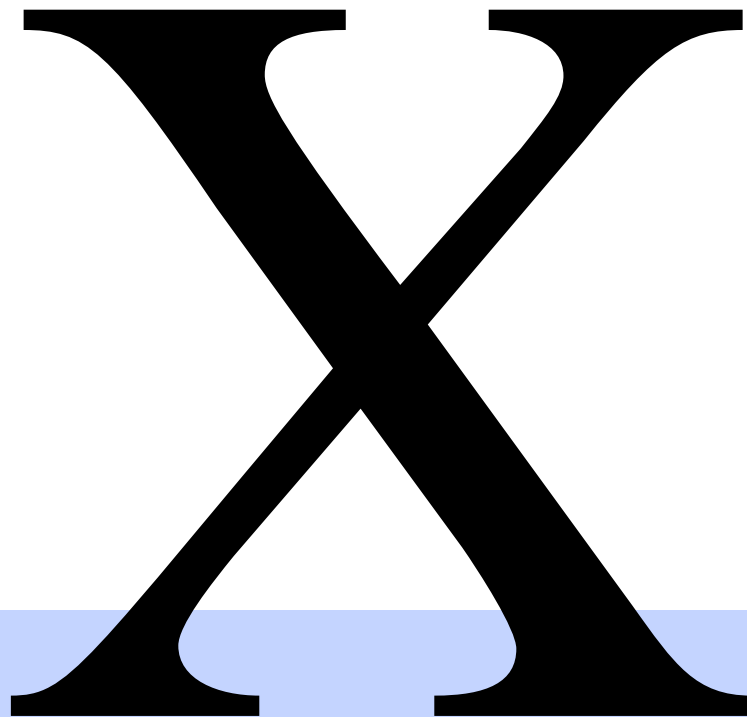


Efficient Website Publishing  
with XML



# **XPublish 1.0**

**The Tutorial**

**by Media Design in•Progress**

© 1995-98

**Media Design in•Progress and Terje Norderhaug.** All rights reserved.

This publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, provided that the same proprietary and copyright notices must be affixed to any copies as were affixed to the original. This exception does not allow additional copies to be sold to others. This publication and the software described in it may not be licensed to others. The specific and complete license agreement is available in the README file in the same folder as the Software.

Interaction, Interaction/IP, XPublish, Cascade and in•Progress are trademarks of Terje Norderhaug and Media Design in•Progress.

Apple is a trademark of Apple computer, Inc., registered in the United States and elsewhere. MCL is a trademark of Digttool, Inc. WebStar is a trademark of StarNine. Web Server 4D is a trademark of MDG Inc. Quid Pro Quo is a trademark of Chris Hawk. MacHTTP is a trademark of Chuck Shotton. NetPresenz is a trademark of Peter Lewis. Other product names and company names mentioned in this publication may be trademarks or registered trademarks of their respective companies and are hereby acknowledged as such

**LIMITED WARRANTY:**

MEDIA DESIGN IN•PROGRESS MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS PROVIDED "AS IS", AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY. IN NO EVENT WILL MEDIA DESIGN IN•PROGRESS OR TERJE NORDERHAUG BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised about the possibility of such damage.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Media Design in•Progress dealer, agent or employee is authorized to make any modification, extension, or addition to this warranty. Some states do not allow the exclusion or limitation of implied warranties or liability of incidental or consequential damages, so the above limitations might not apply to you. This warranty gives you specific legal rights, and you might also have other rights which vary from state to state.

# Introduction

XPublish is an XML website publishing system for efficient authoring, publishing and maintenance of medium-sized to large Web sites. The latest version of XPublish is available from:

<http://interaction.in-progress.com/xpublish>

## THE XPUBLISH TUTORIAL

This tutorial gives step-by-step examples for basic tasks of publishing a website using XML and XPublish. To facilitate a smooth transition for those familiar with HTML, the lessons focus on how to extend HTML with XML markup. You are recommended to complement the tutorial with a book or documentation on XML.

The first lesson demonstrates how to publish a site as HTML pages for standard browsers. Lesson 2-5 teach you the basics in using XPublish for authoring, included use of general entities (includes) to eliminate repeated information, design of style sheets for consistent appearance of the site, and use of your own tags. Lesson 6-9 introduces how you can customize the authoring and publishing environment by declaring your own XML elements, entities, and document types.

## BENEFITS OF XML IN WEB PUBLISHING

XML is very well suited as the source format for content. Generally, it is preferable to maintain your content in XML rather than in page layout formats including HTML. Some of the benefits of keeping your content in XML includes:

- Reach a wide audience by authoring the content once and publish to multiple formats in addition to the web. For example, the same XML documents can be cross-published for print, on-line viewing and for the web with better results than other formats.
- Maintain flexibility by not binding the content to a specific presentation. The documents can be presented in an unlimited number of ways by applying different style sheets. The presentation style can be designed or changed at any time.
- Ensure consistency by focusing the creativity of authors on content rather than presentation. With XML, the authors will be able to concentrate on the message, leaving the presentation of the site to the publisher. The presentation will be the same no matter the author.
- Allow interchange of content by keeping it in standard XML format. XML is technically a subset of the more comprehensive International SGML standard, allowing XML documents to be reused with SGML compliant software. Also, an increasing number of software programs can import or export XML.

## THE XML WEBSITE PUBLISHING MODEL

Many of the benefits of using XPublish stems from its publishing model. Rather than authoring pages directly in the presentation format (HTML), the site is constructed using rich XML markup that is converted automatically into HTML based on a style sheet.

### Descriptive Markup

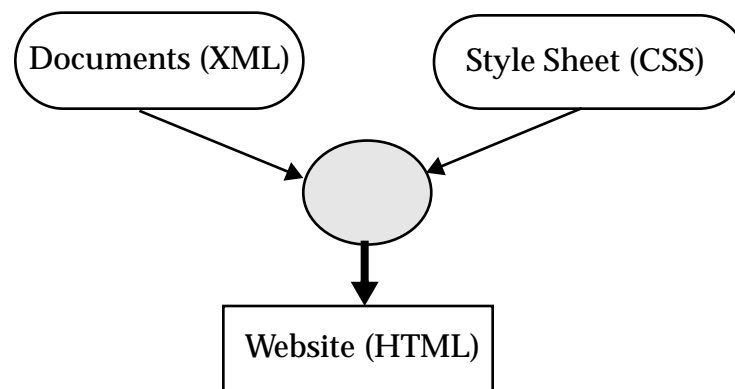
The core idea of XML is that markup should describe the logical structure of content rather than how it is presented. XML allows the author to explicitly describe the roles of the various parts of a document, allowing software to process the content in powerful ways.

