Voyager® 7.0 WebVoyáge® Architecture and Configuration User's Guide

April 2008
Draft

Ex Libris (USA) Inc.
This document contains proprietary information of
Ex Libris (USA) Inc. and is not to be disclosed or used
except in accordance with applicable agreements.

Copyright © 2008 Ex Libris (USA) Inc.
Unpublished and Not for Publication
All Rights Reserved

Copyright © 2008 Ex Libris (USA) Inc.

This material is protected by the copyright laws of the United States and other countries. It may not be reproduced, distributed, or altered in any fashion by any entity (either internal or external to Ex Libris (USA) Inc.), except in accordance with applicable agreements, contracts or licensing.

CONFIDENTIAL INFORMATION

The information herein is the property of Ex Libris (USA) Inc. or its affiliates and any misuse or abuse will result in economic loss.

This document is provided for limited and restricted purposes in accordance with a binding contract with Ex Libris (USA) Inc. or an affiliate. The information herein includes trade secrets and is confidential.

Notice

Every effort was made to ensure that the information in this document was complete and accurate at the time of printing. However, information is subject to change.

Trademarks

Voyager is a registered trademark of Ex Libris (USA) Inc.
WebVoyáge is a trademark of Ex Libris (USA) Inc.
UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.
Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft PowerPoint, Visual Basic, Visual C++, Win32, Microsoft Windows, the Windows logo, Microsoft Notepad, Microsoft Windows Explorer, Microsoft Internet Explorer, and Windows NT are registered trademarks and ActiveX is a trademark of the Microsoft corporation in the United States and/or other countries.
Sun, Sun Microsystems, the Sun logo, Solaris, ONC+, NFS, SunVTS, VIS, JumpStart, Sun Fire, StorEdge, Sun Fireplane Interconnect, WebNFS, Java, Solstice, Solstice AdminSuite, Sun Management Center, Sun Cluster, JVM, ShowMeHow, ShowMe TV, Solaris Webstart, Sun Remote Systems Control, Solaris PC Netlink, and Open Windows are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.
The Mozilla trademarks include, among others, the names Mozilla®, mozilla.org®, Firefox®, Thunderbird™, Bugzilla™, Camino®, Sunbird®, Seamonkey®, Foxkeh™ and XUL™, as well as the Firefox logo, Thunderbird logo, SeaMonkey logo, Camino logo, Foxkeh logo and the Mozilla red lizard logo.
Unicode and the Unicode logo are registered trademarks of Unicode, Inc.
Google is a trademark of Google Inc.
WorldCat is a registered trademark provided by OCLC Online Computer Library Center, Inc. on behalf of its member libraries.
Amazon.com is a registered trademark of Amazon.com, Inc.
Other products and corporate names mentioned herein which are trademarks of a third party are used only for explanation and for the owners' benefit and with no intent to infringe.
Use of third party products or information referred to herein is at the user's risk.

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Contents

DRAFT

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Figures

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Figures

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Figures

DRAFT

Figures

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Procedures

DRAFT

Procedures

DRAFT

Procedures

DRAFT

Procedures

DRAFT

Ex Libris (USA) Inc.
See notice on first page

About This Document

DRAFT

Purpose

The purpose of the *WebVoyáge Architecture and Configuration User's Guide* is to describe WebVoyáge files, their relationship, and configuration options by example.

Given the programming design used for the new user interface, there is considerable flexibility in customizing the online public access catalog (OPAC) to your preferences. Key to this customization is experience with coding cascading style sheets (CSS), XSL, XML, and/or JavaScript.

The design of this user's guide is intended for learn by example. As a result, the *WebVoyáge Architecture and Configuration User's Guide* incorporates several chapters of specific "how to" instructions.

Intended Audience

This document is intended for programmers who are customizing WebVoyáge in CSS, XSL, XML, and/or JavaScript.

Reason for Reissue

This is the first release of the *WebVoyáge Architecture and Configuration User's Guide*.

Ex Libris (USA) Inc.
See notice on first page

Document Summary

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Conventions Used in This Document

DRAFT

The following conventions are used throughout this document:

- Names of commands, variables, stanzas, files, and paths (such as `/dev/tmp`), as well as selectors and typed user input, are displayed in `constant width` type.
- Commands or other keyboard input that must be typed exactly as presented are displayed in **`constant width bold`** type.
- Commands or other keyboard input that must be supplied by the user are displayed in ***`constant width bold italic`*** type.
- System-generated responses such as error messages are displayed in `constant width` type.
- Variable *portions* of system-generated responses are displayed in *`constant width italic`* type.
- Keyboard commands (such as **Ctrl** and **Enter**) are displayed in **bold**.
- Required keyboard input such as "Enter **`vi`**" is displayed in **`constant width bold`** type.
- Place holders for variable portions of user-defined input such as **`ls -l`** ***`filename`*** are displayed in ***`italicized constant width bold`*** type.
- The names of menus or status display pages and required selections from menus or status display pages such as "From the **Applications** drop-down menu, select **System-wide**," are displayed in **bold** type.
- Object names on a window's interface, such as the **Description** field, the **OK** button, and the **Metadata** tab, are displayed in **bold** type.
- The titles of documents such as *Acquisitions User's Guide* are displayed in *italic* type.

- Caution, and important notices are displayed with a distinctive label such as the following:

NOTE:
Extra information pertinent to the topic.

IMPORTANT:
Information you should consider before making a decision or configuration.

CAUTION:
Information you must consider before making a decision, due to potential loss of data or system malfunction involved.

TIP:
Helpful hints you might want to consider before making a decision.

RECOMMENDED:
Preferred course of action.

OPTIONAL:
Indicates course of action which is not required, but may be taken to suit your library's preferences or requirements.

Document Reproduction/Photocopying

Photocopying the documentation is allowed under your contract with Ex Libris (USA) Inc. It is stated below:

All documentation is subject to U.S. copyright protection. CUSTOMER may copy the printed documentation only in reasonable quantities to aid the employees in their use of the SOFTWARE. Limited portions of documentation, relating only to the public access catalog, may be copied for use in patron instruction.

Comment on This Document

Please contact Customer Support to provide us with your feedback. For Customer Support contact information, see SupportWeb at:

DRAFT

`http://www.exlibrisgroup.com/support_center.htm`.

To Submit Comments by E-mail

To submit comments by e-mail, please send your message to:

`docmanager@exlibrisgroup.com`

DRAFT

Ex Libris (USA) Inc.
See notice on first page

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Getting Started

1

Contents

DRAFT

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Getting Started 1

Purpose of this Chapter

The purpose of this chapter is to describe what you need to get started to effectively work with/customize WebVoyáge and use this guide.

Prerequisite Skills and Knowledge

To use this document effectively, you should have a working knowledge of the following:

- Microsoft Windows operating environment.
- CSS.
- XML.
- XSL.
- JavaScript.
- Text editor(s) for working with CSS, XML, XSL, and so on.
- UNIX operating system commands and file system (depending on your environment).
- Basic MARC records formats.
- Local procedures.

DRAFT

Before You Begin

Before you can begin, you need to do the following:

- Have the Voyager WebVoyáge 7.0 and corresponding Voyager 7.0 integrated library system software installed.
- Have access to an internet browser on your PC.
- Obtain the URL and/or the IP and port address for accessing your instance of Voyager WebVoyáge 7.0.
- Obtain your user ID and password for logging in to Voyager WebVoyáge 7.0. For login steps, refer to the *WebVoyáge Basic User's Guide*.
- Set up your PC to display Unicode-specific data as needed. See the *WebVoyáge Basic User's Guide* for instructions.

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Architecture

2

Contents

DRAFT

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Architecture 2

WebVoyáge Architecture Overview

The purpose of this section is to provide an overview description of the architecture of WebVoyáge for displaying information.

WebVoyáge has a modular design to control formatting for the broadest number of page displays.

To display search results, patron information, and other dynamically generated information, WebVoyáge combines information from the Voyager database (or from an outside resource like Google^{TM} Book Search) with formatting properties from the WebVoyáge `.css`, `.xsl`, and `.xml` files to render a display page in HTML. See Figure 2-1 on page 2-2.

For a flowchart example of how these files work together, see Flowchart Example - myAccount Page on page 2-2.

For an example and description of the HTML page components, see Page Components Example - Basic Search on page 2-5.

