

# Chapter 7

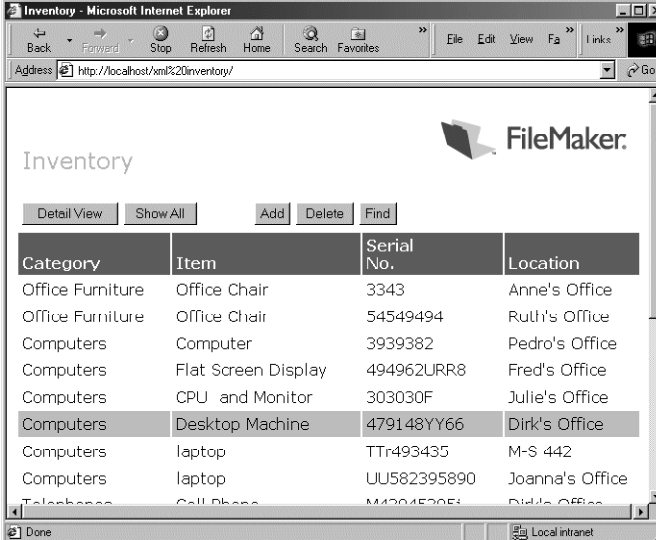
## *Using FileMaker Pro XML to deliver your data*

With the enhanced Web Companion, FileMaker Pro can now deliver data from your database to the Web in Extensible Markup Language (XML) format. In the same way that HTML has become the standard display language for communication on the World Wide Web, XML promises to become the standard language for structured data interchange.

In XML format, FileMaker Pro data can be populated in your web page programmatically instead of downloading statically from the server. This gives you more flexibility and a more web-like application that allows your web users to interact with the data after it has been downloaded. This also allows the web server to handle more requests as more processing is done by the browser on web users' machines. Using FileMaker Pro XML capabilities, you are no longer limited to the FileMaker proprietary CDML to publish your database on the Web.

### ***About the XML examples***

FileMaker Developer 5 includes an XML example that demonstrates how you can publish your database on the Web using XML and Dynamic HTML (including JavaScript). This XML example is designed to work in the Microsoft Internet Explorer 5 for Windows web browser. For step-by-step instructions, see “Looking at the XML Inventory example” on page 7-17.



Category	Item	Serial No.	Location
Office Furniture	Office Chair	3343	Anne's Office
Office Furniture	Office Chair	54549494	Ruth's Office
Computers	Computer	3939382	Pedro's Office
Computers	Flat Screen Display	494962URR8	Fred's Office
Computers	CPU and Monitor	303030F	Julie's Office
Computers	Desktop Machine	479148YY66	Dirk's Office
Computers	laptop	TTr493435	M-S 442
Computers	laptop	UU582395890	Joanna's Office
Telephones	Cell Phone	M420452051	Diane's Office

List View of the Inventory.fp5 database in the XML Inventory example

For examples showing the differences between using stylesheets or scripting (Dynamic HTML) with your FileMaker Pro XML documents, see “Comparing CSS, XSLT, and JavaScript” on page 7-11.

For general information on XML (including a glossary of XML terms), additional examples that use XML, and links to XML resources, see the FileMaker, Inc. web site at [www.filemaker.com](http://www.filemaker.com). As a shortcut to the site, double-click FileMaker on the Web (installed in the FileMaker Developer 5 folder).

## ***General process for custom web publishing using XML***

Here's a simple overview of the process for publishing your FileMaker Pro database on the Internet or an intranet using XML:

1. You send a FileMaker Pro CGI request (such as to find records in the database) to the Web Companion through an HTML form, an HREF link, or a script on your web page. The request can also be made by typing the URL in the web browser.

See "Generating FileMaker Pro CGI requests for an XML document" on page 7-8 and appendix B, "Valid names used in CGI requests for FileMaker XML data."

2. The Web Companion generates an XML document containing the results of your request in XML format (for example, a found set of records from the database and an XML-stylesheet processing instruction) and returns it to your web browser.

See "Generating an XML document" next.

3. The web browser, with the help of an XML parser, applies any instructions that you've specified via a stylesheet and displays the data in HTML format.

See "Using style sheets with your XML document" on page 7-10.

Once the XML document is downloaded to your web browser, you can use stylesheets (such as CSS or XSL) to apply text formatting styles and object positioning, or scripting (such as JavaScript) to manipulate the data however you want. See "Comparing CSS, XSLT, and JavaScript" on page 7-11.

### ***Generating an XML document***

When you specify an XML format parameter in your FileMaker Pro CGI request, the Web Companion generates an XML document containing data from your database that is formatted by one of two types of XML grammars (or schemas).

One type (called FMPDSO) gives you more flexibility and control over individual elements and is ideally suited for use with cascading style sheets (CSS) or Extensible Stylesheet Language (XSL). The FMPDSO grammar can also be used with the Microsoft XML Data Source Object (DSO) in Internet Explorer 4.0 to publish read-only databases. (The Microsoft XML DSO lets you view but not update data in XML format.)

The other type of grammar (called FileMaker Extended XML or FMPXML) provides a broader, richer XML that defines FileMaker Pro layouts, fields, and value list information. These grammars can be combined with XSL documents or scripting (such as JavaScript) to publish dynamic databases on the Web.

All XML data generated by the Web Companion is well-formed and compliant with the XML 1.0 specification. The document type definitions (DTDs) for the grammars are provided in HTML documents for your convenience (installed in the FileMaker Developer 5 folder).

FileMaker Developer 5 > External FileMaker APIs > XML > Documentation

Two of the grammars generated by the Web Companion are used for retrieving query results and a third is used for retrieving layout information. Depending on what you specify in your FileMaker Pro CGI request, the Web Companion will generate an XML document using one of these grammars:

- the FMPDSORESULT grammar
- the FMPXMLRESULT grammar
- the FMPXMLLAYOUT grammar

Each XML document contains a default XML namespace declaration for the grammar. (See "About XML namespaces" next.) You can also specify that the document contain an XML-stylesheet processing instruction. (See "Using style sheets with your XML document" on page 7-10.)

Use one of these grammars in your document or web page to display and work with FileMaker data in XML format.

