

# Using XML in ColdFusion

## A starting point



By Jeff Houser

**X**ML is over 8 years old. How many of you have been using it that long? (Feel free to raise your hand.) I don't see a lot of hands, which doesn't surprise me. However, I bet a lot of you have used XML without even

knowing it.

XML has great flexibility and there are numerous uses. Its use is becoming more and more common with every passing day. You've probably come across it in some form even if you didn't know it was XML. If you haven't used it yet, you will soon. This article is all about XML and XML within ColdFusion.

### What Is XML?

An explanation seems like a good way to start this article, so what is XML exactly? XML stands for eXtensible Markup Language. This markup language is tag-based similar to HTML or (most of) CFML. Whereas HTML is used to define the look and feel of a Web page and CFML is designed to issue commands to the ColdFusion application server, XML is used to define data. The primary use of XML is to share data between different systems. Sometimes these systems are radically different.

The main advantage to using XML is that if properly designed, it can be easily read by both a computer and a person (even some non-technical people). It supports Unicode so many types of information can be translated. XML is usually provided as plain text and is free from any license restrictions and easily readable from any language. It does have some drawbacks, however. The syntax is fairly verbose. That can make for good readability, but can decrease processing efficiency especially when dealing with

large amounts of data. Binary data formats are known to be much more efficient. As an example, Flash Remoting is more efficient than Web services for this very reason. XML does not have any inherent support for advanced data types, so you'll have to run some processing to turn an XML object into a native ColdFusion structure or array. As you have seen in this article it can be cumbersome to access individual elements in an XML document.

There are two requirements of an XML document: they must be well formed and they must be valid. A well-formed document is one that conforms to the syntax rules. To make sure that you conform you should follow some of these simple rules:

- Close all tags. Even tags that do not contain data between the open and close tag need to be closed. It's fine to use the shorthand syntax, which is to close it before the close bracket, but it must be closed.
- Make sure there is only a single root element. Multiple root elements are invalid in XML.
- Put all attribute values in quotes, either single or double.
- Tags must not overlap. You can nest tags, but make sure you properly close all the inner tags before you close the outer tags.

Follow these instructions to make sure that your XML is well formed.

To determine validity, we compare the XML document against the DTD (Document Type Definition) or XSD (XML Schema Definition). The DTD, or XSD, define the tags and attributes to those tags that make up your XML syntax. Both HTML and CFML have a specific set of defined tags and the attributes that can be used with those tags, but XML does not. It's up to you to define the tags that make up your XML in a

DTD or XSD. You can use a DTD or XSD to define things like the names of tags, their data types, their attributes, whether they are required or optional, and their children if any. To determine if an XML document is valid, compare the XML to the rules in the DTD or XSD.

Unfortunately, in my experience with the corporate world, most developers do not bother to create either of these types of documents, so verifying for validity is often



not an option. But, there are some common implementations of XML that are in use. You've probably heard or used many of them. Here are a few:

- **WDDX:** WDDX is a specific dialect of XML and one that integrated with ColdFusion even before CF had native XML capabilities. If you ever used WDDX, you were using XML without even knowing it.

- **RSS:** RSS is used for the syndication of Web data. Its usage is very common in blog sites. CFDJ offers RSS feeds for their articles. This example shows an RSS feed of all the articles that I've written: <http://coldfusion.sys-con.com/author/3566Houser.rss>.
- **SOAP:** SOAP is the language of Web services. It stands for Simple Object Access Protocol and is XML.

- **XHTML:** XHTML is an XML version of HTML. It has a stricter syntax than HTML (for example, all tags must be closed). It is expected to be the long-term successor to HTML.
- **MXML:** MXML is the language of Macromedia Flex and it's an XML implementation.

Next, I'm going to show you how to create and use XML objects in ColdFusion.

## Creating an CF XML Object

There are two different ways that you can create an XML object inside ColdFusion. The first is to use the CFXML tag, and the second is to use the XMLParse function. CFXML is good if you are going to create an XML document from scratch within ColdFusion. The XMLParse function will take a string that contains an XML document and return the XML document.

The cfxml tag contains two attributes:

- **Variable:** The variable attribute defines the variable that will contain the resulting XML object. It is required.
- **CaseSensitive:** The CaseSensitive attribute specifies whether the case used in the XML document should be maintained. It is an optional Boolean attribute with the default value of no.