### Style Sheets

XML maintains a strict separation between content and presentation. The presentation is designed in one or more style sheets. The same content can be presented based on various style sheets. The presentation is not determined at authoring time but is applied at the time of publishing or reading. XML thus goes far beyond the WYSIWYG paradigm (What You See Is What You Get), allowing for any number of presentations of the same content rather than limiting the presentation to how the content appears for the author.

### Automated Processing

At publishing time, XML documents will be combined with a style sheet and automatically processed according to various rules to generate the presented pages:



Separating content authoring, style design, and final output resembles the publishing procedure common in print media. It allows authors to focus on writing the content, leaving the design of the presentation to the specialists.

## HOW CAN WE MAKE XPUBLISH MORE USEFUL?

Your feedback is crucial in making XPublish as useful as possible for your publishing needs. We appreciate hearing from you about what improvements you would like to get for upcoming releases. The email address for feedback and suggestions is:

[feedback@in-progress.com](mailto:feedback@in-progress.com)

## TROUBLESHOOTING

If you encounter any inconsistencies with XPublish, the first step is to quit the application, trash the file called “Cache Database”, and start the application again. This will make XPublish rebuild its cache (an object database) from the files in the Documents folder.

If trashing the cache doesn't solve the problem, you are advised to check our website for a fix or to download the latest version of the application. We will post “Update” plug-ins as soon as a problem has been detected and eliminated. The following URL provides the latest information:

<http://interaction.in-progress.com/xpublish/update>

We appreciate reports about flaws or inconsistencies so we can improve XPublish. Please include a detailed description of the problem as well as the version code and eventual update of XPublish. Often we will be able to provide a fix within the same day. Reports should be emailed to:

[bugs@in-progress.com](mailto:bugs@in-progress.com)

## ACKNOWLEDGMENTS

XPublish is a result of almost five years of making website publishing systems based on extensible markup. I am grateful to Erik Naggum for introducing me to SGML while we were hanging out at the computer lab at the Institute of Informatics, University of Oslo. Bjørn Remseth is appreciated among other things for handing me a copy of Goldfarb' SGML Handbook while I was designing my first systems for automated website generation in 1993.

Håkon Lie deserves credit for the way style sheets are used in XPublish as well as for improving my concepts about personalization and adaption.

Everybody involved in moving XML to a standard has my deepest appreciation. It has been a long time since 1993/94 when some of us started to suggest that extensible markup should be used on the web. I am pleased about the result.

I have great gratitude to the folks at DigiTool for Macintosh Common LISP, a premier rapid development environment for implementing professional publishing systems.

I would like to dedicate XPublish in the memory of Yuri Rubinsky. He has been an inspiration, and also personally influenced the use of entities in XPublish.

*Terje Norderhaug*

*President  
Media Design in•Progress  
San Diego, March 18, 1998*

# Lesson 1: Publishing the Site

This first tutorial goes straight to the essence of what XPublish is about: publishing a website. After completing the tutorial, you will know how to generate a website with XPublish.

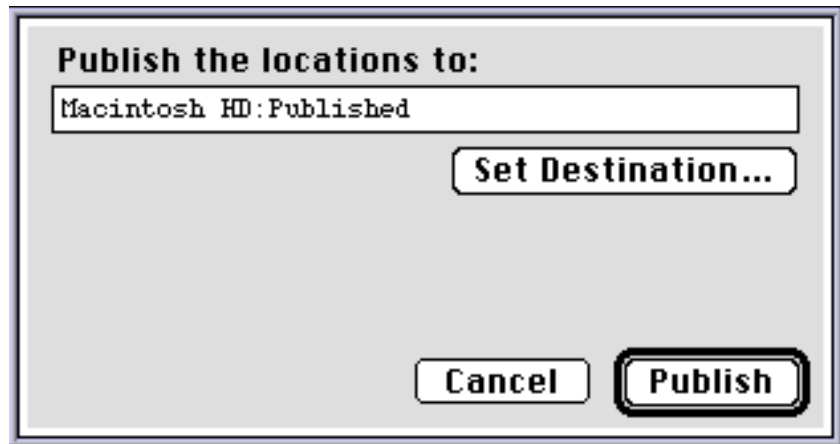
Follow these steps to generate a website with XPublish:

## 1. Open the Locations dialog by choosing “Locations” from the File menu of XPublish

The Locations dialog contains an overview of the location hierarchy of the site as it appears for visitors. Each Location corresponds to a page in the site. Double clicking any of the locations opens the associated document for editing.

## 2. Choose “Publish” from the File menu of XPublish

The Publish dialog lets you select the folder for the resulting pages. The path for the folder will depend on where XPublish is placed on your computer, so the example path in the dialog to the right may be different for you.



## 3. Click the “Publish” button

Answer “yes” if you are asked whether it is OK to overwrite an existing folder. XPublish will build an HTML page for each location by processing the XML documents. Any file in the Documents folder that doesn’t contain markup will be copied to the destination folder.

Your site is now ready to be browsed. Open the “index.html” page in your favorite browser to check out the result.

## LEARNING MORE

You may want to compare the markup of the Index location with the HTML in the published page. To open the document of the index location, double-click on “index” in the Locations dialog of the File menu. To view the markup of the corresponding published page, select “view source” or equivalent in the browser.

Note the special markup in the document such as `&footer;`. These are called entity references, and are replaced by a value at publishing time. **Lesson 5: Using General Entities** tells you more about how to use such entity references to maintain content that is used in multiple locations.

You will recognize that the markup of the “index” location is essentially HTML with some additional markup constructs. XPublish allow you to extend HTML with XML. Thus you can import your current HTML files into XPublish and start making use of the XML features when you are ready.

## Lesson 2: Creating a Location

As you may recall from **Lesson 1: Publishing the Site**, each location is converted into a page in the published site. By adding locations, you are in effect expanding the final website with new pages.

This lesson tells you how to add a location to your website. You will be creating a location based on an default HTML document with XML entity references.

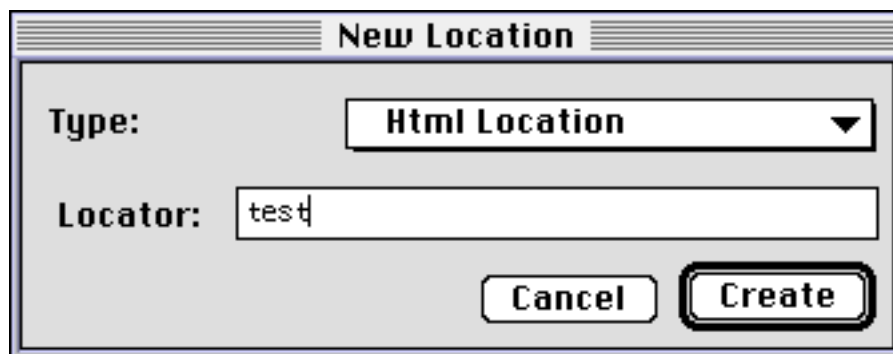
The following steps will add a new HTML location called “test”:

### 1. Open the Locations dialog from the File menu of XPublish

The Locations dialog lists all available locations. Each location corresponds to a page available on your site.

