

# Manual de Muse

---

un entorno de autoría y publicación  
para GNU Emacs y XEmacs

---

Este es el manual de Emacs Muse versión 3.20.

La traducción está en fase beta.

Copyright © 2004, 2005, 2006, 2007, 2008, 2009, 2010 Free Software Foundation, Inc.

Copyright © de la traducción, 2009, 2010 Free Software Foundation, Inc. La traducción está realizada por el Departamento de Traducciones técnicas de Urban Planet, de la empresa Timijiraque, S.L., en su proyecto de Emacs en español <http://gnu.manticore.es>. La empresa cede el copyright a la FSF.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover texts being “A GNU Manual”, and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled “GNU Free Documentation License” in this manual.

(a) The FSF’s Back-Cover Text is: “You have freedom to copy and modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development.”

This document is part of a collection distributed under the GNU Free Documentation License. If you want to distribute this document separately from the collection, you can do so by adding a copy of the license to the document, as described in section 6 of the license.

Todo el código Emacs Lisp contenido en este documento puede usarse, distribuirse y modificarse sin restricciones. — [NdT: Por indicaciones precisas de la FSF, se deja la nota anterior en inglés, así como el texto oficial de la Licencia. La traducción se acoge también a los mismos términos.]

# Índice General

<b>1</b>	<b>Acerca de la documentación</b> .....	<b>1</b>
<b>2</b>	<b>¿Qué es Muse?</b> .....	<b>2</b>
<b>3</b>	<b>Cómo obtener Muse y los cambios en el desarrollo</b> .....	<b>3</b>
3.1	Versiones publicadas de Muse .....	3
3.2	Últimos cambios en el desarrollo, no publicados .....	3
<b>4</b>	<b>Compilación e instalación de Muse</b> .....	<b>6</b>
<b>5</b>	<b>Comienzo</b> .....	<b>7</b>
5.1	Cómo cargar Muse .....	7
5.2	Cómo editar ficheros en Muse .....	7
5.3	Publicación de un solo fichero o proyecto .....	8
5.4	Empleo de extensiones de fichero distintas .....	8
<b>6</b>	<b>Creación y gestión de proyectos Muse</b> .....	<b>9</b>
6.1	Ejemplo de proyecto en solitario .....	9
6.2	Ejemplo de múltiples proyectos .....	9
6.3	Publicación de sub-directorios de proyectos .....	10
6.4	Listado de opciones disponibles para los proyectos .....	10
<b>7</b>	<b>Teclas usadas en el modo Muse</b> .....	<b>13</b>
<b>8</b>	<b>Reglas de uso del marcado</b> .....	<b>15</b>
8.1	Párrafos: centrado y citas .....	15
8.2	Niveles de las cabeceras .....	16
8.3	Directivas al comienzo de los documentos .....	16
8.4	Texto en negrita, cursivas y subrayadas .....	16
8.5	Notas a pie de página .....	17
8.6	Indicación de estrofas poéticas .....	17
8.7	Listas de elementos .....	17
8.8	Generación de tablas de datos .....	19
8.9	Hiperenlaces y direcciones de correo con descripciones .....	20
8.10	Enlaces a URL desnudas, NombresWiki e InterWiki .....	20
8.11	Publicación y visualización de imágenes .....	21
8.12	Inserción de líneas horizontales o anclas .....	22
8.13	Evaluación de código Emacs Lisp en los documentos, para la extensibilidad .....	22

8.14	Soporte para citar otros recursos .....	22
8.15	Líneas omitidas en la salida publicada.....	23
8.16	Etiquetas reconocidas por Muse .....	24
<b>9</b>	<b>Publicación de diversos tipos de documentos</b> .....	<b>28</b>
9.1	Integración de Muse con pyblosxom.cgi.....	28
9.1.1	Otras herramientas necesarias para el estilo Blosxom.....	28
9.1.2	Formato de las entradas Blosxom y automoción.....	29
9.1.3	Estilos y opciones proporcionados para Blosxom.....	30
9.2	Publicación de entradas en una compilación .....	30
9.3	Publicación de documentos ConTeXt .....	31
9.4	Publicación en formato DocBook XML.....	34
9.5	Publicación en formato HTML o XHTML.....	35
9.6	Integración con ikiwiki.....	37
9.7	Mantenimiento de un diario o bitácora (o blog) .....	38
9.8	Publicación de documentos LaTeX .....	41
9.9	Publicación de poemas en LaTeX o PDF.....	44
9.10	Publicación de entradas en formato Texinfo o PDF.....	46
9.11	Publicación de entradas en XML.....	47
<b>10</b>	<b>Creación de estilos de publicación propios</b> .....	<b>49</b>
10.1	Especificación de funciones para marcar texto .....	49
10.2	Reglas de marcado para la publicación .....	49
10.3	Cadenas específicas de un estilo de publicación .....	51
10.4	Especificaciones de etiquetas para marcados especiales.....	56
10.5	Parámetros empleados para la definición de estilos.....	57
10.6	Derivación de estilos nuevos a partir de otro existente .....	58
<b>11</b>	<b>Complementos misceláneos, por ejemplo un modo menor</b> .....	<b>60</b>
11.1	Edición fácil de listas en otros modos mayores .....	60
<b>12</b>	<b>Obtención de ayuda y reporte de fallos ....</b>	<b>62</b>
<b>13</b>	<b>Historia de este documento .....</b>	<b>64</b>
<b>14</b>	<b>Contribuidores a esta documentación .....</b>	<b>65</b>
<b>Apéndice A</b>	<b>Licencia de Documentación Libre de GNU</b> .....	<b>66</b>
	ADDENDUM: How to use this License for your documents .....	72
<b>Índice</b>	.....	<b>73</b>

# 1 Acerca de la documentación

Este documento describe Muse, que fue escrito originalmente por John Wiegley y ahora lo mantiene Alex Ott. Está disponible en línea en varios formatos.

- <http://www.gnu.org/software/emacs-muse> Página web oficial del paquete.
- <http://gnu.manticore.es/manual-muse-emacs> Página web dedicada completamente a Emacs en español, responsable de la traducción de este manual y en donde se mantienen las últimas correcciones.

## 2 ¿Qué es Muse?

Emacs Muse (también conocido como “Muse” o “Emacs-Muse”) es un entorno de autor y de publicación para Emacs. Simplifica el proceso de escritura de documentos y de su publicación en varios formatos de salida.

Muse consiste en dos partes principales: un modo texto mejorado para la creación de documentos y la navegación a través de proyectos Muse, y un conjunto de estilos de publicación para la generación de diferentes tipos de salida.

Lo que hace a Muse distinto de otros sistemas de publicación de texto es un entorno modular, con un núcleo bastante simple, en el cual los "estilos" se derivan unos de otros. La mayor parte de la funcionalidad general de Muse es optativa. Por ejemplo, se puede usar el publicador sin el modo mayor, o el modo sin hacer ninguna publicación; o si no se cargan los módulos Texinfo o LaTeX, esos estilos no estarán disponibles.

El código base de Muse proviene de ‘`emacs-wiki.el`’ versión 2.44. El código ha sido reestructurado y reescrito, especialmente sus funciones de publicación. El foco de esta revisión está en los aspectos de autor y de publicación, y la "wikinería" ha sido removida como comportamiento predeterminado (pero está disponible en el módulo optativo ‘`muse-wiki`’). Las palabras TipoCamello no siguen siendo especiales por defecto.

Una de las pretensiones principales del desarrollo de Muse es hacer muy fácil la producción de documentos con muy buena vista y conformes a los estándares.

## 3 Cómo obtener Muse y los cambios en el desarrollo

### 3.1 Versiones publicadas de Muse

Para minimizar los riesgos es preferible instalar una versión liberada.

Los errores se corrigen primero en la versión en desarrollo. Los cambios visibles al usuario se anuncian en la lista de correo [muse-el-discuss@gn.org](mailto:muse-el-discuss@gn.org). Véase [Capítulo 12 \[Ayuda y fallos\]](#), página 62.

Los usuarios de Debian y Ubuntu pueden obtener Muse por medio de `apt-get`. El paquete `'muse-el'` está disponible tanto en el repositorio APT de Michael Olson como en los repositorios oficiales de Debian y Ubuntu. Para usar el primero se añade la siguiente línea al fichero `'/etc/apt/sources.list'` y se ejecuta `apt-get install muse`.

```
deb http://mwolson.org/debian/ ./
```

Los usuarios de Ubuntu también pueden obtener Muse por medio de `apt-get`. El paquete `'muse-el'` está disponible tanto en el repositorio APT de Michael Olson como en el oficial de Ubuntu. Para hacer uso del primero, se añade la línea siguiente al fichero `'/etc/apt/sources.list'` y se ejecuta `apt-get install muse`.

```
deb http://mwolson.org/ubuntu/ ./
```

La razón por la que construimos paquetes separados para Debian y Ubuntu es que este manual está bajo la GFDL y Debian podría no permitir su distribución en el repositorio principal. Ubuntu, por su parte, permite que este manual se incluya con el paquete `'muse-el'`.

De manera alternativa se puede descargar la última entrega en

<http://download.gna.org/muse-el/>.

### 3.2 Últimos cambios en el desarrollo, no publicados

Cuando se desea vivir en el límite probar nuevas características antes de su publicación, se elige la versión en desarrollo de Muse.

El sistema de control de versiones git permite mantenerse actualizado con respecto a los últimos cambios en la versión de desarrollo de Muse. También permite contribuir cambios (compulsándolos si se tiene acceso de desarrollador al repositorio o, si no, por medio de parches). Si se quiere contribuir al desarrollo de Muse, se recomienda encarecidamente el uso de git.

El usuario nuevo en git puede encontrar de utilidad esta guía:

<http://www.kernel.org/pub/software/scm/git/docs/tutorial.html>.

La descarga del módulo Muse con git y mantenerse actualizado implica los siguientes pasos.

1. Instalación de git.

- Debian y Ubuntu: `apt-get install git-core`.
- Windows: <http://git.or.cz/gitwiki/WindowsInstall>.
- Otros sistemas operativos: descargar, compilar e instalar las fuentes de <http://www.kernel.org/pub/software/scm/git/> o encontrar un paquete git para el sistema operativo en cuestión.

2. Descarga de la rama de desarrollo de Muse.

Si se tiene acceso de desarrollador a Muse, se hace:

```
git clone ssh://repo.or.cz/srv/git/muse-el.git muse
```

De lo contrario:

```
git clone git://repo.or.cz/muse-el.git muse
```

El usuario que se encuentre detrás de un cortafuegos restrictivo y no tenga acceso de desarrollador puede hacer esto otro:

```
git clone http://repo.or.cz/r/muse-el.git muse
```

3. Listado de los cambios principales que faltan en la copia local. Esto se hace siempre que se quiera ver si hay nuevos cambios compulsados en Muse. Si se desea, este paso se puede saltar y proceder directamente al paso de “actualización”.

```
# Cambio al directorio de fuentes en el que se está interesado.
cd muse
```

```
# Obtención de los cambios nuevos en el repositorio, pero sin
# aplicarlos aún.
git fetch origin
```

```
# Visualización de los mensajes de log de los nuevos cambios
git log HEAD..origin
```

El “origen” es el nombre que le da git a la ubicación desde donde se obtuvo Muse originalmente. Esta ubicación se puede cambiar en cualquier momento editando el fichero ‘.git/config’ en el directorio donde se pusieron las fuentes de Muse.

4. Actualización a la última versión mediante la inclusión de cualquier cambio ausente.

```
cd muse
git pull origin
```

git mostrará la cantidad de ficheros que hayan cambiado y proporcionará una muestra visual de cuántas líneas se cambiaron en cada fichero.

Hay otras vías para interactuar con el repositorio Muse.

- Navegación del repositorio git:  
<http://repo.or.cz/w/muse-el.git>
- Última instantánea del desarrollo:  
<http://mwolson.org/static/dist/muse-latest.tar.gz>
- Ídem (archivo zip):  
<http://mwolson.org/static/dist/muse-latest.zip>

La última instantánea del desarrollo puede demorarse bajo el repositorio git hasta 20 minutos, pero nunca más de eso.

## Cómo convertirse en desarrollador Muse

Quien desee compulsar cambios en el repositorio compartido de Muse, entonces hay que registrar una cuenta en <http://repo.or.cz> (y también asegurarse de añadir una clave SSH), y contactar al mantenedor actual. Sería mejor enviar primero algunos parches a



la lista de correo [muse-el-discuss@gn.org](mailto:muse-el-discuss@gn.org), tal que el mantenedor compruebe que el usuario sabe lo que está haciendo. Véase [Capítulo 12 \[Ayuda y fallos\]](#), página 62, para las instrucciones para suscribirse a la lista de correo.