**Note** XML data generated by the Web Companion is encoded using UTF-8 format (Unicode Transformation Format 8). For information, see “About UTF-8 encoded data” on page 7-8.

### **About XML namespaces**

To avoid name collisions, unique XML namespaces help distinguish XML tags by the application they were designed for. For example, if your XML document contains two DATABASE elements, one for FileMaker Pro XML data and another for Oracle XML data, the namespaces will identify the DATABASE element for each.

The FileMaker Pro Web Companion generates a default namespace for each grammar. For example, for the FMPDSORESULT grammar, the following namespace is generated:

```
xmlns="http://www.filemaker.com/fmpdsoreult"
```

### **About FileMaker Pro database error codes**

The FileMaker Pro Web Companion generates an error code at the beginning of each grammar based on the current error status of the database. A value of zero (0) is returned for no error.

```
<ERRORCODE>0</ERRORCODE>
```

See appendix C, “FileMaker Pro values for error codes” for information.

## **Using the FMPDSORESULT grammar**

When you specify “-dso\_xml” as the format for a FileMaker Pro CGI request, the Web Companion will generate XML data based on a database-specific grammar that uses field names as element names. The FMPDSORESULT grammar is useful for publishing databases on web pages that are formatted with cascading style sheets or XSLT. (See “Comparing CSS, XSLT, and JavaScript” on page 7-11 for information.) The FMPDSORESULT grammar is compatible with the Microsoft XML Data Source Object used by Internet Explorer 4.0.

The Web Companion will also generate the document type definition for the grammar if you specify “-dso\_xml\_dtd” as the format. This is useful if you want an XML parser to validate the XML before your document goes to production.

**Note** Internet Explorer 4.0 directly supports XML with no additional software required. The XML can be displayed using dynamic data binding features available in the browser. This is accomplished with a Java applet that ships with Internet Explorer 4.0, which presents the XML as a Data Source Object (DSO) to the browser. With the DSO, the Internet Explorer 4.0 browser exposes XML data to scripting languages such as JavaScript or VBScript via the Microsoft Document Object Model (DOM). Keep in mind that the Microsoft XML DSO applet does not provide a mechanism for updating the data, nor does it know anything about FileMaker Pro database layouts or value lists.

The following is an example of a Microsoft XML DSO applet tag that you might use in your web page to query FileMaker Pro for XML data using the FMPDSORESULT grammar—where the “url” parameter can be any valid FileMaker Pro CGI request containing a -format parameter equal to “-dso\_xml” or “-dso\_xml\_dtd.” (See “Generating FileMaker Pro CGI requests for an XML document” on page 7-8 for a list of valid FileMaker CGI requests.)

```
<applet code=com.ms.xml.dso.XMLDSO.class width=0 height=0
id=xmldso MAYSCRIPT=true>
  <PARAM NAME="url" VALUE=fmpro?-db=PhoneList.fp5&
  -format=-dso_xml&-find=>
</applet>
```

### Description of elements in the FMPDSORESLT grammar

Each ROW element in the generated FMPDSORESLT grammar contains a number of FIELD elements that correspond to the field names in the specified layout.

Spaces or single colons in field names are converted to underscores in the element names (for example, <FIRST\_NAME>). Double colons in portal fields are converted to periods (for example, <PHONE.PHONE\_NUMBER>). This is done because colons are reserved in XML for specifying namespaces and spaces are not allowed in XML element names.

For repeating and portal fields, each FIELD element will contain a DATA element that corresponds to each repetition or portal record.

**Note** The content of container fields in the database will be generated in the form of the relative URL used for retrieving the content instead of the actual content (such as an image).

To qualify the XML elements for the FileMaker Pro application, the names of all elements and attributes in this grammar are associated with the unique XML namespace <http://www.filemaker.com/fmpdsoreresult>. This namespace is declared in the grammar as the default namespace.

The following is an example of XML data generated with the FMPDSORESLT grammar.

### Example of XML data in the FMPDSORESLT grammar

```
<?xml version="1.0" encoding="UTF-8"?>
<FMPDSORESLT xmlns="http://www.filemaker.com/fmpdsoreresult">
  <ERRORCODE>0</ERRORCODE>
  <DATABASE>PhoneList.fp5</DATABASE>
  <LAYOUT>Web Layout</LAYOUT>
  <ROW RECORDID="3" MODID="23">
    <FIRST_NAME>John</FIRST_NAME>
    <LAST_NAME>Smith</LAST_NAME>
    <PHONE.PHONE_NUMBER>
      <DATA>555-444-3333</DATA>
      <DATA>555-222-9999</DATA>
    </PHONE.PHONE_NUMBER>
  </ROW>
  <ROW RECORDID="6" MODID="32">
    <FIRST_NAME>Barbara</FIRST_NAME>
    <LAST_NAME>Jones</LAST_NAME>
    <PHONE.PHONE_NUMBER>
      <DATA>555-666-7777</DATA>
      <DATA>555-333-0000</DATA>
      <DATA>555-111-7654</DATA>
    </PHONE.PHONE_NUMBER>
  </ROW>
</FMPDSORESLT>
```

**Note** If the -lay parameter is not specified in the FileMaker Pro CGI request, the LAYOUT element is empty and data for every field in the database is returned. (See “Generating FileMaker Pro CGI requests for an XML document” on page 7-8 for information.)

## Using the FileMaker Pro Extended XML grammars

The FileMaker Pro Extended XML grammars contain additional information about field types, value lists and layouts that is not found in the FMPDSORESLT grammar. Use the FMPXMLRESULT and FMPXMLLAYOUT grammars if you require layout information or want the METADATA information provided by these grammars.

**Note** These grammars are not well suited for cascading style sheets with positioning. See “Using the FMPDSORESLT grammar” on page 7-3 if you want to use CSS with your XML data.

When you specify “-fmp\_xml” as the format for a FileMaker Pro CGI request, the Web Companion will generate XML data using either the FMPXMLRESULT or FMPXMLLAYOUT grammar, depending on the request you specify in the CGI command:

- The Web Companion will generate the FMPXMLRESULT grammar when you specify -edit, -delete, -find, -new, -dbnames, -layoutnames, -scriptnames or -dbopen as the FileMaker CGI request.
- The Web Companion will generate the FMPXMLLAYOUT grammar when you specify -view as the FileMaker CGI request.