DRAFT

Ex Libris (USA) Inc.
See notice on first page

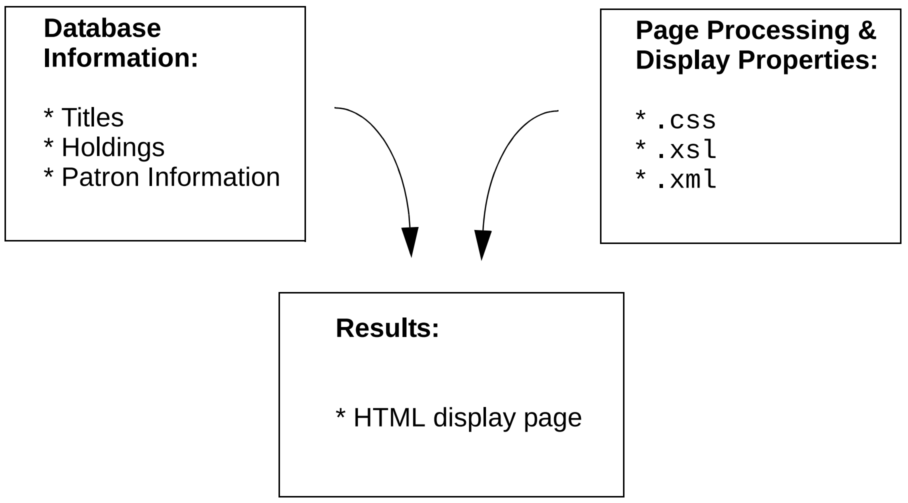


Figure 2-1. Display page build overview

DRAFT

Flowchart Example - myAccount Page

The purpose of this section is to describe and illustrate the relationship of the WebVoyáge files used for building and displaying content using the myAccount page as an example.

Flowchart Example Description

The content of this section describes the the flowchart example highlighted in Figure 2-2 on page 2-4.

To build the display page for myAccount page, the primary `.xsl` file is used. For this example, it is the `myAccount.xsl` file that is used. The name of the .xsl file used for this process generally represents the page being built such as `displayRecord.xsl`. See the .xsl files located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/`.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name

that is used at your site. The default skin path provided is en_US as in the following:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

The `myAccount.xsl` imports the following:

- `stdImports.xsl`.

  This is an important part of the page processing. Every page uses it. It imports `frameWork.xsl`.
- `cl_myAccount.xsl`.

  This is a key component. The `cl` stands for content layout.
- `myAccountLinks.xsl`.
- `myAccount.css`.

NOTE:
The file naming convention used is intentional to show relationships between files used to build the HTML page as with **`myAccount`**`.xsl`, **`myAccount`**`Links.xsl`, `cl_`**`myAccount`**`.xsl`, and **`myAccount`**`.css`.

The `myAccount.xsl` calls the following templates which are used to construct every page:

- `buildHtmlPage`.
- `buildContent`.

The `frameWork.xsl` takes input from the following components:

- `.xml` files.
- `.js` files.
- `.css` files.
- `buildHtmlPage` template.
- Timeout mechanism.

And subsequently, the `frameWork.xsl` generates HTML output with the following page components:

- Header.
- Main content based on the `buildContent` template.
- Footer.

For more information regarding page components, see Page Components Example - Basic Search on page 2-5.

DRAFT

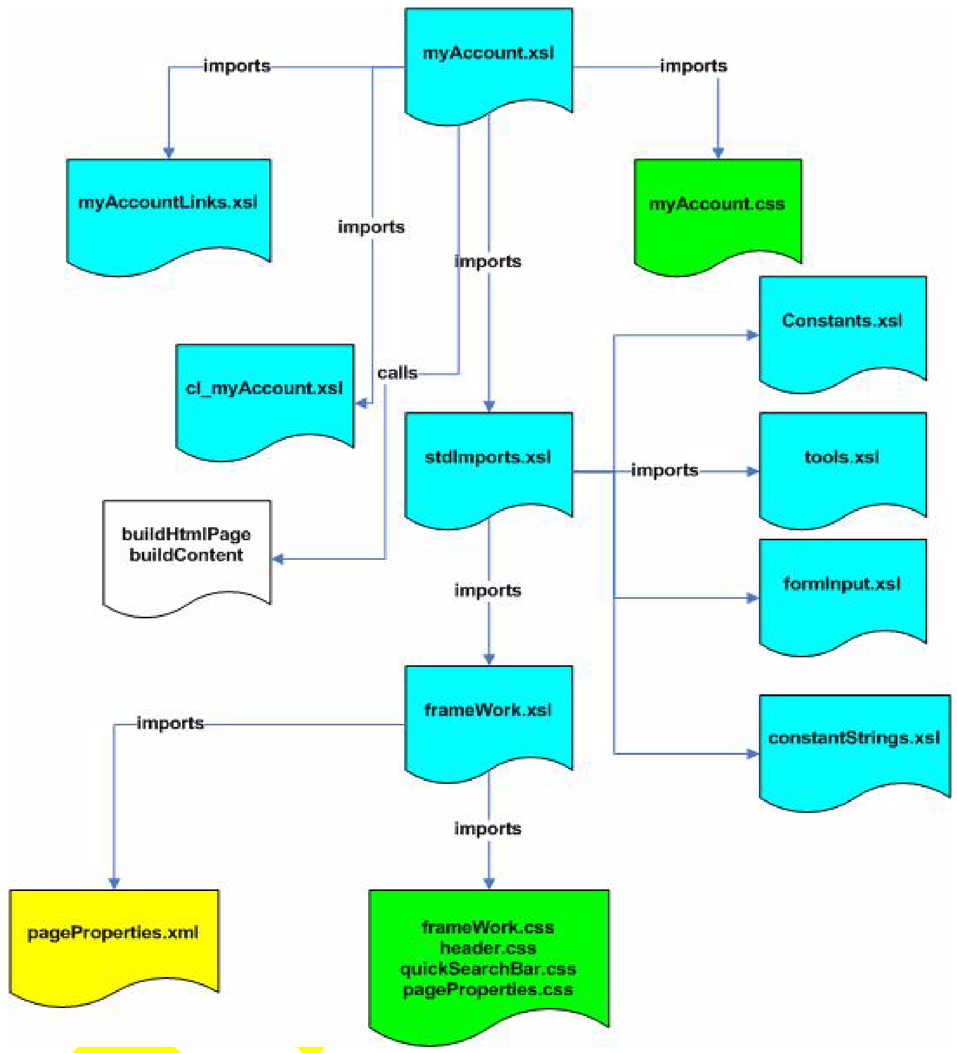


Figure 2-2. Flowchart example for myAccount page

.css Processing

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document. CSS is commonly used to apply style to web pages or more specifically colors, fonts, layout, and other aspects of document presentation. These are processed in a specified priority scheme or hierarchy that determines which style rules apply. As a result, pieces of one stylesheet may be overridden by another stylesheet.

In WebVoyáge, the baseline defaults are provided by the following `.css` files that are used by almost every page that is generated:

1. `frameWork.css`.
2. `header.css`.
3. `quickSearchBar.css`.
4. `pageProperties.css`.

In the above hierarchy, the items #1 through #3 have a higher priority scheme and take precedence while `pageProperties.css` provides overriding characteristics to #1 through #3. Specific page .css files like `myAccount.css` always override the baseline defaults.

Variations in a specific page `.css` file only applies to that page. For more global changes like a font change for all pages, the files controlling the baseline defaults need to be modified.

TIP:
A tool like Mozilla® Firebug enables you to view these variations/ interactions and edit/debug CSS, JavaScript, and so on live in any web page.

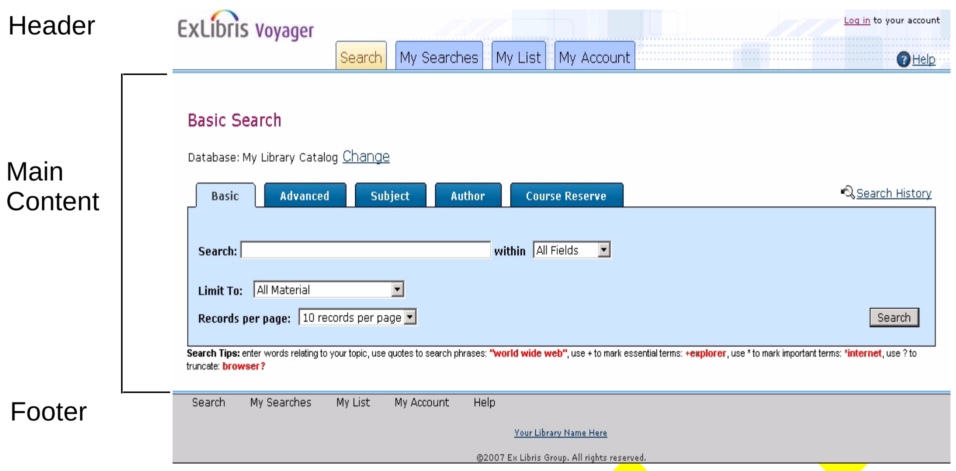
DRAFT

Page Components Example - Basic Search

As described, each page includes the following major components:

- Header.
- Footer.
- Main content.

See Figure 2-3.



Search Tips: enter words relating to your topic, use quotes to search phrases: "world wide web", use + to mark essential terms: +explorer, use * to mark important terms: *internet, use ? to truncate: browser?

Figure 2-3. Page components

Any element of WebVoyáge that is used by multiple pages such as the footer component, the search navigation bar, or the login link is defined independently of the pages on which they appear. An independent definition allows the element to be called by any and every page. This provides greater flexibility; but as a result, there isn't a single `.xsl` stylesheet for each page that WebVoyáge renders.

The page shown in Figure 2-3 is built with the following files that are used by all pages.