También es necesario firmar la asignación de copyright para los cambios que se hagan a Muse, ya que Muse es un proyecto GNU. El mantenedor actual proporciona asistencia en esta materia si se ponen en contacto con él.

Para obtener información acerca de la compulsa de cambios en Muse y la realización de tareas de desarrollo, consúltese <http://emacswiki.org/cgi-bin/wiki/MuseDevelopment>.

## 4 Compilación e instalación de Muse

Muse puede ser compilado e instalado en la máquina local.

### Compilación

Este es un paso optativo, ya que el código fuente de Emacs Lisp no ha de ser necesariamente byte-compilado. Sin embargo, eso permitirá un incremento de velocidad.

Se necesita una copia operativa de Emacs o XEmacs para poder compilar Emacs Muse. Por omisión se usará el programa que esté instalado con el nombre `emacs`.

Si se quiere compilar usando el binario `xemacs`, se debe copiar `'Makefile.defs.default'` en `'Makefile.defs'` en el directorio de máximo nivel y después editar `'Makefile.defs'` como sigue. Se puede poner ya sea una ruta completa a un binario Emacs o XEmacs o simplemente el nombre del comando, siempre que esté en la PATH.

```
EMACS      = xemacs
SITEFLAG  = -no-site-file
# Se edita la sección según sea necesario
install_info = install-info --section "XEmacs 21.4" $(1).info \
              $(INFODIR)/dir || :
```

La ejecución de `make` en el directorio de máximo nivel debería compilar los ficheros fuente Muse en el directorio `'lisp'` y generar un fichero de auto-carga en `'lisp/muse-autoloads.el'`.

### Instalación

Muse puede instalarse en la jerarquía de ficheros haciendo lo siguiente.

Se copia `'Makefile.defs.default'` de `'Makefile.defs'` en el directorio de máximo nivel, si aún no se ha hecho. Entonces se edita el fichero `'Makefile.defs'` tal que `ELISPDIR` apunte a donde se quiera instalar los ficheros fuente y compilados de Muse e `INFODIR` indica dónde hay que poner el manual de Muse. Se puede usar una combinación de `DESTDIR` y de `PREFIX` para determinar dónde deben ser puestos los ficheros instalados. Como se mencionó antes, se querrá editar `EMACS` y `SITEFLAG` como se muestra en la sección de Compilación si se está usando XEmacs.

Si se está instalando Muse en un sistema Debian o Ubuntu, es posible que se quiera modificar el valor de `INSTALLINFO` como se especifica en `'Makefile.defs'`.

Si se desea instalar Muse en ubicaciones diferentes a las especificadas por defecto, se edita `'Makefile.defs'` adecuadamente.

Se ejecuta `make` como usuario normal, si no se ha hecho aún.

Se ejecuta `make install` como usuario root si se han elegido ubicaciones para la instalación que lo requieran.

### ELPA

Para aquellos acostumbrados a instalar paquetes de software, habrá disponible un paquete `muse` en el Emacs Lisp Package Archive (abreviado en "ELPA") a partir de la publicación de la versión 3.10 de Muse. Este paquete se compilará e instalará automáticamente en una ubicación específica del usuario. Para más información acerca de ELPA, véase <http://tromeey.com/elpa/>.

## 5 Comienzo

### 5.1 Cómo cargar Muse

Para usar Muse se añade el directorio que contenga sus ficheros a la variable `load-path`, en el fichero `‘.emacs’`. Entonces se carga el modo de autor y los estilos en los que se quiere publicar. A continuación un ejemplo.

```
(add-to-list 'load-path "<ruta a Muse>")

(require 'muse-mode)      ; carga el modo de autor

(require 'muse-html)     ; carga los estilos de publicación que uso
(require 'muse-latex)
(require 'muse-texinfo)
(require 'muse-docbook)

(require 'muse-project)  ; publica los ficheros de los proyectos
```

Una manera fácil de ver las opciones disponibles y de modificarlas es emplear la interfaz de personalización de Muse. Para hacerlo se escribe *M-x customize-group muse RET*. Cada una de las opciones tiene su propia documentación. Van agrupadas lógicamente de acuerdo con el efecto que producen.

### 5.2 Cómo editar ficheros en Muse

El modo Muse debería activarse automáticamente cuando se visite un fichero con extensión `‘.muse’`. Un fichero de ese tipo es `‘QuickStart.muse’`, que está disponible en el directorio `‘examples’` de la distribución Muse. Se puede ver que el modo Muse ha sido activado buscando el texto `‘Muse’` en la línea de modo. Si el modo Muse no se ha activado, se puede hacer escribiendo *M-x muse-mode RET*.

Se comprobará que el resaltado en los ficheros Muse es muy simple. Los enlaces están coloreados de azul, las cabeceras van en texto más grande y en negrita y las etiquetas `<example>` se colorean en gris.

Hay varias vías distintas para editar cosas como los enlaces, que ocultan el marcado subyacente de Muse. Una manera es desactivar el resaltado presionando *C-c C-l*, que también es *M-x font-lock-mode*, hacer cambios y después presionar *C-c C-l* de nuevo para conmutar de vuelta el resaltado. Otra vía es moverse al texto y editarlo. El marcado también se puede remover mediante los métodos normales de eliminación, aunque algunos efectos colaterales podrían requerir una segunda eliminación.

Para el caso particular de la edición de los enlaces, lo más fácil es moverse al enlace y hacer *C-c C-e*, que también es *M-x muse-edit-link-at-point*. Con esto se pide el enlace y su descripción, empleando el contenido previo del enlace como valores iniciales. Se puede crear un enlace a otro fichero Muse presionando *C-c TAB l*. Se puede crear un enlace a un URL presionando *C-c TAB u*. Los enlaces se pueden seguir presionando *RET* sobre ellos.

Si se quiere añadir elementos nuevos a una lista, se puede conseguir presionando *M-RET*. Esto pondrá un guión y algunos espacios en la pantalla. El guión es la marca Muse que

indica un elemento de lista. También es posible crear listas anidadas con este comando mediante el ajuste del número de espacios frente a los guiones. Si se tienen listas con líneas largas, el usuario se puede mover a un elemento de lista y presionar *M-q* para ajustarlo en varias líneas.

### 5.3 Publicación de un solo fichero o proyecto

El comando *M-x muse-project-publish-this-file* publica el documento en uso en alguno de los estilos de publicación disponibles (los estilos de publicación son formatos de salida, como HTML o Docbook), poniendo la salida generada en el directorio en curso. Si se está en modo Muse, este comando quedará asociado a *C-c C-t*. Si el fichero ha sido publicado recientemente y su contenido no ha cambiado, la ejecución de nuevo de *C-c C-t* no publicará el fichero. Para forzar la publicación en estos casos, se hace *C-u C-c C-t*.

Si se han establecido proyectos y se está visitando un fichero que es parte de un proyecto, entonces *C-c C-t* restringirá los formatos de salida a aquellos empleados por el proyecto y lo publicará automáticamente en el directorio de salida definido por el proyecto. Si se lo quiere publicar en un directorio diferente o emplear un formato distinto, entonces se usa *C-c M-C-t*, que también es *M-x muse-publish-this-file*.

Si el fichero visitado es parte de un proyecto definido en *muse-project-alist*, puede ser publicado (y el resto de los ficheros cambiados en el proyecto) empleando *C-c C-p*.

### 5.4 Empleo de extensiones de fichero distintas

Lo predeterminado es que Muse espere que todos los ficheros de proyecto tengan la extensión de fichero `‘.muse’`. Los ficheros sin esta extensión no serán asociados al modo Muse y no se considerarán parte de ningún proyecto, incluso si estuvieran dentro de un directorio de proyecto.

Si no se quiere usar `‘.muse’`, se puede personalizar la extensión estableciendo el valor de `muse-file-extension`.

Si no se quiere usar ninguna extensión y aún así se quiere que Muse autodetecte los ficheros de proyectos basándose en su ubicación, entonces se añade lo siguiente al fichero de configuración de Muse.

```
(setq muse-file-extension nil
      muse-mode-auto-p t)
```

Si se elige tener `muse-file-extension` puesta en `nil`, se pueden tener problemas si el fichero `‘.emacs’` o algún otro fichero de inicialización intenta visitar un fichero Muse. (Un ejemplo muy común de esto es si se usa Planner con Muse y se ejecuta (`plan`) desde `‘.emacs’`.) Si se quiere visitar ficheros Muse desde `‘.emacs’`, hay que asegurarse de añadir también el código adicional antes de que ocurran esas visitas:

```
(add-hook 'find-file-hooks 'muse-mode-maybe)
```

## 6 Creación y gestión de proyectos Muse

Con frecuencia se querrá publicar automáticamente todos los ficheros de un directorio en un conjunto particular de estilos de salida. Para permitirlo, Muse habilita la creación de "proyectos".

### 6.1 Ejemplo de proyecto en solitario

Este es un proyecto de muestra, que puede definirse en el fichero `.emacs`.

```
(setq muse-project-alist
  '(("Sitioweb" ("~/Páginas" :default "index")
    (:base "html" :path "~/public_html")
    (:base "pdf" :path "~/public_html/pdf"))))
```

El de arriba define un proyecto llamado "Sitioweb", cuyos ficheros están localizados en el directorio `~/Páginas`. La página predeterminada para su visita es `index`. Cuando este proyecto es publicado, cada página saldrá como HTML al directorio `~/public_html`, y como PDF al directorio `~/public_html/pdf`. Dentro de cualquier página del proyecto se pueden crear enlaces a otras páginas usando la sintaxis `[[nombre-de-página]]`.

Si se quisiera incluir sólo algunos de los ficheros de un directorio en un proyecto Muse, se puede usar una expresión regular en lugar del `~/Páginas` del ejemplo.

### 6.2 Ejemplo de múltiples proyectos

También es posible definir múltiples proyectos. Lo siguiente es un ejemplo de tres proyectos: un sitio web genérico, un área de proyectos y un planificador diario (la parte del planificador requiere Planner, véase <http://wjsullivan.net/PlannerMode.html> para obtenerlo).

```
(setq muse-project-alist
  '(("Sitio-web" ("~/Páginas" :default "index")
    (:base "html" :path "~/public_html"))
    ("Proyectos" ("~/Proyectos" :default "index")
    (:base "xhtml"
      :path "~/public_html/proyectos"
      :exclude "/TopSecret")
    (:base "pdf"
      :path "~/public_html/proyectos/pdf"
      :exclude "/TopSecret"))))
    ("Plans" ("~/Planes"
      :default "BolsaTareas"
      :major-mode planner-mode
      :visit-link planner-visit-link)
    (:base "planner-xhtml"
      :path "~/public_html/planes"))))
```

El atributo `:major-mode` especifica qué modo mayor hay que usar cuando se visiten los ficheros de este directorio.

El atributo `:visit-link` especifica la función que hay que llamar cuando se visite enlaces.

El atributo ‘`exclude`’ tiene una expresión regular que concordará con los ficheros que no han de publicarse nunca.

### 6.3 Publicación de sub-directorios de proyectos

Si se quiere publicar un directorio y todos sus sub-directorios, Muse proporciona dos funciones de conveniencia que, juntas, generan las reglas apropiadas. Véase que empleamos la comilla trasera para empezar esta definición de `muse-project-alist`, en vez del apóstrofo.

```
(setq muse-project-alist
  '(("Sitio-Web" ("~/Páginass" :default "index")
    (:base "html" :path "~/public_html"))
    ("Blog" (,@(muse-project-alist-dirs "~/Blog")
             :default "index")
     ;; Publica este directorio y sus sub-directorios. Los
     ;; argumentos son como sigue. También se necesita la parte
     ;; 'muse-project-alist-dirs' de arriba.
     ;; 1. Directorio de origen
     ;; 2. Directorio de salida
     ;; 3. Estilo de publicación
     ;; recordatorio: otras cosas a poner en cada estilo generado
     ,@(muse-project-alist-styles "~/Blog"
                                       "~/public_html/blog"
                                       "blosxom"))))
```

La función `muse-project-alist-dirs` toma un directorio y lo devuelve con todos sus sub-directorios en una lista.

La función `muse-project-alist-styles` se explica en los comentarios de arriba.

El texto “blosxom” es el nombre de otro estilo de publicación, muy parecido a “html”. Véase [Sección 9.1 \[Blosxom\], página 28](#), para mayor información acerca de él. Se puede usar cualquier estilo de publicación que se desee para el tercer argumento de `muse-project-alist-styles`.

### 6.4 Listado de opciones disponibles para los proyectos

Este es el listado de todas las opciones (con más precisión: atributos) que se pueden especificar en `muse-project-alist`.