The Web Companion will also generate the document type definition for the grammar if you specify “-fmp\_xml\_dtd” as the format. This is useful if you want an XML parser to validate the XML before your document goes to production.

For a list of valid FileMaker CGI requests, see “Generating FileMaker Pro CGI requests for an XML document” on page 7-8.

## Description of elements in the FMPXMLRESULT grammar

In the generated FMPXMLRESULT grammar, the DATABASE element contains attributes for the name of the database, the number of records in the database, the name of the layout that was used to generate the result set, and the format of dates and times in the XML document.

The DATEFORMAT attribute specifies the format of dates in the XML document.

Field	Full form	Short form
Year	yyyy (4 digits)	yy (2 digits)
Month	mm (2 digits)	M (1 or 2 digits)
Day	dd (2 digits)	d (1 or 2 digits)

The TIMEFORMAT attribute specifies the format of times in the XML document.

Field	Full form	Short form
Hour (1 – 12)	hh (2 digits)	h (1 or 2 digits)
Hour (1 – 24)	kk (2 digits)	k (1 or 2 digits)
Minute	mm	
Second	ss	
AM/PM	a	

The METADATA element contains one or more FIELD elements, each containing information for one of the fields/columns of the result set—including the name of the field as defined in the database, the field type, the Yes or No allowance for empty fields (EMPTYOK attribute) and the maximum number of repeating values (MAXREPEAT attribute). Valid values for field types are TEXT, NUMBER, DATE, TIME, and CONTAINER.

The RESULTSET element contains all of the ROW elements returned as the result of a query and an attribute for the total number of records found. Each ROW element contains the field/column data for one row in the result set—including the record ID for the row, the modification ID for the row, and the COL element containing the data for one field/column in the row (where multiple DATA elements represent one of the values in a repeating or portal field).

**Note** The content of container fields in the database will be generated in the form of the relative URL used for retrieving the content, instead of the actual content (such as an image).

To qualify the XML elements for the FileMaker Pro application, the names of all elements and attributes in this grammar are associated with the unique XML namespace <http://www.filemaker.com/fmpxmlresult>. This namespace is declared in the grammar as the default namespace.

The following is an example of XML data generated with the FMPXMLRESULT grammar.

### *Example of XML data in the FMPXMLRESULT grammar*

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="yourstylesheet.xsl"?>
<FMPXMLRESULT xmlns="http://www.filemaker.com/fmpxmlresult">
  <ERRORCODE>0</ERRORCODE>
  <PRODUCT NAME="Web Companion" VERSION="5.0"
  BUILD="10/23/99"/>
  <DATABASE NAME="Employees.fp5" RECORDS="23"
  DATEFORMAT="MM/dd/yy" TIMEFORMAT="hh:mm:ss"
  LAYOUT="summary"/>
  <METADATA>
    <FIELD NAME="First Name" TYPE="TEXT"
    EMPTYOK="NO" MAXREPEAT="1"/>
    <FIELD NAME="Last Name" TYPE="TEXT"
    EMPTYOK="NO" MAXREPEAT="1"/>
```

```
<FIELD NAME="Department" TYPE="TEXT"
EMPTYOK="YES" MAXREPEAT="1"/>
</METADATA>
<RESULTSET FOUND="5">
  <ROW RECORDID="34" MODID="47">
    <COL>
      <DATA>Joe</DATA>
    </COL>
    <COL>
      <DATA>Smith</DATA>
    </COL>
    <COL>
      <DATA>Engineering</DATA>
    </COL>
  </ROW>
  <ROW RECORDID="78" MODID="89">
    <COL>
      <DATA>Susan</DATA>
    </COL>
    <COL>
      <DATA>Jones</DATA>
    </COL>
    <COL>
      <DATA>Marketing</DATA>
    </COL>
  </ROW>
</RESULTSET>
</FMPXMLRESULT>
```

The order of the COL elements corresponds with the order of the FIELD elements in the METADATA element—for example, where the “First Name”, “Last Name”, and then “Department” elements are listed in the METADATA, “Joe”, “Smith”, and then “Engineering” are listed in the same order in the RESULTSET ROW.

**Note** If the `-lay` parameter is not specified in the FileMaker Pro CGI request, the LAYOUT attribute in the DATABASE element is empty and data for every field in the database is returned. (See “Generating FileMaker Pro CGI requests for an XML document” on page 7-8 for information.)

### *Description of elements in the FMPXMLLAYOUT grammar*

In the generated FMPXMLLAYOUT grammar, the LAYOUT element contains the name of the layout, the name of the database, and FIELD elements for each field found in the corresponding layout in the database. Each FIELD element describes the style type of the field, and contains the VALUELIST attribute for any associated value list of the field.

The VALUELISTS element contains one or more VALUELIST elements for each value list found in the layout—each including the name of the value list and a VALUE element for each value in the list.

To qualify the XML elements for the FileMaker Pro application, the names of all elements and attributes in this grammar are associated with the unique XML namespace `http://www.filemaker.com/fmpxmllayout`. This namespace is declared in the grammar as the default namespace.

The following is an example of XML data generated with the FMPXMLLAYOUT grammar.

### *Example of XML data in the FMPXMLLAYOUT grammar*