- `frameWork.xsl`.
- `constants.xsl`.
- `tools.xsl`.
- `formInput.xsl`.
- `constantStrings.xsl`.
- `frameWork.css`.
- `header.css`.
- `quickSearchBar.css`.
- `pageProperties.css`.

In addition, the **Basic Search** page uses the following:

- `cl_searchBasic.xsl`.

DRAFT

- `searchFacets.xsl`.
- `searchPages.css`.
- `searchBasic.css`.
- `pageInputFocus.js`.

Specific to the **Basic** tab, it uses the following:

- The font family is from `frameWork.css`.
- The font size, color, weight, and alignment of the tab label are from the `searchPages.css`.
- The font size, color, weight, and alignment of the tab contents such as Limit To are from `frameWork.css`.
- The font family, size, color, weight, and alignment of the search tips are from `pageProperties.css`.
- The text values are from `webvoyage.properties`.
- The images that make up the **Basic** tab are from the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/` directory.
- The cursor placement is determined by `pageInputFocus.js`.

In summary, this example illustrates the hierarchy described in Flowchart Example - myAccount Page on page 2-2 where baseline defaults are used in combination with page-specific, overriding controls for formatting that includes the following:

- Fonts.
- Color.
- Images.
- Content placement on the page.
- And so on.

DRAFT

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Build A Separate Display For Serials?

3

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Build A Separate Display For Serials?

3

DRAFT

Description For "How Do I Build A Separate Display For Serials?" Example

By default, WebVoyáge uses a single `displaycfg.xml` file to display all MARC bibliographic records.

The instructions for this example allow you to create and use a separate configuration file for serials based on the MARC leader value.

This model can also be used to create different MARC views for any material type.

Files

The example in this chapter uses the following files:

- `sdisplaycfg.xml` (new for this example).
- `cl_displayRecord.xsl`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Ex Libris (USA) Inc.
See notice on first page

Procedure 3-1. Building Separate Display for Serials

Use the following procedure to build a separate display for serials.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Create and save a new file in the /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/ directory named sdisplaycfg.xml to be used to specify the display fields for your serials records. Use the sample lines of code shown in Figure 3-1 for this example.

```
#(c)#==================================================================
#(c)#
#(c)#       Copyright 2008 ExLibris Group
#(c)#                       All Rights Reserved
#(c)#
#(c)#==================================================================
-->
<!--
**        Product : WebOpac : displaycfg
**        Version : 7.0
**        Created : 17-OCT-2007
**    Orig Author : David Sellers
**  Last Modified : 11-APR-2008
**Last Modified By: ASP

-->
<display>
        <titleTags>
```

Figure 3-1. Sample line of code for sdisplaycfg.xml

DRAFT

```
        <displayTag field="245" indicator1="X" indicator2="X"
   subfield="ab"/>
</titleTags>


<displayTags  label="Title:">
        <displayTag field="245" indicator1="X" indicator2="X"
   subfield="abcfknps" />
</displayTags >


<displayTags  label="Also Called:">
        <displayTag field="246" indicator1="X" indicator2="X"
   subfield="abfnp" />
</displayTags >


<displayTags  label="Continues:">
        <displayTag field="780" indicator1="0" indicator2="0"
   subfield="at" />
        <displayTag field="780" indicator1="0" indicator2="1"
   subfield="at" />


</displayTags >
<displayTags  label="Supersedes:">
        <displayTag field="780" indicator1="0" indicator2="2"
   subfield="at" />
        <displayTag field="780" indicator1="0" indicator2="3"
   subfield="at" />
</displayTags >


<displayTags label="Publisher:">
        <displayTag field="260" indicator1="X" indicator2="X"
   subfield="abc"/>
</displayTags>


<displayTags label="ISSN:">
        <displayTag field="022" indicator1="X" indicator2="X"
   subfield="a"/>
</displayTags>


<displayTags label="Format:">
        <displayTag field="000" indicator1="0" indicator2="6"
   subfield="2"/>
</displayTags>
```

Figure 3-1. Sample line of code for sdisplaycfg.xml (Continued)

DRAFT

```
        <displayTags label="Subjects:">
                <displayTag field="600" indicator1="X" indicator2="X"
           subfield="abcdefghkjlmnopqrstuvxyz">
                        <subfield value="v" preText="--"/>
                        <subfield value="x" preText="--"/>
                        <subfield value="y" preText="--"/>
                        <subfield value="z" preText="--"/>
                </displayTag>
        </displayTags>

        <displayTags label="Online Journal:">
                <displayTag field="3000"/>
        </displayTags>

    <displayTags label="Holdings Information" notFound="No holdings available -
           - check at the Circulation Desk.">
                <displayTag field="9000"/>
        </displayTags>

</display>
```

Figure 3-1. Sample line of code for sdisplaycfg.xml (Continued)

2. Make a backup copy of `cl_displayRecord.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

3. Edit `cl_displayRecord.xsl`.

   a. Add a document holders declaration (see Figure 3-3) after the lines of code shown in Figure 3-2.

```
<!-- ## Our Document Holders ## -->
<xsl:variable name="MConfig" select="document('./configs/displaycfg.xml')"/>
```

Figure 3-2. Preceding lines of code

DRAFT

```
<!-- added for serial config display -->
<xsl:variable name="SConfig" select="document('./configs/sdisplaycfg.xml')"/>
<xsl:variable name="recType" select="substring($bibRecord/hol:marcRecord/
          slim:leader,8,1)" />
```

Figure 3-3. Document holders declaration to add

b. Add logic for looking at the record leader by replacing the lines of code seen in Figure 3-4 with the example lines of code in Figure 3-5.

```
                                <!-- ## Bibliographic Data ## -->
                                                <xsl:for-each select="$Config">
                                                        <div
           class="bibliographicData">
                                                           <xsl:call-template
           name="buildMarcDisplay">
                                                                    <xsl:with-
           param name="recordType" select="'bib'"/>
                                                          </xsl:call-template>
                                                        </div>
                                                </xsl:for-each>
```

Figure 3-4. Lines to be replaced

```
<!-- ## Bibliographic Data ## -->
<xsl:choose>
              <xsl:when test="$recType='s'">
                  <xsl:for-each select="$SConfig">
                      <div class="bibliographicData">
                          <xsl:call-template name="buildMarcDisplay">
                              <xsl:with-param name="recordType"
           select="'bib'"/>
                          </xsl:call-template>
```

Figure 3-5. Replacement lines of code

DRAFT

```
                        <br />Record type:&#160;<xsl:value-of
        select="$recType"/>

                    </div>
                </xsl:for-each>
            </xsl:when>

            <xsl:otherwise >
                <xsl:for-each select="$MConfig">
                    <div class="bibliographicData">
                        <xsl:call-template name="buildMarcDisplay">
                            <xsl:with-param name="recordType"
        select="'bib'"/>
                        </xsl:call-template>


                        <br />Record type:&#160;<xsl:value-of
        select="$recType"/>

                    </div>
                </xsl:for-each>

            </xsl:otherwise>
</xsl:choose>
```

Figure 3-5. Replacement lines of code (Continued)

4. Save and test the changes you made to `cl_displayRecord.xsl`.

OPTIONAL:

5. *Back out your changes, if necessary, by deploying your backup copy of* `cl_displayRecord.xsl`*.*

DRAFT

How Do I Add Static Links To The Header Or Footer?

4

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Add Static Links To The Header Or Footer?

4

DRAFT

Description For "How Do I Add Static Links To The Header Or Footer?" Example

Both the header and footer page elements can accommodate links to external web sites, web applications, and so forth.

In this chapter, the header example demonstrates hyperlinks; and the footer example demonstrates adding a new tab.

Files

The examples in this chapter use the following files:

- `header.xsl`.
- `footer.xsl`.

Instructions

This section provides the instructions for creating the examples described in this chapter.

Procedure 4-1. Creating Header Links

Use the following procedure to create header links.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Make a backup copy of the `header.xsl` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/`.