Cada entrada en `muse-project-alist` luce así:

```
(NOMBRE-DEL-PROYECTO (ORIGEN)
  SALIDA)
```

Abajo hablamos de estos nombres.

Los *atributos*, que se componen de `SOURCES` y de `OUTPUTS`, son pares de valores. El primer valor es una palabra clave, como ‘`default`’. La segunda parte es el valor asociado con esa palabra clave, como el texto “index”. Si se está familiarizado con las listas de propiedades de Emacs Lisp, el concepto es similar a ellas, excepto que en la sección `SOURCES`, los directorios pueden entrecruzarse con los atributos de dos valores.

## Nombre del proyecto

Se trata de una cadena que indica el nombre del proyecto. Se usa principalmente para la publicación de los enlaces interwiki con el módulo `'muse-wiki.el'`.

## Origen

Esta parte de una entrada en `muse-project-alist` consiste en atributos de dos valores y también en nombres de directorios. Si se está publicando un libro, el orden de los directorios y de los atributos es significativo.

El contenido mínimo de la sección Origen es una lista de directorios.

`'book-chapter'`

Indica un capítulo nuevo de un libro. El título del capítulo va inmediatamente después de esta palabra clave.

`'book-end'`

Indica el final de un libro. Los directorios listados después se ignoran en el momento de la publicación. El valor `t` (sin las comillas) debería ir inmediatamente después de esta palabra clave.

`'book-funcall'`

Función a llamar mientras se publica un libro. Es útil para hacer algo justo después de un capítulo particular.

`'book-part'`

Indica el comienzo de una parte nueva del libro. El texto del título ha de ir inmediatamente después de esta palabra clave.

`'book-style'`

Indica un estilo de publicación particular a usar para esta parte del libro. Si se especifica, ha de ir justo después de un atributo `'part'`.

`'default'`

La página predeterminada a visitar cuando se navega por un proyecto. También, si se está usando el módulo `'muse-wiki.el'`, la publicación de un enlace a nombre de un proyecto causará que dicho enlace vaya a este fichero predeterminado.

`'force-publish'`

Esto especifica una lista de páginas que deberían publicarse cada vez que se publique un proyecto (empleando `C-c C-p`, por ejemplo), sin considerar si su contenido ha cambiado. Es útil para la actualización de las páginas Index, páginas que usan la etiqueta `<include>` y otras páginas que tienen contenido generado dinámicamente.

`'major-mode'`

Especifica el modo mayor a usar cuando se visita ficheros en este proyecto. Lo predeterminado es `muse-mode`.

`'nochapters'`

Esto indica que, mientras se publica un libro, no se crean automáticamente los capítulos. Los valores que pueden seguir son `nil` (el predeterminado, que significa que los capítulos se crearán automáticamente), o `no-nil`, que significa que especificaremos manualmente los capítulos con el atributo `'book-chapter'`.

- `‘:publish-project’`  
Indica la función que habrá de llamarse cuando se publique un proyecto.
- `‘:set’`  
Especifica una lista de variables y valores a poner cuando se publique un proyecto. La lista debería ser una lista de propiedades, con la forma:  
(VAR1 VALOR1 VAR2 VALOR2 ...)
- `‘:visit-link’`  
Especifica la función a llamar cuando se visite un enlace. La predeterminada es `muse-visit-link-default`. Los argumentos de esa función deberían ser (1) el enlace y (2) si hay que visitarlo en una ventana nueva.

## Salida

Esta parte de una entrada en `muse-project-alist` se compone de listas de atributos. Cada lista se denomina un “estilo de salida”.

El contenido mínimo para un estilo de salida es un atributo `‘:base’` y un atributo `‘:path’`.

- `‘:base’`  
Estilo de publicación a usar, como “html”, “docbook” o “pdf”.
- `‘:base-url’`  
Un URL externo que se pueda usar para acceder a los ficheros publicados. Esto se usa principalmente por el módulo `‘muse-wiki’` cuando publica enlaces entre dos proyectos separados, si los proyectos se sirven en dominios diferentes.  
También lo usa el módulo `‘muse-journal’` para crear la salida RSS o RDF.
- `‘:exclude’`  
Excluye la publicación de los elementos que coincidan con una expresión regular. La expresión regular debería empezar habitualmente con `"/`.
- `‘:include’`  
Sólo incluye los elementos que coincidan con un expresión regular cuando se produzca la publicación. La expresión regular debería comenzar habitualmente con `"/`.
- `‘:path’`  
El directorio en el que almacenar los ficheros publicados.
- `‘:timestamps’`  
Fichero que contiene las marcas de tiempo de los ficheros del proyecto. Puede usarla el módulo `‘muse-blosxom’`, pero esta opción no está en uso en ninguna parte del código Muse.



## 7 Teclas usadas en el modo Muse

Este es un resumen de las combinaciones de teclas disponibles en cada búfer Muse.

<i>C-c C-a</i> ( <i>muse-index</i> )	Muestra un índice de todas las páginas Muse conocidas.
<i>C-c C-b</i> ( <i>muse-find-backlinks</i> )	Encuentra todas las páginas que enlazan a ésta.
<i>C-c C-e</i> ( <i>muse-edit-link-at-point</i> )	Edita el enlace en el punto.
<i>C-c C-f</i> ( <i>muse-project-find-file</i> )	Abre otra página Muse. Pide el nombre.
<i>C-c C-i l</i> , <i>C-c TAB l</i> ( <i>muse-insert-relative-link-to-file</i> )	Inserta interactivamente un enlace a un fichero.
<i>C-c C-i t</i> , <i>C-c TAB t</i> ( <i>muse-insert-tag</i> )	Inserta una etiqueta interactivamente.
<i>C-c C-i u</i> , <i>C-c TAB u</i> ( <i>muse-insert-url</i> )	Inserta interactivamente un URL.
<i>C-c C-l</i> ( <i>font-lock-mode</i> )	Conmuta el resaltado del búfer en uso.
<i>C-c C-p</i> ( <i>muse-project-publish</i> )	Publica cualquier página Muse que haya cambiado.
<i>C-c C-s</i> ( <i>muse-search</i> )	Busca texto en todos los ficheros del proyecto en uso.
<i>C-c C-t</i> ( <i>muse-project-publish-this-file</i> )	Publica el fichero visitado. Pide el estilo si el fichero en cuestión puede usar más de uno.
<i>C-c C-S-t</i> , o <i>C-c C-M-t</i> ( <i>muse-publish-this-file</i> )	Publica el fichero visitado. Pide tanto el estilo como el directorio de salida.
<i>C-c C-v</i> ( <i>muse-browse-result</i> )	Muestra el resultado publicado de la página.
<i>C-c =</i> ( <i>muse-what-changed</i> )	Hace las Diff entre la página y la última copia de seguridad.
<i>TAB</i>	Mueve a la siguiente referencia Wiki.
<i>S-TAB</i>	Mueve a la referencia Wiki previa.
<i>M-TAB</i>	Completa el nombre de una página del proyecto en uso en el punto.
<i>M-RET</i>	Inserta un nuevo elemento de lista en el punto, sangrándolo apropiadamente.
<i>C-&lt;</i>	Disminuye el sangrado del elemento de lista en el punto.
<i>C-&gt;</i>	Aumenta el sangrado del elemento de lista en el punto.

*M-x muse-colors-toggle-inline-images RET*

Conmuta la visualización de las imágenes incrustadas.

*M-x muse-update-values RET*

Actualiza diversos valores que se generan automáticamente.

Se llama a esto después de cambiar `muse-project-alist`.

## 8 Reglas de uso del marcado

Un documento Muse usa reglas de marcado especiales y contextuales para determinar cómo formatear la salida del resultado. Por ejemplo, si un párrafo está sangrado, Muse asume que debe ser entrecomillado.

No hay demasiadas reglas de marcado, y todas ellas procuran ser tan simples como sea posible tal que uno se pueda centrar en la creación del documento, en lugar de en el formateado.

### 8.1 Párrafos: centrado y citas

Los párrafos en Muse deben separarse con una línea en blanco.

#### Párrafos y citas centradas

Una línea que empieza con seis o más columnas de espacios en blanco (ya sean tabulaciones o espacios) indica un párrafo centrado. Alternativamente, se puede usar la etiqueta `<center>` para envolver regiones que vayan a ser publicadas como párrafos centrados.

Pero si una línea comienza con espacios en blanco, pero menores a seis columnas, eso indica un párrafo entrecomillado. Alternativamente, se puede usar la etiqueta `<quote>` para envolver regiones que vayan a ser publicadas como párrafos entrecomillados.

#### Párrafos literales

La etiqueta `<example>` se usa para ejemplos, donde los espacios en blanco deban preservarse, el texto procesado en monospace, y todos los caracteres especiales para el estilo de la salida, escapados.

También está la etiqueta `<literal>`, que causa que un bloque marcado sea dejado tal cual. Esto puede usarse para insertar bloques de código HTML dentro de la salida HTML, por ejemplo.

Si se quiere que un texto sólo se inserte cuando se esté publicando en un estilo de publicación particular, se usa el atributo `'style'` para la etiqueta `<literal>`. Sigue un ejemplo.

```
<literal style="latex">  
Se empleó un estilo basado en LaTeX para la publicación de este  
documento.  
</literal>
```

Esto dejará la región en solitario si el estilo de publicación presente es “latex” o está basado en “latex”, como el “pdf”, y eliminará la región de lo contrario. También es posible dejar el texto en solitario sólo para un estilo en particular, en vez de sus derivaciones, añadiendo `exact="t"` a la etiqueta.

#### Ruptura de líneas

Si se necesita una ruptura de línea se usa la etiqueta `<br>`. La mayor parte de las veces esta etiqueta es innecesaria, porque Muse detectará automáticamente los párrafos por medio de las líneas en blanco. Si se quieren preservar los saltos de línea en varias línea de texto, entonces se usa en su lugar el marcado de versos (véase [Sección 8.6 \[Versos\]](#), página 17).

## 8.2 Niveles de las cabeceras

Una cabecera se convierte en capítulo o en sección en la salida impresa, dependiendo del estilo. Para indicar una cabecera se comienza un nuevo párrafo con uno o más asteriscos, seguidos por un espacio y el título de la cabecera. Entonces se comienza otro párrafo para introducir el texto de esa sección.

Se publicarán todos los niveles de cabecera. Sin embargo, la mayoría de los estilos de publicación sólo distinguen entre los primeros 4 niveles.

```
* Primer nivel

** Segundo nivel

*** Tercer nivel

**** Cuarto nivel
```

## 8.3 Directivas al comienzo de los documentos

Las directivas son líneas que comienzan con el carácter ‘#’ que vayan antes de cualquier párrafo o sección en el documento. Las directivas adoptan la forma de “#directiva contenido de la directiva”. Se puede usar cualquier combinación de mayúsculas y minúsculas para las directivas, incluso si la directiva no está en la lista de abajo.

La función `muse-publishing-directive` puede usarse en textos de cabecera y al pie para acceder a las directivas. Por ejemplo, para acceder a la directiva ‘#title’ se usa (`muse-publishing-directive "title"`).

La siguiente es una lista de directivas que usa Muse.

```
#author    El autor del documento.
            Si no se especifica, Muse intentará figurárselo desde la variable user-full-name.

#date      La fecha de la última modificación del documento.
            Esto se usa en los estilos de publicación que pueden insertar la información de fecha.

#desc      Una descripción breve del documento.
            Se usa en el estilo de publicación journal para insertar información dentro de una semilla RSS/RDF.

#title     El título del documento.
            Si no se especifica, se usará el nombre del fichero.
```

## 8.4 Texto en negrita, cursivas y subrayadas

Para enfatizar texto, se lo envuelve con ciertos caracteres especiales.

```
*destacado*
**destacado fuerte**
***destacado muy fuerte***
```

```
_subrayado_
=verbatim y monospace=
```

Mientras se edita un documento Muse en modo Muse, estas formas de énfasis serán resaltadas en una manera WYSIWYG. Cada una de estas formas pueden abarcar múltiples líneas.

El texto literal quedará en color gris por defecto. Para cambiar esto se configura `muse-verbatim-face`.

También se puede usar la etiqueta `<code>` para indicar texto literal y monospace. Esto es útil para regiones que tengan un “=” en ellas.

## 8.5 Notas a pie de página

Una referencia a una nota al pie de página es simplemente un número entre corchetes. Para definir una nota al pie, se sitúa esta definición al final del fichero. ‘`footnote-mode`’ puede usarse para facilitar grandemente la creación de estos tipos de notas al pie.

Las notas al pie quedan definidas por el mismo número entre corchetes puesto al principio de una línea. Se usa el comando del `footnote-mode` `C-c ! a`, para insertar con mucha sencillez notas al pie mientras se escribe. Se usa `C-x C-x` para regresar al punto de inserción.

