost` for the hostname in the URL should allow the application to launch.

10. The Oscars sample application is created with a few sample documents. You can enter search terms or click on the browse links to narrow the results displayed.



11. If you want to load the full content set, click the Load Full Sample Dataset link toward the bottom of the page. The full dataset is around 20 MB to download. After clicking the link, the data will download and load automatically (internet connection is required). A spinning icon will appear until the load is complete. When it is done, you will notice that the counts have changed and additional facet values are visible.

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

1.4 Using the Oscars Sample Application

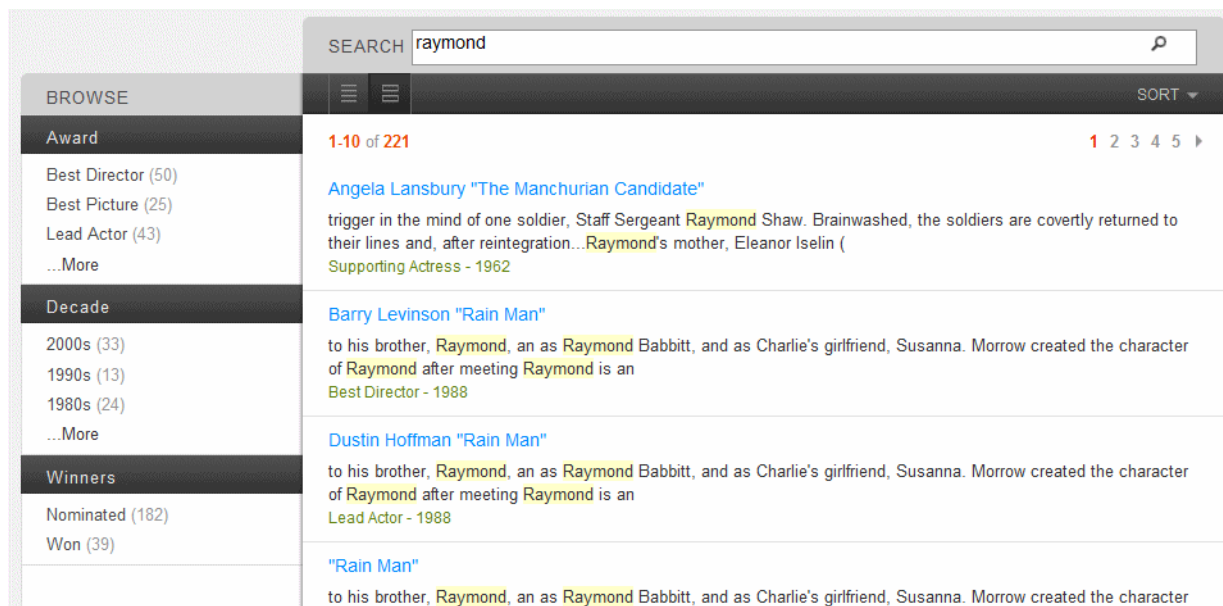
The Oscars sample application allows you to search, browse, and display articles about Oscar award winners from the last nine decades. It is built using the standard features of the Search API, including query text parsing, faceted navigation, snippeting, and many other features. For details about the Search API, see [Search API: Understanding and Using](#) in the *Search Developer's Guide*.

The Oscars sample application is like many search applications, and you should be able to just play around with it to use it. This section walks through the main features of it to highlight some of the things you can create in Application Builder, and includes the following parts:

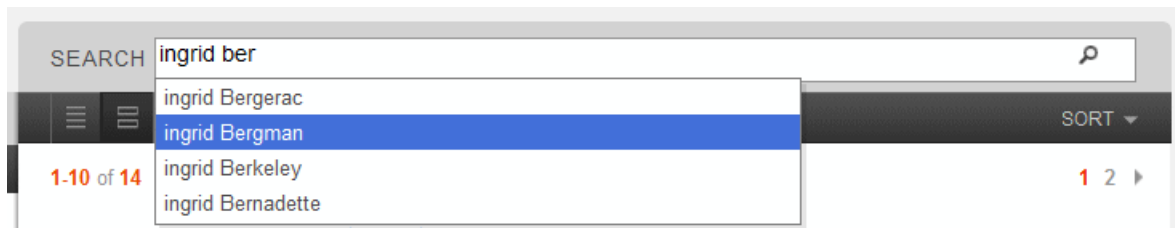
- [Keyword Searching, Search Suggestions, and Parsing](#)
- [Browsing with Facets](#)
- [Search Result Page](#)
- [Displaying Content Detail](#)

1.4.1 Keyword Searching, Search Suggestions, and Parsing

At its most basic, you can enter keywords into the search box and press return to search for matches in the database. For example, a search for raymond shows snippets for the first 10 of 221 results.



As you search, the application provides suggestions for things that might match your search. For example, as you start typing Ingrid Bergman's name, you might see something like the following:



Because the application is built using the Search API, it uses 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 do all of the advanced things such as searching for 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 has the content `Dustin Hoffman`. You can combine the constraint with other terms to further narrow the results:

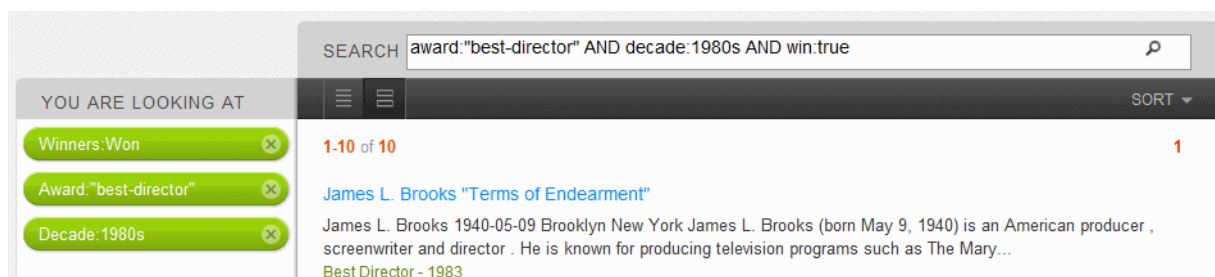
```
buck actor:"Dustin Hoffman"
```

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

1.4.2 Browsing with Facets

On the left side of the application are facets that allow you to browse through the content. Each time you click on a facet, it narrows the results to that category, as well as any other categories or search terms that are active.

For example, if you first click on the Best Director browse link, then on the 1980s link, and then on the Won link, you will find all of the Best Director winners from the 1980s.



Each of the browse facets has a count that shows you how many results match for your current query.

1.4.3 Search Result Page

The page displaying the search result shows a link with a text summary of the content, highlighted snippets of the content matching your search, and some other information about the search match. When you click on the result link, it takes you to the content detail.

1.4.4 Displaying Content Detail

The content detail includes the complete content for the search result. The rendering is based on what was configured in the Content Display page in Application Builder. Styling is customized based on any which skin you choose and based on any custom CSS entered from the Appearance page.

1.5 Using Application Builder to Modify the Oscars Sample Application

You can modify the Oscars sample application with Application Builder. This section describes how to add another facet to the application: a year facet. With the year facet, you can drill-down on the decade facet, then drill down on those results with the year facet. The year facet will use the same index as the decade facet.

Perform the following steps to create a year facet in the Oscars sample application.

1. Start Application Builder (for example, open <http://localhost:8002> in a browser).
2. On the Application Builder Applications page, click edit on the application you used for your Oscars sample application (for example, `Oscars`).
3. Click the Search tab. The Search page appears.

ApplicationBuilder
MarkLogic Application Services

Appearance **Search** Sorting Results Content Deploy NewApplication Home Help

Constraints and Facets

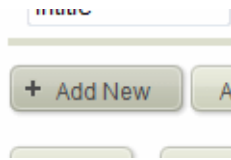
Configure constraints for search and facets for navigation.

Name	Label	Type	Source	Faceted	
award	Award	Range	Index: nominee/@award	<input checked="" type="checkbox"/>	Options
film-title	Film title	Range	Index: film-title	<input checked="" type="checkbox"/>	Options
name	Name	Range	Index: name	<input checked="" type="checkbox"/>	Options
winner	Winner	Range	Index: nominee/@winner	<input checked="" type="checkbox"/>	Options
year	Year	Range	Index: nominee/@year	<input checked="" type="checkbox"/>	Options

+ Add New Advanced Settings

Back Resample Next

- On the Search page, click the Add New Button.