2. Edit the `header.xsl` file to add a new template.

   a. Add your template (see Figure 4-1) immediately before the `<xsl:stylesheet>` element at the bottom of the file.

```
<!--  #################
  -->
<!--  ## buildHeaderLinks ##
  -->
<!--  ############################################################-->
<xsl:template name="localHeaderLinks">
<div id="headerLinks" style="font-family:Arial;font-size:.70em;">
  <span style="color:#9F175E;padding: 0 .20em;margin:0 .2em;">University
           Library Catalog</span>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">My library</a>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">SFX</a>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">Primo</a>
  <a href="#" style="padding: 0 .20em;margin:0 .2em;">Chat with a librarian</a>
```

Figure 4-1. Header links template

DRAFT

```
   </div>
   </xsl:template>
 <!-- ############################################################-->
 <!-- ############################################################ -->
```

Figure 4-1. Header links template (Continued)

b. Call the template that you created in Step a by adding the instruction (see Figure 4-2, line 7) to the buildHeader section located at the top of the header.xsl file.

```
Line#
1   <!-- ################# -->
2   <!-- ## buildHeader ## -->
3   <!-- ############################################################ -->
4
5   <xsl:template name="buildHeader">
6      <xsl:for-each select="/page:page/page:pageHeader">
7        <xsl:call-template name="localHeaderLinks" />
8         <div id="headerRow">
9            <div id="logo" title="{$headerText/logo}">
```

Figure 4-2. Call instruction

3. Save and test your changes to header.xsl.

OPTIONAL:

4. Back out your changes, if necessary, by deploying your backup copy of header.xsl.

DRAFT

Procedure 4-2. Creating Footer Tabs

Use the following procedure to create a footer tab.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

`/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`

1. Make a backup copy of the `footer.xsl` file that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/`.

2. Edit the `footer.xsl` file to add a new tab by adding the new tab code (see Figure 4-3, lines 15 -19) after the `<ul class="navbar">` in the `buildFooter` section in the file.

```
Line#
 1      <!-- ################# -->
 2      <!-- ## buildFooter ## -->
 3      <!-- ############################################################-->
 4
 5      <xsl:template name="buildFooter">
 6
 7         <xsl:for-each select="/page:page/page:pageFooter">
 8            <div id="pageFooter">
 9               <xsl:for-each
                   select="page:tabs[@nameId='page.footer.buttons']">
10
11                  <div id="footerTabs" title="{$footerText/footerTabs}">
12                     <a name="navFooter"></a>
13                    <h2 class="navFooter"><xsl:value-of select="$footerText/
                   footerTabs"/></h2>
14                     <ul class="navbar">
15      <!--  extra footer tabs  -->
16        <li>
17        <a href="http://www.exlibrisgroup.com/" target="_newWin">Ex Libris</
                   a>
18        </li>
19      <!--  end extra footer tabs  -->
20                             <xsl:for-each select="$Configs/pageConfigs/
                   footerTabDisplayOrder/tab">
```

Figure 4-3. Footer tab code example

DRAFT

3. Save and test your changes to `footer.xsl`.

OPTIONAL:

4. *Back out your changes, if necessary, by deploying your backup copy of* `footer.xsl`*.*

DRAFT

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Remove Information From A Page?

5

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Remove Information From A Page?

5

Description For “How Do I Remove Information From A Page?” Example

There may be some page elements you want to disable or prevent from displaying. This example describes how to remove the **Item Type** column from the **Charged Items** table on the **My Account** page.

NOTE:
It's important to remove all the relevant pieces of a page element. In this example, the instructions comment out both the heading and table cell pieces of the **Item Type** column. Commenting out only one isn't sufficient.

Files

The example in this chapter uses the `cl_myAccount.xsl` file.

Instructions

This section provides the instructions for completing the example described in this chapter.

DRAFT

Procedure 5-1. Removing Information From a Page

Use the following procedure to remove information from a page.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Make a backup copy of `cl_myAccount.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

2. Edit `cl_myAccount.xsl`.

   a. Find the `displayChargedItems` template section marked by the comment shown in Figure 5-1.

```
<!-- ###################### -->
<!-- ## displayChargedItems ## -->
<!-- ############################################################ -->
<xsl:template name="displayChargedItems">
```

Figure 5-1. Section comment to locate

   b. Comment out the table heading and table cell lines of code relevant to item type. See Figure 5-2 and Figure 5-3.

```
<!-- <th id="cellChargedType"><xsl:value-of select="page:element/
           page:heading[@nameId='type']"/></th> -->
```

Figure 5-2. Table heading example

DRAFT

```
<!-- <td id="tableCell" headers="cellChargedType"><xsl:value-of
           select="page:itemType"/>&#160;</td> -->
```

Figure 5-3. Table cell example

3. Save and test your changes.

OPTIONAL:

4. *Back out your changes, if necessary, by deploying your backup copy of* `cl_myAccount.xsl`*.*

DRAFT

Ex Libris (USA) Inc.
See notice on first page

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Add A Map Or Other Information To A Location?

6

Contents

DRAFT

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Add A Map Or Other Information To A Location?

6

DRAFT

Description For "How Do I Add A Map Or Other Information To A Location?" Example

This example allows you to direct your patrons to add a map to a location and/or other applicable information like the hours of the reading room. The information offered is based on the MFHD location (852 |b).

Files

The example in this chapter uses the following files:

- local_locMapLink.xsl (new for this example).
- `display.xsl`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 6-1. Adding a Map to a Location and/or Other Applicable Information

Use the following procedure to add a map to a location and/or other applicable information.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Create and save a new file in the /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/ directory named local_locMapLink.xsl to define how to build the hyperlink. Use the sample lines of code shown in Figure 6-1 for this example.

NOTE:
Notice the template name locMapLink in this example.

```
<!--
**                  Note: sample link to map based on loc code
**                  Version : 1.0
**                  Created : 16-Nov-2007
**                  Created By : EJW
-->

<xsl:stylesheet version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
              xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/
          page"
              xmlns:fo="http://www.w3.org/1999/XSL/Format">

<!-- ###################################################################### --
          >
```

Figure 6-1. Example code for building hyperlink

DRAFT

```
<xsl:template name="locMapLink">
<xsl:param name="mfhd"/>
              <xsl:variable name="locCode">
                   <xsl:call-template name="BMDProcessMarcTags">
                    <xsl:with-param name="field" select="'852'"/>
                    <xsl:with-param name="indicator1" select="'X'"/>
                    <xsl:with-param name="indicator2" select="'X'"/>
                    <xsl:with-param name="subfield" select="'b'"/>
                    <xsl:with-param name="mfhdID" select="$mfhd"/>
                    <xsl:with-param name="recordType" select="'mfhd'"/>
                </xsl:call-template>
              </xsl:variable>



              <!-- you must create your web site to display maps -->
              <xsl:variable name="baseURL">http://www.exlibrisgroup.com/
           ?loc=</xsl:variable>

                      <div class="locationMap">
                          Show me a&#160;<a id="locMap"
           href="{$baseURL}{$locCode}" target="_new">map</a>.
                      </div>



</xsl:template>

<!-- ###################################################################### --
           >

</xsl:stylesheet>
```

Figure 6-1. Example code for building hyperlink (Continued)

2. Make a backup copy of `display.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/display/`.

DRAFT

3. Edit `display.xsl`.

   a. Add a statement at the top of the `display.xsl` file that contains the path to the `local_locMapLink.xsl` file. See Figure 6-2.

```
<xsl:import href="../display/marc21slim.xsl"/>
<xsl:import href="../configs/104X_display.xsl"/>
<xsl:import href="../local_locMapLink.xsl"/>
```

Figure 6-2. Example code for adding path statement

   b. Call the `locMapLink` template from within the `BMD100` template. See Figure 6-3.

```
<!--################################################################### -->


<xsl:template name="BMD1000">
<xsl:param name="mfhdID"/>


    <xsl:for-each select="$HoldXML/hol:holdingsRecord/hol:mfhdCollection/
           mfhd:mfhdRecord[@mfhdId = $mfhdID]/
           mfhd:mfhdData[@name='locationDisplayName']">
        <xsl:if test="string-length(.)">
            <xsl:value-of select="."/>
                  <!-- ## add a map link ## -->
                  <xsl:call-template name="locMapLink" >
                  <xsl:with-param name="mfhd" select="$mfhdID"/>
                  </xsl:call-template>
                  <br/>
        </xsl:if>
    </xsl:for-each>
</xsl:template>


<!-- ################################################################### -->
```

Figure 6-3. Example of call for locMapLink

4. Save and test your changes.

DRAFT

OPTIONAL:

5. *Back out your changes, if necessary, by deploying your backup copy of* `display.xsl`*.*

DRAFT

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Create An External Search From A Bibliographic Record Display?

7

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Create An External Search From A Bibliographic Record Display?

7

DRAFT

Description For "How Do I Create An External Search From A Bibliographic Record Display?" Example

It is common to use the ISBN as a parameter to a new search in a different application such as Amazon.com®, WorldCat®, and so on.

The instructions in this chapter describe how to parse out the ISBN from a bibliographic record display and insert it into a URL.

The URL is displayed in the Action Box on the item display page.

This model can also be used to extract different pieces of the MARC record and construct different or multiple URLs.

Files

The example in this chapter uses the following files:

- `isbnSearch.xsl` (new for this example).
- `displayFacets.xsl`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 7-1. Create External Search From Bibliographic Record Display

Use the following procedure to create an external search from a bibliographic record display.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new file in the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/` directory named `isbnSearch.xsl` to be used to specify how to extract the ISBN from the bibliographic record. Use the sample lines of code shown in Figure 7-1 for this example.

NOTE:
Notice the template name `recordIsbnSearch` in this example.