```
<?xml version="1.0" encoding="UTF-8"?>
<FMPXMLLAYOUT xmlns="http://www.filemaker.com/fmpxmllayout">
  <ERRORCODE>0</ERRORCODE>
  <PRODUCT NAME="Web Companion" VERSION="5.0v1"
  BUILD="10/24/99"/>
  <LAYOUT NAME="Web Layout" DATABASE="employees.fp5">
    <FIELD NAME="First Name">
      <STYLE TYPE="EDITTEXT" VALUELIST="" />
    </FIELD>
    <FIELD NAME="Last Name">
      <STYLE TYPE="EDITTEXT" VALUELIST="" />
    </FIELD>
    <FIELD NAME="Department">
      <STYLE TYPE="POPUPMENU"
      VALUELIST="Departments" />
    </FIELD>
  </LAYOUT>
  <VALUELISTS>
    <VALUELIST NAME="Departments">
      <VALUE>Engineering</VALUE>
      <VALUE>Marketing</VALUE>
    </VALUELIST>
  </VALUELISTS>
</FMPXMLLAYOUT>
```

## About UTF-8 encoded data

All XML data generated by the Web Companion is encoded in UTF-8 (Unicode Transformation 8 Bit) format. This format compresses data from the standard Unicode format of 16 bits to 8 bits for ASCII characters. XML parsers are required to support Unicode and UTF-8 encoding.

UTF-8 encoding includes direct representations of most of the characters used in English using values of 0-127 for the standard ASCII set of characters, and provides multibyte encodings for Unicode characters with higher values. UTF-8 encoded data is compressed almost in half (lower ASCII characters are compressed from 2 bytes to 1 byte), which helps data download faster.

**Note** Because your XML data is UTF-8 encoded, some upper ASCII characters will be represented by 2 or 3 characters in the text editor—they will appear as single characters only in the XML parser or browser.

The UTF-8 encoding format includes the following features:

- All ASCII characters are one-byte UTF-8 characters. A legal ASCII string is a legal UTF-8 string.
- Any non-ASCII character (i.e., any character with the high-order bit set) is part of a multibyte character.
- The first byte of any UTF-8 character indicates the number of additional bytes in the character.
- The first byte of a multibyte character is easily distinguished from the subsequent bytes. Thus, it is easy to locate the start of a character from an arbitrary position in a data stream.
- It is easy to convert between UTF-8 and Unicode.
- The UTF-8 encoding is relatively compact. For text with a large percentage of ASCII characters, it is more compact than Unicode. In the worst case, a UTF-8 string is only 50% larger than the corresponding Unicode string.

## Generating FileMaker Pro CGI requests for an XML document

You use FileMaker Pro CGI (Common Gateway Interface) commands to generate requests for XML data from your database.

For example, to generate a `-find` request to display all employees from a database, web users might click on a link containing the following FileMaker Pro CGI command:

```
FMPPro?-db=employees.fp5&-format=-dso_xml&-styletype=text/css&-stylehref=stylesheet.css&-find
```

### Request and parameter names

The following tables list the request and parameter names in name/value pairs you can use in a FileMaker Pro CGI command when requesting data in XML format.

For more information and examples, see appendix B, “Valid names used in FileMaker CGI requests for XML data.”

Use this request name	To generate this request
<code>-new</code>	New record
<code>-edit</code>	Edit record
<code>-delete</code>	Delete record
<code>-find</code>	Find record(s)
<code>-findall</code>	Find all records
<code>-findany</code>	find a random record
<code>-view</code>	View layout info (in FMPXMMLAYOUT grammar)
<code>-dbnames</code>	Retrieve names of all open and web-shared databases
<code>-layoutnames</code>	Retrieve names of all available layouts for a specified open, web-shared database

Use this request name	To generate this request
–scriptnames	Retrieve names of all available scripts for a specified open, web-shared database
–dbopen	Open a database that’s in the Web folder with Remote Administration enabled
–dbclose	Close a database that’s in the Web folder with Remote Administration enabled

Use these parameter names	To go with these requests
–db (database name)	Required for all requests except –dbnames
–lay (layout name)	Required for –view, and with –edit or –new requests for data in related fields and portals. Optional for –find, –findall
–format	Required for all requests. (Use one of these formats: –dso_xml, –dso_xml_dtd, –fmp_xml, or –fmp_xml_dtd)
–recid (record I.D.)	Required for –edit and –delete. Optional for –find
–modid (modification I.D.)	Optional for –edit
–lop (logical operator)	Optional for –find
–op (operator)	Optional for –find
–max (maximum records)	Optional for –find
–skip (skip records)	Optional for –find
–sortorder (sort order)	Optional for –find, –findall
–sortfield (sort field)	Optional for –find, –findall
–script (perform script)	Optional for –find, –findall
–script.prefind (perform script before –find)	Optional for –find, –findall

Use these parameter names	To go with these requests
–script.presort (perform script before sort)	Optional for –find, –findall
–styletype (stylesheet type)	Optional for all requests
–stylehref (stylesheet HREF)	Optional for all requests
–password	Optional for –dbopen requests. Specifies the database’s password.
field name (no hyphen)	At least one field name is required for –new and –edit. Optional for –find. See “field name (Name of specific field)” on page B-10 for more information.

### Requests for adding records to a portal

When you make an –edit request or a –new request that includes data for a portal of related database records, you must specify the layout and the relationship name for the related database.

**Note** You can only add one record at a time to a portal, and therefore must make separate –new requests to add more rows to the portal.

The following is an example of a –new request for adding a record to a portal, where “Address::” is the name of the database relationship, and “City.0” is the related field name in the portal:

```
FMPPro?–db=employees.fp5&–lay=LayoutOne&FirstName=Sam
&LastName= Smith&Address::City.0=Seattle&–format= –fmp_xml&–new
```

### Requests for editing multiple records in a portal

You only need to make one –edit request to edit multiple records in a portal. You specify each row (or record) in the portal by adding a period and a consecutive number (starting with number 1) to the end of the related field name.

The following is an example of an `-edit` request for editing records in a portal, where “Address:” is the name of the relationship, “City.1” is the first row in the portal, and “City.2” is the second row in the portal:

```
FMPPro?-db=employees.fp5&-lay=LayoutOne&recid=11&
FirstName=Sam&LastName=Smith&Address::City.1=Seattle
&Address::City.2=New York&-format=-fmp_xml&-edit
```

The following is an example of another `-edit` request for editing records in a portal, in an HTML form:

```
<FORM ACTION="fmp" METHOD="POST">
  <INPUT TYPE="HIDDEN" NAME="-db" VALUE="employees.fp5">
  <INPUT TYPE="HIDDEN" NAME="-lay" VALUE="LayoutOne">
  <INPUT TYPE="HIDDEN" NAME="-format" VALUE="-fmp_xml">
  <INPUT TYPE="HIDDEN" NAME="-recid" VALUE="11">
  <INPUT TYPE="TEXT" NAME="FirstName" VALUE="Joe">
  <INPUT TYPE="TEXT" NAME="LastName" VALUE="Smith">
  <INPUT TYPE="TEXT" NAME="Address::City.1" VALUE="San
  Jose">
  <INPUT TYPE="TEXT" NAME="Address::City.2" VALUE="Santa
  Clara">
  <INPUT TYPE="SUBMIT" NAME="-edit" VALUE="Edit Record">
</FORM>
```

## *Using style sheets with your XML document*

The Web Companion will generate an XML-stylesheet processing instruction with each grammar if the FileMaker CGI request includes the `-styletype` and `-stylehref` parameters. This allows you to use cascading style sheets (CSS) or Extensible Stylesheet Language (XSL) documents for displaying your XML document.

The `-styletype` parameter is used for setting the value of the type attribute (`type=text/css` or `type=text/xml`).

The `-stylehref` parameter is used for setting the value of the HREF attribute (`href=document.css` or `href=document.xml`).

Here is an example of what a possible FileMaker CGI command might look like:

```
http://localhost/fmp?db=employees.fp5&-format=-fmp_xml&-find=&
-styletype=text/xml&-stylehref=document.xml
```

Based on this command, the Web Companion will include the following processing instruction in the XML document:

```
<?xml-stylesheet type="text/xml" href="document.xml"?>
```

The following text is an example of a possible XSL document used with the FMPXMLRESULT grammar. In this example, the XSL document converts the XML document into an HTML document by inserting HTML tags. It builds an HTML table that contains a header row for all the field names from the METADATA element in the FMPXMLRESULT grammar, and table rows for all the field data in the ROW elements of the RESULTSET.

**Note** This is an example of XSLT that was written to work with Internet Explorer 5.0 for Windows, not with other browsers using later versions of XSLT.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
xmlns:HTML="http://www.w3.org/Profiles/XHTML-transitional">
  <xsl:template>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="text()">
    <xsl:value-of/>
```

```

</xsl:template>

<xsl:template match="/">
  <HTML>
    <BODY>
      <CENTER>
        <TABLE BORDER="0">
          <xsl:apply-templates/>
        </TABLE>
      </CENTER>
    </BODY>
  </HTML>
</xsl:template>

<xsl:template match="ERRORCODE">
</xsl:template>

<xsl:template match="METADATA">
  <TR>
    <xsl:apply-templates/>
  </TR>
</xsl:template>

<xsl:template match="FIELD">
  <TD ALIGN="CENTER" BGCOLOR="#336666">
    <FONT FACE="Verdana"
      COLOR="#FFFFFF"><B><xsl:value-of
        select="@NAME"/></B></FONT>
  </TD>
</xsl:template>

<xsl:template match="RESULTSET/ROW">
  <TR>
    <xsl:apply-templates/>
  </TR>
</xsl:template>

<xsl:template match="COL">
  <TD BGCOLOR="#00CCFF">
    <xsl:apply-templates/>
  </TD>
</xsl:template>

<xsl:template match="DATA">
  <P><xsl:value-of/></P>
</xsl:template>

</xsl:stylesheet>

```

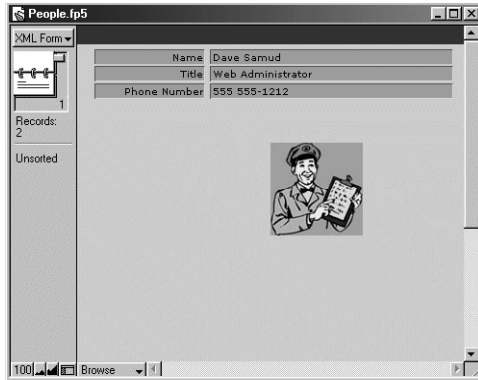
## Comparing CSS, XSLT, and JavaScript

FileMaker Developer includes three simple examples that demonstrate the differences between using cascading style sheets (CSS), Extensible Stylesheet Language–Transformations (XSLT), and JavaScript scripting language with your XML documents.

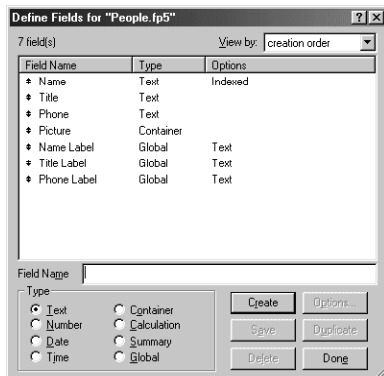
These examples are installed in the XML folder:

FileMaker Developer 5 > External FileMaker APIs > XML > Simple Examples

All three examples use a simple database named People.fp5.



The People.fp5 database contains seven fields — three text fields for data, three global fields for field labels, and one container field for pictures.



The seven fields in the People.fp5 database used for these examples

These three examples were designed to be viewed in the Internet Explorer 5.0 for Windows web browser. For information on new browsers that can be used to view the examples, double-click FileMaker on Web (installed in the FileMaker Developer 5 folder) to go to the FileMaker Developer product support pages in your browser.

To view the examples:

1. Place a copy of the Simple Examples folder and its files in the Web folder inside the FileMaker Pro 5 application folder on your hard disk.

FileMaker Pro 5 > Web > Simple Examples

**Note** For security, when actually publishing a database on the Web, you should not keep the database file in the Web folder unless you plan to administer it remotely. See “Opening password-protected databases remotely” on page 6-15 for more information.

2. In FileMaker Pro, open the People.fp5 database and make sure it's shared via the Web Companion.

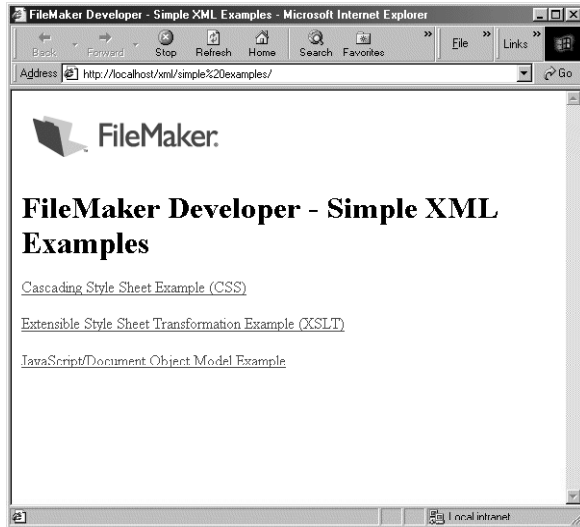
See “Enabling the Web Companion” on page 6-3 and “Sharing the database via the Web” on page 6-5 for information.