## 8.6 Indicación de estrofas poéticas

La Poesía requiere que los espacios en blanco se preserven, pero sin convertirse en monospace. Para indicar esto se usa la siguiente marca, reminiscencia de los correos citados.

```
> En Emacs se hace poesía;
>   disculpen mi gran alegría.
```

También se puede usar la etiqueta `<verse>`, si se prefiere.

```
<verse>
En Emacs se hace poesía;
   disculpen mi gran alegría.
</verse>
```

Se pueden incluir múltiples estrofas en un solo conjunto de etiquetas `<verse>`, como sigue.

```
<verse>
En Emacs se hace poesía;
   disculpen mi gran alegría.

Estos versos de mi autoría,
   son una gran porquería.
</verse>
```

## 8.7 Listas de elementos

Las listas se crean usando caracteres especiales al comienzo de la línea. Debe haber espacios en blanco entre los símbolos o números de los elementos, para distinguirlos de la posibilidad de que esos caracteres también se encuentren en frases reales.

Lo siguiente se interpreta como una lista con símbolos.

Texto normal.

- entrada de elemento uno
- entrada de elemento dos

Lo siguiente como una lista numerada.

Texto normal.

1. Enumerado elemento uno
2. Enumerado elemento dos

Lo siguiente es una lista de definiciones.

Term1 ::

Esto es una primera definición  
Y tiene dos líneas;  
no, pongámosle tres.

Term2 ::

Esta es una segunda definición

## Listas anidadas

Es posible anidar listas de tipos iguales o distintos. El “nivel” de la lista queda determinado por la cantidad de espacio en blanco inicial.

Texto normal.

- Nivel 1, entrada de elemento uno
    1. Nivel 2, enumerado elemento uno
    2. Nivel 2, enumerado elemento dos
  - Nivel 1, entrada de elemento dos
    1. Nivel 2, enumerado elemento tres
    2. Nivel 2, enumerado elemento cuatro
- term :: definición

## Separación de elementos de las listas

Si se quiere separar una línea dentro de cualquier tipo de lista, basta con poner una línea en blanco entre el final de la línea previa y el comienzo de la siguiente, empleando la misma cantidad de sangría inicial.

- elemento 1, línea 1

elemento 1, línea 2

1. Enumerado línea 1

Enumerado línea 2

- elemento 2, línea 1

elemento 2, línea 2

## 8.8 Generación de tablas de datos

Sólo se admiten tablas muy simples. La sintaxis es como sigue.

Barras dobles || Separan campos de cabecera

Barras simples | Separan campos de cuerpo  
Aquí hay más | campos de cuerpo

Barras triples ||| Separan campos al pie

Algunos estilos de publicación requieren que primero se proporcionen los campos de cabecera, después los campos al pie, y después los campos del cuerpo. Se puede usar cualquier orden que se desee para estas secciones y Muse las reordenará en el momento de la publicación.

Si se desea desactivar la generación de tablas en un fichero Muse, se añade la directiva `#disable-tables t` al comienzo del fichero.

### Otros formatos de tablas

Es posible publicar tablas muy básicas al estilo de Orgtbl-mode.

```
| org | estilo | tabla |
|-----+-----+-----|
| uno |          | uno   |
| dos | dos     |      |
|     | tres    | tres  |
|-----+-----+-----|
| más | cosas  |      |
```

Si el lector está acostumbrado a la manera en que el modo Org publica estas tablas, entonces puede personalizar `muse-html-table-attributes` en lo siguiente, para obtener un tipo de salida similar.

```
border="2" cellspacing="0" cellpadding="6" rules="groups" frame="hsides"
```

También se admiten las tablas al estilo de `table.el`, en la medida en que `table.el` en sí le de soporte a la salida de tablas para un estilo de publicación en particular. En el momento de escribir esto, los estilos “html”, “latex” y “docbook” tienen soporte en `table.el`. Los estilos derivados de estos otros también funcionarán.

```
+---+-----+---+
|   | uno | 1 |
+---+-----+---+
| b | dos |   |
+---+-----+---+
| c |     | 3 |
+---+-----+---+
```

## 8.9 Hiperenlaces y direcciones de correo con descripciones

Un hiperenlace puede referenciar un URL u otra página dentro de un proyecto Muse. Además se puede especificar texto descriptivo, que puede ser mostrado en lugar del texto del enlace en los estilos de salida que admiten descripciones de enlace. La sintaxis es como sigue.

```
[[destino del enlace][descripción del enlace]]
[[destino del enlace sin descripción]]
```

Así, la página web de Muse en español se puede encontrar

‘[[<http://gnu.manticore.es/manual-muse-emacs>][aquí]]’,

o en ‘[[<http://gnu.manticore.es/manual-muse-emacs>]]’.

## 8.10 Enlaces a URL desnudas, NombresWiki e InterWiki

Una dirección URL o de correo electrónico localizada en el texto de entrada se publica como hiperenlace. Este tipo de enlaces se denominan *enlaces implícitos* porque no quedan separados del resto del documento Muse.

Algunos caracteres en los URL pueden prevenir que Muse los reconozca como enlaces implícitos. Si se quiere enlazar a un URL que contiene espacios o cualquiera de los caracteres “[, ”“( )<>^”, habrá que convertir el enlace en explícito. Los caracteres de puntuación “.,;:” tampoco se reconocen como parte de un URL cuando aparecen al final. Para información acerca de cómo hacer un enlace explícito, véase [Sección 8.9 \[Hiperenlaces y direcciones de correo electrónico con descripciones\]](#), página 20.

Si se carga el módulo `muse-wiki`, quedará disponible otra forma de enlace implícito. Los NombresWiki, que se escriben en LetraCamello, son resaltados y publicados como enlaces, dado que exista el fichero al que remiten.

Se puede personalizar el reconocimiento de los NombresWiki editando la opción `muse-wiki-wikiword-regexp` y ejecutando seguidamente `(muse-configure-highlighting 'muse-colors-markupmuse-colors-markup)`. Si se usa la interfaz de Customize, lo último se hará automáticamente.

El módulo `muse-wiki` también permite los enlaces InterWiki. Estos son similares a las PalabrasWiki, pero en este caso especifican tanto el proyecto como la página de un fichero. Se usará el nombre de las entradas de proyecto en `muse-project-alist` como nombre InterWiki predeterminado. Siguen varios ejemplos.

```
Blog::DocumentarMuse
Proyectos#EmacsMuse
Sitioweb
```

En el primer caso, el delimitador interwiki es ‘::’, ‘Blog’ es el nombre del proyecto, y ‘DocumentingMuse’ es el nombre de la página. En el segundo ejemplo, ‘#’ es el delimitador interwiki. Si el nombre de un proyecto se presenta como tal en el texto, como en el tercer caso, quedará colorizado y publicado como enlace a la página predeterminada del proyecto dado.

Se puede conseguir la personalización de los enlaces interwiki editando la opción `muse-wiki-interwiki-alist`.

También es posible enlazar a un ancla en un documento interwiki. A esto se le denomina un “enlace de tres partes”. Siguen ejemplos de esto.



```
Blog::DocumentarMuse#anchor1
Proyectos#EmacsMuse#anchor2
```

## 8.11 Publicación y visualización de imágenes

### Enlaces a imágenes

Se pueden usar enlaces a imágenes tanto en el destino como en la descripción o en ambos. Así, el siguiente código se publicará como imagen pulsable que apunte a <http://gnu.manticore.es/>.

```
[[http://gnu.manticore.es/][themes/chameleon/marvin/logo.png]]
```

Normalmente, las imágenes en la parte del enlace estarán internalizadas.

Si se quiere que estas imágenes se publiquen como enlaces, se pone el texto “URL:” inmediatamente al frente del texto del enlace. Sigue un ejemplo.

```
[[URL:http://gnu.manticore.es/themes/chameleon/marvin/logo.png]]
```

### Visualización de imágenes en el modo Muse

Si se encuentra en la descripción del enlace el vínculo a una imagen disponible localmente, el modo Muse intentará mostrarla si la versión de Emacs lo permite.

Este comportamiento puede conmutarse con `C-c C-i`, o desactivarse permanentemente estableciendo la opción `muse-colors-inline-images` en `nil`.

Es posible alterar el método para localizar imágenes personalizando la opción `muse-colors-inline-image-method`. Un valor útil para esta opción es `muse-colors-use-publishing-directory`, que le dice al modo Muse que mire en el directorio donde el fichero en uso será publicado. Lo predeterminado es mirar en el directorio en uso. Las rutas relativas como `../pics/` deberían funcionar para cualquier configuración.

Se espera que Muse sea capaz de copiar imágenes desde el directorio de “origen” al directorio de publicación personalizando `muse-project-alist`, pero esto aún no se ha implementado.

### Publicación simple de imágenes

El siguiente ejemplo se mostrará y publicará correctamente si un fichero PNG llamado `TestLogo.png` existe en el directorio `../pics/`. Si hay texto en la misma línea que la imagen, también se mantendrá en la salida.

```
[[../mi-imagen.png]]
```

### Publicación de imágenes con títulos

Si se quiere añadir un pie de foto a una imagen, se usa la siguiente sintaxis. Esto centrará la imagen (si el formato de salida lo admite) y añadirá un pie centrado debajo de la foto. Los formatos que no admitan el centrado de la imagen la dejarán en el margen izquierdo.

```
[[../pics/mi-gato.png][Mi gato Derbe]]
```

Las imágenes con pies solo pueden presentarse en sus propios párrafos, sin texto en la misma línea. De lo contrario, la salida publicada no será sintácticamente correcta.

## 8.12 Inserción de líneas horizontales o anclas

### Reglas horizontales

Cuatro o más guiones indican una línea horizontal. Hay que asegurarse de poner líneas en blanco en torno suyo, o será considerada parte del párrafo precedente o siguiente.

### Anclas

Si se comienza una línea con un "#ancla" -donde "ancla" puede ser cualquier palabra que no contenga espacios- se define un ancla en ese punto documento. Este punto puede referenciarse empleando "página#ancla" como el objetivo en un enlace Muse.

## 8.13 Evaluación de código Emacs Lisp en los documentos, para la extensibilidad

Se pueden conseguir tipos arbitrarios de marcado empleando la etiqueta `<lisp>`. Con dicha etiqueta se puede generar cualquier texto de salida que se desee. La salida insertada quedará marcada si la etiqueta `<lisp>` aparece dentro del texto principal del documento.

```
<lisp>(concat "Esta forma queda " "insertada")</lisp>
```

Téngase presente que no se debe usar el comando `insert` dentro de un conjunto de etiquetas `<lisp>`, ya que el valor de retorno desde esas etiquetas `<lisp>` sería insertado automáticamente en el documento.

También es posible tratar la salida como si estuviera rodeada por las etiquetas `<example>`, `<src>` o `<verse>`, especificando "example", "src" o "verse" como el atributo 'markup' de la etiqueta `<lisp>`.

```
<lisp markup="ejemplo">
(concat "Inserta" " me")
</lisp>
```

Hay otros lenguajes que también tienen etiquetas que provocan que su código fuente sea evaluado. Véase [Sección 8.16 \[Sumario de etiquetas\], página 24](#), para los detalles.

## 8.14 Soporte para citar otros recursos

### Ejemplo

Este es un ejemplo de cómo lucen las citas en un documento Muse.

```
#bibsource REFDB
```

```
* Título
** Subtítulo
```

```
Algo de texto antes de <cite>Miller1999</cite> y tras la cita.
```

```
Esta es una cita sólo de autor <cite type="author">Miller1999</cite>.
```

```
Y esta es una cita sólo de año <cite type="year">Miller1999</cite>.
```

Finalmente, esta es una cita multi-cabecera  
`<cite>Miller1999,Andrews2005</cite>.`

## General

La directiva `#bibsource` define la fuente bibliográfica. Son posibles las siguientes fuentes.

- DocBook + RefDB: la cadena "REFDB"
- LaTeX + bibtex: el nombre de un fichero bibtex apropiado
- LaTeX + RefDB: si el fichero de entrada se llama "foo.muse", entonces pone esto en "foo.bib"

Las citas se codifican como elementos `<cite>` que encierran las claves de citado como están definidas en el fichero de bibliografía o en el banco de datos. En las citas con cabeceras múltiples, las claves de citado han de estar separadas por signos de dos puntos o por puntos y comas. Los estilos `latex` y `docbook` los traducen automáticamente en el separador apropiado.

Los elementos `<cite>` toman un atributo "type" optativo que define cómo se procesa la cita. Si el atributo no está presente, se obtendrá una cita normal de acuerdo al estilo bibliográfico, e.g. "(Miller et al., 1999)". Si el atributo está puesto como "author", sólo se procesarán los nombres de los autores. De acuerdo con esto, "year" provocará que se imprima el año. Es útil para citas como esta:

Miller et al. ya han demostrado en una publicación anterior (1999) que esto no funciona.