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

<!--
**                  Note: Creates a link to WorldCat using the ISBN
**                  Version : 1.0
**                  Created : 15-APR-08
**                  Created By : ASP
-->
```

Figure 7-1. Sample code to extract ISBN from bibliographic record

DRAFT

```
<xsl:stylesheet version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
              xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/
           page"
              xmlns:fo="http://www.w3.org/1999/XSL/Format">


<!-- ###################################################################### --
           >


<xsl:template name="recordIsbnSearch">
              <xsl:variable name="isbn">
              <xsl:call-template name="BMDProcessMarcTags">
                    <xsl:with-param name="field" select="'020'"/>
                    <xsl:with-param name="indicator1" select="'X'"/>
                    <xsl:with-param name="indicator2" select="'X'"/>
                    <xsl:with-param name="subfield" select="'a'"/>
                    <xsl:with-param name="mfhdID" select="$bibID"/>
                    <xsl:with-param name="recordType" select="'bib'"/>
                </xsl:call-template>
              </xsl:variable>


<a target="_blank">
   <xsl:attribute name="href">http://worldcatlibraries.org/XXXX/isbn/
           <xsl:value-of select="$isbn"/>&amp;loc=united+states</
           xsl:attribute>
Check Other Local Libraries</a>


</xsl:template>
</xsl:stylesheet>
```

Figure 7-1. Sample code to extract ISBN from bibliographic record (Continued)

DRAFT

2. Make a backup copy of `displayFacets.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/`.

3. Edit `displayFacets.xsl`.

   a. Define where the `isbnSearch.xsl` can be found with a line of code immediately after the namespace declarations. See Figure 7-2.

```
<xsl:stylesheet version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
              xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/
           page"
              xmlns:fo="http://www.w3.org/1999/XSL/Format">


              <xsl:include href="./isbnSearch.xsl"/>
```

Figure 7-2. Define isbnSearch.xsl location example

b. Call the template defined in `isbnSearch.xsl`.

For this example, it is added to the bottom of the Action Box.

Working from the end of the file, add the call before the final `</div>`. See Figure 7-3.

```
                      <!--/ul-->
                      <!-- ## add the other libraries search ## -->
                          <xsl:call-template name="recordIsbnSearch" />


                  </div>
              </xsl:for-each>



</xsl:template>

<!-- ###################################################################### --
           >

</xsl:stylesheet>
```

Figure 7-3. Call for isbnSearch.xsl in displayFacets.xsl

4. Save and test your changes.

DRAFT

OPTIONAL:

5. *Back out your changes, if necessary, by deploying your backup copy of* `displayFacets.xsl`*.*

DRAFT

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?

8

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?

8

DRAFT

Description For "How Do I Dynamically Disable Limits And Change Search Tips Based On The Selected Search Index?" Example

The **Basic Search** page includes a **Limit To** dropdown list that is compatible with keyword searches. The instructions in this chapter explain how to install a JavaScript that disables the **Limit To** dropdown list when you select an index that is incompatible with limits such as headings and call number.

The JavaScript also changes the search tips displayed to the user based on the index selected. This provides you with the option to offer hints on improving search strategies.

Files

The example in this chapter uses the following files:

- `searchBasic.js` (new for this example).
- `cl_searchBasic.xsl`.
- `pageProperties.xml`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 8-1. Disable Limits and Change Search Tips

Use the following procedure to disable limits and change search tips dynamically based on the selected search index.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new JavaScript file in the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/jscripts/` directory named `searchBasic.js` to be used to disable the quick limits dropdown list. Use the sample lines of code shown in Figure 8-1 for this example.

```
/*
#(c)#===========================================================================
#(c)#
#(c)# Copyright 2007 ExLibris Group
#(c)# All Rights Reserved
#(c)#
#(c)#===========================================================================

** Product : WebVoyage :: disable/enable limits on basic search, adjust search
           tips
** Version : 7.0
** Created : 23-JAN-2008
** Orig Author : EJW
```

Figure 8-1. Example code for searchBasic.js

DRAFT

```
** Last Modified : 18-APR-2008
** Last Modified By : ASP
*/

////////////////////////////////////////////////////////////////////////
function updateSearchTip ()
{
        // added to searchCode.onchange to customize search tip based on
    index

        // save the defautl search tip the first time through
        defaultTip = window.defaultTip  ||
    document.getElementById('customSearchTip').innerHTML;

        currentSearchCode = document.getElementById('searchCode').value

        switch (currentSearchCode)
        {
        case 'CMD':
        case 'CMD*':
            document.getElementById('customSearchTip').innerHTML =
    "build a simple Boolean search: <span class=\"example\">(cats or
    dogs) and therapy</span>"
        break;

        case 'NAME+':
        case 'AUTH+':
            document.getElementById('customSearchTip').innerHTML =
    "search by personal or corporate author: last name first <span
    class=\"example\">pessl marisha</span>, or company name <span
    class=\"example\">jung seed</span>"
        break;

        case 'CALL+':
            document.getElementById('customSearchTip').innerHTML =
    "enter as much of the call number that you know: <span
    class=\"example\">PR 1297</span>"
        break;


        default:
```

Figure 8-1. Example code for searchBasic.js (Continued)

D R A F T

```
                    // use the default tip if not otherwise overridden
                    document.getElementById('customSearchTip').innerHTML =
           defaultTip;
               }



}

function searchCodeChanged ()
{
/*
** Disable the limits drop for searches that do not support it.
*/

currentSearchCode =
           document.getElementById('searchCode').value.substring(0,4);

if (!document.getElementById('limitTo').disabled) {
currentLimit = document.getElementById('limitTo').value;
}

switch(currentSearchCode)
 {
// OPAC HEADING INDEXES
case 'SUBJ':
case 'TITL':
case 'NAME':
case 'AUTH':
// MAIN CALL NUMBER INDEX
case 'CALL':
// JOURNAL INDEX WITH PRELIMITS
case 'JKEY':
case 'JALL':

 document.getElementById('limitTo').value="none";
 document.getElementById('limitTo').disabled=true;
 break;
```

Figure 8-1. Example code for searchBasic.js (Continued)

DRAFT

```
default:
 document.getElementById('limitTo').disabled=false;
 document.getElementById('limitTo').value=currentLimit;
}


}


function addSearchCodeChanged ()
{
               document.getElementById('searchCode').onchange = function()
           {searchCodeChanged(); updateSearchTip();}
               // run the script to catch default index doesn't support limits
           or edit search
               searchCodeChanged ();
               updateSearchTip();
}



function addLoadEvent(func) {
  var oldonload = window.onload;
  if (typeof window.onload != 'function') {
    window.onload = func;
  } else {
    window.onload = function() {
      if (oldonload) {
        oldonload();
      }
      func();
    }
  }
}

addLoadEvent(addSearchCodeChanged);
```

Figure 8-1. Example code for searchBasic.js (Continued)

2. Make a backup copy of `pageProperties.xml` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/userTextConfigs/`.

3. Edit `pageProperties.xml`.

DRAFT

Locate the comment `<!-- ## Start Search Tips ## -->` and identify the `<page name="page.searchBasic" position="belowContent">` element.

Add a `<div>` element as in Figure 8-2.

```
    <page name="page.searchBasic" position="belowContent">
        <div class="searchTip">
            <span class="label">Search Tips: </span>
             <!-- special div to enable javascript to swap out help text -->
                  <div  id="customSearchTip" style="display:inline">
                enter words relating to your topic, use quotes to search
           phrases: <span class="example">"world wide web"</span>,
                use + to mark essential terms:  <span
           class="example">+explorer</span>,
                use * to mark important terms:  <span
           class="example">*internet</span>,
                use ? to truncate:  <span class="example">browser?</span>
               </div>
        </div>
    </page
```

Figure 8-2.   Example <div> element

4. Save your changes.

5. Make a backup copy of `cl_searchBasic.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

6. Edit `cl_searchBasic.xsl`.

   At the end of the `buildBasicSearch` template (see Figure 8-3), load the `searchBasic.js` JavaScript file you created.

DRAFT

```
<!-- ##################### -->
<!-- ## buildSearchForm ## -->
<!-- ########################################################## -->

<xsl:template name="buildBasicSearch">

   <div id="searchParams">
      <div id="searchInputs">
         <xsl:call-template name="buildFormInput">
            <xsl:with-param name="eleName"  select="'searchArg'"/>
            <xsl:with-param name="size"  select="'51'"/>
            <xsl:with-param name="accesskey"  select="'s'"/>
         </xsl:call-template>
         <xsl:call-template name="buildFormDropDown">
            <xsl:with-param name="eleName"  select="'searchCode'"/>
         </xsl:call-template>
      </div>
      <div id="quickLimits">
         <xsl:call-template name="buildFormDropDown">
            <xsl:with-param name="eleName"  select="'limitTo'"/>
         </xsl:call-template>
      </div>

      <!-- load javascript file for handling limits enable/disable -->
      <script type="text/javascript" src="{$jscript-loc}searchBasic.js"/>

   </div>

   <xsl:call-template name="buildSearchButtons"/>

</xsl:template>

<!-- ########################################################## -->
```

Figure 8-3. Locate buildBasicSearch template

7. Save and test your changes.

DRAFT

OPTIONAL:

8. *Back out your changes, if necessary, by deploying your backup copies of* `pageProperties.xml` *and* `cl_searchBasic.xsl`*.*

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Disable AutoComplete? 9

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Disable AutoComplete? 9

DRAFT

Description For “How Do I Disable AutoComplete?” Example

AutoComplete is a feature of certain web browsers that stores information on the computer's hard drive that a user types into web page forms. When you begin filling in another form, the browser suggests possible answers from information stored on the hard drive.

Particularly at public workstations, you may want to disable the browser's AutoComplete capability. This is a feature of both Internet Explorer® and Firefox®.

Files

The example in this chapter uses the following files:

- `login.xsl`.
- `searchFacets.xsl`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 9-1. Disable AutoComplete

Use the following procedure to disable AutoComplete.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

`/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`

1. Make a backup copy of `searchFacets.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/`.

2. Edit `searchFacets.xsl`.

   a. Locate the `buildTheSearchForm` section near the top of the file.

   b. Add `autocomplete=”off”` to the `<form action>` element. See Figure 9-1.