### 2. Choose “New Location” from the File menu

The New Location dialog allows you to specify a locator as well as a type for the new location:



### 3. Write “test” in the field for Locator

The Locator is the part of the URL that follows the domain name, excluding the suffix. The URL that corresponds to the locator “test” above is thus:

```
http://www.yoursite.com/test.html
```

### 4. Click the “Create” button

XPublish will create a new location and automatically open the associated document in the editor. The document will have default content, such as:

```
<?XML version="1.0"?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
    "http://www.w3.org/TR/REC-html4/strict.dtd">
<HTML>
  <HEAD><TITLE>test</TITLE></HEAD>
  <BODY>
    <H1>Heading</H1>
```

```
<P>XML Entity References such as time (&time;) and date (&date;)
are available from the Entities dialog.</P>
```

```
&footer;
</BODY>
</HTML>
```

### **5. Save the document using Save from the File menu**

XPublish will save the document to a file in the Documents folder, and update its Cache Database with the content.

### **6. Publish the site as explained in Lesson 1: Publishing the Site**

The new location is now available as a page in the published site.

## **ALTERNATIVE WAYS OF ADDING LOCATIONS**

XPublish provides several other ways to add documents to your site. Existing files can be imported using “Import” from the File menu. Alternatively, drag & drop files or folders onto the Locations dialog.

If you have many existing files that you want to add, you have the option of placing them directly in the Documents folder. However, this requires that you rebuild the database of XPublish by quitting the application, trashing the file called “Cache Database”, then restart the application.

## Lesson 3: Marking Up Documents

This lesson demonstrates how to markup a document according to XML. XML differs from HTML by more focus on describing the content and structure rather than presentation.

Descriptive markup is a key to successful use of XML. It requires that you liberate yourself from the paradigm that a document defines how the content should be presented. Instead, tags are used to declare the role of the various parts of the document, leaving the presentation to be designed in style sheets (see “Designing a Presentation Style” on page 13).

In this lesson you will type the markup by hand. **Lesson 7: Declaring Custom Elements** will teach you how to customize the XPublish to provide buttons and dialogs for efficient markup.

The following steps creates a simple XML document with descriptive markup. The example uses HTML elements in an XML manner, together with custom elements that describe the content:

### 1. Choose “New Location” from the File menu of XPublish

### 2. Choose XML for the Type of the new location

An XML location is defined with an XML document. Use the HTML type only if it is based on existing HTML content that isn’t well-formed XML (**Lesson 3: Marking Up Documents** introduces the basic requirements for markup to be well-formed according to XML, such as that each start tag has a corresponding end-tag).

### 3. Fill in “shakespeare” as the locator of the new location, and click the Create button

You have now created a new location called “shakespeare.” XPublish will open an XML document that is empty except for a prolog that declares that the document adheres to XML:

```
<?XML version="1.0"?>
```

### 4. Type in <HTML> after the XML prolog

The <HTML> start tag declares that what follows is marked up with HTML tags.

### 5. Type in the following HTML header:

```
<H1>Shakespeare’s Plays</H1>
```

The H1 element declares that “Shakespeare’s Plays” is a top level header. Be aware that XML tags are case-sensitive (i.e. that it makes a difference whether you use upper or lower case). You are recommended to **always use upper-case letters** in the identifier of tags for compatibility reasons.

### 6. Type in the following paragraph:

```
<P>Shakespeare is one of the best known playwrights of all times.</P>
```

The <P> start-tag **starts** the paragraph element, and the</P> end-tag ends the paragraph element. This markup stating that what is in between is a paragraph. XML

**requires** that all start tags have a corresponding nested end-tag to specify when the element ends.

### 7. Type in the following element:

```
<BLOCKQUOTE>To be or not to be...</BLOCKQUOTE>
```

A blockquote element is used to declare that its content is a quote. The BLOCKQUOTE element is never used to make an indent. In XML, presentation is always left to be designed in a style sheet.

### 8. Type in the following paragraph element:

```
<P>Shakespeare wrote many plays, including <PLAY>Hamlet</PLAY>, <PLAY>Macbeth</PLAY> and <PLAY>Romeo and Juliet</PLAY>.</P>
```

This paragraph introduces the custom element PLAY. Note how each of Shakespeare's plays are marked up with the PLAY element, with a <PLAY> start-tag before and a </PLAY> end-tag after. This markup allow XPublish, Interaction, and other XML processors to make sense of the content and process it appropriately.

Use such content specific tags to better describe the structure and meaning of your documents. Custom element types allows advanced processing and more efficient and powerful web publishing. **Lesson 7: Declaring Custom Elements** tells how to formally declare such custom elements, but declaring the elements aren't strictly required to use XML.

### 9. Type in </HTML> in the end of the document

The HTML end-tag corresponds to the <HTML> start-tag that opened the document. The document is thus one HTML element. XML documents always have such a root element that contains all the content of the document.

Congratulations! You should now have the following XML document:

```
<?XML version="1.0"?>
<HTML>
  <H1>Shakespeare's Plays</H1>
  <P>Shakespeare is one of the best known playwrights of all times.</P>
  <BLOCKQUOTE>To be or not to be...</BLOCKQUOTE>
  <P>Shakespeare wrote many plays, including
    <PLAY>Hamlet</PLAY>,
    <PLAY>Macbeth</PLAY> and
    <PLAY>Romeo and Juliet</PLAY>.
  </P>
</HTML>
```

Follow the steps in **Lesson 1: Publishing the Site** to publish the result.

## ABOUT XML

Basic XML markup is perhaps easier than HTML, as you can create your own tags that matches your content. There are only a few things you need to know in order to markup any content with XML:

- A document is marked up with elements that describe the document structure.
- You can use **custom elements** to optimally describe the structure of your documents. Elements should be labeled with a noun that describes the content of the element.
- An element starts with a start-tag, e.g. `<PARAGRAPH>`, followed by content, ending with a corresponding **mandatory end-tag**, e.g. `</PARAGRAPH>`. Elements can be nested by enclosing other elements within an element.
- The start-tag of an element can contain attributes with characteristic qualities of the element. You can use **custom attributes**. Attributes should be nouns or adjectives that describe significant characteristics.
- An attribute has a name and a value with an equal sign in between. **Attribute values are always in quotes**. Here is an example of an element with an “author” attribute:  
`<PARAGRAPH AUTHOR="Terje">This is a paragraph authored by Terje</AUTHOR>`
- XML is **case sensitive** (i.e. it makes a difference whether an element identifier or attribute name is upper case or lower case). You are advised to use upper case in element identifiers and attribute names for backward compatibility with HTML.