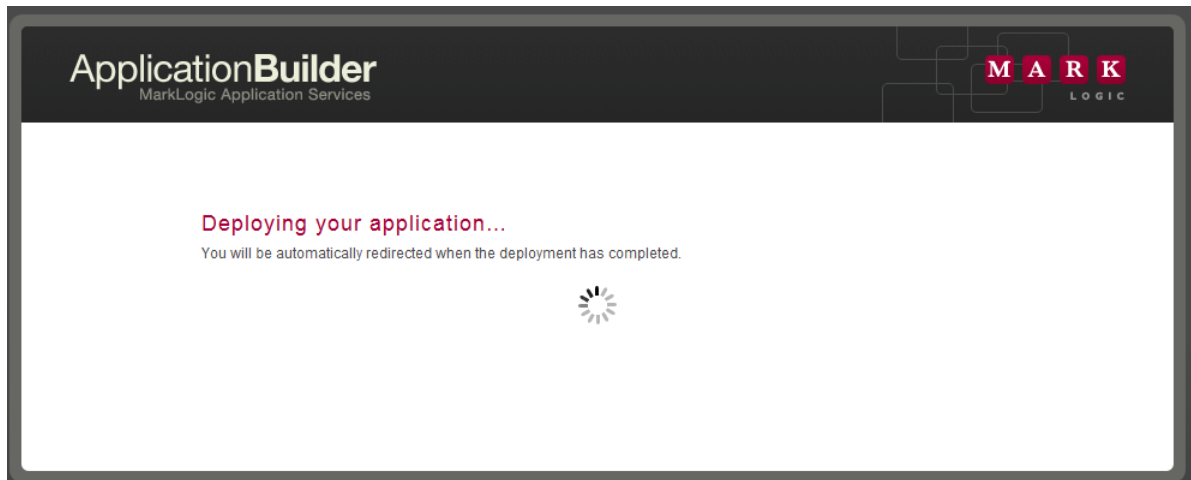
- On the New Constraint dialog, click Range.
- On the New Constraint dialog, enter `year` for the Name and select `year` for the Source Index.

A screenshot of the 'New Constraint' dialog box. It has a title bar 'New Constraint' and six tabs: 'Range', 'Word', 'Value', 'Collection', 'Element Scope', and 'Properties Scope'. The 'Range' tab is selected. Below the tabs, there are descriptions for each constraint type. Under the 'Range' tab, there is a section titled 'Range Constraint' with a 'Name' field containing 'year' and a 'Source Index' dropdown menu also set to 'year'. A note below the dropdown says 'Requires a range index in the source database'. At the bottom, there are 'Cancel' and 'Create Range Constraint' buttons.

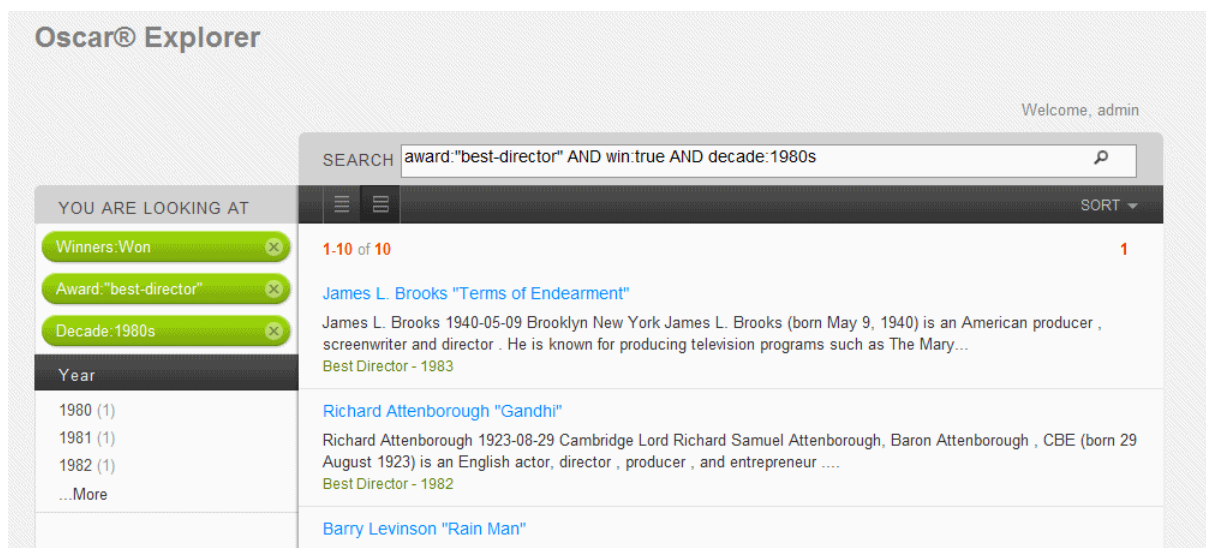
- Click Create Range Constraint. Application Builder creates the constraint.
- In the application name menu, select Deploy Now.



9. Application Builder compiles and deploys the new application code to the modules database of your App Server. In a new window, the following screen appears while it is deploying.



When it is done, the newly modified application replaces the status page. Your new year facet appears in the new application page. If you select a particular decade, you can then select the year from the year facet to find the results for a single year from that decade.



Note: The year facet in this example will be available at all times, whether or not you have clicked on the decade facet. If you want the year facet to only display after you have selected a decade, you need to add some display logic that only displays the year facet after specifying a decade facet value. The Oscars sample application is not set up to do this, but it is possible to modify the built application to create such hierarchical logic. For details on modifying the application, see “Extending Applications Built With Application Builder” on page 41.