3. In your web browser, type localhost (or your computer's IP address) followed by /simple examples/ and press Enter.

<http://localhost/simple examples/>

<http://17.17.17.17/simple examples/>

For information on setting up your computer as the localhost, see “Testing your site without a network connection” on page 6-13.



4. On the Default.htm page, click the links to the CSS, XSL, and Scripting/DOM examples, or view the source to see the FileMaker Pro CGI requests for the links.

See “Generating FileMaker Pro CGI requests for an XML document” on page 7-8 for information.

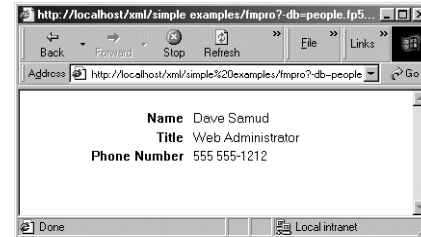
### *Cascading style sheets (CSS) example*

Cascading style sheets (CSS) documents format the text style (font, size, color, etc.) and positioning of your data in an XML document.

A cascading style sheet can only be applied to the existing data in an XML document—it cannot be used to transform the data (such as transforming a URL for a container field into its corresponding image), or to add additional information (such as labels for each field) to the XML document.

This example demonstrates the use of a CSS document with an XML document. The following CGI command was used to generate the FMPDSORESLT grammar for the People.fp5 database and to apply the People\_form.css stylesheet to the generated XML data:

```
fmproweb?db=people.fp5&-lay=xml form&-format=dso_xml&-styletype=text/css&-stylehref=people_form.css&-max=1&-find=
```



To simulate field labels, the CSS document applies boldface and right-alignment to data in the three global fields

The picture in the container field is not displayed in this example because the CSS document can only apply styles to the existing data (the relative URL to the image). It’s not possible to transform the URL into its corresponding image using cascading style sheets.

The following text is the people\_form.css stylesheet, which adds formatting and positioning to the XML:

```
/*
 * Since FMPDSORESLT is the root element of the XML document,
 * any formatting applied to this element will cascade down to all
 * other elements in the document. This is a good place to set the
 * default formatting options for your page.
 */
FMPDSORESLT
{
    font-family: sans-serif;
    font-size: 10pt;
}
/*
 * Don't display the data for these elements.
 */
```

```

ERRORCODE, DATABASE, LAYOUT, Picture
{
    visibility: hidden;
}
/*
 * Since it is not possible to add additional text to the output
 * of the XML document via CSS, we must rely on the XML document to
 * provide the labels when displaying the field data. In this case,
 * the labels are from global fields defined in the database.
 */
Name_Label
{
    position: absolute;
    left: 20px;
    top: 20px;
    width: 150px;
    text-align: right;
    font-weight: bold;
}
Title_Label
{
    position: absolute;
    left: 20px;
    top: 40px;
    width: 150px;
    text-align: right;
    font-weight: bold;
}
Phone_Label
{
    position: absolute;
    left: 20px;
    top: 60px;
    width: 150px;
    text-align: right;
    font-weight: bold;
}
/*
 * The following styles are applied to the actual field data.
 */
Name
{
    position: absolute;
    left: 180px;
    top: 20px;

```

```

}
Title
{
    position: absolute;
    left: 180px;
    top: 40px;
}
Phone
{
    position: absolute;
    left: 180px;
    top: 60px;
}

```

### ***Extensible Stylesheet Language–Transformations (XSLT) example***

Extensible Stylesheet Language–Transformations (XSLT) is a language for transforming XML documents into other XML documents. XSLT is part of the overall XSL specification, which also includes an XML vocabulary for formatting the transformed XML document (such as applying text styles).

This example demonstrates the use of an XSL document with an XML document. The following CGI command was used to generate the FMPDSORESULT grammar for the People.fp5 database and to apply the People\_form.xml stylesheet to the generated XML data:

```

fmpro?-db=people.fp5&-lay=xml form&-format=dso_xml&
-styletype=text/xsl&-stylehref=people_form.xml&-max=1&-find=

```



XSL lets you display images in container fields, format the data, and add field labels and a logo to the XML document

The three global fields for field labels in the People.fp5 database are not necessary when you're using an XSL document. You can use XSLT to add labels after the XML document has been generated by FileMaker Pro.

**Note** You can also include scripting (such as JavaScript) in your XSL document. See “JavaScript scripting language example” next.

The following text is the people\_form.xsl stylesheet, which adds labels and transforms the URL in the Picture field into an image:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
xmlns:fm="http://www.filemaker.com/fmpdsoreult">
  <!--
  The following two templates have been added for IE5 Windows,
  since that browser's XSL processor does not provide defaults for
  them.
  -->
  <xsl:template><xsl:apply-templates/></xsl:template>
  <xsl:template match="text()"><xsl:value-of select="."/;></
xsl:template
  <xsl:template match="fm:FMPDSOREULT">
    <html>
```

```
<head>
  <style type="text/css">
    table {font-family: sans-serif; font-size:
      10pt; }
    td.label { text-align: right; vertical-align:
      text-top; font-weight: bold; }
  </style>
</head>
<body>
  