This is almost all you need to know in order to author with extensible markup. XML has many features to formally declare your own markup constructs, but you aren't required to use these when authoring documents with XML.

### ABOUT THE MARKUP EDITOR

The markup editor of XPublish is a Macintosh clone of the powerful Emacs editor. The editor provides numerous shortcuts that are the same as its UNIX/Windows counterpart. At the same time, the markup editor of XPublish has Macintosh ease of use, with standard editing tools, pane splitting, drag & drop, and more.

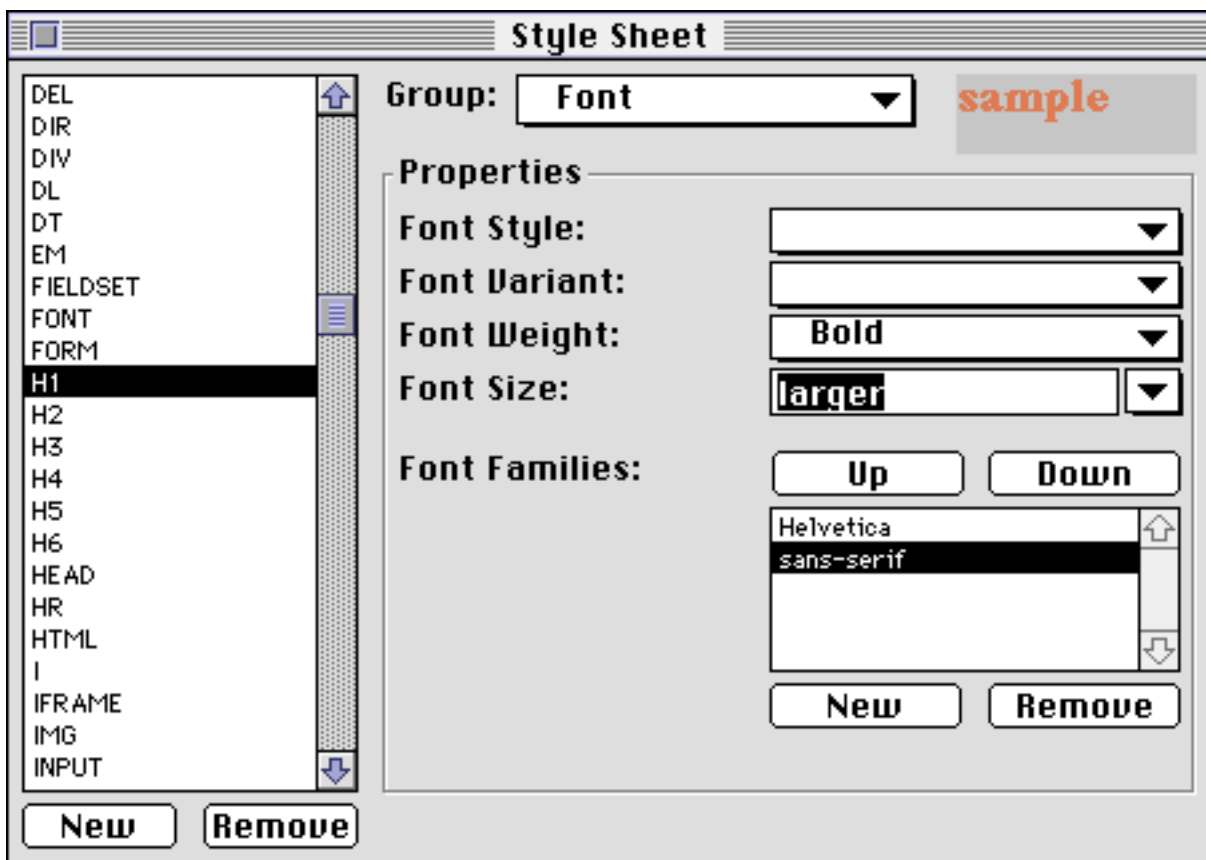
Some Emacs short-cuts useful for power users are CTRL-k to kill a line, CTRL-y to yank back a killed line, and CTRL-a to get to the start of a line. Just as Emacs, the editor of XPublish can be infinitely customized and extended using LISP. New editor commands can for example be added to the Tools menu of XPublish by using the Edit Tools option.

# Lesson 4: Designing a Presentation Style

A style sheet contains the design of the look & feel of your website. It facilitates consistent presentation, and frees the document authors from the demands of layout and typography. Changes to the style sheet will take effect for the full site as it is published, making it easy to modify the presentation or to experiment with various designs.

XPublish supports Cascading Style Sheets, a style sheet standard that has been developed by the W3C to serve the needs of the web. This lesson introduces how to change the presentation style of the H1 element type of HTML using Cascading Style Sheets.

## 1. Open the Style Sheet dialog from the Windows menu of XPublish



## 2. Select H1 from the list of selectors

There will be a selector for any element type used in the documents or in a document type definition.

## 3. Select Bold for Font Weight. Type in “larger” for Font Size.

## 4. Click the “New” button for Font Family, and select “Helvetica” in the dialog. Repeat with “sans-serif” for Font Family.

It is always a good idea to list more than one font for font family. This improves the chance that the font is supported by the computer of the visitor to your site. You are

recommended to choose a general font family such as sans-serif as last option in case the browser doesn't have any of the fonts.

**5. Make a style for the P element with Palatino as font face and sans-serif as alternative.**

Select P from the list of selectors. Click the “New” button for Font Family, and select “Palatino” in the dialog. Repeat with “sans-serif” for Font Family.

**6. Make a style for the BLOCKQUOTE element with Smaller font size, Courier as font face and monospace as alternative.**

Select BLOCKQUOTE from the list of selectors. Click the “New” button for Font Family, and select “Courier” in the dialog. Repeat with “monospace” for Font Family. Set the Font Size to “Smaller” by choosing Relative Size in the pop-up menu for font size.

**7. Save the Style Sheet by closing the dialog**

Saving makes XPublish aware of the changes.

**8. Publish the site as described in Lesson 1: Publishing the Site**

The top headers all pages of the published site will now appear as designed in the style sheet.

**Lesson 8: Preparing HTML Conversion** will introduce you to designing a presentation style and HTML equivalent for custom elements. Take a sneak preview at the “Preparing HTML Conversion” lesson to learn how to make XPublish process custom elements such as PLAY.

## ABOUT CASCADING STYLE SHEET

Cascading Style Sheets is a style sheet standard specifically developed for the web. It differs from most other style sheet technologies in that it fosters adaptive presentation so that the same documents can be presented optimally depending on the context.