Recuérdese que el modo `refdb` (la interfaz de Emacs a RefDB) puede recuperar referencias con tan solo marcar la clave de citado y ejecutar el comando `refdb-getref-by-field-on-region`. Versiones posteriores de `refdb-mode` también permitirán insertar referencias como citas Muse (lo que ya está implementado para los documentos DocBook, TEI y LaTeX).

Se puede haber notado que no existe elemento para indicar la posición de la bibliografía. Esta última se crea siempre en una posición válida cerca del final del documento. Las funciones `muse-docbook-bibliography` y `muse-latex-bibliography` son llamadas en la cabecera o en el pie para generar este contenido, así que es posible cambiar la posición exacta.

## 8.15 Líneas omitidas en la salida publicada

Para indicar un comentario se usa la siguiente sintaxis. Los comentarios no se publicarán.

`;` El texto del comentario va aquí.

Es decir sólo un punto y como al principio de una línea, seguido por un espacio, causará que la línea sea tratada como comentario.

Como alternativa, se puede rodear la región con la etiqueta `<comment>`.

Si se desea publicar el comentario, pero que quede comentado empleando la sintaxis de comentario del formato de salida, entonces se pone `'muse-publish-comments-p'` en `no-nil`.

## 8.16 Etiquetas reconocidas por Muse

Muse posee varias etiquetas integradas que pueden ser útiles a la hora de la publicación. Véase [\[muse-publish-markup-tags\]](#), página 56, para ver cómo personalizar las etiquetas que Muse usa, así como para crear etiquetas propias.

En el texto de cabecera y al pie sólo hay disponible un pequeño subconjunto de estas etiquetas. La opción `muse-publish-markup-header-footer-tags` lista las etiquetas permitidas en las cabeceras y pies.

### Sintaxis

Si una etiqueta toma argumentos, lucirá así:

```
<nombre-etiqueta arg1="cadena1" arg2="cadena2">
```

Si se quiere que la etiqueta luzca como si viniera directamente de un documento XHTML, se puede hacer lo siguiente como alternativa.

```
<nombre-etiqueta arg1="cadena1" arg2="cadena2" />
```

Si una etiqueta rodea algún texto, se presentará así:

```
<nombre-etiqueta>Algo de texto</nombre-etiqueta>
```

Si una etiqueta rodea una región larga, se presentará así.

```
<nombre-etiqueta>
Algo de texto.
Algo más de texto.
</nombre-etiqueta>
```

### Listado de etiquetas

Esta es la lista completa de etiquetas que Muse acepta, incluyendo aquellas mencionadas en secciones anteriores.

- ‘<br>’      Inserta una ruptura de línea.  
Muse detecta automáticamente los párrafos durante la publicación por medio de líneas en blanco, así que esta etiqueta habitualmente es innecesaria.
- ‘<cite>’    Inserta una cita de otra fuente.  
Toma el argumento ‘`type`’, que indica el tipo de cita. Los tipos válidos son "author" y "year". Si se omite este argumento, se incluyen tanto el autor como el año en la cita.  
El uso de una bibliografía para la cita puede especificarse con la directiva ‘`#bibsource`’.  
Véase [Sección 8.14 \[Citas\]](#), página 22, para información adicional.
- ‘<class>’   Si se publica en HTML, rodea el texto dado con una etiqueta <span>. Toma un argumento llamado *name* que especifica el atributo `class` de la etiqueta <span>.  
Si se está publicando en un formato diferente, no hace nada extra al texto.
- ‘<code>’    Trata el texto rodeado por la etiqueta como si estuviera incluido entre signos de igual, esto es lo convierte en monospace.

**<command>**

Ejecuta un comando sobre la región, reemplazándola con el resultado del comando. El comando se especifica con el argumento “interp”. Si no se proporciona un valor para “interp”, se pasa la región entera a la shell.

El argumento “markup” controla cómo se marcará la sección.

Si se omite, se publica la región con las reglas normales de Muse.

Si es `nil`, no marca para nada la región, pero previene que Muse siga interpretándola.

Si es `example`, trata a la región como si estuviera rodeada por la etiqueta `<example>`.

Si es `src`, trata al texto incluido como si estuviera rodeado por la etiqueta `<src>`. También se debería especificar el atributo “lang” si se hace esto.

Si es `verse`, trata a la región como si estuviera rodeada por la etiqueta `<verse>`, para preservar las nuevas líneas.

Por lo demás, debería ser el nombre de una función a la que se invoque, con el búfer estrechado a la región.

**<comment>**

Considera la región entera como comentario. Si la opción `muse-publish-comments-p` es `nil`, elimina la región, de lo contrario la publica usando la sintaxis de comentario del estilo de publicación en uso.

**<contents>**

Publica una Índice de Contenidos. Esto se insertará ya sea en ese punto o al principio del documento, dependiendo del estilo de publicación que se esté usando. No tiene una etiqueta delimitadora.

Por omisión, sólo se incluirán 2 niveles de cabeceras en el Índice de Contenidos que se genere. Para modificar esto globalmente, se personaliza la opción `muse-publish-contents-depth`. Para cambiarlo sólo para la etiqueta del momento se usa el argumento “depth”.

**<div>**

Inserta una etiqueta `<div>` en los documentos HTML y no inserta nada especial para otros formatos de publicación no-HTML.

Si se proporciona el argumento “style”, se incluye con la etiqueta `<div>` publicada. Igualmente para el argumento “id”.

**<example>**

Publica la región en monoespacio, preservando las nuevas líneas en la región. Esto es útil para snippets de código.

**<include>**

Inserta el fichero dado en la ubicación del momento durante la publicación. El uso básico de esta etiqueta es como sigue, reemplazando “fichero\_incluido” por el nombre del fichero que se desea incluir.

```
<include file="fichero_incluido">
```

El argumento “markup” controla cómo se marca esta sección.

Si se omite, publica el texto incluido con las reglas Muse normales.

Si es "nil", no marca para nada el texto incluido.

Si es "example", trata al texto incluido como si estuviera rodeado por la etiqueta `<example>`.

Si es "src", trata al texto incluido como si estuviera rodeado por la etiqueta `<src>`. También se debería especificar el atributo "lang" si se hace esto.

Si es "verse", trata el texto incluido como si estuviera rodeado por la etiqueta `<verse>`, para preservar las nuevas líneas.

De lo contrario, debería ser el nombre de una función a la que invocar después de insertar el fichero con el búfer estrechado a la sección insertada.

`<lisp>` Evalúa las expresiones Emacs Lisp entre las etiquetas inicial y final. El resultado se insertará en el documento, tal que no se necesite llamar explícitamente a `insert`. Todas las propiedades del texto quedan removidas del texto resultante. Esta etiqueta toma el argumento "markup". Véase la descripción de `<command>` para los detalles.

`<literal>` Se asegura que el texto rodeado por esta etiqueta se publica sin ningún tipo de escape. Esto es útil para la inserción de marcas directamente en el documento publicado, cuando Muse no proporciona la funcionalidad deseada.

`<markup>` Marca el texto entre las etiquetas de inicio y de fin. El comando de marcado a usar se puede especificar con el argumento "function". Se usarán las rutinas estándares de marcado de Muse si no se proporciona ningún argumento "function".

Esto es útil para el marcado de regiones en cabeceras y pies. Un ejemplo que vendría a la cabeza es generar un índice publicado de todos los ficheros del proyecto haciendo lo siguiente.

```
<markup><lisp>(muse-index-as-string t t)</lisp></markup>
```

`<perl>` Ejecuta el intérprete del lenguaje `perl` en la región, reemplazando la región con los resultados del comando.

Esta etiqueta toma el argumento "markup". Véase la descripción de `<command>` para los detalles.

`<python>` Ejecuta el intérprete del lenguaje `python` en la región, reemplazando la región con los resultados del comando.

Esta etiqueta toma el argumento "markup". Véase la descripción de `<command>` para los detalles.

`<quote>` Publica la región como una cita. Esto será insertado ya sea en el punto donde se encuentre o al comienzo del documento, dependiendo del estilo de publicación. No tiene una etiqueta delimitadora.

`<ruby>` Ejecuta el intérprete del lenguaje `ruby` en la región, reemplazando la región con los resultados del comando.

Esta etiqueta toma el argumento "markup". Véase la descripción de `<command>` para los detalles.

- ‘<src>’ Publica la región empleando `htmlize`. El lenguaje a utilizar puede especificarse con el atributo `lang`.  
Muse buscará una función llamada *lang-mode*, donde *lang* es precisamente el valor del atributo `lang`.  
Esta etiqueta requiere `htmlize` 1.34 o posterior para que funcione. Si no se satisface este requisito o el estilo de publicación en uso no está basado en HTML, Muse publicará la región como una etiqueta `<example>`.
- ‘<verbatim>’ Se usa cuando se quiere prevenir que Muse intente interpretar alguna marca. Se rodea la marca entre `<verbatim>` y `</verbatim>`, y no será interpretada.  
Esta etiqueta se usó con frecuencia en versiones previas de Muse porque no soportaba los escapes especiales en todo el documento. Ahora sólo será necesaria para otras etiquetas, y quizás también para notas al pie.
- ‘<verse>’ Preserva las nuevas líneas en la región. En formatos como HTML, las nuevas líneas se remueven por defecto, por ello la necesidad de esta etiqueta. En otros estilos de publicación, esta etiqueta puede causar que el texto sea sangrado ligeramente de una manera agradable para la poesía y la prosa.

## 9 Publicación de diversos tipos de documentos

Una de las características principales de Muse es la habilidad de publicar un único texto de entrada en una variedad de estilos de salida diferentes. Muse también hace fácil crear nuevos estilos, o derivarlos de alguno ya existente.

### 9.1 Integración de Muse con `pyblosxom.cgi`

El estilo de publicación Blosxom publica un árbol de ficheros categorizados en un árbol espejo de historias para que sean servidas por `blosxom.cgi` o `pyblosxom.cgi`. En otras palabras, cada entrada de blog corresponde con un fichero.

#### 9.1.1 Otras herramientas necesarias para el estilo Blosxom

Se necesitará tener `pyblosxom.cgi` o `blosxom.cgi` instalado en una máquina a la que se tenga acceso de subida.

La mayor dificultad con estos dos programas es la especificación de las fechas de las entradas. Ambos se basan en la hora de modificación del fichero en vez de en los datos contenidos en las propias entradas. Se necesitará un complemento para que estos programas sean capaces de obtener la fecha correcta.

#### PyBlosxom

Hay dos maneras de conseguirlo con `pyblosxom`. La primera implica la recolección de las marcas de tiempo (como se especifique en la directiva `#date`) en un fichero y después el envío de ese fichero junto con las entradas publicadas al servidor web.

La segunda leerá cada fichero en el momento de su procesamiento y analizará la directiva `#postdate`. Muse traducirá la directiva `#date` en `#postdate` en el momento de la publicación, tal que el usuario no tenga que realizar ningún trabajo extra.

#### Cómo poner marcas de tiempo en los ficheros

Hacen falta los siguientes componentes adicionales para hacer que la fecha de las entradas de blog se muestren como algo sensible.

1. Un script para recopilar directivas de fecha desde el blog completo en un único fichero. El fichero debe asociar una entrada de blog con una fecha.
2. Un plugin para `(py)blosxom` que lea este fichero.

Estas 2 cosas las proporciona `pyblosxom.cgi` en el subdirectorio `'contrib/pyblosxom'`. `'getstamps.py'` proporciona el primer servicio, mientras que `'hardcodedates.py'` proporciona el segundo.

Aquí ponemos una lista de ejemplo de mi fichero `'timestamps'`, que mapea cada fichero hacia una fecha. Realmente pueden estar en cualquier formato, en la medida en que el script de recopilación de fecha y el plugin puedan entenderlo ambos.

```
2005-04-01-14-16 personal/paper_cranes
2005-03-21 personal/spring_break_over
2004-10-24 personal/finished_free_culture
```



## Obtención de la marca de tiempo de la entrada durante su procesamiento

Como alternativa se puede usar el complemento `metadate` para `pyblosxom`. Entre las ventajas, no hay necesidad de ejecutar un script para recoger la fecha. Entre las desventajas, cada entrada se lee dos veces en vez de sólo una cuando se procesa la página. Se pone el valor de `muse-blosxom-use-metadate` en `no-nil` para activar el añadido de la directiva `#postdate` a todos los ficheros publicados. Esto se puede hacer con:

```
M-x customize-variable RET muse-blosxom-use-metadate RET
```

Con el complemento `metadate` instalado en `pyblosxom`, se usará el conjunto de fechas de esta directiva en vez de la hora de modificación del fichero. El complemento va incluido en Muse en `'contrib/pyblosxom/metadate.py'`.