  <xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="fm:ROW">
  <table>
    <tr>
      <td class="label">Name</td>
      <td><xsl:value-of select="fm:Name"/></td>
    </tr>
    <tr>
      <td class="label">Title</td>
      <td><xsl:value-of select="fm:Title"/></td>
    </tr>
    <tr>
      <td class="label">Telephone Number</td>
      <td><xsl:value-of select="fm:Phone"/></td>
    </tr>
    <tr>
      <td class="label">Picture</td>
      <td><xsl:element name="img"><xsl:attribute
name="src"><xsl:value-of select="fm:Picture"/>
</xsl:attribute></xsl:element></td>
    </tr>
  </table>
</xsl:template>
<!--
  Don't display anything for the following elements.
-->
<xsl:template
match="fm:ERRORCODE|fm:DATABASE|fm:LAYOUT">
</xsl:template>
</xsl:stylesheet>
```

## JavaScript scripting language example

Using HTML and a scripting language with your XML document can allow your web users to interact with the database after it has been downloaded. For example, a simple onClick scripting event handler can allow web users to click a button and see different records in the database.

This example demonstrates the use of the JavaScript scripting language with an XML document to publish the People.fp5 database on a web page. It starts with an HTML file, named People\_form.htm, that 1) references the JavaScript library, FMP.js, 2) contains a simple HTML form with a table for the field and picture rows, and 3) builds and executes a CGI command to FileMaker Pro to find and download all the records in the People.fp5 database.



With JavaScript, you can manipulate the data in the XML document to display different records in the database—after the data has been downloaded

**Note** The FMP.js JavaScript library was created for the XML Inventory example, described next, which is a more complex and sophisticated example of the use of JavaScript and the W3C Document Object Model with your XML documents.

Once the people\_form.htm page is loaded in the browser, the onLoad event handler performs the “initialize” function, creating an ActiveXObject and building the FMPFindRequest:

```
function initialize ( )
{
    var xmlDocument = new ActiveXObject ("Microsoft.XMLDOM");
    xmlDocument.async = false;
    var findRequest = new FMPFindRequest ("people.fp5", null, AND,
    false);
    if (xmlDocument.load(findRequest.getRelativeURL( ) ) )
    {
        foundSet = new FMPFoundSet (xmlDocument);
        populateFields( );
    }
    else
        alert ("Error retrieving records.");
}
```

The xmlDocument and findRequest variables are referenced from the FMP.js JavaScript library. The values in the new FMPFindRequest are people for the name of the database, null for no layout, AND for the find criteria’s logical operator, and false for no XML DTD to be generated as a result of the request.

In the People\_form.htm page, the nextRecord and previousRecord functions are assigned to the onClick event handler of the Next and Previous buttons respectively, so that when web users click the buttons, the function is performed and the fields are re-populated with new data.

```

function nextRecord( )
{
    if (foundSet.nextRecord( ))
        populateFields( );
}

function previousRecord( )
{
    if (foundSet.previousRecord( ))
        populateFields( );
}

function populateFields( )
{

    document.getElementsByName("Name") . item(0) . value =
        foundSet.getFieldByName ("Name");

    document.getElementsByName("Title") . item(0) . value =
        foundSet.getFieldByName ("Title");

    document.getElementsByName("Phone") . item(0) . value =
        foundSet.getFieldByName ("Phone");

    document.getElementsByName("Picture") . item(0) . src =
        foundSet.getFieldByName ("Picture");