To benefit from the adaptive capabilities of Cascading Style Sheets, you are advised to use relative values (e.g percentage, ex, em) rather than absolute values (px, cm, in). This allows the presentation to be adjusted for the presentation medium while still maintaining the designed look & feel.

## Lesson 5: Using General Entities

An entity is a unit of information that is referred to with an entity reference. Any entity-reference in a document will be replaced by its entity value when the location is published. Entities are typically used for content fragments that are repeated more than once or are frequently changed.

Entities make it more efficient to maintain a site. Instead of using search-replace on the site to change repeated content, you can modify the entity value in one place and have the change immediately take effect for the full site.

The following tutorial demonstrates how to create a new general entity and place an entity reference to it in a document:

### 1. Open the Entities dialog from the File menu of XPublish

The Entities dialog lists all entities that can be used in **any** document of the site. The dialog shows a short documentation for each entity. Any of the entities in the editor can be dropped in a document to create a reference.

### 2. Choose “New Entity” from the File menu

The dialog for creating a new entity lets you choose between various entity types. Text Entities are simply replacement text or markup.



### 3. Type in “webmaster” as name for the new entity

The name is used to refer to the entity in documents. It can contain characters and digits, as well as one or more hyphens or periods.

### 4. Click the “Create” button

XPublish creates the entity and opens it in the editor. The new entity is listed in the Entities dialog.

### 5. Type in your email address in the entity editor

The entity value will be used as replacement text for a reference to the entity. Feel free to use markup such as making the email address into a link using HTML.

### 6. Open the “test” Location by double-clicking on it in the Locations dialog of the File menu

The editor will display the document of the location called “test”.

### 7. Type in the following paragraph After the last paragraph in the document:

```
<P>Our webmaster is &webmaster;</P>
```

### 8. Save the document

Saving the document makes XPublish aware of the changes.

### 9. Publish the website as explained in Lesson 1: Publishing the Site

The entity reference &webmaster; will be replaced with your name as the site is published.

You can use a general entity reference everywhere in the site, saving yourself from typing or copying the same text more than once. If you ever want to change the text, simply change the value of the webmaster entity by opening it in the Entities dialog. The change will take effect for all pages that refer to the entity.

## ABOUT ENTITIES

Using entities is key to efficient content management. The demonstration in this lesson is just one of many ways that entities can be used with XPublish. Here are a few hints for exploring the possibilities.

Entities can be created by dropping content onto the Entities dialog:

- Dragging a file onto the Entities dialog makes it possible to embed the content of the file with an entity reference.
- Dragging text from the editor onto the Entities dialog creates a text entity and automatically replaces the original content with a reference.
- Dragging and dropping an entity back on the Entities dialog creates an Alias (alternative name) for the entity.

The general entities can be used in any document. You can also create entities that are specific for a document or document type. How to create such local entities is described in **Lesson 6: Declaring Internal Entities**.

## Lesson 6: Declaring Internal Entities

Recall that an Entity is a unit of information and provides replacement text for an Entity Reference. An Entity Declaration advises about the existence of an entity, and either provides the replacement text or tells where the content can be found.

**Lesson 5: Using General Entities** demonstrated how to make use of general entities that are shared between all documents. This lesson introduces how to use XML markup to declare Internal Entities, entities that can be used only in one specific document. Declaring such entities has several purposes, including:

- Customize the Entity catalog palette with frequently used entities.
- Save typing by declaring internal entities for repeated content.
- Maintain frequently modified phrases as entities in the top of the document for easier updating of the content without the potential complications of search/replace.
- Override general entities to provide a document-specific replacement text.

Here is how to define an internal entity that provides the name of the author of the document:

### 1. Open the XML document for the Location called “shakespeare”

This is the document that you created in **Lesson 3: Marking Up Documents**. Open the Locations dialog from the File menu, and double-click the item labeled “shakespeare” to edit the document.

### 2. Select the word “Shakespeare” in the document

### 3. Open the Declare Entity dialog from the Markup menu

The Declare Entity dialog is used to declare a new entity or change an existing entity declaration. The selected text will automatically be presented as the default text for the entity.

The internal value is replacement text for the entity. The replacement text will substitute any reference to the entity in the document.

### 4. Type in lowercase “playwriter” in the field for Name

The entity name is used to reference the entity.

### 5. Change the Internal Value to “William Shakespeare”

The internal value is the replacement text of the entity.

### 6. Click the Insert button to add the entity declaration in the document

The selected text will be substituted with the entity reference `&playwriter;` The entity reference will be replaced with the entity value at publishing time.

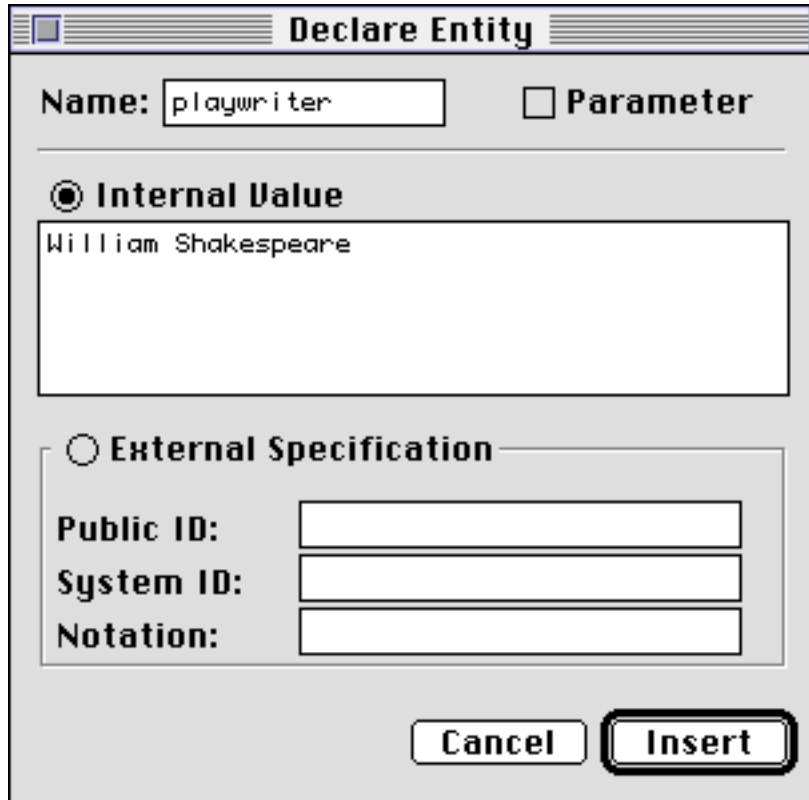
XPublish will automatically place the entity declaration in a document type declaration in the beginning of the document. The document type declaration formally specifies which markup is allowed in the document. If the document doesn't have a document type declaration, one will be created automatically before any other declarations are added.