## Blosxom

También es posible usar `Blosxom`, escrito en Perl, para servir entradas de blog que hayan sido publicadas con Muse. Los pasos son los siguientes.

1. Se descarga e instala `blosxom` desde <http://blosxom.sourceforge.net/>.
2. Se instala el complemento `metadate`. Está disponible en `'contrib/blosxom/metadate_0_0_3'`.
3. Cada vez que se hace una nueva entrada de blog, se cambia al directorio de datos de `blosxom` y se ejecuta el script `'contrib/blosxom/getstamps.pl'`. Este script hace poco que está creado y puede contener algunos fallos, así que hay que usarlo con precaución.

### 9.1.2 Formato de las entradas `Blosxom` y automoción

Cada fichero `Blosxom` debe incluir `'#date yyyy-mm-dd'`, o a discreción el formato largo `'#date yyyy-mm-dd-hh-mm'`, un título (usando la directiva `#title`), además de cualquier contenido normal que se desee.

La directiva de fecha no se usa directamente por `pyblosxom.cgi` o este programa. Se necesita tener los dos elementos adicionales de la sección anterior para hacer uso de esta característica.

Hay una función llamada `muse-blosxom-new-entry` que automatiza el proceso de crear una nueva entrada de blog. Para hacer uso de ella, se hace lo siguiente.

- Se configura `muse-blosxom-base-directory` a la ubicación en la que se almacena las entradas de blog.
- Se asigna la función `muse-blosxom-new-entry` a una secuencia de teclas. Yo uso el siguiente código para asignar esta función a `C-c p l`.

```
(global-set-key "\C-cpl" 'muse-blosxom-new-entry)
```

- Se debe crear la estructura del directorio aparte de la hora bajo el directorio base. Estos directorios, que se corresponden con los nombres de categorías, pueden estar anidados.
- Cuando se introduce la secuencia de teclas, se pedirá la categoría de la entrada y su título. Después de introducir esta información, se creará un nuevo fichero que se corresponda con el título, pero en minúsculas y convirtiendo los caracteres especiales en guiones bajos. Las directivas de título y de fecha se insertarán automáticamente.

### 9.1.3 Estilos y opciones proporcionados para Bloxom

Están disponibles los siguientes estilos y opciones en el estilo de publicación Bloxom.

#### Estilos proporcionados para Bloxom

`bloxom-html`

Publica las entradas Bloxom en forma HTML.

`bloxom-xhtml`

Publica las entradas Bloxom en forma XHTML.

#### Opciones proporcionadas para Bloxom

`muse-bloxom-extension`

Extensión de fichero predeterminada para la publicación de ficheros Bloxom.

`muse-bloxom-header`

Cabecera usada para publicar ficheros Bloxom.

Esto puede ser texto o un nombre de fichero.

`muse-bloxom-footer`

Pie usado para la publicación de ficheros Bloxom.

Esto puede ser texto o un nombre de fichero.

`muse-bloxom-base-directory`

Directorio base de las entradas de blog, usado por `muse-bloxom-new-entry`.

Este es el directorio de nivel superior local donde las entradas de blog pueden encontrarse.

## 9.2 Publicación de entradas en una compilación

Este estilo de publicación se usa para la salida de “libros” en formato LaTeX o PDF.

Cada página se convertirá en un capítulo separado en el libro, a menos que se use la palabra clave de estilo ‘`:nochapters`’, en cuyo caso se ejecutarán todos juntos como si fueran un capítulo gigante.

Una manera de publicar un libro es crear un proyecto para él, añadirlo a `muse-project-alist` y usar el estilo `book-pdf` con un valor ‘`:include`’ muy específico para determinar alguna página cuyo contenido será comprobado para los valores del `#title` y de la `#date`, y cuyo nombre se usará en el fichero de salida. Entonces para publicar el libro, se visita la página mencionada y se presiona `C-c C-t` o `C-c C-p` para disparar el proceso de publicación. A continuación sigue un ejemplo de `muse-project-alist` para este método.

```
(setq muse-project-alist
  '(("MisNotas" (:nochapters t ; añade capítulos automáticamente
                :book-chapter "Ciencias de la Computación"
                "~/Notas/cs"
                :book-chapter "Matemáticas"
                "~/Notas/mates"
                :book-chapter "Emacs"
                "~/Notas/emacs"))
```

```

:book-end t ; el resto no se pondrá en el libro
"/Notas"    ; tal que podamos encontrar la página
            ; de antología de notas
"/Notas/privadas"
:force-publish ("index")
:default "index")
(:base "book-pdf"
 :include "/notas-antología[~/]*$"
 :path "~/public_html/notas")
;; otros estilos de publicación para cada directorio van aquí,
;; si se desea
)))

```

En este ejemplo habría un fichero llamado ‘~/Notas/notas-antología.muse’, que contendría justo lo que le sigue. El libro resultante se publicaría como ‘~/public\_html/notes/notas-antología.pdf’.

```
#title Mis paseos tecnológicos
```

Otra vía es llamar a la función `muse-book-publish-project` manualmente, con una entrada de proyecto personalizado. Se puede encontrar un ejemplo en el fichero de configuración de John Wiegley en ‘examples/johnw/muse-johnw.el’, en la función `muse-publish-my-books`.

## Estilos proporcionados para la compilación

### `book-latex`

Publica un libro en forma LaTeX. La cabecera y pie son diferentes a los del modo de publicación LaTeX normal.

`book-pdf` Publica un libro en forma PDF. La cabecera y pie son diferentes a los del modo de publicación PDF normal.

## Opciones proporcionadas para la compilación

### `muse-book-before-publish-hook`

Gancho a ejecutar en el búfer del libro antes de su marcado.

### `muse-book-after-publish-hook`

Gancho a ejecutar en el búfer del libro después de su marcado.

### `muse-book-latex-header`

Cabecera usada para publicar libros en LaTeX.

Puede ser texto o un nombre de fichero.

### `muse-book-latex-footer`

Pie usado para publicar libros en LaTeX.

Puede ser texto o un nombre de fichero.

## 9.3 Publicación de documentos ConTeXt

Este estilo de publicación es capaz de producir documentos ConTeXt o PDF.

Si se desea publicar documentos PDF basados en ConTeXt, habrá de instalarse este último. En Debian y Ubuntu se puede conseguir instalando el paquete “texlive”.

## Estilos proporcionados para ConTeXt

`context` Publica un documento ConTeXt.

`context-pdf`  
Publica un documento PDF, empleando una herramienta de conversión externa para documentos ConTeXt.

`context-slides`  
Produce pases a partir de un documento ConTeXt.  
Este es un ejemplo de un pase.

\* Primer pase

[[Alguna-imagen-guapa-aquí.png]]

\*\* Sub-cabecera

- Una entrada.
- Otra entrada.

\* Segundo pase

... y así sucesivamente

`context-slides-pdf`  
Publica un documento PDF compuesto de pases ConTeXt.

## Opciones proporcionadas para ConTeXt

`muse-context-extension`  
Extensión de fichero predeterminada para la publicación de ficheros ConTeXt.

`muse-context-pdf-extension`  
Extensión de fichero predeterminada para la publicación de ficheros ConTeXt en PDF.

`muse-context-pdf-program`  
El programa que se llamará para generar el contenido PDF a partir del ConTeXt.

`muse-context-pdf-cruft`  
Extensiones de los ficheros a remover después de la generación correcta de la salida en PDF.

`muse-context-header`  
Cabecera para la publicación de ficheros ConTeXt.  
Puede ser texto o un nombre de fichero.

`muse-context-footer`  
Pie usado para la publicación de ficheros ConTeXt.  
Puede ser texto o un nombre de fichero.

**muse-context-markup-regexps**

Lista de expresiones regulares de marcado para identificar regiones en una página Muse.

Para más información acerca de la estructura de esta lista, Véase [[muse-publish-markup-regexps](#)], página 49.

**muse-context-markup-functions**

Una a-lista de tipos de estilos para personalizar funciones para ese tipo de texto.

Para más información acerca de la estructura de esta lista, Véase [[muse-publish-markup-functions](#)], página 49.

**muse-context-markup-strings**

Cadenas usadas para el marcado del texto.

Cubren los tipos de marcado más básicos, cuya manipulación difiere poco entre los distintos estilos.

**muse-context-slides-header**

Cabecera para la publicación de una presentación (pase) empleando ConTeXt.

Se puede usar cualesquier de los módulos predefinidos, disponibles en el directorio `tex/context/base`, para la escritura de una directiva "module" al comienzo del fichero Muse; si no se proporciona dicha directiva, se usará el módulo `pre01`. También se puede usar un estilo propio ("miestilo", en este ejemplo) reemplazando "`\usemodule[]`" con "`\input miestilo`".

Puede ser texto o un nombre de fichero.

**muse-context-slides-markup-strings**

Cadenas usadas para el marcado de texto en los pases ConTeXt.

**muse-context-markup-specials-document**

Tabla de los caracteres que deban representarse de manera especial. Se aplican al documento entero, sin regiones ya escapadas.

**muse-context-markup-specials-example**

Tabla de los caracteres que deban representarse de manera especial. Se aplican a las regiones `<example>`.

Con la interpretación predeterminada de las regiones `<example>`, no hay necesidad especial de escapes.

**muse-context-markup-specials-literal**

Tabla de los caracteres que deban representarse de manera especial. Esto se aplica a `=monospaced text=` y a las regiones `<code>`.

**muse-context-markup-specials-url**

Tabla de los caracteres que deban representarse de manera especial. Se aplica a los URL.

**muse-context-markup-specials-image**

Tabla de los caracteres que deban representarse de manera especial. Se aplica a los nombres de ficheros de imágenes.

**muse-context-permit-contents-tag**

Si es `nil`, se ignoran las etiquetas `<contents>`. De lo contrario, inserta el índice de contenidos.

La mayor parte de las veces, es mejor tener el índice de contenidos en la primera página, a la que le siga inmediatamente una nueva página. Para hacer que esto funcione con documentos publicados tanto en HTML como en ConTeXt, necesitamos ignorar la etiqueta `<contents>`.

Si el lector no está de acuerdo con esto, entonces puede poner esta opción en `no-nil`, y hará lo que se espera.

## 9.4 Publicación en formato DocBook XML

Este estilo de publicación se usa para generar ficheros DocBook XML.

### Estilos proporcionados para DocBook XML

`docbook` Publica un fichero en formato Docbook.

### Opciones proporcionadas para DocBook XML

Este estilo de publicación emplea las mismas opciones de marcado especial de caracteres que el estilo de publicación “xml”. Véase [Sección 9.11 \[XML\], página 47](#), par los detalles.

**muse-docbook-extension**

Extensión de fichero predeterminada para la publicación de los ficheros DocBook XML.

**muse-docbook-header**

Cabecera usada para publicar ficheros DocBook XML.

Puede ser texto o un nombre de fichero.

**muse-docbook-footer**

Pie usado para publicar ficheros DocBook XML.

Puede ser texto o un nombre de fichero.

**muse-docbook-markup-regexps**

Lista de reglas de marcado para publicar una página Muse en DocBook XML.

**muse-docbook-markup-functions**

Una a-lista de tipos de estilos para personalizar funciones para ese tipo de texto.

**muse-docbook-markup-strings**

Cadenas usadas para marcar el texto.

Cubren los tipos más básicos de marcado, cuyo manipulación difiere poco entre los distintos estilos.

**muse-docbook-encoding-default**

La codificación predeterminada del búfer de Emacs para su empleo en los ficheros publicados. Esto se usará si no se encuentran caracteres especiales.

**muse-docbook-charset-default**

El conjunto de caracteres predeterminado de DocBook XML a usar si no se encuentran traducciones en `muse-docbook-encoding-map`.

## 9.5 Publicación en formato HTML o XHTML

Este estilo de publicación es capaz de producir documentos HTML y XHTML.

### Estilos proporcionados para HTML/XHTML

`html` Admite la publicación en HTML 4.0 y HTML 4.01, estricto o transitorio.

`xhtml` Admite la publicación en XHTML 1.0 y XHTML 1.1, estricto o transitorio.

### Opciones proporcionadas para HTML/XHTML

Si una opción HTML no tiene una opción XHTML correspondiente, será empleada para ambos estilos de publicación.

Estos estilos de publicación usan las mismas opciones para el marcado de los caracteres especiales que el estilo de publicación “xml”. Véase [Sección 9.11 \[XML\]](#), página 47, para los detalles.

#### `muse-html-extension`

Extensión de fichero predeterminada para la publicación de ficheros HTML.