}

```

## Looking at the XML Inventory example

The XML Inventory example is a demonstration of a web-published database that uses the FMPXMLRESULT and FMPXMLLAYOUT grammars and Dynamic HTML (including JavaScript and the W3C Document Object Model) to display the XML data on the Web.

This example was designed to be viewed in the Internet Explorer 5.0 for Windows web browser. For information on new browsers that can be used to view the example, double-click FileMaker on the Web (installed in the FileMaker Developer 5 folder) to go to the FileMaker Developer product support pages in your browser.

The example uses an inventory database for office equipment and provides two methods for displaying records from the database—in a list or in a detailed view of each record. Web site visitors can switch between List View and Detail View, and add, edit, delete, or find records in the database.

The XML Inventory example includes:

- a text file containing the JavaScript library used for this demonstration, named FMP.js
- the Inventory.fp5 database
- HTML files for viewing, searching, and adding records to the database



The Inventory.fp5 database used in the XML Inventory example

To view the XML Inventory example:

1. If necessary, install the Inventory example files from the FileMaker Developer 5 CD.

FileMaker Developer 5 > External FileMaker APIs > XML > Inventory Example

See “Installing the FileMaker Developer software package” on page 1-2 for information.

2. Place a copy of the Inventory Example folder and its files in the Web folder inside the FileMaker Pro 5 application folder on your hard disk.

FileMaker Pro 5 > Web > Inventory Example

**Note** For security, when actually publishing a database on the Web, you should not keep the database file in the Web folder unless you plan to administer it remotely. See “Opening password-protected databases remotely” on page 6-15 for information.

3. In FileMaker Pro, open the Inventory.fp5 database and make sure it's shared via the Web Companion.

See “Enabling the Web Companion” on page 6-3 and “Sharing the database via the Web” on page 6-5 for information.

4. In your web browser, type localhost (or your computer's IP address) followed by /inventory example/ and press Enter.

<http://localhost/inventory example/>

<http://17.17.17/inventory example/>

For information on setting up your computer as the localhost, see “Testing your site without a network connection” on page 6-13.

The XML Inventory example opens in List View in the browser.

5. Select a record and click the Detail View button.

The currently selected record displays in a separate Detail View window. It is also possible to open the Detail View window by double-clicking a record item in the list. To close the Detail View window, click the Cancel button. To apply edits made to a record in Detail View, click the Update button.

Item	Category	Location	Cost
Desktop Machine	Computers	Dirk's Office	1499.00
		Date Purchased	Serial Number
		1/16/1997	479148YY66
		Model	DC1
Information			
Development Machine			

Update Delete Cancel

Detail View of a selected record in the browser

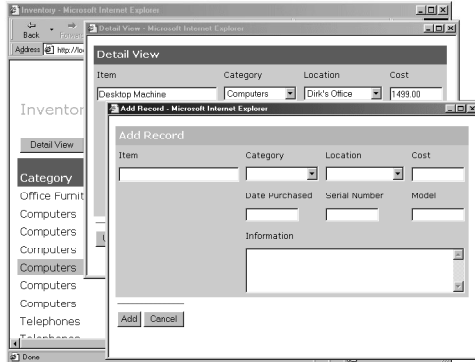
6. To add a record, click the Add button in the List View window. A separate Add Record window opens.

Item	Category	Location	Cost
		Date Purchased	Serial Number
		Model	
Information			

Add Cancel

7. Enter data for the new record and click Add to add it to the bottom of the list.

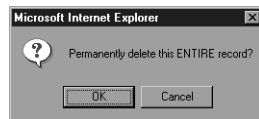
It is possible to have both the Detail View and Add Record windows open at the same time. The Detail View and Add Record windows will automatically close when a visitor clicks the Back or Forward button in the browser.



Detail View and Add Record windows open at the same time

8. To delete a record, click the Delete button in the Detail View or List View window.

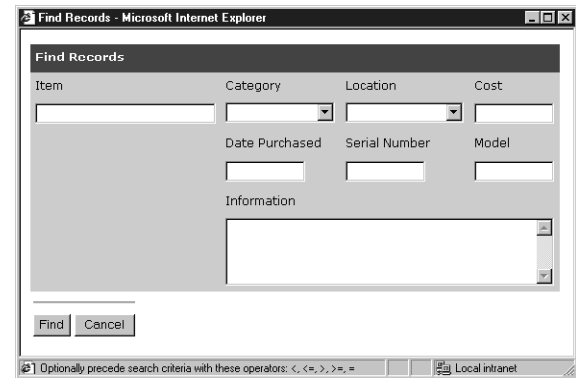
A confirmation dialog box provides options to cancel or delete the currently selected record.



9. Click Cancel.

10. To search for data, click the Find button in the List View window.

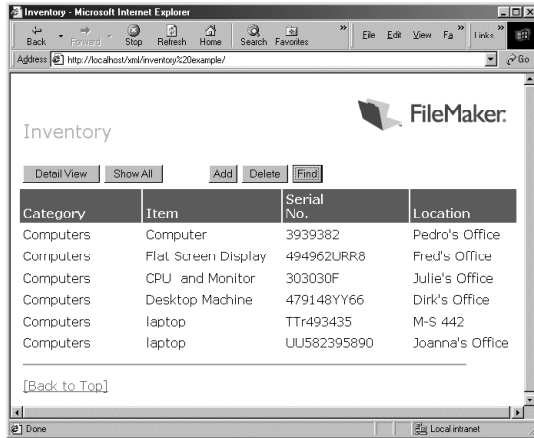
A detail view of an empty record opens in a separate window with a blinking insertion point in the Item field. When web users click inside the Item, Cost, Date Purchased, Serial Number, or Notes text boxes, a help string appears in the status bar at the bottom of the window. The default operator for a search is “begins with” or web users can type one of the <, <=, >, >=, or = operator symbols in these boxes.



Click in text boxes to display data entry instructions at bottom of window

11. Click in the Cost text box and type >=2000. Then click Find.

Any record containing the amount of \$2,000 or greater in the Cost field is found and displayed in List View.



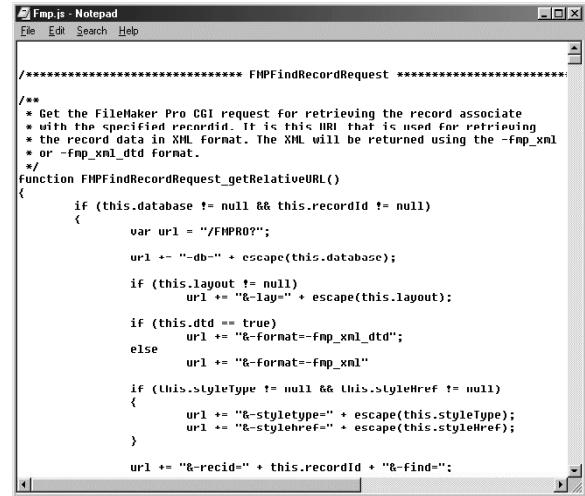
The found set of records is displayed in the List View window

12. In List View, click Show All to display all the records in the database, including any records you added.

If you want, you can examine the XML Inventory example's JavaScript library as you view the source for the example pages.

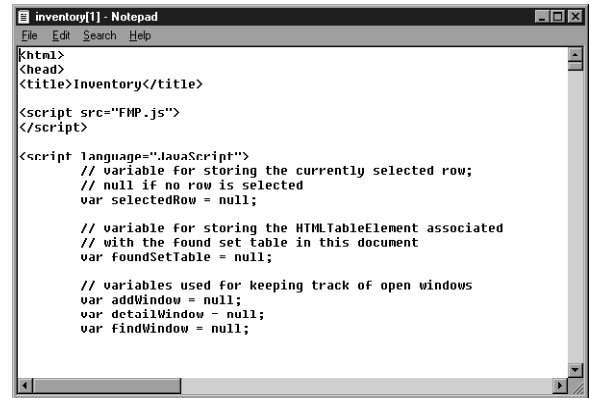
1. Start the Microsoft Notepad application (or a similar text editor), and choose File menu > Open. Locate and open the FMP.js file. (If necessary, choose All Files (\*.\*) as the Type.)

FileMaker Pro 5 > Web > Inventory Example > FMP.js



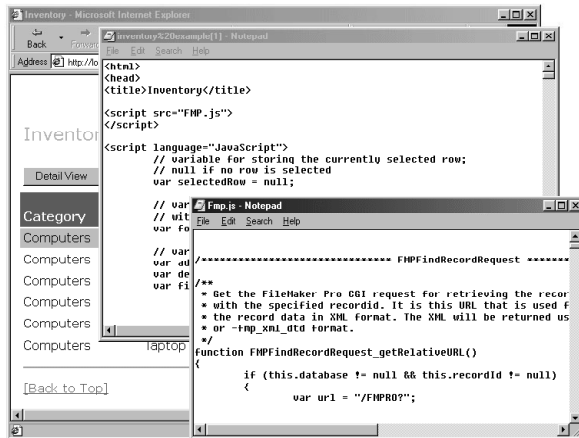
FMP.js text file containing the JavaScript library for this example

2. In the browser window, choose View menu > Source to see the source markup for the example pages in the Notepad application.



Source markup of the Default.htm page (List View)

The scripts and functions are well commented in the source and in the FMP.js library, providing information about each step. It's useful to arrange the windows so you can switch between the example pages in the browser, the source, and the script library.



**Tip** Use code snippets from the FMP.js JavaScript library for scripting your own web sites.

For more information about delivering your FileMaker Pro data using XML, double-click FileMaker on the Web (installed in the FileMaker Developer 5 folder).