The entity declaration should now appear similar to this:

```
<!DOCTYPE HTML [
  <!ENTITY playwriter "William Shakespeare">
]>
```

### 7. Open the Entity Catalog from the Palettes of the Windows menu

The Entity Catalog lists Entities that are specific for the document (in contrast to the general entities in **Lesson 5: Using General Entities** that can be referenced in any document). The new entity will be listed in the Entities Catalog.



### 8. Replace any occurrence of “Shakespeare” in the document with the entity reference `&playwriter;`

Click on the item in the Entities catalog to insert a reference in the document at the position of the cursor, replacing eventually selected content. You can also drag & drop the entity item onto the editor, or type a reference directly in the text.

### 9. Save the Document

Saving the document makes XPublish aware of the changes.

### 10. Publish the site as described in Lesson 1: Publishing the Site

All instances in the document of the `&playwriter;` entity reference will be replaced with the full name of the playwright in the final page.

You may find it useful to review **Lesson 5: Using General Entities** to see how internal and general entities can be combined for efficient and flexible content management.

## Lesson 7: Declaring Custom Elements

A major advantage of XML is that you can make your own element types. Recall from **Lesson 3: Marking Up Documents** that elements are used to describe the role of various parts of the document. Formally declaring the elements has several advantages, including:

- XPublish can provide buttons to quickly insert custom tags and elements in the documents, saving the author from repeated typing.
- XPublish can provide dialogs to define the various attributes of an element, making authoring more efficient and typing errors less likely.
- XPublish can provide on-line help advising the author about which attributes can be used in an element, as well as the structure of the content of the element.
- The document can be validated, warning about inconsistencies such as paragraphs within an header and other constructs that don't make sense.

The lesson demonstrates how to define a new document-specific element type. You are advised to complete the previous lessons in advance, included **Lesson 6: Declaring Internal Entities** which is a good warm-up for declaring custom elements. What you learn in this lesson is also useful for defining your own document types, as described in **Lesson 9: Defining a Document Type**.

In **Lesson 3: Marking Up Documents** we introduced a new `<PLAY> . . . </PLAY>` element to mark up the various plays of Shakespeare. Now is time to formally define the element:

### 1. Open the “shakespeare” location from the Locations dialog

The document of the location should now be available in the editor.

### 2. Open the Declare Element dialog from the Markup menu

The Declare Element dialog facilitates the declaration of custom elements.

Each element has a name and a content model that specifies the valid structure of the content of the element.

### 3. Type the noun “PLAY” as the name for the element

The name is the same as in the start tag `<PLAY>`.

XML Elements should have names that describe the content rather than its presentation. This gives you optimal flexibility to efficiently maintain the presentation in a

style sheet, opens for advanced processing, and allows the content to be reused and processed by a wide range of software.

As a rule of thumb, **use nouns as the name of your custom elements**. Good names includes HINT, PERSON, and HEADER. Avoid names that represent colors, sizes, fonts or other presentational aspects. Also, avoid using verbs as element names, as elements are not instructions or commands.

#### 4. Check the radio button for Model, and type in ANY as content model

ANY declares that the element can contain any other elements or content. XML allows you to specify which elements can be contained in an element so that XPublish can validate that the markup is correct. Further details are found in documentation for XML.

#### 5. Click the Insert button to complete the dialog

The Element Declaration is inserted in the Document Type Declaration of the document, which now should appear as follows:

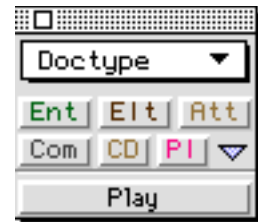
```
<!DOCTYPE HTML [  
  <!ELEMENT PLAY ANY>  
>
```

To modify the element declaration, place the cursor in the declaration and select Inspect from the Markup menu. This will open the declaration in the same dialog that you originally used to declare the element.

#### 6. Save the document

Saving the document will advise XPublish that a new element type has been defined. You will find the new element listed in the Tags Catalog, available from the Windows menu under Palettes.

Click the triangle to view the button that inserts a Play element in documentd. Select the name of a play in the document and click the Play button to mark it up.



A custom element will not be processed by XPublish until you associate it with processing rules in a style sheet. **Lesson 8: Preparing HTML Conversion** tells you how to use the style sheet to define the presentation style of your custom element and how the element should be converted into HTML.

## DECLARING ATTRIBUTES

Attributes are characteristic qualities of an element. The following steps will declare an attribute “year” that can be used in a play element to specify which year the play was written:

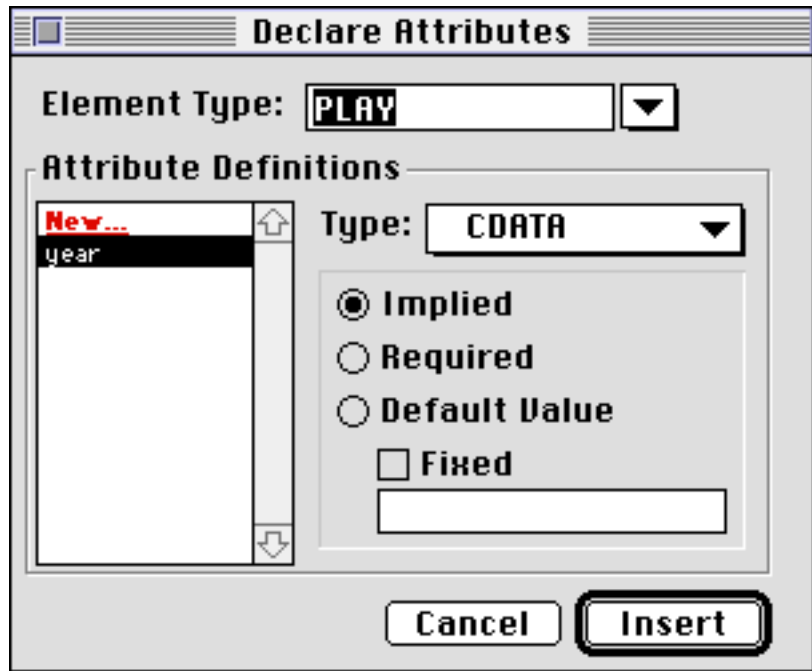
**7. Select “Declare Attribute” from the markup menu**

A dialog opens with fields to specify the attributes of an element.

**8. Select the PLAY element as the Element Type for the attributes**

**9. Click the New item of the list of Attribute definitions, and give the new attribute the name “year”.**

The attribute definition list has an item for each attribute that can be used in an element. See XML documentation for further details about how to specify the attribute type and value.