Between the open cfxml tag and the close cfxml tag, specify the XML. For our example, I'm going to use some XML borrowed from the *CFDJ* RSS feed (see above for the link (see Listing 1)).

I cut and pasted the XML directly from the *CFDJ* feed, then modified it to include fewer records. I set it to not be case sensitive, and stored the XML object in the variable "MyXML". The last line of code dumps the object. Inside the cfxml tags you can use any sort of ColdFusion processing that you need to create your XML object. For example, suppose you wanted to loop over the records of a database and use those records to create an XML object? We can do that with the CFXML tag.

XMLParse is a function used to turn a string into an XML object. Perhaps you are consuming a Web service that returns XML or using cfhttp to get an XML document. The XMLParse function

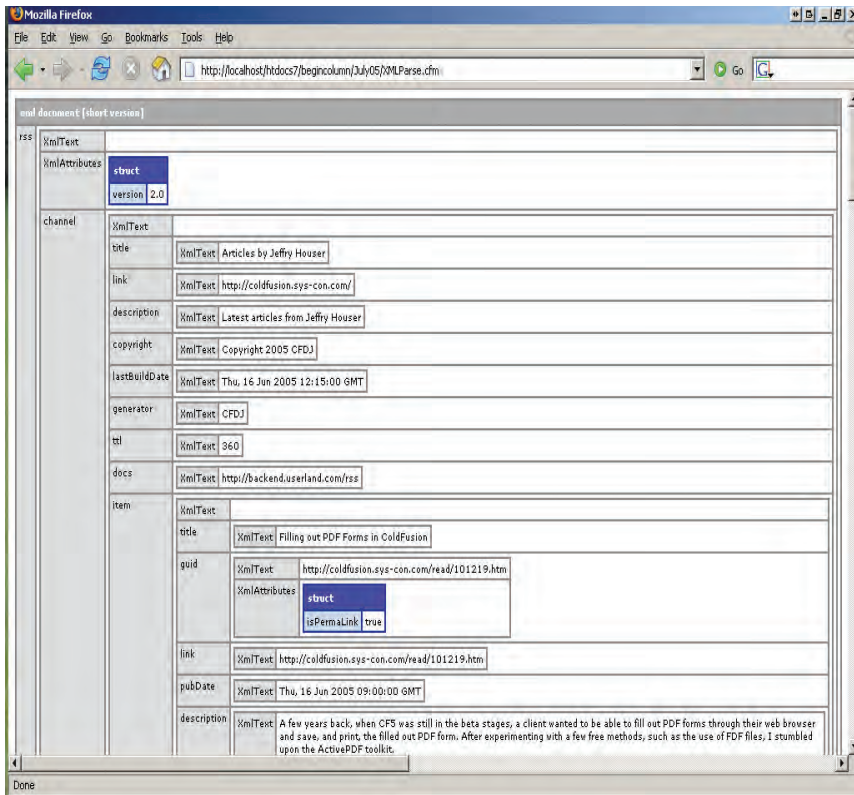


Figure 1: cfdump an XML object

XML Structure	Data
<?xml version="1.0" encoding="UTF-8"?>	No data to Access
<rss version="2.0">	MyXML.rss.xmlattributes.version
<channel>	No data to Access
<title>Articles by Jeffrey Houser</title>	MyXML.rss.channel.title.xmltext
<link>http://coldfusion.sys-con.com/</link>	MyXML.rss.channel.link.xmltext
<description>Latest articles from Jeffrey Houser</description>	MyXML.rss.channel.description.xmltext
...	...
<item>	No data to access
<title>Filling out PDF Forms in ColdFusion</title>	MyXML.rss.channel.item[1].title.xmltext
<guid isPermaLink="true">http://coldfusion.sys-con.com/read/101219.htm</guid>	MyXML.rss.channel.item[1].guid.xmlattributes.ispermalink MyXML.rss.channel.item[1].guid.xmltext
<link>http://coldfusion.sys-con.com/read/101219.htm</link>	MyXML.rss.channel.item[1].link.xmltext
...	...
</item></channel></rss>	No data to access

Table 1: How to access XML

has one argument, which is the string that contains the XML.

```
<cfhttp url="http://coldfusion.sys-con.com/author/3566Houser.rss">
</cfhttp>

<cfset MyXML = XMLParse(cfhttp.FileContent)>

<cdump var="#MyXML#">
```

The code grabs the CFDJ RSS feed using the cfhttp tag. The cfhttp.filecontent variable is a string. The XMLParse function is used to turn that string into an XML variable. The results dump the XML object to the screen, as shown in Figure 1.