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 Through the User Interface](#)
- [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:8002
```

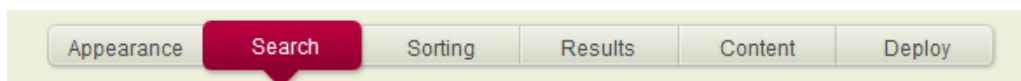
If your instance of MarkLogic Server is running on a different host, or if Application Services is configured on a different port, substitute the appropriate values for host and port.

In order to use Application Builder, you must have the `app-builder` role assigned to your login account. To use Information Studio, you must have the `infostudio-user` role. Users with the `admin` role have access to both applications.

Log in as a user with the `app-builder` role (or as a user with the `admin` role). For details about the `app-builder` role, see “Predefined Roles for Application Builder” on page 39.



2.2 Navigating Through the User Interface

The user interface for Application Builder is a straightforward tabbed interface, where each tab allows you to configure certain functionality in the application.



You can click on any tab to go to that page, and you can click the next and back buttons to navigate to the adjacent screens. The following table describes the navigation portions of Application Builder User Interface that appear on most of the pages.

Action	Description
Tabs	Click on a tab to change to display a page in Application Builder. Changing tabs will automatically save the state of the application.

Action	Description
Application name menu	<p>If you click on the name of the application towards the upper right corner of the screen, a menu of options displays:</p> 
	The Save option saves the application to the database.
	The Deploy Now option immediately deploys the application. This option is available only after the application has been deployed for the first time.
	The Support Package option generates a zip file of the application and the application (including all of the application code) in case you need to contact MarkLogic Support. The support zip file also includes a small sample of documents from your content database.
	The Application Configuration option displays the XML of the current application in a new window. The application XML includes the Search API options element, which is helpful if you are using Application Builder to help you generate Search API code.
Home	Takes you back to the Application Services page.
Help	Help for the current page.
Next/Back	<p>Navigate to the next page or back to the previous page.</p> 

Action	Description
Resample	<p>When available, samples random documents in the content database for use in building constraints, search results, and rendering the content.</p> <p>Note: Resampling replaces many of the current settings on the page, so certain customizations you have configured will be lost after resampling.</p>

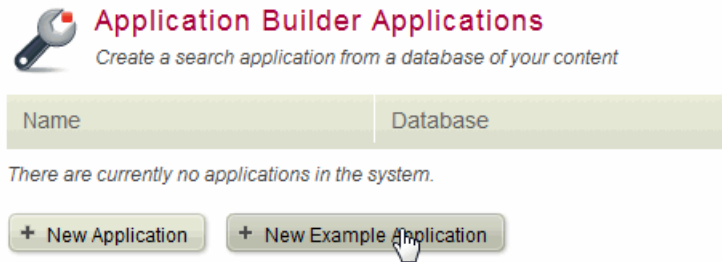
2.3 Page-By-Page Walkthrough

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

- [Application Builder section of the Application Services Page](#)
- [Appearance Page](#)
- [Search Page](#)
- [Sorting Page](#)
- [Results Page](#)
- [Content Page](#)
- [Deploy Page](#)

2.3.1 Application Builder section of the Application Services Page

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



The Application Builder Applications section lists all of the applications in the `App-Services` database and allows you to create a new applications, create a new sample applications (the Oscars sample application), or modify existing applications. An application stores all of the information Application Builder uses to generate an application, such as information about constraints you have configured, setting for the results page, deployment options, and so on. You can click the application name to modify an existing application as you iteratively develop a search application.

When you create an application, you specify the database to use with the application (or, if you are creating a new sample application to build the Oscars sample application, you can have Application Builder create a new database for the application). You must load a representative sample of content and set up any indexes for the database outside of the scope of Application Builder. The content should be indexed and ready to be searched. Application Builder looks at the indexes configured in your database and uses that information to help you configure the application. If your database structure is still evolving, that is OK, because you can iterate.

Note: Application Builder assumes there is no fragmentation in the database. If your database is fragmented, you might need to modify the generated application for it to work properly with fragmentation. For details on modifying the generated application, see “Extending Applications Built With Application Builder” on page 41.

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 name of the application and choose an existing database to use for the application.
New Sample 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 9.
Edit	Open an existing application to modify settings and/or deploy the application.
Delete	Permanently delete an application from the <code>App-Services</code> database. Deletes the application, but not the database or App Server used by the application.

2.3.2 Appearance Page

The Appearance page is where you can configure some look-and-feel aspects of your application. For example, you can specify a title, add a logo, and add custom CSS code. Default values are supplied for all of the options, so you can simply take the defaults if they suit 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 on the main page.
Skin	Affects the look and feel of the application. Select one of the available skins, and by clicking the Customize button, you can enter arbitrary CSS that the application uses in addition to the selected skin CSS.
Site Info	Metadata about the site. The page title appears in the top bar of the browser window for the application (the <code>html/head/title</code> element), and the copyright information appears in the footer.

2.3.3 Search Page

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

Name	Label	Type	Source	Faceted	Options
award	Award	Range	Index: nominee/@award	<input checked="" type="checkbox"/>	Options
film-title	Film title	Range	Index: film-title	<input checked="" type="checkbox"/>	Options
name	Name	Range	Index: name	<input checked="" type="checkbox"/>	Options
winner	Winner	Range	Index: nominee/@winner	<input checked="" type="checkbox"/>	Options
year	Year	Range	Index: nominee/@year	<input checked="" type="checkbox"/>	Options

[+ Add New](#)
[Advanced Settings](#)

[Back](#)
[Resample](#)
[Next](#)

You can configure existing constraints or add new constraints on the Search page. 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 that are from the 1980s:

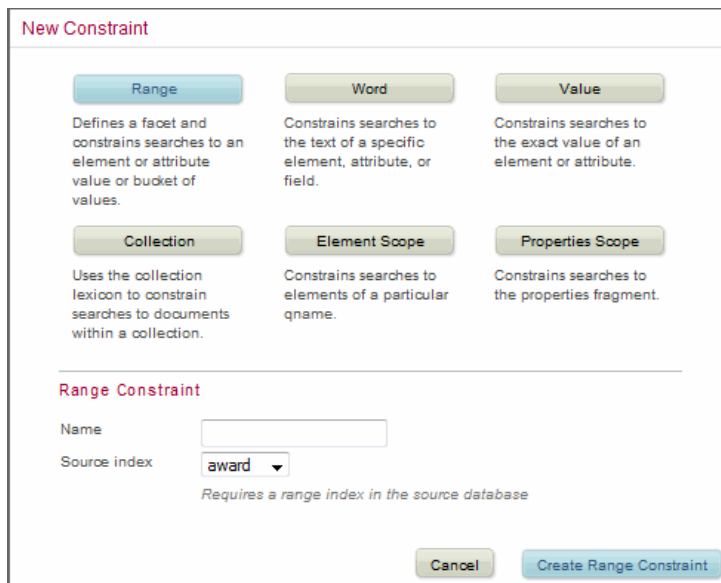
```
decade:1980s
```

There are several kinds of constraints you can add or modify on the Search page. The following parts describe those constraints as well as modifying other options to search in the application:

- [Add/Modify Range Constraints](#)
- [Add/Modify Value Constraints](#)
- [Add/Modify Word Constraint](#)
- [Add/Modify Collection Constraint](#)
- [Modifying Search Options](#)

2.3.3.1 Add/Modify Range Constraints

A range constraint uses range indexes to support queries and create facets. Click the Add New button to add a new constraint, or click the options link to edit an existing constraint.



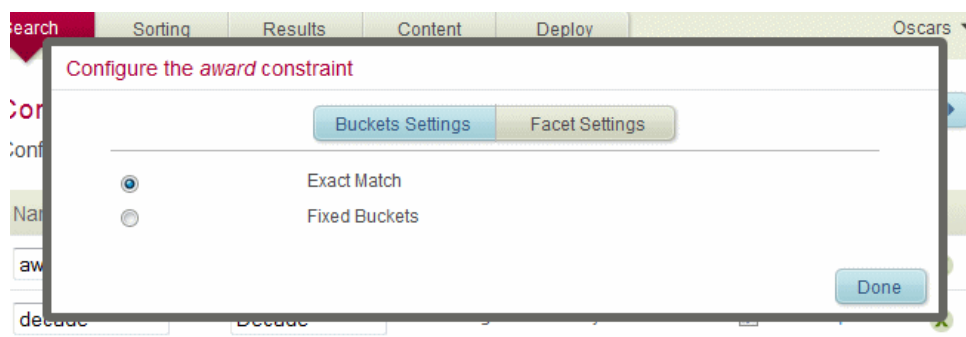
The "New Constraint" dialog box contains six buttons: "Range", "Word", "Value", "Collection", "Element Scope", and "Properties Scope". Each button has a description of its function. The "Range" button is highlighted. Below the buttons, the "Range Constraint" section includes a "Name" text field, a "Source index" dropdown menu set to "award", and a note "Requires a range index in the source database". At the bottom are "Cancel" and "Create Range Constraint" buttons.

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

decade:1980s

There are two types of Range constraints: Exact Match and Fixed Buckets. Additionally, for date ranges, there is a third type of Range constraint: Relative Buckets.

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



The "Configure the award constraint" dialog box has two tabs: "Buckets Settings" and "Facet Settings". Under "Buckets Settings", there are two radio buttons: "Exact Match" (selected) and "Fixed Buckets". A "Done" button is at the bottom right.

Fixed buckets constraints allow you to specify labels that correspond to ranges of values. Fixed buckets are useful when you want to specify ranges of values (decades on time-based ranges, for example) for queries and facets.

Configure the decade constraint

Buckets Settings | Facet Settings

☐ Exact Match
☒ Fixed Buckets

Bucket name

Match items greater than
 and less than

Name	Label	Greater Than	Less Than	
2000s	2000s	2000		X
1990s	1990s	1990	2000	X
1980s	1980s	1980	1990	X
1970s	1970s	1970	1980	X
1960s	1960s	1960	1970	X
1950s	1950s	1950	1960	X
1940s	1940s	1940	1950	X
1930s	1930s	1930	1940	X
1920s	1920s	1920	1930	X

Relative buckets constraints are used with date ranges (that is, elements or attributes of type `xs:date` or `xs:dateTime`). You can create constraints that are for various ranges of dates. To create a new bucket that represents a range of dates, fill in the items in the form and click the Add Bucket button. A new row is added to the bucket table with the range you specified.

You can also modify existing buckets by editing the fields in the table that lists the buckets. The Anchor field in the relative buckets configuration screen specifies what part of the date or `dateTime` value to start counting where the bucket boundaries are. The On or After and the Before fields in the relative buckets table must be represented as durations. Durations are used to specify time periods, and it is possible to do arithmetic on durations. For some information about the syntax of durations, see http://en.wikipedia.org/wiki/ISO_8601#Durations or see 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.

Configure the day constraint

Buckets Settings **Facet Settings**

☐ Exact Match
☐ Fixed Buckets
☒ Relative Buckets

Bucket name

Relative to

match items later than

and earlier than

[+ Add Bucket](#)

Name	Label	Anchor	On or After	Before	
<input type="text" value="yesterday"/>	<input type="text" value="Yesterday"/>	<input type="text" value="Start of Day"/>	<input type="text" value="-P1D"/>	<input type="text" value="P0D"/>	<input checked="" type="checkbox"/>
<input type="text" value="today"/>	<input type="text" value="Today"/>	<input type="text" value="Start of Day"/>	<input type="text" value="P0D"/>	<input type="text" value="P1D"/>	<input checked="" type="checkbox"/>

[Done](#)

Each range constraint, if faceted, also has Facet Settings, which control things like how many facets appear in the results.

Configure the decade constraint

Buckets Settings **Facet Settings**

Order

Direction

Number of results

Include on front page ☐

Include in sidebar ☒

[Done](#)

2.3.3.2 Add/Modify Value Constraints

A value constraint uses element or attribute values to create a constraint. Click the Add New button to add a new constraint, or click the options link to edit an existing constraint. On the New Constraint page, enter a name for the constraint and enter an element or attribute name to match on. When you click the Find button, Application Builder searches through sample documents in your database to find wildcard-matching elements or attributes.

New Constraint

Range Defines a facet and constrains searches to an element or attribute value or bucket of values.	Word Constrains searches to the text of a specific element, attribute, or field.	Value Constrains searches to the exact value of an element or attribute.
Collection Uses the collection lexicon to constrain searches to documents within a collection.	Element Scope Constrains searches to elements of a particular qname.	Properties Scope Constrains searches to the properties fragment.

Value Constraint

Name

Element or attribute **Find**

Find matching elements or attributes in the source content.

Cancel **Create Value Constraint**

Value constraints differ from range constraints in the following ways:

- You do not need a range indexes for value constraints, you need range indexes for range constraints.
- You cannot create facets from value constraints, 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 set up to be case-insensitive (which is the default).

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

2.3.3.3 Add/Modify Word Constraint

A word constraint uses elements, attributes, or fields to find words. When the constraint is used, it matches documents with the word specified in the constraint. For example, if you have create a word constraint on the `ABSTRACT` element, the following search would find documents with the word `hello` in an `ABSTRACT` element:

```
abstract:hello
```

Click the Add New button to add a new constraint, or click the options link to edit an existing constraint.

New Constraint

Range Defines a facet and constrains searches to an element or attribute value or bucket of values.	Word Constrains searches to the text of a specific element, attribute, or field.	Value Constrains searches to the exact value of an element or attribute.
Collection Uses the collection lexicon to constrain searches to documents within a collection.	Element Scope Constrains searches to elements of a particular qname.	Properties Scope Constrains searches to the properties fragment.

Word Constraint

Name

☒ Source field

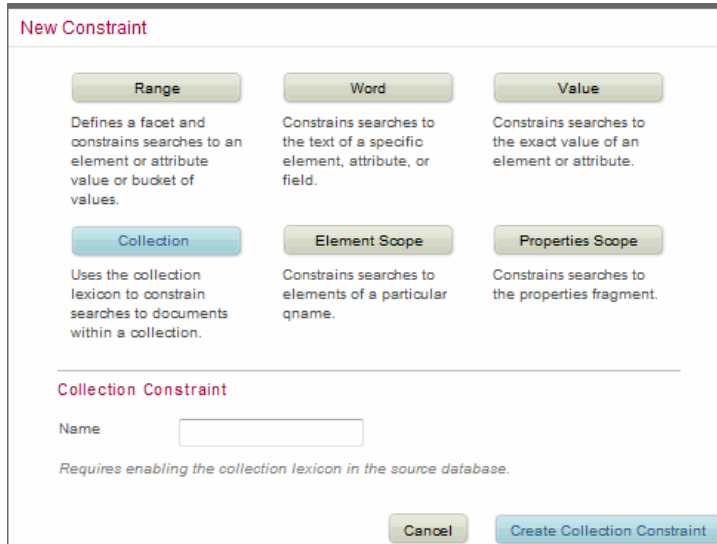
☐ Element or attribute

Find matching elements or attributes in the source content.

Word constraints cannot be faceted. No special indexes are required for a word constraint, although if you specify an element, attribute, or field that does not exist, queries using the constraint will return no results.

2.3.3.4 Add/Modify Collection Constraint

A collection constraint constrains a search to items in the specified collection and allows facets on the collection. Click the Add New button to add a new constraint, or click the options link to edit an existing constraint.



New Constraint

<p>Range</p> <p>Defines a facet and constrains searches to an element or attribute value or bucket of values.</p>	<p>Word</p> <p>Constrains searches to the text of a specific element, attribute, or field.</p>	<p>Value</p> <p>Constrains searches to the exact value of an element or attribute.</p>
<p>Collection</p> <p>Uses the collection lexicon to constrain searches to documents within a collection.</p>	<p>Element Scope</p> <p>Constrains searches to elements of a particular qname.</p>	<p>Properties Scope</p> <p>Constrains searches to the properties fragment.</p>

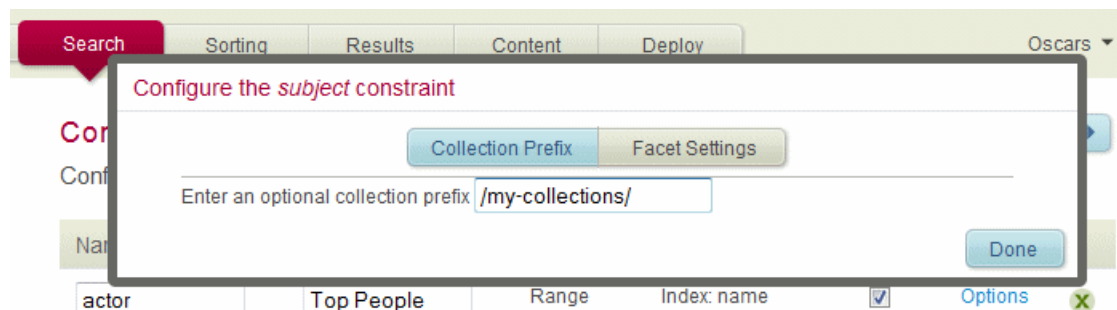
Collection Constraint

Name

Requires enabling the collection lexicon in the source database.

A Collection constraint requires that the collection lexicon is enabled in the database. and you can create facets with a collection constraint.

You can optionally set up a prefix that is used to concatenate with the collection constraint value by clicking the Options link for the constraint and setting a prefix.



Configure the subject constraint

Enter an optional collection prefix

Assuming that all documents in your database have collection URIs that begin with the string `/my-collections/` like the following:

```

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

```

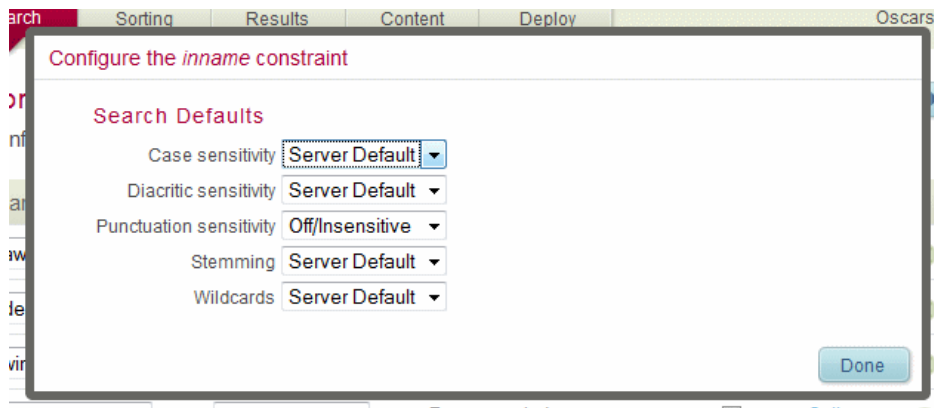
Then the following query text examples will match documents in the corresponding collections:

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

You can check the Faceted button to create facets on the collections.

2.3.3.5 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 options link for the constraint. By default, most of the options are set to the MarkLogic Server defaults. 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*.



Additionally, the Advanced settings button displays a page allowing you to modify the search grammar and search options (case sensitive/insensitive, diacritic sensitive/insensitive, and so on) for everything besides the constraints in your application (some constraints allow you to set their search options independently). By default, many of the options are set to the Search API and MarkLogic Server defaults. For details on the various search options in MarkLogic Server, see the *Search Developer's Guide*.

The screenshot shows the 'Advanced Settings' dialog box. It is divided into three main sections: 'Search Grammar', 'Suggestions', and 'Search Defaults'. In the 'Search Grammar' section, there are four checked checkboxes for 'Support AND queries', 'Support OR queries', 'Support grouping', and 'Support negation', each with a corresponding example query. Below these is a dropdown for 'Empty query behavior' set to 'Return All Results' and an unchecked checkbox for 'Filter results'. The 'Suggestions' section has a 'Default Source' dropdown set to 'Off' with a note 'Requires a range index in the source database'. The 'Search Defaults' section contains five dropdown menus for 'Case sensitivity', 'Diacritic sensitivity', 'Punctuation sensitivity', 'Stemming', and 'Wildcards', all set to 'Server Default'. A 'Done' button is located at the bottom right of the dialog.

Section	Option	Value	Example
Search Grammar	Support AND queries	<input checked="" type="checkbox"/>	foo AND bar
	Support OR queries	<input checked="" type="checkbox"/>	foo OR bar
	Support grouping	<input checked="" type="checkbox"/>	foo OR (bar AND baz)
	Support negation	<input checked="" type="checkbox"/>	-foo
Search Grammar	Empty query behavior	Return All Results	
	Filter results	<input type="checkbox"/>	
Suggestions	Default Source	Off	Requires a range index in the source database
Search Defaults	Case sensitivity	Server Default	
	Diacritic sensitivity	Server Default	
	Punctuation sensitivity	Off/Insensitive	
	Stemming	Server Default	
	Wildcards	Server Default	

The Filter results button specifies whether the searches should be run unfiltered (option unchecked) or filtered. Filtered searches are more accurate in some cases, and unfiltered searches are always faster.

2.3.4 Sorting 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 allows you to configure other operators for different sorting orders.

Name	Label	Sorting Criteria
relevance	Relevance	Score X + Add X
year	Year	nominee/@year X Score X + Add X

+ Add New

Back Resample Next

In the application that is generated, you can use the name of the sorting criteria with sort operator to specify the order of your results. For example, the following specifies sorting by year:

```
sort:year
```

To create different sort orders, you need range indexes on the keys on which you want to sort. Each line in the sort table represents a sort operator, and the green rounded controls represent a key on which to sort. Click the Add button to add a new sort key to an existing sort operator. The new field is placed after any existing sort keys. To remove a key, click the X in the key. To remove the whole sort operator, click the X at the right of the row.

year Year nominee/@year X Score X + Add X

By adding new keys or deleting existing ones, you can specify a custom sort order. You can edit the source index or sort order of an existing key by clicking it.

Edit Order Item

Source index: year (Requires a range index in the source database)

Sort direction: Descending

Cancel Done

2.3.5 Results Page

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

The screenshot shows the 'Results' configuration page in the MarkLogic Application Builder. The interface includes a top navigation bar with tabs for Appearance, Search, Sorting, Results (selected), Content, and Deploy. Below the navigation bar, the 'Search Results' section allows users to configure the contents of individual search results. A 'Live Preview' area shows a sample result for 'Gentleman's Agreement - Twentieth Century Fox - Gregory Peck'. The configuration sections include:

- Title:** The text for the result link. It shows a configuration with 'film-title', 'distributor', and 'name' fields.
- Snippet:** The elements from the result that the snippeting will favor. It shows a configuration with 'p', 'a', 'i', and 'li' fields.
- Metadata:** Additional information about the result. It shows a configuration with 'name', 'film-title', 'nominee@year', 'nominee@award', and 'nominee@winner' fields.

At the bottom of the configuration area, there are 'Back', 'Resample', and 'Next' buttons.

As you make changes, the changes appear in the preview area. Not all changes will 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 test the snippeting, it is best to test 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 you 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 the X button corresponding to the item.

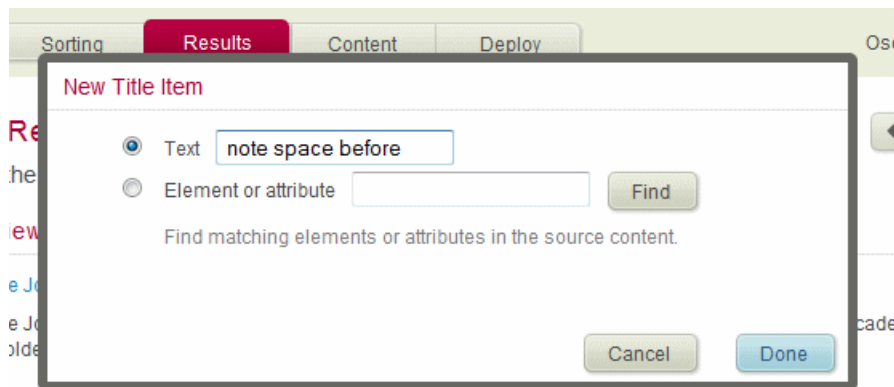
Title

The text for the result link.

name X " X film-title X " X

+ Item

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



You can configure snippets by adding one or more elements that the snippeting 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.

Snippet

The elements from the result that the snippeting will favor.

p X

+ Element

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

Metadata

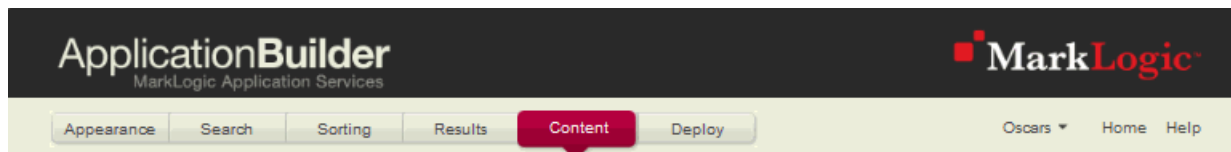
Additional information about the result.

nominee/@award X - X nominee/@year X

+ Item

2.3.6 Content Page

The content page allows you to control the rendering of the search content. This controls how the content is rendered to xhtml in the generated application. Application Builder looks at the content and divides it up into *container element* (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.



Content Display

Control how the application renders content as XHTML for web browsers.

Default Custom XSLT

☒ XHTML pass-through

Container Elements

Element	Render as
nominee	Div
person	Div
oscar	None
birth	None
death	None
place	None
date	None

Live Preview

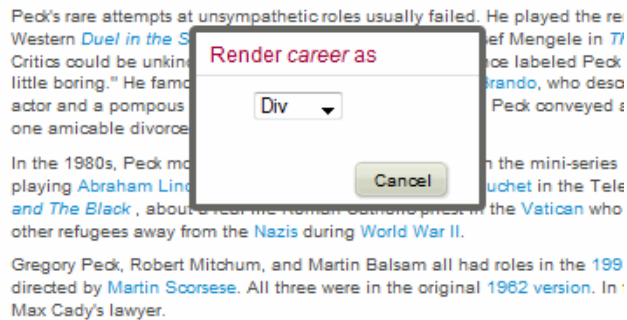
Click to edit inline.

Gregory Peck

Gregory Peck (5 April 1916 – 12 June 2003) was an American film actor. He was one of 20th Century Fox's most popular film stars, from the 1940s to the 1960s, and played important roles well into the 1990s. One of his most notable performances was as Atticus Finch in the 1962 film version of *To Kill a Mockingbird*, for which he won his Academy Award. President Lyndon Johnson honored Peck with the Presidential Medal of Freedom in 1969 for his lifetime humanitarian efforts. In 1999, the American Film Institute named Peck among the Greatest Male Stars of All Time, ranking at #12.

Peck was born Eldred Gregory Peck in San Diego, California's seaside community of La Jolla, the son of Missouri-born Bernice Mae "Bunny" Ayres and Gregory Pearl Peck, who was a chemist and pharmacist. Peck's father was of English (paternal) and Irish (maternal) heritage, and his mother was of Scots (paternal) and English (maternal) ancestry. Peck's father was a Catholic and his mother converted upon marrying his father. Peck's Irish-born paternal grandmother, Catherine Ashe, was related to Thomas Ashe, who took part in the Easter Rising less than three weeks after Peck's birth and died while on hunger strike in 1917. Peck's parents divorced by the time he was six years old and he spent the next few years being raised by his grandmother.

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

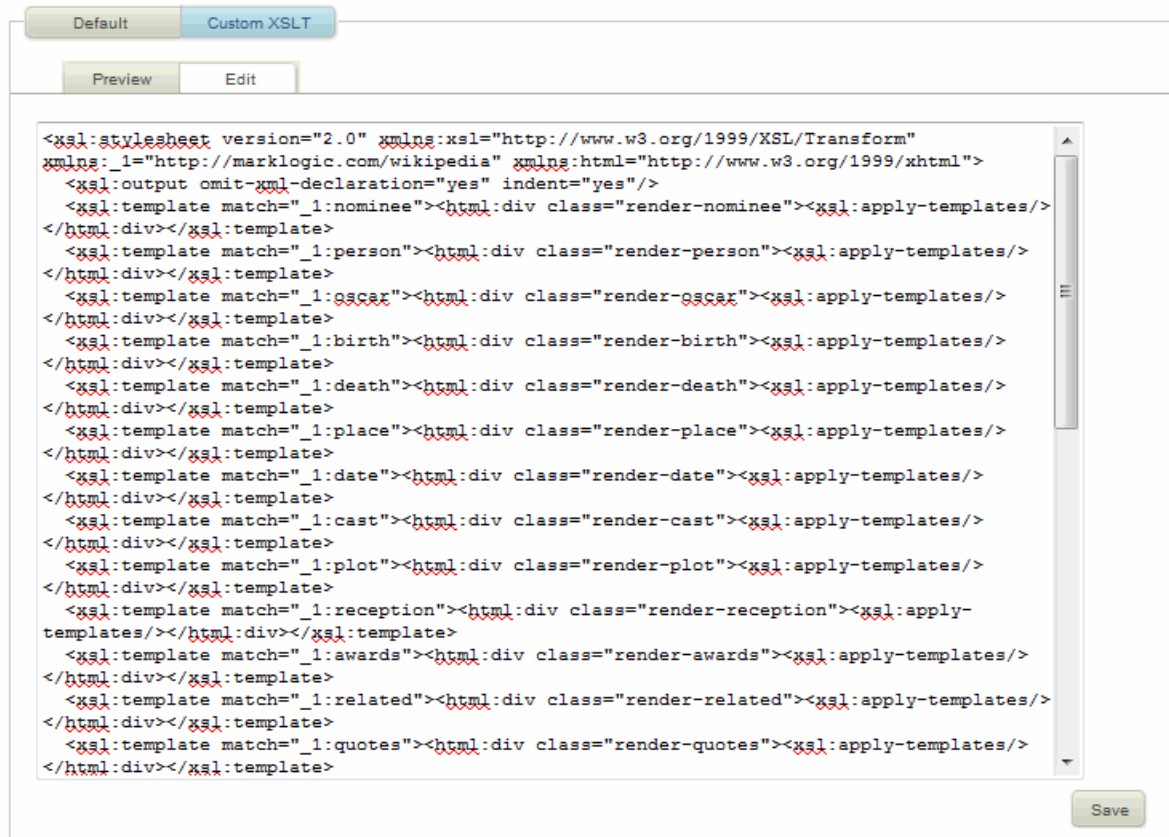


If the XHTML passthru box is checked, then any content in the XHTML namespace (<http://www.w3.org/1999/xhtml>) is passed through without change to the displayed application.

Selecting the Custom XSLT tab displays the XSL stylesheet used to render the content page. Here you can make more detailed modifications to the XSL.

Content Display

Control how the application renders content as XHTML for web browsers.



Click the Preview tab to preview the results of your changes to the XSL stylesheet. When you are satisfied with the results, click Edit to return to the XSL stylesheet and click Save. If you navigate away from the Custom XSLT pages without first saving your changes, they will be lost.



If you want to further customize the rendering, you can customize the application after it is built, by means of the XSL files in the `/application/custom` directory, as described in “Extending Applications Built With Application Builder” on page 41.

2.3.7 Deploy Page

The deploy page is where you select or configure the App Server on MarkLogic Server to which your generated application will deploy. When you click the Deploy button, Application Builder performs the following actions:

- It compiles the application based on the application settings.
- It deploys the compiled application code to the modules database of the specified App Server.
- It 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/`; Application Builder will not deploy code to an App Server with any other root or with its root on the filesystem.

If you create a new App Server, Application Builder will also create a modules database for the new App Server, and it will configure it with a root of `/application/`.

ApplicationBuilder
MarkLogic Application Services

Appearance Search Sorting Results Content **Deploy** Oscars Home Help

Compile and Launch Application

Specify the App Server on which to deploy your configured application.

☒ Existing App Server
Overwrite this App Server's configuration and its modules database with the compiled application.
Oscars (8003)

☐ New App Server
Create a new HTTP App Server and modules database.
Name: Oscars-1
Port: 8005

Deploy