```
<form action="{$formAction}" method="GET" accept-charset="UTF-8"
          id="{$formName}" autocomplete="off">
```

Figure 9-1. Example of autocomplete=”off” code

3. Save your changes.

4. Make a backup copy of `login.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/`.

5. Edit `login.xsl`.

   a. Locate the `buildContent` section.

   b. Add `autocomplete=”off”` to the `<form action>` element. See Figure 9-2.

DRAFT

```
<form action="{/page:page//page:element[@nameId='page.logIn.logIn.button']/
           page:buttonAction}" method="GET" accept-charset="UTF-8"
           name="selectDatabases" autocomplete="off">
```

Figure 9-2. Additional example of autocomplete=”off” code

6. Save and test your changes.

OPTIONAL:

7. *Back out your changes, if necessary, by deploying your backup copies of `searchFacets.xsl` and/or `login.xsl`.*

DRAFT

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Display A Favicon? 10

Contents

DRAFT

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Display A Favicon? 10

DRAFT

Description For “How Do I Display A Favicon?” Example

In Internet Explorer and Firefox, a favicon (favorite icon) displays in the address bar, the favorites menu, bookmarks, and page tabs.

A favicon is a way to brand your catalog for your patrons.

NOTE:
There are multiple methods for installing a favicon. The method described in this chapter is specific to WebVoyáge and allows you to use any type of image files such as `.jpg`, `.gif`, or `.png` in addition to favicon `.ico` files.

For further information regarding favicons, see the following sites:

- `http://msdn2.microsoft.com/en-us/library/ms537656.aspx`.
- `http://en.wikipedia.org/wiki/Shortcut_icon`.

Files

The example in this chapter uses the `frameWork.xsl` file.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 10-1. Display favicon

Use the following procedure to display your favicon.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

`/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`

1. Create a 16x16 pixel icon.
2. Save the icon to the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/` directory.
3. Make a backup copy of `frameWork.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageTools/`.
4. Edit `frameWork.xsl`.
   a. Locate the `buildHtmlPage` template.
   b. Insert the example code shown in Figure 10-1 after the <head> element.

```
<link rel="shortcut icon" href="{$image-loc}favicon.ico" type="image/x-icon" /
                >
<link rel="icon" href="{$image-loc}favicon.ico" type="image/x-icon " />
```

Figure 10-1. Example favicon code for frameWork.xsl

   c. Replace `favicon.ico` with the name of the icon file that you created at the begining of this procedure and saved in .../`images/`.

DRAFT

NOTE:
By default, the `{$image-loc}` notation is a path to the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/images/` directory. This path is defined in `constants.xsl`.

5. Save and test your changes.

OPTIONAL:

6. *Back out your changes, if necessary, by deploying your backup copy of* `frameWork.xsl`*.*

DRAFT

Ex Libris (USA) Inc.
See notice on first page

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Hide Limits On The Advanced Search Page? 11

Contents

DRAFT

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Hide Limits On The Advanced Search Page?

11

Description For “How Do I Hide Limits On The Advanced Search Page?” Example

This chapter describes how to hide the various limits options on the **Advanced Search** page until a user clicks a **More** link to display them.

Files

The example in this chapter uses the following files:

- `searchAdvanced.js` (new for this example).
- `searchAdvanced.css`.
- `cl_searchAdvanced.xsl`.

Instructions

This section provides the instructions for creating the example described in this chapter.

DRAFT

Procedure 11-1. Hide Limits on the Advanced Search Page

Use the following procedure to hide limits on the **Advanced Search** page.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new JavaScript file in the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/jscripts/` directory named `searchAdvanced.js` to be used to hide limits on the **Advanced Search** page. Use the sample lines of code shown in Figure 11-1 for this example.

```
/*
#(c)#=====================================================================
#(c)#
#(c)# Copyright 2007 ExLibris Group
#(c)# All Rights Reserved
#(c)#
#(c)#=====================================================================

** Product : WebVoyage :: disable/enable limits on advanced search
** not an accessible technique
** Version : 7.0
** Created : 23-JAN-2008
** Orig Author : EJW
** Last Modified : 23-JAN-2008
** Last Modified By : EJW
*/

// hide the limitList div
```

Figure 11-1. Example code for searchAdvanced.js

DRAFT

```
function hideLimits() {
               document.getElementById('limitList').style.display='none';
}

// display the limitList div
function showLimits() {
               document.getElementById('limitList').style.display='';
}

// toggle the limitList div
// we'll check the class of the toggle switch
function toggleLimits () {
               showLimits();
               document.getElementById('limitToggle').style.display='none';
}


function addLoadEvent(func) {
  var oldonload = window.onload;
  if (typeof window.onload != 'function') {
    window.onload = func;
  } else {
    window.onload = function() {
      if (oldonload) {
        oldonload();
      }
      func();
    }
  }
}

addLoadEvent(
               function () {
                       // hide the the limits on page load and show the toggle
           switch
                       hideLimits();
                       document.getElementById('limitToggle').style.display='';
               });
```

Figure 11-1. Example code for searchAdvanced.js (Continued)

DRAFT

2. Make a backup copy of `searchAdvanced.css` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/css/`.

3. Edit `searchAdvanced.css`.

   Go to the end of the file and add the example code shown in Figure 11-2.

```
/* display link to show limits */
#limitToggle {
        font-size: smaller;
        font-family: Verdana;
        margin:10px 10px 0.5em;
}
```

Figure 11-2. Example code for searchAdvanced.css

4. Save your changes.

5. Make a backup copy of `cl_searchAdvanced.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

6. Edit `cl_searchAdvanced.xsl`.

   a. Locate the `buildSearchForm` section.

   b. Add a call to the `searchAdvanced.js` file immediately before the `<!-- line 6 - year label, radio button & selection box -->` comment. See Figure 11-3.

DRAFT

```
<!-- add java script to hide/show limits on page -->
                    <script type="text/javascript" src="{$jscript-
          loc}searchAdvanced.js"/>


                    <div id="limitToggle" class="limitsOff"
          style="display:none"><a href="javascript:toggleLimits();">more</
          a></div>
                    <div id="limitList">
            <!-- end - other end of limitList div was added below  -->
```

Figure 11-3. Example coding change for cl_searchAdvanced.xsl

c. Add the closing `<div>` element above the `buildSearchButtons` template at the bottom of the file. See Figure 11-4.

```
                    </xsl:for-each>
                 </div>


                <xsl:call-template name="buildSearchButtons"/>


        </div>
        <!-- search advanced form - end -->
    </xsl:template>


</xsl:stylesheet>
```

Figure 11-4. Additional coding change to cl_searchAdvanced.xsl

7. Save your changes and test.

8. Back out your changes, if necessary, by deploying your backup copies of `searchAdvanced.css` and `cl_searchAdvanced.xsl`.

DRAFT

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Build And Display A Persistent Link To A Bibliographic Record?

12

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Build And Display A Persistent Link To A Bibliographic Record? 12

DRAFT

Description For "How Do I Build And Display A Persistent Link To A Bibliographic Record?" Example

Patrons can use the persistent link to bibliographic records for bookmarking, tagging, emailing, blogging, and so forth.

Files

The example in this chapter uses the following files:

- `local_PersistentLink.xsl` (new for this example).
- `cl_displayRecord.xsl`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 12-1. Building Persistent Link to Bibliographic Record

Use the following procedure to create a persistent link to a bibliographic record.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

```
/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/
```

1. Create and save a new file in the `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/` directory named `local_PersistentLink.xsl` to store the template that defines how to build the hyperlink. Use the sample lines of code shown in Figure 12-1 for this example.

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


<!--
**                  Note: create persistent link based on bib ID
**                  Version : 1.0
**                  Created : 16-Nov-2007
**                  Created By : EJW
**                  Last Modified : 14-Apr-2008
**                  Last Modified By : ASP
-->


<xsl:stylesheet version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
              xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/
          page"
              xmlns:fo="http://www.w3.org/1999/XSL/Format">


<!-- ############################################################ -->
```

Figure 12-1. Sample code for local_PersistentLink.xsl

DRAFT

```
<xsl:template name="persistentLink">
<xsl:param name="bibID"/>

    <!-- :doc: you must set this to match your public URL -->
       <xsl:variable name="baseURL">http://xxx.xxx.xxx.xxx:7008/vwebv/
           holdingsInfo?bibId=</xsl:variable>

              <a id="persistentLink" href="{$baseURL}{$bibID}" target="_new" >
                             Persistent Link
                         </a>
</xsl:template>

<!-- ############################################################ -->

</xsl:stylesheet>
```