#### `muse-xhtml-extension`

Extensión de fichero predeterminada para la publicación de ficheros XHTML.

#### `muse-html-style-sheet`

Aquí se almacenan las definiciones de hojas de estilos.

Se usa en `muse-html-header`. Aquí se puede poner CSS puro o una etiqueta `<link>` a una hoja de estilos externa. Este texto puede contener etiquetas de marcado `<lisp>`.

Si se está publicando en XHTML, entonces se personaliza la opción `muse-xhtml-style-sheet`.

#### `muse-xhtml-style-sheet`

Aquí se almacenan las definiciones de la hoja de estilos.

Se usa en `muse-xhtml-header`. Se puede poner CSS en bruto o una etiqueta `<link>` a una hoja de estilos externa. Este texto puede contener etiquetas de marcado `<lisp>`.

#### `muse-html-header`

Cabecera para la publicación de ficheros HTML.

Puede ser texto o un nombre de fichero.

#### `muse-html-footer`

Pie usado para publicar ficheros HTML.

Puede ser texto o un nombre de fichero.

#### `muse-xhtml-header`

Cabecera usada para la publicación de ficheros XHTML.

Puede ser texto o un nombre de fichero.

#### `muse-xhtml-footer`

Pie usado para la publicación de ficheros XHTML.

Puede ser texto o un nombre de fichero.

**muse-html-anchor-on-word**

Cuando es cierta, las anclas rodean la palabra más cercana.

Esto permite seleccionarlas en un navegador (i.e. para su pegado), pero tiene el efecto lateral de marcar las cabeceras en colores múltiples si el estilo de la cabecera es distinto al estilo del enlace.

**muse-html-table-attributes**

Atributo a usar con las etiquetas HTML `<table>`.

Si se quiere hacer tablas más complicadas en HTML, se rodea el HTML con la etiqueta `literal`, tal que que no se escapen.

**muse-html-markup-regexps**

Lista de reglas de marcado para la publicación de páginas Muse en HTML.

**muse-html-markup-functions**

Una a-lista de tipos de estilos para personalizar funciones para ese tipo de texto.

**muse-html-markup-strings**

Cadenas usadas para marcar el texto como HTML.

Cubren los tipos de marcados más básicos, cuya manipulación difiere poco entre los distintos estilos.

**muse-xhtml-markup-strings**

Cadenas usadas para marcar el texto como XHTML.

Cubren los tipos de marcados más básicos, cuya manipulación difiere poco entre los distintos estilos.

**muse-html-markup-tags**

Lista de especificaciones de etiquetas, para el marcado especial en HTML. Véase [\[muse-publish-markup-tags\]](#), página 56, para más información.

**muse-html-meta-http-equiv**

El atributo `http-equiv` usado para la etiqueta HTML `<meta>`.

**muse-html-meta-content-type**

El tipo de contenido usado para la etiqueta HTML `<meta>`.

Si uno se está esforzando en mantener la conformidad con XHTML 1.1, podría querer cambiar esto a “`application/xhtml+xml`”.

**muse-html-meta-content-encoding**

El conjunto de caracteres a añadir a la etiqueta HTML `<meta>`.

Si se pone en el símbolo `'detect`, se usa `muse-xml-encoding-map` para probar y determinar el conjunto de caracteres HTML a partir de la codificación de Emacs. Si se establece en una cadena, esta cadena se usará para forzar un conjunto de caracteres particular.

**muse-html-charset-default**

El conjunto de caracteres meta predeterminado de HTML a usar si no se encuentra ninguna traducción en `muse-xml-encoding-map`.

**muse-html-encoding-default**

La codificación predeterminada del búfer de Emacs para su empleo en los ficheros publicados. Se usará si no se encuentran caracteres especiales.



## 9.6 Integración con ikiwiki

Ikiwiki es un compilador wiki (<http://ikiwiki.info/>). Emacs Muse puede usarse como formato fuente de páginas Ikiwiki con el complemento ‘IkiWiki::Plugin::muse’.

El fichero ‘lisp/muse-ikiwiki.el’ proporciona funciones y estilos de publicación para Ikiwiki. El complemento para Ikiwiki que reconoce los ficheros Muse se proporciona en el fichero ‘contrib/ikiwiki/IkiWiki/Plugin/muse.pm’. Hay disponibles dos ficheros de inicialización en el directorio ‘examples/ikiwiki’. Se configura el fichero ‘ikiwiki.setup’ tal que la variable `muse_init` tenga la ubicación de fichero init de Muse.

Si se está usando, el directorio ‘contrib/ikiwiki/IkiWiki’ debe copiarse en el mismo directorio que el script CGI que genera Ikiwiki. Cuando se publica la wiki, la variable de entorno `PERL5LIB` debe contener la ruta al directorio ‘contrib/ikiwiki/IkiWiki’.

### Estilos proporcionados para Ikiwiki

`ikiwiki` Le da soporte a la publicación en XHTML que Ikiwiki pueda comprender.

### Opciones proporcionadas para Ikiwiki

#### `muse-ikiwiki-header`

Cabecera para la publicación de ficheros de salida Ikiwiki.

Puede ser texto o un nombre de fichero.

#### `muse-ikiwiki-footer`

Pie para la publicación de ficheros de salida Ikiwiki.

Puede ser texto o un nombre de fichero.

### Otras opciones importantes

#### `muse-colors-evaluate-lisp-tags`

Especifica si hay que evaluar el contenido de las etiquetas `<lisp>` en el momento de la visualización. Si es `nil`, no las evalúa. Si es `no-nil`, las evalúa.

El contenido del búfer no cambia, sólo el texto mostrado.

#### `muse-html-src-allowed-modes`

Modos donde permitimos que la etiqueta `<src>` los colorice. Si es `t`, permite que dicha etiqueta colorice cualquier modo.

Si es una lista de nombres de modos, tales como `'("html" "latex")`, y el argumento `lang` de `<src>` no está en la lista, entonces empleará en su lugar el modo fundamental.

#### `muse-publish-enable-dangerous-tags`

Si es `no-nil`, publica etiquetas como `<lisp>` y `<command>` que pueden llamar a programas externos o exponer información sensible. De lo contrario, ignora las etiquetas como estas.

Es útil poner esto en `nil` cuando el fichero que se publica proviene de un origen no confiable.

## 9.7 Mantenimiento de un diario o bitácora (o blog)

El módulo facilita la conservación y publicación de diarios. Cuando se publica en HTML, asume la forma de una bitácora o blog.

El formato de entrada para cada entrada de diario es como sigue.

```
* 20040317: Título de la entrada
```

```
texto de la entrada.
```

```
<qotd>
"El teatro es poesía que se sale del libro para hacerse humana." --
Federico García Lorca
</qotd>
```

La "qotd", o Cita del día, es completamente optativa. Cuando se genere en HTML, esta entrada se procesa como sigue.

```
<div class="entry">
  <div class="entry-qotd">
    <h3>Frase del día:</h3>
    <p>"El teatro es poesía que se sale del libro para
      hacerse humana." -- Federico García Lorca
    </p>
  </div>
  <div class="entry-body">
    <div class="entry-head">
      <div class="entry-date">
        <span class="date">17 de Marzo de 2004</span>
      </div>
      <div class="entry-title">
        <h2>Título de la entrada</h2>
      </div>
    </div>
    <div class="entry-text">
      <p>Texto de la entrada.</p>
    </div>
  </div>
</div>
```

La diversidad de etiquetas "div" posibilita mostrar las entradas en la forma que se desee, usando estilos CSS.

También se puede generar un fichero .RDF a partir del diario, mediante su publicación en estilo "rdf". Emplea las dos primeras frases del primer párrafo de cada entrada como su "descripción" y auto-genera las etiquetas para el enlazado a las distintas entradas.

### consideraciones respecto a muse-project-alist

Si se desea publicar una canal RDF o RSS, es importante incluir el atributo 'base-url' en la entrada muse-project-alist para los proyectos personales de Bitácora. Sigue un ejemplo.

```
(setq muse-project-alist
  '(("Diario" ("~/Diario/"
             :default "diario")
    (:base "journal-rss"
     :base-url "http://ejemplo.org/diario/"
     :path "~/public_html/diario"))))
```

## Estilos proporcionados para bitácoras

### journal-html

Publica las entradas de diario como documento HTML.

### journal-xhtml

Publica las entradas de diario como documento XHTML.

### journal-latex

Publica las entradas de diario como documento LaTeX.

### journal-pdf

Publica las entradas de diario como documento PDF.

### journal-book-latex

Publica las entradas de diario como libro LaTeX.

### journal-book-pdf

Publica las entradas de diario como libro PDF.

### journal-rdf

Publica las entradas de diario como fichero RDF (RSS 1.0).

### journal-rss

Publica las entradas de diario como fichero RSS (RSS 2.0).

### journal-rss-entry

Usado internamente por `journal-rss` y `journal-rdf` para la publicación de entradas individuales.

## Opciones proporcionadas para bitácoras

### muse-journal-heading-regexp

Expresión regular que concuerde con la cabecera del diario.

El grupo 1 es la fecha ISO; el grupo 2 es la categoría optativa y el grupo 3 es la cabecera optativa para la entrada.

### muse-journal-date-format

Formato de fecha para las entradas del diario.

### muse-journal-html-heading-regexp

Expresión regular que concuerde con una cabecera de diario de un documento HTML.

El grupo 1 es la fecha ISO; el grupo 2 es la categoría optativa y el grupo 3 es la cabecera optativa para la entrada.

- muse-journal-html-entry-template**  
Plantilla para la publicación de entradas individuales de diario en HTML.  
Puede ser texto o un nombre de fichero.
- muse-journal-latex-section**  
Plantilla para la publicación de una sección LaTeX.
- muse-journal-latex-subsection**  
Plantilla para la publicación de una sub-sección LaTeX.
- muse-journal-markup-tags**  
Lista de especificaciones de etiquetas, para el marcado especial de LaTeX.  
Véase [[muse-publish-markup-tags](#)], página 56, para más información.  
Esto lo usan `journal-latex` y sus estilos relacionados, así como el estilo `journal-rss-entry`, empleado por `journal-rdf` y `journal-rss`.
- muse-journal-rdf-extension**  
Extensión predeterminada de fichero para la publicación de ficheros RDF (RSS 1.0).
- muse-journal-rdf-base-url**  
El URL base del sitio web referenciado por el fichero RDF.
- muse-journal-rdf-header**  
Cabecera para la publicación de ficheros RDF (RSS 1.0).  
Puede ser texto o un nombre de fichero.
- muse-journal-rdf-footer**  
Pie para la publicación de ficheros RDF (RSS 1.0).  
Puede ser texto o el nombre de un fichero.
- muse-journal-rdf-date-format**  
Formato de fecha para las entradas RDF.
- muse-journal-rdf-entry-template**  
Plantilla para publicar entradas individuales de diario como RDF.  
Puede ser texto o un nombre de fichero.
- muse-journal-rdf-summarize-entries**  
Si es `no-nil`, incluye sólo los sumarios en el fichero RDF, no los datos completos.  
Lo predeterminado es `nil`, porque esto molesta a algunos suscriptores.
- muse-journal-rss-heading-regexp**  
Expresión regular que concuerde con una cabecera de diario a partir de un documento HTML.  
El grupo 1 es la fecha ISO, el 2 es la categoría optativa y el 3 es cabecera optativa para la entrada.
- muse-journal-rss-extension**  
Extensión de fichero predeterminada para la publicación de ficheros RSS 2.0.
- muse-journal-rss-base-url**  
El URL base del sitio web referenciado por el fichero RSS.

**muse-journal-rss-header**

Cabecera para la publicación de ficheros RSS 2.0.

Puede ser texto o un nombre de fichero.

**muse-journal-rss-footer**

Pie para la publicación de ficheros RSS 2.0.

Puede ser texto o un nombre de fichero.

**muse-journal-rss-date-format**

Formato de fecha para las entradas RSS 2.0.

**muse-journal-rss-entry-template**

Plantilla para publicar entradas individuales de diario como RSS 2.0.

Puede ser texto o un nombre de fichero.

**muse-journal-rss-enclosure-types-alist**

Tipos de ficheros que se aceptan como envoltorios RSS.

Se trata de una a-lista que mapea extensiones de ficheros a tipos de contenidos.

Útil para podcasting.

**muse-journal-rss-summarize-entries**

Si es `no-nil`, sólo incluye los sumarios en el fichero RSS, no los datos completos.

Lo predeterminado es `nil`, porque esto molesta a algunos suscriptores.

**muse-journal-rss-markup-regexps**

Lista de reglas de marcado para la publicación de páginas de diario Muse en RSS.

Para más información acerca de la estructura de esta lista, Véase [[muse-publish-markup-regexps](#)], página 49.