**10. Click Insert to complete the attribute declaration**

An Attribute List Declaration is now inserted in the Document Type Declaration of the document. The prolog of the document should now be as follows:

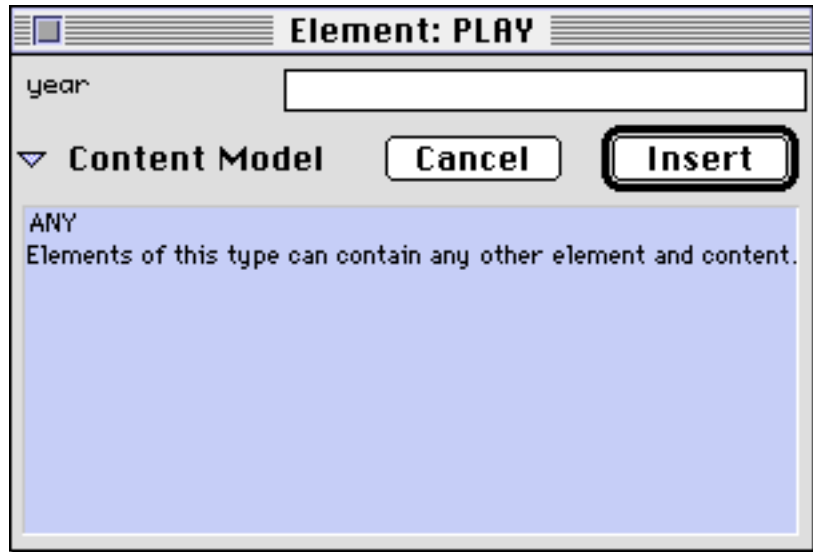
```
<?XML version="1.0"?>
<!DOCTYPE HTML [
  <!ENTITY playwright "William Shakespeare">
  <!ELEMENT PLAY ANY>
  <!ATTLIST PLAY
    year CDATA #IMPLIED>
]>
```

The Attribute List Declaration can be modified by placing the cursor in the markup and select Inspect from the Markup menu. You can also change the declaration directly as long as the document is saved afterwards.

### 11. Place the cursor in the start-tag of any **PLAY** element of the document, and select **Inspect from the Markup menu**

This will bring up a dialog to configure the attributes of the element. There will be a field for year.

Disclose the Content Model for information about valid content for the element. The explanation is created based on your custom content model.



### 12. Type in a year for when the play was written

Any year will do.

### 13. Click **Insert** to modify the start tag of the element with the changes

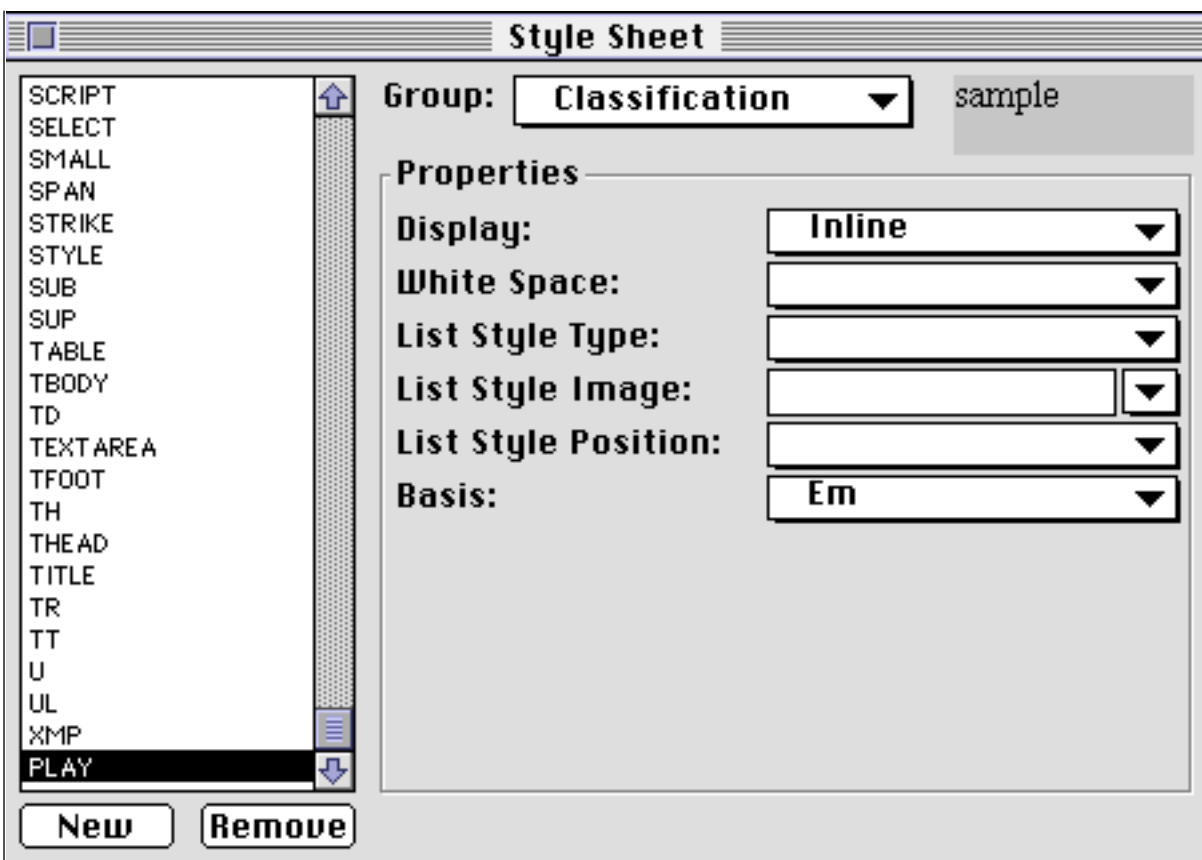
You are now familiar with declaring elements and their attributes. This lesson described how to declare elements for a specific document. **Lesson 9: Defining a Document Type** tells how you can use the same method to declare elements for many documents that share the same Document Type Definition.

## Lesson 8: Preparing HTML Conversion

XML allows you to use your own elements (**Lesson 3: Marking Up Documents**) as well as to define custom elements (see **Lesson 7: Declaring Custom Elements**). However, such custom elements don't make a difference unless you have specified the rules for how they should be converted into HTML and presented.

In this lesson you will specify how to process PLAY element into HTML.

1. Open the Style Sheet editor of the Windows menu
2. Select the PLAY item (create it if necessary)
3. Select Classification in the pop-up menu for Group



4. Set the Display property to Inline

**All custom element types require a Display setting to be processed into HTML by XPublish.** "Inline" means that the content of the element is part of the content of the surrounding element (similar to for example the EM element in HTML). "Block" declares that the element should be formatted as a separate block (similar to a paragraph). List Items declare that the element is used for elements in lists (similar to the LI element in HTML).