Figure 12-1. Sample code for local_PersistentLink.xsl (Continued)

2. Make a backup copy of `cl_displayRecord.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/`.

3. Edit `cl_displayRecord.xsl`.

   a. Locate the namespace declarations at the top of the file and include a reference to the `local_PersistentLink.xsl` file after the `<xsl:stylesheet>` element. See Figure 12-2.

```
<xsl:stylesheet version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
              xmlns:page="http://www.exlibrisgroup.com/voyager/webvoyage/
           page"
              xmlns:fo="http://www.w3.org/1999/XSL/Format">


    <xsl:include href="../pageFacets/local_PersistentLink.xsl"/>
```

Figure 12-2. Include file reference example

DRAFT

b. Call the persistent link template from the `Bibliographic Data` section. See Figure 12-3.

```
<!-- ## Bibliographic Data ## -->
               <xsl:for-each select="$Config">
                   <div class="bibliographicData">
                       <xsl:call-template name="buildMarcDisplay">
                               <xsl:with-param name="recordType"
           select="'bib'"/>
                       </xsl:call-template>
                   <!-- add a persistent link -->
                       <xsl:call-template name="persistentLink" >
                               <xsl:with-param name="bibID" select="$bibID"/>
                       </xsl:call-template>


                   </div>
               </xsl:for-each>
```

Figure 12-3. Persistent link template call in Bibliographic Data section

4. Save and test your changes.

OPTIONAL:

5. *Back out your changes, if necessary, by deploying your backup copy of* `cl_displayRecord.xsl`*.*

DRAFT

How Do I Change The Format Of The Record Display Page?

13

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Change The Format Of The Record Display Page?

13

DRAFT

Description For “How Do I Change The Format Of The Record Display Page?” Example

For greater formatting control of elements on the record display page, class attributes may be added to the XML and subsequently adjusted in the appropriate style sheet. This allows you, for example, to change the font characteristics of the Title and Author lines without affecting other data on the page.

See Procedure 13-1, Add Class Attributes For Formatting, on page 13-2 for instructions regarding class attributes.

Files

The example in this chapter uses the following files:

- `displaycfg.xml`.
- `display.xsl`.
- `displayRecord.css`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 13-1. Add Class Attributes For Formatting

Use the following procedure to implement class attributes for formatting.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/

1. Make a backup copy of `displaycfg.xml` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/configs/`.

2. Edit `displaycfg.xml`.

   a. Locate the `displayTag` to which you want to apply formatting.

   b. Add a class attribute after the `label=` string. Use a name of your own choosing. See Figure 13-1 for examples.

```
<displayTags label="Title:" localcssclass="tagTitle">
<displayTags label="Author:" localcssclass="tagAuthor">
```

Figure 13-1. Class attribute creation example

3. Save your changes to the `displaycfg.xml`.

4. Make a backup copy of `display.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/contentLayout/display/`.

5. Edit `display.xsl`.

DRAFT

a. Locate the template for buildMarcDisplay.

b. Add the lines shown in Figure 13-2 to assign one of two classes to the bibliographic tag data.

```
                                           </a><br/>


                                       </xsl:for-each>
                                   </div>
                               </li>
                           </xsl:if>
                       </xsl:if>




<!-- assign class-->
                       <li>
                           <xsl:attribute name="class">
                               <xsl:choose>
                                   <xsl:when test="string-
           length(@localcssclass)">
                                       <xsl:value-of select="@localcssclass" />
                                   </xsl:when>
                                   <xsl:otherwise>bibTag</xsl:otherwise>
                               </xsl:choose>
                           </xsl:attribute>
                           <xsl:copy-of select="$bibTag"/>
                       </li>
<!-- assign class dinking -->

<!-- comment out these lines if using localcssclass
                   <li class="bibTag">
                      <xsl:copy-of select="$bibTag"/>
                   </li>
-->
                </xsl:if>
```

Figure 13-2. Example code to add to buildMarcDisplay

DRAFT

```
            </xsl:for-each>


            </ul>
         </div>
      </xsl:when>
```

Figure 13-2. Example code to add to buildMarcDisplay (Continued)

6. Save your changes to `display.xsl`.

7. Make a backup copy of `displayRecord.css` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/css/`.

8. Add style directives to the `localcssclass` tags that you defined in `displaycfg.xml` and place at the bottom of the `displayRecord.css` file.

   See example code in Figure 13-3.

```
.tagTitle, .tagAuthor
{
    margin-bottom: .75em;
    font-size: larger;
    color: #ff0000;
}
```

Figure 13-3. Example code for displayRecord.css

9. Save and test your changes.

10. Back out your changes, if necessary, by deploying your backup copies of `displaycfg.xml`, `displayRecord.xsl`, and `displayRecord.css`.

DRAFT

How Do I Add Tracking Codes? 14

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Add Tracking Codes?

14

DRAFT

Description For “How Do I Add Tracking Codes?” Example

Various companies offer web page tracking/analytic services such as Google Analytics. The general practice is to include tagging on all the pages you want tracked.

The instructions in this chapter may be use to add the relevant code to one of two `.xsl` files that are called by all WebVoyáge pages.

NOTE:
You must establish a relationship with the tracking service first and consider use of such a tool in relationship to your institution's privacy policy.

Files

The example in this chapter can apply to either of the following files:

- `frameWork.xsl`.
- `footer.xsl`.

Instructions

This section provides the instructions for creating the example described in this chapter.

Procedure 14-1. Add tracking codes

Use the following procedure to add tracking codes (in coordination with an outside service).

Specifically, this example inserts the Google Analytics tracking code into the `footer.xsl`. Coordinate with other services regarding their specific instructions.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

`/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`

1. Make a backup copy of `footer.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/[skin]/xsl/pageFacets/`.

2. Edit `footer.xsl`.

   a. Copy the script snippet the third-party vendor generated for you upon signup. This example uses a Google Analytics script snippet.

   b. Paste the script snippet into the `buildFooter` template. See Figure 14-1.

```
<!-- ################# -->
<!-- ## buildFooter ## -->
<!-- ############################################################### -->

<xsl:template name="buildFooter">

   <xsl:for-each select="/page:page/page:pageFooter">
```

Figure 14-1. Example of script snippet add to buildFooter template

DRAFT

```
      <div id="pageFooter">
         <xsl:for-each select="page:tabs[@nameId='page.footer.buttons']">


            <div id="footerTabs" title="{$footerText/footerTabs}">
               <a name="navFooter"></a>
               <h2 class="navFooter"><xsl:value-of select="$footerText/
          footerTabs"/></h2>
               <ul class="navbar">
                   <xsl:for-each select="$Configs/pageConfigs/
          footerTabDisplayOrder/tab">
                     <xsl:variable name="tempName" select="@name"/>
                    <xsl:variable name="newWin" select="@clickOpensNewWindow"/>
                     <xsl:call-template name="buildFooterTab">
                        <xsl:with-param name="displayTab" select="$tempName"/>
                        <xsl:with-param name="newWin" select="$newWin"/>
                     </xsl:call-template>
                   </xsl:for-each>
               </ul>
            </div>


         </xsl:for-each>


         <div id="libraryLink">
            <span>
               <xsl:call-template name="buildLinkType">
                  <xsl:with-param name="eleName"
          select="'page.footer.library.link'"/>
               </xsl:call-template>
            </span>
         </div>


         <div id="copyright" title="{$footerText/copyright}"><span><xsl:value-
           of select="page:element[@nameId='page.footer.copyright.message']/
           page:messageText"/></span></div>


      </div>
   </xsl:for-each>


<!-- Google Analytics snippet -->
```

Figure 14-1. Example of script snippet add to buildFooter template (Continued)

D R A F T

```
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." :
           "http://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/
           ga.js' type='text/javascript'%3E%3C/script%3E"));
</script>
<script type="text/javascript">
var pageTracker = _gat._getTracker("UA-xxxxxx-x");
pageTracker._initData();
pageTracker._trackPageview();
</script>


</xsl:template>
```

Figure 14-1. Example of script snippet add to buildFooter template (Continued)

3. Save and test your changes.

OPTIONAL:

4. *Back out your changes, if necessary, by deploying your backup copy of `footer.xsl`.*

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Implement Google Book Search?

15

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Implement Google Book Search? 15

Description For "How Do I Implement Google Book Search?"

The Voyager 7.0 version of WebVoyáge provides code to interface with the Google Book Search API. This enables users of WebVoyáge to display Google Book Search information.

This feature is enabled at installation. To disable it, see Procedure 15-1, Disable Google Book Search Feature, on page 15-3.

Files

The Google Book Search feature uses the following files:

- `googleBooksAvail.js`.
- `local_googleBooksAvail.xsl`.
- `displayFacets.xsl`.
- `displayGoogleBooks.css`.

DRAFT

Google Book Search Implementation

This section describes the WebVoyáge implementation for displaying Google Book Search information in Voyager 7.0.

NOTE:
Directory path references to xxxdb implies that you need to substitute your database path name; and where [skin] is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/.

DRAFT

googleBooksAvail.js

The googleBooksAvail.js script in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/jscripts/ executes the call to the Google Book Search service. (This service is transparent to the user.)