## Accessing the XML Object

Now that you've created an XML object, how do you use it within ColdFusion? This section will show you how. An XML object in ColdFusion is often a group of embedded structures and arrays. You can tell what's what by viewing the long version of the cfdump. Sometimes it can be tricky to parse this data because of its complexity. This is a list of elements that we'll need to access our data in the XML object:

- **XMLRoot:** The XMLRoot is stored as an "XMLElement". You'll use this to access most of the data in your object. Each additional element I discuss here is part of the XML Element object. This keyword "XMLRoot" is interchangeable with the name of the top-level node. In the RSS feed, we could use "MyXML.xmlroot" or "myXML.rss".
- **XmlName:** The name of the XML Element and it's a string.
- **XmlText:** This is the content that resides between an open and close tag. It's a string.
- **XmlAttributes:** The XMLAttributes is a structure. It contains the name, and values, of each attribute of the XML Element.
- **XmlChildren:** XMLChildren is an array of all the children's elements. Each child is treated as an XML Element of its own.

A complete list of what is associated with an XML document in ColdFusion is located here <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00001510>.

[htm#1119477](http://htm#1119477). Table 1 shows how to access each line of the original XML file.

This is an example of what we would use to loop over the XML document and display a link to each article in the RSS feed:

```
<cfloop from="1" to="#arrayLen(MyXML.rss.channel.item)#" index="CurrentLink">
<cfoutput>
  <a href="#MyXML.rss.channel.item[CurrentLink].link.xmltext#">
    #MyXML.rss.channel.item[CurrentLink].xmltext#
  </a><br>
</cfoutput>
</cfloop>
```

You've probably run this type of code many times against a ColdFusion query object. The code loops over the array of items. In the XML object items is an array of "XMLChildren". It displays a link using the "link" and displays the title as the link text, using a type of access similar to what is in the table.

## Where to Go from Here

With this basic introduction to XML, where do you go from here? Well, you'll definitely want to read more about ColdFusion's XML functions: <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000372.htm#3468770>. You may want to investigate XPath to allow you to search through an XML object to find more information. XSL (Extensible Style sheets) are also intriguing. They allow you to transform an XML document using a style sheet to make it ready for Web display. ColdFusion supports XSL with the XMLTransform function. Information XSL is located here: <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00001520.htm#1209889> and the XMLTransform function is here: <http://livedocs.macromedia.com/coldfusion/7/htmldocs/00000673.htm#140093>. This article should give you a good starting point for using XML in your applications.



## About the Author

*Jeff Houser has been working with computers for over 20 years and in Web development for over 8 years. He owns a consulting company and has authored*

*three separate books on ColdFusion, most recently ColdFusion MX: The Complete Reference (McGraw-Hill Osborne Media).*

[jeff@instantcoldfusion.com](mailto:jeff@instantcoldfusion.com)

## Listing 1

```
<cfxml casesensitive="no" variable="MyXML">
  <?xml version="1.0" encoding="UTF-8"?>
  <rss version="2.0">
    <channel>
      <title>Articles by Jeffrey Houser</title>
      <link>http://coldfusion.sys-con.com/</link>
      <description>Latest articles from Jeffrey Houser</description>
      <copyright>Copyright 2005 CFDJ</copyright>
      <lastBuildDate>Thu, 16 Jun 2005 09:32:00 GMT</lastBuildDate>
      <generator>CFDJ</generator>
      <ttl>360</ttl>
      <docs>http://backend.userland.com/rss</docs>

      <item>
        <title>Filling out PDF Forms in ColdFusion</title>
        <guid isPermaLink="true">http://coldfusion.sys-con.com/read/101219.htm</guid>
        <link>http://coldfusion.sys-con.com/read/101219.htm</link>
        <pubDate>Thu, 16 Jun 2005 09:00:00 GMT</pubDate>
        <description>
          A few years back, when CF5 was still in the beta stages, a client wanted to be able to fill out PDF forms through their Web browser and save and print the filled out PDF form. After experimenting with a few free methods, such as the use of FDF files, I stumbled upon the ActivePDF toolkit.
        </description>
      </item>
    </channel>
  </rss>
</cfxml>

<cdump var="#MyXML#">
```

Download the Code...  
Go to [www.coldfusionjournal.com](http://www.coldfusionjournal.com)