Back Resample Next

Note: When you deploy an application, it opens the newly generated application in a new browser window using a URL with the hostname that is stored in the MarkLogic Server configuration files (the result of an `xdrm:host-name` call). If you are running on a laptop computer that is changing networks, it is possible that the hostname will 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 that is not available on your current network). In these cases, substituting `localhost` for the hostname in the URL should allow the application to launch.

3.0 Controlling Access to Application Builder and to Generated Applications

Application Builder allows you to configure and generate search applications without the need to write any code. Both Application Builder and the applications it generates use the MarkLogic Server security model to control access. This chapter describes the security roles needed to run Application Builder and to run the applications it generates, and includes the following parts:

- [Predefined Roles for Application Builder](#)
- [Permissions on Documents](#)
- [Adding Other Roles or Modifying the app-user Role or to Meet Your Requirements](#)

3.1 Predefined Roles for Application Builder

There are three predefined roles that are used by Application Builder:

- [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 that is needed to run any application that Application Builder generates. You must grant this role to all users who are allowed to run the generated application.

3.1.2 app-builder Role

The `app-builder` role provides the privileges needed to run Application Builder. Application Builder performs many administrative tasks on MarkLogic Server (for example, creating databases and App Servers), and this role provides the privileges to perform those tasks. While the privileges are minimized to the needed functions and to amped functions, it still allows users with the role to create these resources on MarkLogic Server, and 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. Assign the `app-builder` role to users who are allowed to generate applications with Application Builder.

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 the `app-builder-internal` role to any user; it is only for internal use by Application Builder.

3.2 Permissions on Documents

In order for users to be able to search the applications generated from Application Builder, all documents in that database must have a read permission for a role the users have. If you want all users to be able 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 access to content by having permissions on documents 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 of the 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 as in the following example:

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

For your applications, you should come up with a security policy then add the appropriate permissions to the documents in your database to implement that security policy.

For more details on permissions, see [Permissions on Documents](#) in the *Understanding and Using Security Guide*.

3.3 Adding Other Roles or Modifying the `app-user` Role or to Meet Your Requirements

For the applications that Application Builder generates, the `app-user` role provides sufficient privileges for any user to run the application. If you modify the application to perform other actions, however, you might need to provide users additional privileges. There are two basic techniques for providing additional privileges to your users: adding another role with the privileges and assigning that role to the application user, or modifying the existing `app-user` role.

Depending on what your requirements are, it might make sense to add those privileges to another role and then grant that role to your users. This is the safer technique, as it leaves the `app-user` role intact and will not affect future applications you might create with Application Builder.

In some cases, if you are comfortable about the implications, it might make sense to modify the `app-user` role by adding other permissions to it. If you do modify the `app-user` role, use caution, as that role is used by all applications Application Builder generates.

4.0 Extending Applications Built With Application Builder

Application Builder generates search applications according to certain templates. The applications are designed to be extended and modified in a number of ways. This chapter describes how to extend and modify the generated applications and contains the following sections:

- [Finding 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 Finding the Generated Code

When you deploy an application, it deploys the code to the modules database for the App Server specified on the Deploy page of Application Builder. To view your code, look in the modules database in the `/application/` directory. You can look at the documents in the modules database by setting up a WebDAV server to the modules database or by using any other method for looking at documents in a database. The entire application is there, including all of the images, libraries, and other components.

Another way to view the code is to generate a support package from Application Builder, as described in “Navigating Through the User Interface” on page 17. The support package is a zip file that contains the entire generated application, and the application code is under the `application` directory of the support package.

The following tables list the files of interest in the `lib`, `css` and `js` directories under the `application` directory.

Warning Do not update the files in any of these directories. Otherwise, the next time you make changes to your application in App Builder, the changes you made to these modules will be overwritten. Instead, modify the files described in “The Custom Directory” on page 44.

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

File	Description
<code>config.xqy</code>	Contains the variable declarations that use function values to specify the code to run for various features in the application. Many of the variables in this file can be overridden by the <code>/application/custom/appfunctions.xqy</code> module.
<code>standard.xqy</code>	Contains the functions that are called from <code>config.xqy</code> .
<code>transform-detail.xsl</code>	Contains the XSLT used to render the detail page in your application.
<code>transform-abstract-metadata.xsl</code>	Contains the XSLT used to render the meta section of individual search results in your application.
<code>transform-abstract-title.xsl</code>	Contains the XSLT used to render the title section of individual search results in your application.

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

File	Description
<code>master.css</code>	Contains the generic style information for applications
<code>custom.css</code>	Contains the style information customized in App Builder
<code>ie_6.css</code>	Contains the style information specific the Internet Explorer, Version 6.