The lines shown in Figure 15-1 define the text that displays in the Action Box.

```
function listBookEntries(booksInfo)
{
   var bookPreviewFull = 'Full text available';
   var bookPreviewPartial = 'Limited Preview';
   var bookPreviewNoView = '"About This Book"';
```

Figure 15-1. Action Box text

local_googleBooksAvail.xsl

The local_googleBooksAvail.xsl file in /m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/ defines a template named googleBooksAvail. This template describes how to identify the ISBN, LCCN, or OCLC numbers which are the standard numbers used to do the lookup executed with googleBooksAvail.js.

displayFacets.xsl

The `displayFacets.xsl` file in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/pageFacets/` calls the `googleBooksAvail` template. See local_googleBooksAvail.xsl on page 15-2.

displayGoogleBooks.css

The `displayGoogleBooks.css` file in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/css/` manages the display of the Google information within the Action Box.

Disable Google Book Search

This section provides the instructions for disabling Google Book Search described in this chapter.

DRAFT

Procedure 15-1. Disable Google Book Search Feature

Use the following procedure to disable Google Book Search.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

`/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`.

1. Make a backup copy of `displayFacets.xsl` that is located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/pageFacets/`.

2. Edit `displayFacets.xsl`.

   a. Locate the line of code near the bottom of the file as shown in Figure 15-2.

```
## mdp add the google book template ##
```

Figure 15-2. Line of code to locate

b. Comment out the lines of code as shown in Figure 15-3.

```
           <!-- ## mdp add the google book template ##
                         <div id="googleBooksRow">
                           <xsl:call-template name="googleBooksAvail"/>
                         </div>
                  -->
```

Figure 15-3. Comment out code to disable Google Book Search

3. Save and test your changes.

OPTIONAL:

4. *Back out your changes, if necessary, by deploying your backup copy of* `displayFacets.xsl`*.*

DRAFT

How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions? 16

Contents

DRAFT

Contents

DRAFT

Ex Libris (USA) Inc.
See notice on first page

How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?

16

DRAFT

Description For “How Do I Display Cover Images From Services Like Amazon.com and Syndetics Solutions?”

WebVoyáge is preconfigured with the capability to display cover images on the results and holdings pages.

The ISBN or ISSN is generally used to do the lookup at the remote service.

NOTE:
You must have a pre-existing relationship or agreement with a service that provides this cover art.

Files

The Syndetics example described in this chapter uses the following files:

- `pageProperties.xml`.
- `resultsFacets.xsl`.
- `resultsTitles.xsl`.
- `imageUtils.js`.
- `displaycfg.xml`.

- `display.xsl`.
- `displayRecord.xsl`.

Syndetics Solutions Implementation

This section describes as an example the WebVoyáge implementation of displaying Syndetics Solutions cover images for results and holdings pages in Voyager 7.0.

NOTE:
Directory path references to `xxxdb` implies that you need to substitute your database path name; and where `[skin]` is referenced, substitute the path name that is used at your site. The default skin path provided is en_US as in the following:

`/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/en_US/`.

pageProperties.xml

The `pageProperties.xml` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/userTextConfigs/` defines how the URL to the cover image for the titles results page is built to include the pre- and post-link text and the alternate name for the image. See Figure 16-1.

```
<!-- each library is responsible for creating a relationship with a provider of
           cover images -->
<!-- these options should not be enabled by default -->


<!-- <resultsCoverTag nameIdMatch="page.searchResults.item.type.isbn"
           linkPRE_TEXT="http://images.sample.com/images/"
           linkPOST_TEXT=".jpg" altText="Cover Image"/> -->


<!--resultsCoverTag nameIdMatch="page.searchResults.item.type.isbn"
           linkPRE_TEXT="http://www.syndetics.com/hw7.pl?isbn="
           linkPOST_TEXT="/SC.jpg" altText="Cover Image"/ -->
```

Figure 16-1. Example of pageProperties.xml

The `/SC.jpg` in Figure 16-1 is the Syndetics Solutions syntax for small cover (SC). If you prefer a medium or large cover image, use MC or LC, respectively.

DRAFT

resultsFacets.xsl

The `resultsFacets.xsl` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/userTextConfigs/` defines a template named `buildResultsCoverImage`. This template builds on the URL components defined in `pageProperties.xml`. See Figure 16-2.

```
<!-- ############################ -->
<!-- ## buildResultsCoverImage ## -->
<!--
           ###################################################################
           #####################################################  -->
<xsl:template name="buildResultsCoverImage">
<xsl:param name="tag"/>
<xsl:param name="tagType"/>


              <xsl:for-each select="$Configs/pageConfigs/
           resultsCoverTag[@nameIdMatch=$tagType]">
                  <div class="resultListCoverImageCell">
                      <img src="{@linkPRE_TEXT}{$tag}{@linkPOST_TEXT}"
           class="resultListCoverImage" alt="{@altText}" style="display:none"
           onload="checkImage(this)"/>
                  </div>
              </xsl:for-each>
</xsl:template>
```

Figure 16-2. Example of buildResultsCoverImage template

The template is used later in the file for constructing the URL with an ISBN or ISSN.

The `trimData` template strips parenthetical references from the standard numbers. See Figure 16-3.

DRAFT

```
<!-- ## cover image ## --
<xsl:choose>
            <xsl:when test="string-length(page:option/
        page:element[@nameId='page.searchResults.item.type.isbn']/
        page:value)">
            <!-- ## cover image from isbn ## -->
                <xsl:call-template name="buildResultsCoverImage">
                    <xsl:with-param name="tag">
                        <xsl:call-template name="trimData">
                            <xsl:with-param name="sData"
        select="page:option/
        page:element[@nameId='page.searchResults.item.type.isbn']/
        page:value"/>
                        </xsl:call-template>
                    </xsl:with-param>
                    <xsl:with-param name="tagType"
        select="'page.searchResults.item.type.isbn'"/>
                </xsl:call-template>
            </xsl:when>
            <xsl:when test="string-length(page:option/
        page:element[@nameId='page.searchResults.item.type.issn']/
        page:value)">
            <!-- ## cover image from issn ## -->
                <xsl:call-template name="buildResultsCoverImage">
                    <xsl:with-param name="tag">
                        <xsl:call-template name="trimData">
                            <xsl:with-param name="sData"
        select="page:option/
        page:element[@nameId='page.searchResults.item.type.issn']/
        page:value"/>
                        </xsl:call-template>
                    </xsl:with-param>
                    <xsl:with-param name="tagType"
        select="'page.searchResults.item.type.isbn'"/>
                </xsl:call-template>
            </xsl:when>
</xsl:choose>
```

Figure 16-3. Example of trimData template

DRAFT

resultsTitles.xsl

The `resultsTitles.xsl` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/` builds the titles results page. It uses a JavaScript file named `imageUtils.js`.

imageUtils.js

The `imageUtils.js` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/jscripts/` performs the functions shown in Figure 16-4.

```
 var setDefaultErrImg=""; // Default image to be displayed on error
   var setDefaultErrTxt=""; // Default text to be displayed on error

//////////////////////////////////////////////////////////////////////


function checkImage(obj)
{

      if(obj.complete==true)
      {
         if(obj.width==0)
         {
            obj.src=setDefaultErrImg;
            obj.setAttribute("alt",setDefaultErrTxt);
         }
         else
         {
            obj.setAttribute("style","display:block");
         }
      }
}

//////////////////////////////////////////////////////////////////////
```

Figure 16-4. Example of imageUtils.js

DRAFT

The Syndetics service (and possibly other remote cover services) does not allow you to pre-check whether an image exists before you build the link. Therefore, `imageUtils.js` checks to see if the image has been retrieved and loaded. If not, it displays a default **no image available** image or **no image available** text of your choice.

displaycfg.xml

The `displaycfg.xml` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/contentLayout/configs/` defines how the URL to the cover image for the record display page is built that includes the pre- and post-link text and the bib field used to build the link. See Figure 16-5.

```
<!--
  <coverTags altText="Cover Image" linkPRE_TEXT="http://www.syndetics.com/
           hw7.pl?isbn=" linkPOST_TEXT="/MC.jpg" singleInstance="true">
        <displayTag field="020" indicator1="X" indicator2="X" subfield="a"/>
    </coverTags>
-->
```

Figure 16-5. Example displaycfg.xml code

The `singleInstance="true"` syntax indicates that the first ISBN or ISSN found is used for building the link.

display.xsl

The `display.xsl` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/contentLayout/display/` defines the following templates for retrieving cover images:

- `buildCoverImage`.
- `buildCoverImageLinks`.

These templates provide function similar to the templates described in resultsFacets.xsl on page 16-3.

DRAFT

displayRecord.xsl

The `displayRecord.xsl` file located in `/m1/voyager/xxxdb/tomcat/vwebv/context/vwebv/ui/{skin}/xsl/` uses a JavaScript file named `imageUtils.js` to build the record display page.

DRAFT

DRAFT

Ex Libris (USA) Inc.
See notice on first page

Index

DRAFT

DRAFT

DRAFT

DRAFT