### 5. Set the Basis property to “EM”

The optional Basis property tells XPublish which HTML element comes closest to the meaning of your custom element. At publishing time, XPublish will convert the custom element to the HTML element defined with the basis property. This property is not part of the CSS1 standard.

### 6. Follow the description in [Lesson 4: Designing a Presentation Style to design a presentation style for the element PLAY](#)

### 7. Publish the site as described in [Lesson 1: Publishing the Site](#)

XPublish will automatically convert your custom tags into HTML at publishing time, adding the appropriate markup to maintain its presentation style as defined in the style sheet (see [“Designing a Presentation Style”](#) on page 13).

## ABOUT THE STYLIZER

XPublish comes with an optional Stylizer component that emulates the settings in the style sheet for older browsers that don't support CSS. It allows you to design with Cascading Style Sheets while maintaining a look & feel for visitors who haven't upgraded to more recent browsers. This liberates you from authoring your documents with presentation oriented HTML tags such as FONT and CENTER, allowing you to move on to strict HTML and XML.

The style sheet is emulated using presentational markup. As you may expect, the resulting HTML tends to be rather complex. To disable CSS emulation, remove the component called “Stylizer” from the Components folder of XPublish and restart the application.



## Lesson 9: Defining a Document Type

Documents often have a similar structure. XML allows you to define document types that contain the elements and rules for describing the structure of a document. HTML is an example of a general document type with many elements and entities. Using XML, you can define document types that are specialized for your content.

The lesson is based on **Lesson 6: Declaring Internal Entities** and **Lesson 7: Declaring Custom Elements**. In those lessons you practiced how to declare document specific entities and elements. The same declarations can be used to define a document type, so that you don't have to repeat the same declarations in multiple documents.

In this lesson, you will create a simple Document Type Definition (DTD):

### 1. Choose “New” from the Document Types submenu of the Markup menu

XPublish displays a dialog to type in the name of a new document type.

### 2. Type in “memo” as the name of your custom document type

XPublish will create a new file in its Document Types folder, and open the file in the editor.

### 3. Select Declare Element from the Markup menu

### 4. Type MEMO as the name of the element, type in (SENDER, RECEIVER, SUBJECT, P\*) for content model, and click the Insert button

An Element Declaration for HTML is now inserted in the Document Type Definition.

The content model declares what is valid content for elements of the type. The content model above declares that a MEMO should contain a sequence of the elements SENDER, RECEIVER, and SUBJECT, followed by any number of paragraphs.

The star '\*' after the P means that a paragraph element can occur zero or more times. A plus '+' in the same position would signify that a paragraph can occur once or more. A questionmark '?' would make the paragraph optional.

The different items in the content model for MEMO are separated by commas, which signifies that they should occur sequentially. They can alternatively be separated by a bar ( '| ' ) which signifies that the items can occur in any order.

The Element Declaration will be placed in the cursor position of the DTD, as a DTD naturally doesn't have any Document Type Declaration.

### 5. Select Declare Element from the Markup menu

### 6. Type SENDER as the name of the element, type in (#PCDATA) as content model, and click the Insert button

The #PCDATA represents character data. Character data is regular text without any markup. Thus, the content model for SENDER states that it can only contain plain text and no markup.

7. Repeat to declare the RECEIVER and SUBJECT elements to have (#PCDATA) as content model
8. Declare an element P with ANY as content model

The content model ANY means that the element can contain any element or data.

The DTD should now be as follows:

```
<?XML version="1.0"?>
<!ELEMENT MEMO (SUBJECT, SENDER, RECEIVER, P*)>
<!ELEMENT SUBJECT (#PCDATA)>
<!ELEMENT SENDER (#PCDATA)>
<!ELEMENT RECEIVER (#PCDATA)>
<!ELEMENT P ANY>
```

To modify any of the declarations, place the cursor in the markup and select Inspect from the Markup dialog.

9. Save the Document Type Definition

Saving the DTD will make XPublish aware of the changes. The name of the document type should now be listed in the Document Types submenu of the markup menu.

## USING THE MEMO DOCUMENT TYPE

You are now ready to make use of the new document type.

10. Create a new XML location called “memo1”

Use the New Location dialog of the File menu.

11. Select “memo” from the Document Type submenu of the Markup menu

The document is now declared to be of the simple document type defined previously in this tutorial. Answer yes if you are asked to confirm the change of document type. The prolog of the document now looks as follows:

```
<?XML version="1.0"?>
<!DOCTYPE name SYSTEM "memo">
```

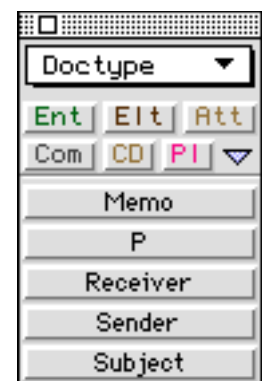
12. Open the Tag Catalog from the Palettes of the Windows menu (if not already shown)

The tag catalog can also be opened using the <> button in the upper right corner of the editor.

13. Click the small triangle in the Tag Catalog to disclose the element buttons for the document

The Tag Catalog lists the same elements as you included in the document type definition.

14. Place the cursor on the first line after the Document Type Declaration. Click the MEMO button of the Tag Catalog to insert the root element. Use the other buttons to build the following document:



```
<?XML version="1.0"?>
<!DOCTYPE MEMO SYSTEM "memo">
<MEMO>
  <SUBJECT>Defining Document Types</SUBJECT>
  <SENDER>Terje Norderhaug</SENDER>
  <RECEIVER>Web Publishers</RECEIVER>
  <P>See, defining document types wasn't that hard!</P>
  <P>You are now ready to define some on your own.</P>
</MEMO>
```

## DEFINING CONVERSION AND PRESENTATION

You are now ready to define how memos should be converted into HTML. This is specified in the Style Sheet of the document type, as described in **Lesson 8: Preparing HTML Conversion**. Here are the steps to define how a memo should be formatted as HTML, assuming that you already are familiar with defining conversion rules in the style sheet:

### 15. Open the Style Sheet editor from the Windows menu

### 16. Select the MEMO element, set its Display property of the Classification group to Block and its Basis property to "HTML"

This specifies that the memo element should be converted into an HTML element.

### 17. Select the SUBJECT element, and set its Display property to Block and Basis property to H1

The subject of the memo will be presented as a top level header.

### 18. Select the SENDER element, and set its Display property to Block and Basis property to P

The sender of the memo will be presented as a paragraph. You may also consider using the style sheet to design a suited appearance.

### 19. Select the RECEIVER element, and set its Display property to Block and Basis property to P

The receiver of the memo will also be presented as a paragraph. As with the sender, you may consider to design an appearance for the element by setting various properties of the style sheet.

### 20. Publish the site as described in Lesson 1: Publishing the Site

The Memo document will be converted into HTML.