<code>ie_7.css</code>	Contains the style information specific the Internet Explorer, Version 7.

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

File	Description
<code>application.js</code>	<p>Contains the JavaScript code used by your applications.</p> <p>You can extend on the JavaScript in this file by adding your extensions to the <code>application/custom/appjs.js</code> file. If you want to override the JavaScript in this file, do the following:</p> <ul style="list-style-type: none">• Copy the <code>application.js</code> file to the <code>/application/custom</code> directory and modify the copied file as needed.• Uncomment the <code>app.js</code> function in the <code>application/custom/appfunctions.xqy</code> file.• Change the script tag in the <code>app.js</code> function in the <code>appfunctions.xqy</code> file from: <pre><script src="/js/application.js"</pre> to: <pre><script src="/custom/application.js"</pre> <p>Note: The script tag looks different for the sample application.</p>
<code>sample-application.js</code>	Contains the JavaScript code used by the Sample applications.

4.2 The Custom Directory

The files in the `/application/custom` directory enable you to override the code in the `config.xqy` and `standard.xqy` files and preserve those changes when the application is redeployed by App Builder.

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

File	Description
<code>appfunctions.xqy</code>	<p>This module contains most of the variables in the <code>config.xqy</code> module and the same functions as the <code>standard.xqy</code> module. Unlike the <code>standard.xqy</code> or <code>config.xqy</code> modules, any changes you make to a variable or function in the <code>appfunctions.xqy</code> module will survive redeployment.</p> <p>Modifying a variable in this module will override the respective variable in the <code>config.xqy</code> module.</p> <p>Each function in the <code>appfunctions.xqy</code> module is commented out. Removing the comments from a function in the <code>appfunctions.xqy</code> module will override the respective function in the <code>standard.xqy</code> module.</p>
<code>appcss.css</code>	<p>Styles added to this file override those in the CSS files located in the <code>/application/css</code> directory.</p> <p>Note that the <code>\$ADDITIONAL-CSS</code> variable in the <code>appfunctions.xqy</code> module points to this file. If you want to add other CSS files for your application, you can put the files in <code>/custom</code> and update the <code>\$ADDITIONAL-CSS</code> variable to load them in the desired order.</p>
<code>appjs.js</code>	<p>JavaScripts added to this file extend (but do not override) the scripts located in the <code>/application/js</code> directory.</p> <p>Note that the <code>\$ADDITIONAL-JS</code> variable in the <code>appfunctions.xqy</code> module points to this file. If you want to support jQuery or other frameworks, you can put the files in <code>/custom</code> and update the <code>\$ADDITIONAL-JS</code> variable to load them in the desired order.</p>
<code>apptransform-detail.xsl</code>	<p>XSLT added to this file overrides the XSLT in the <code>/application/lib/transform-detail.xsl</code> file. This XSLT is used to render the detail page in your application.</p>

File	Description
apptransform-abstract-metadata.xml	XSLT added to this file overrides the XSLT in the /application/lib/transform-abstract-metadata.xml file. This XSLT is used to render the meta section of individual search results in your application.
apptransform-abstract-title.xml	XSLT added to this file overrides the XSLT in the /application/lib/transform-abstract-title.xml file. This XSLT is used to render the title section of individual search results in your application.

4.3 Customizing Applications Generated by Application Builder

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

- [Basic Design Pattern](#)
- [Accessing the Code in the Custom Directory](#)
- [Road Map for Application Page Functions](#)
- [Customizing the Footer](#)
- [Customizing the Content Display](#)
- [Customizing the Detail Page](#)
- [Customizing the Generated Search Options Node](#)

Note: Modifications you make to the generated application must be made to the files located in the /application/custom directory, as described in “The Custom Directory” on page 44.

4.3.1 Basic Design Pattern

The applications generated from Application Builder are designed to be extensible so you can easily modify them. The basic design pattern to modify the generated application is as follows:

- Find the variables and functions in `/application/custom/appfunctions.xqy` that correspond to the part of the application you want to modify.
- Remove the comments from the function in the `/application/custom/appfunctions.xqy` file and modify. This function will override the respective function in the `/application/lib/standard.xqy` file. For examples, see “Customizing the Footer” on page 50 and “Customizing the Content Display” on page 51.
- Modify the `/application/custom/appfunctions.xqy` variable declaration(s) to override the variables in the `/application/lib/config.xqy` file. For examples, see “Customizing the Generated Search Options Node” on page 53.
- Similar modifications can be made to override the other files generated by Application Builder. For example, you can override the XSL in the `/application/lib/transform-detail.xsl` file used to render the detail page by adding XSL to the `/application/custom/apptransform-detail.xsl` file, as shown in “Customizing the Detail Page” on page 52.

4.3.2 Accessing the Code in the Custom Directory

To modify the code for the built 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 that WebDAV App Server 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*.

Another option is to copy the files in `/application/custom` to your filesystem, then create an HTTP App Server using the same content database as your Application Builder-generated application. Set the root of the new App Server to the location of your copied code.

4.3.3 Road Map for Application Page Functions

This section shows the functions that control the specific sections of the application pages. All of the functions shown in this section can be modified in the `custom/appfunctions.xqy` file.

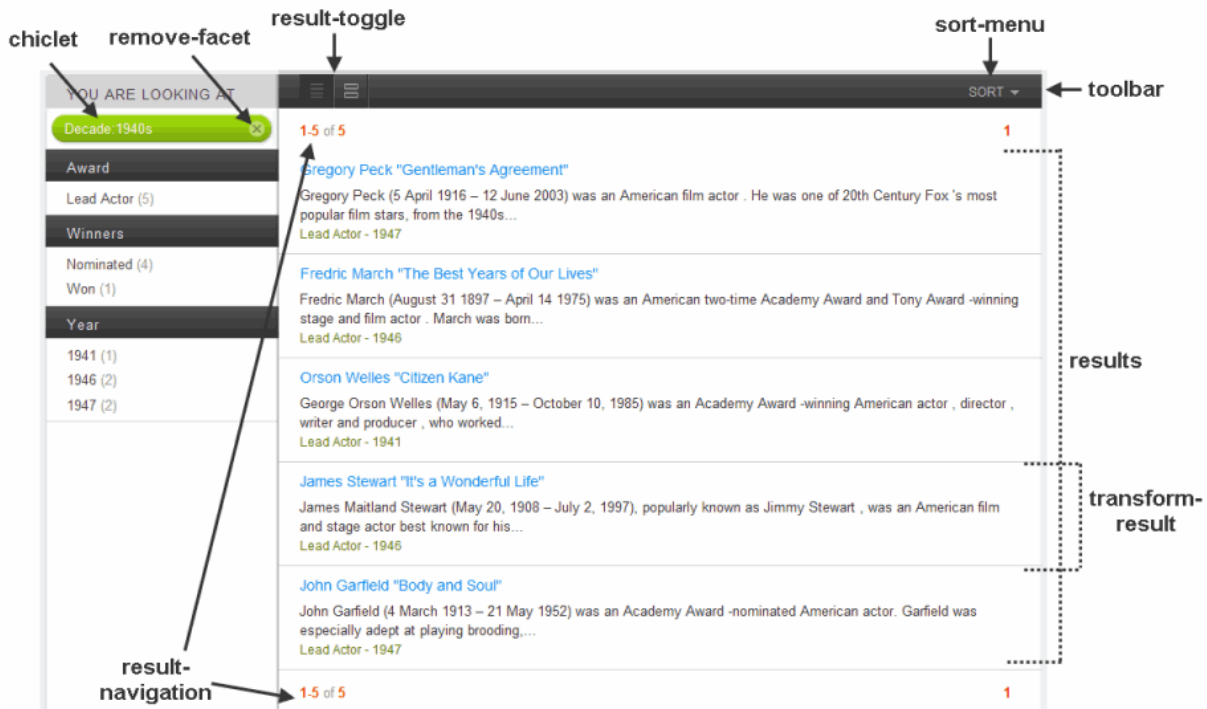
The functions illustrated below are common to all of the application pages. The illustrations that follow show the content sections for the default, search, and detail pages, along with any specific renditions of the sidebar.



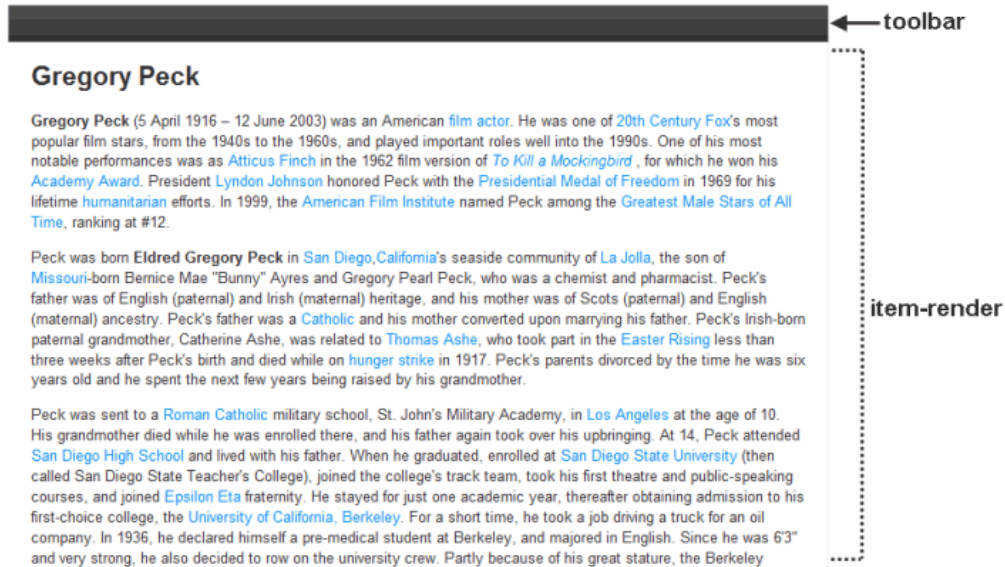
The content portion of the default page is controlled by the functions shown below:



The content and sidebar portions of the search page is controlled by the functions shown below:



The content portion of the detail page is controlled by the functions shown below:



4.3.4 Customizing the Footer

To modify the footer in your generated application, find the `app:footer` function in `/application/custom/appfunctions.xqy`:

```
(:~
: Default footer content

declare function app:footer()
as element(div)
{
  <div class="footer" arysize="0 0 5 5">
    <span class="copyright">&copy;
      {$config:SLOTS/slots:copyright-year/string()},
      {$config:SLOTS/slots:copyright-holder/string()},
      All Rights Reserved.
    </span>
  </div>
};
:)
```

To create your own footer, simply remove the comments from the `app:footer` function definition and add your own `copyright-year` and `copyright-holder` text. For example, to change the footer from the default:

```
© 2010, MarkLogic Corporation, All Rights Reserved.
```

to:

```
© 2011, My Company, All Rights Reserved.
```

Remove the comments around the function and modify as follows:

```
(: Default footer content :)

declare function app:footer()
as element(div)
{
  <div class="footer" arysize="0 0 5 5">
    <span class="copyright">&copy;
      {$config:SLOTS/slots:copyright-year/string("2011")},
      {$config:SLOTS/slots:copyright-holder/string("My Company")},
      All Rights Reserved.
    </span>
  </div>
};
```

When you run your application, it will now use your new footer and the change will persist after redeploying the application in Application Builder.

4.3.5 Customizing the Content Display

To modify the content display in your generated application, find the `app:content` function in `/application/custom/appfunctions.xqy`:

```
(:~
: Page content

declare function app:content()
as element(div)
{
  <div class="content">
    <div class="content-background"><!-- --></div>
      { asc:get-content() }
    </div>
  </div>
};
:)
```

For example, to change the background of the content display to yellow, you can uncomment the function and modify the `<div>` tag as follows:

```
(: Page content :)

declare function app:content()
as element(div)
{
  <div class="content">
    <div class="content-background" style="background: yellow">
      <!-- -->
    </div>
    { asc:get-content() }
  </div>
};
```

4.3.6 Customizing the Detail Page

You can modify the look of the detail page by adding XSL to the `apptransform-detail.xml` file. This overrides the XSL in the `/application/lib/transform-detail.xml` file. For example, you want to change the links in the Oscars application to the Wikipedia pages from this:

James Stewart

James Maitland Stewart (May 20, 1908 – July 2, 1997), popularly known as **Jimmy Stewart**, was an [American film](#) and [stage actor](#) best known for his self-effacing persona. Over the course of his career, he starred in many films widely considered classics and was nominated for five [Academy Awards](#), winning one in competition and one Lifetime Achievement award. He was a major [MGM](#) contract star. He also had a noted military career, rising to the rank of [Brigadier General](#) in the [United States Air Force Reserve](#).

to look like this:

James Stewart

James Maitland Stewart (May 20, 1908 – July 2, 1997), popularly known as **Jimmy Stewart**, was an [American](#) [film](#) and [stage](#) [actor](#) best known for his self-effacing persona. Over the course of his career, he starred in many films widely considered classics and was nominated for five [Academy Awards](#), winning one in competition and one Lifetime Achievement award. He was a major [MGM](#) contract star. He also had a noted military career, rising to the rank of [Brigadier General](#) in the [United States Air Force Reserve](#).

You can add the following code to the `apptransform-detail.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet extension-element-prefixes="xdmp" version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:config="http://marklogic.com/appservices/config"
  xmlns:translate="http://marklogic.com/translate"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns:xdmp="http://marklogic.com/xdmp">
  <xsl:import href="../../lib/transform-detail.xml"/>

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

4.3.7 Customizing the Generated Search Options Node

Application Builder generates an options node which is passed into the various search API calls. You can modify the options node in the generated application by means of the `$OPTIONS` and `$ADDITIONAL-OPTIONS` variables in the `appfunctions.xqy` file. For details about the options node and about the Search API, see [Search API: Understanding and Using](#) in the *Search Developer's Guide*.

This section describes the process for adding options to the `options` node and modifying existing options in the `options` node:

- [Adding a searchable-expression Option](#)
- [Modifying the transform-results Option](#)

4.3.7.1 Adding a searchable-expression Option

If you do not want to search over the whole database, you can add a `searchable-expression` option in the `search:options` node. For example, if you want the application to return `/root/my-node`, you can add a `searchable-expression` to the `options` node by adding it to the `$ADDITIONAL-OPTIONS` variable in the `appfunctions.xqy` file:

```
declare variable $ADDITIONAL-OPTIONS := (  
  <searchable-expression  
    xmlns="http://marklogic.com/appservices/search">  
      /root/my-node  
    </searchable-expression>  
  );
```

It is a good idea for the `searchable-expression` value to be a fragment root. If you do search below a fragment root, it is a good idea to filter your output. These settings will affect the accuracy of the various counts in the search results for your application. You can set whether to filter results in the search options, as described in “Modifying Search Options” on page 30. For more details on filtered and unfiltered searches, see [Fast Pagination and Unfiltered Searches](#) in *Query Performance and Tuning Guide*.

4.3.7.2 Modifying the transform-results Option

Snippeting is often specific to a particular content set or application. You can modify the `transform-results` option in the `options` node to specify that the Search API execute your own snippet code to generate the result display.

For example, the following option specifies that the application use the `my-snippet` function in the library module `/my-module.xqy` under the App Server root.

```
<transform-results apply="my-snippet" ns="my-namespace"
  at="/my-module.xqy" />
```

To modify the `transform-results` option, copy the entire `options` node from the `config.xqy` file and paste it into the `$OPTIONS` variable in the `appfunctions.xqy` file. Then change the `transform-results` option as follows:

```
declare variable $OPTIONS := (
  <options xmlns="http://marklogic.com/appservices/search">
    .....
    <transform-results apply="my-snippet" ns="my-namespace"
      at="/my-module.xqy" />
    .....
  </options>
);
```

For more details about modifying snippets, including the function signature that your code must be compatible with, see [Modifying Your Snippet Results](#) in the *Search Developer's Guide*.

4.4 Making Further Modifications to the Application

The generated application is an XQuery application, and you can make any kind of modifications to it that you want. It is designed to make modifications like the ones already described in “Customizing Applications Generated by Application Builder” on page 45 and “Customizing the Generated Search Options Node” on page 53. There are many more such modifications you can make, and you can also take the application code as a starting point and make more substantial modifications. Consider your application requirements and decide what is level of modification is right for your application.

4.5 Removing Modifications to an Application

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

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

The screenshot shows the 'ApplicationBuilder' interface with the 'MarkLogic Application Services' logo. The top navigation bar includes tabs for 'Appearance', 'Search', 'Sorting', 'Results', 'Content', and 'Deploy' (which is highlighted with a red callout). On the right, there are links for 'Oscars', 'Home', and 'Help'. The main content area is titled 'Compile and Launch Application' and instructs the user to 'Specify the App Server on which to deploy your configured application.' There are two radio button options: 'Existing App Server' (unselected) and 'New App Server' (selected). Below 'Existing App Server' is a dropdown menu. Below 'New App Server' are input fields for 'Name' (containing 'CleanApp') and 'Port' (containing '8007'). At the bottom, there are three buttons: 'Deploy' (with a mouse cursor over it), 'Back', and 'Resample'. A 'Next' button is also visible on the right side of the form.

5.0 Technical Support

MarkLogic provides technical support according to the terms detailed in your Software License Agreement or End User License Agreement. For evaluation licenses, MarkLogic may provide support on an “as possible” basis.

For customers with a support contract, we invite you to visit our support website at <http://support.marklogic.com> to access information on known and fixed issues.

For complete product documentation, the latest product release downloads, and other useful information for developers, visit our developer site at <http://developer.marklogic.com>.

If you have questions or comments, you may contact MarkLogic Technical Support at the following email address:

support@marklogic.com

If reporting a query evaluation problem, please be sure to include the sample XQuery code.