**muse-journal-rss-markup-functions**

Una a-lista de tipos de estilo para personalizar funciones para ese tipo de texto.

Para más acerca de la estructura de esta lista, Véase [[muse-publish-markup-functions](#)], página 49.

## 9.8 Publicación de documentos LaTeX

Este estilo de publicación es capaz de producir documentos LaTeX y PDF.

Si se desea publicar documentos PDF, hará falta una buena instalación TeX. Para Debian y Ubuntu, se puede conseguir instalando los paquetes “tetex-bin” y “tetex-extra”. También son obligatorios los tipos de letras TeX.

Si la instalación de LaTeX tiene un fichero ‘`grffile.sty`’, que puede encontrarse en el paquete ‘`texlive-latex-recommended`’ para Debian y Ubuntu, entonces será bueno considerar su uso añadiendo lo siguiente al fichero cabecera. Esto permite que funcionen los espacios en los nombres de ficheros.

```
\usepackage{grffile}
```

## Estilos proporcionados para LaTeX

- latex** Publica documentos LaTeX.
- pdf** Publica documentos PDF, empleando una herramienta externa de conversión de documentos LaTeX.
- latexcjk** Publica documentos LaTeX con codificación CJK (China).
- pdfcjk** Publica documentos PDF en codificación CJK (China), con el empleo de una herramienta externa de conversión de documentos LaTeX.
- slides** Publica un documento LaTeX que emplea la extensión Beamer. Esto es adecuado para la producción de pases.  
Este es un ejemplo de pase.
- ```
<slide title="Primer pase">
  Todo lo que hay entre las etiquetas slide compone el pase.

  [[Una-imagen-guapa-aquí.png]]

  - Una entrada.
  - Otra entrada.
</slide>
```
- slides-pdf** Publica un documento PDF o pase, empleando la extensión Beamer.
- lecture-notes** Publica un documento LaTeX que emplea la extensión Beamer. Es adecuado para la producción de notas de lecciones.  
También puede usar la etiqueta `<slide>`.
- lecture-notes-pdf** Publica un documento PDF o notas de lecciones, empleando la extensión Beamer.

## Opciones proporcionadas para LaTeX

- muse-latex-extension** Extensión predeterminada de fichero para la publicación de ficheros LaTeX.
- muse-latex-pdf-extension** Extensión predeterminada de fichero para publicar ficheros LaTeX en PDF.
- muse-latex-pdf-browser** El programa a usar cuando se navegue un fichero PDF publicado.  
Debe ser una cadena de formato.
- muse-latex-pdf-program** El programa al que se llama para generar el contenido PDF a partir del LaTeX.
- muse-latex-pdf-cruft** Extensiones de los ficheros a remover después de generar correctamente la salida en PDF.

**muse-latex-header**

Cabecera usada para la publicación de ficheros LaTeX.  
Puede ser texto o un nombre de fichero.

**muse-latex-footer**

Pie usado para la publicación de ficheros LaTeX.  
Puede ser texto o un nombre de fichero.

**muse-latexcjk-header**

Cabecera usada para la publicación de ficheros LaTeX (CJK).  
Puede ser texto o un nombre de fichero.

**muse-latexcjk-footer**

Pie usado para la publicación de ficheros LaTeX (CJK).  
Puede ser texto o un nombre de fichero.

**muse-latex-slides-header**

Cabecera para la publicación de pases usando LaTeX.  
Puede ser texto o un nombre de fichero.  
Se debe tener instalada la extensión Beamer para LaTeX para que esto funcione.

**muse-latex-lecture-notes-header**

Cabecera para la publicación de notas de lecciones usando LaTeX.  
Puede ser texto o un nombre de fichero.  
Se debe tener instalada la extensión Beamer para LaTeX para que esto funcione.

**muse-latex-markup-regexps**

Lista de expresiones regulares de marcado para identificar regiones en una página Muse.  
Para más acerca de la estructura de esta lista, Véase [\[muse-publish-markup-regexps\]](#), página 49.

**muse-latex-markup-functions**

Una a-lista de tipos de estilos para personalizar funciones para ese tipo de texto.  
Para más acerca de la estructura de esta lista, Véase [\[muse-publish-markup-functions\]](#), página 49.

**muse-latex-markup-strings**

Cadenas usadas para marcar el texto.  
Estas cubren los tipos de marcado más básicos, cuya manipulación difiere poco entre los distintos estilos.

**muse-latex-slides-markup-tags**

Lista de especificaciones de etiquetas, para el marcado especial de pases LaTeX.

**muse-latexcjk-encoding-map**

Una a-lista que mapea los sistemas de codificación de Emacs en las codificaciones apropiadas en CJK. Se usa el nombre base del sistema de codificación (ie, sin el -unix).

**muse-latexcjk-encoding-default**

La codificación predeterminada del búfer de Emacs para su uso en los ficheros publicados.

Se empleará si no se encuentran caracteres especiales.

**muse-latex-markup-specials-document**

Tabla de los caracteres que deban ser representados especialmente. Se aplican al documento entero, sin regiones ya escapadas.

**muse-latex-markup-specials-example**

Tabla de los caracteres que deban ser representados de manera especial. Se aplican a las regiones `<example>`.

Con la interpretación predeterminada de las regiones `<example>`, no hay necesidad de escapar ninguna cosa especial.

**muse-latex-markup-specials-literal**

Tabla de los caracteres que deban ser representados de manera especial. Se aplican al `=monospaced text=` y a las regiones `<code>`.

**muse-latex-markup-specials-url**

Tabla de los caracteres que deban ser representados de manera especial. Se aplican a los URL.

**muse-latex-markup-specials-image**

Tabla de los caracteres que deban representarse de manera especial. Se aplican a los nombres de ficheros de imágenes.

**muse-latex-permit-contents-tag**

Si es `nil`, ignora las etiquetas `<contents>`. De lo contrario, inserta el índice de contenidos.

Casi siempre es mejor tener un índice de contenidos en la primera página, siguiéndole inmediatamente una página nueva. Para hacer que esto funcione con documentos publicados tanto en HTML como en LaTeX, necesitamos ignorar la etiqueta `<contents>`.

Si el usuario no está de acuerdo con esto, podrá poner esta opción en `no-nil` y hará lo que se espera.

## 9.9 Publicación de poemas en LaTeX o PDF

El módulo `muse-poem` facilita la publicación y referencia atractiva de poemas en el siguiente formato, empleando el módulo "memoir" para la publicación en LaTeX. También marcará los poemas para todos los demás estilos de salida, aunque no de forma tan bella.

Título

Cuerpo del poema

Anotaciones, historia, notas, etc.



Una vez escrito el poema en este formato, basta con publicarlo en PDF empleando el estilo `poem-pdf`. Para hacer una referencia interna a un poema que uno mismo haya escrito -por ejemplo, en una página de blog-, existe una etiqueta "poem" definida en este módulo.

```
<poem title="nombre.de.la.página.del.poema">
```

Asumamos que la plantilla de arriba se llama 'nombre.de.la.página.del.poema'; por consiguiente la etiqueta de arriba provocaría esta inclusión.

```
** Título
```

```
> Cuerpo del poema
```

John Wiegley utiliza este módulo para publicar todos los poemas de su sitio web, que están en <http://www.newartisans.com/johnw/poems.html>.

## Estilos proporcionados para los poemas

`poem-latex`

Publica los poemas en formato LaTeX.

`poem-pdf` Publica los poemas en documentos PDF.

`chapbook-latex`

Publica un libro de poesías en formato LaTeX.

`chapbook-pdf`

Publica un libro de poesías como documento PDF.

## Opciones proporcionadas para los poemas

`muse-poem-latex-header`

Cabecera usada para la publicación de poemas en LaTeX.

Puede ser texto o un nombre de fichero.

`muse-poem-latex-footer`

Pie usado para la publicación de poemas en LaTeX.

Puede ser texto o un nombre de fichero.

`muse-poem-markup-strings`

Cadenas usadas para el marcado de los poemas.

Cubren los tipos de marcado más básicos, cuya manipulación difiere poco entre los distintos estilos.

`muse-chapbook-latex-header`

Cabecera usada para la publicación de libros de poemas en formato LaTeX.

Puede ser texto o un nombre de fichero.

`muse-chapbook-latex-footer`

Pie usado para la publicación de libros de poemas en formato LaTeX.

Puede ser texto o un nombre de fichero.

`muse-poem-chapbook-strings`

Cadenas usadas para el marcado de libros de poemas.

Cubren los tipos de marcado más básicos, cuya manipulación difiere poco entre los distintos estilos.

## 9.10 Publicación de entradas en formato Texinfo o PDF

Reglas para publicar los ficheros Muse como artículos Texinfo.

### Estilos proporcionados para Texinfo

- `texi` Publica los ficheros en formato Texinfo.  
`info` Genera ficheros Info a partir de ficheros Muse.  
`info-pdf` Publica los ficheros en formato PDF.

### Opciones proporcionadas para Texinfo

- `muse-texinfo-process-natively`  
Si es no-nil emplea el módulo `texinfofmt` de Emacs para crear los ficheros Info.
- `muse-texinfo-extension`  
Extensión de fichero predeterminada para la publicación de los ficheros Texinfo.
- `muse-texinfo-info-extension`  
Extensión de fichero predeterminada para la publicación de los ficheros Info.
- `muse-texinfo-pdf-extension`  
Extensión de fichero predeterminada para la publicación de ficheros PDF.
- `muse-texinfo-header`  
Texto a anteponer en las páginas Muse para ser publicado como Texinfo.  
Puede ser texto o un nombre de fichero. Puede contener etiquetas de marcado `<lisp>`.
- `muse-texinfo-footer`  
Texto a añadir al final en las páginas Muse para ser publicado como Texinfo.  
Puede ser texto o un nombre de fichero. Puede contener etiquetas de marcado `<lisp>`.
- `muse-texinfo-markup-regexps`  
Lista de reglas de marcado para la publicación de las páginas Muse en Texinfo.  
Para más acerca de la estructura de esta lista, Véase [[muse-publish-markup-regexps](#)], página 49.
- `muse-texinfo-markup-functions`  
Una a-lista de los tipos de estilos para personalizar funciones para ese tipo de texto.  
Para más acerca de la estructura de esta lista, véase Véase [[muse-publish-markup-functions](#)], página 49.
- `muse-texinfo-markup-strings`  
Cadenas usadas para el marcado del texto.  
Cubren los tipos de marcado más básicos, cuya manipulación difiere poco entre los distintos estilos.
- `muse-texinfo-markup-specials`  
Tabla de los caracteres que deban ser representados de manera especial.

**muse-texinfo-markup-specials**

Tabla de caracteres que deban representarse de manera especial. Se aplica a los URL.

->[A esto le falta algo, pues es lo mismo que la entrada anterior en cuanto al nombre, pero aquí se habla además de URL]

## 9.11 Publicación de entradas en XML

Muse es capaz de publicar documentos XML, con la ayuda del módulo ‘`muse-xml.el`’.

Hay disponible un esquema RelaxNG como parte de la distribución Muse en el fichero ‘`etc/muse.rnc`’.

### Estilos proporcionados para XML

`xml`            Publica el fichero en formato XML.

### Opciones proporcionadas para XML

**muse-xml-encoding-map**

Una a-lista que mapea los sistemas de codificación de Emacs en los conjuntos de caracteres apropiados de XML. Se usa el nombre base del sistema de codificación (i.e. sin el `-unix`).

**muse-xml-markup-specials**

Tabla de caracteres que deban representarse especialmente en todos los formatos de marcado de tipo XML.

**muse-xml-markup-specials-url-extra**

Tabla de caracteres que deban representarse especialmente en todos los formatos de marcado de tipo XML.

Se trata de caracteres extra que quedan escapados dentro de los URL.

**muse-xml-extension**

Extensión predeterminada de fichero usada para la publicación de ficheros XML.

**muse-xml-header**

Cabecera usada para la publicación de ficheros XML.

Puede ser texto o un nombre de fichero.

**muse-xml-footer**

Pie usado para la publicación de ficheros XML.

Puede ser texto o un nombre de fichero.

**muse-xml-markup-regexps**

Lista de reglas de marcado para la publicación de páginas Muse en XML.

Para más información acerca de la estructura de esta lista, Véase [[muse-publish-markup-regexps](#)], página 49.

**muse-xml-markup-functions**

Una a-lista de tipos de estilos para personalizar funciones para ese tipo de texto.

Para más información acerca de la estructura de esta lista, Véase [[muse-publish-markup-functions](#)], página 49.

**muse-xml-markup-strings**

Cadenas a usar para el marcado del texto.

Cubren los tipos de marcado más básicos, cuyo manejo difiere poco entre los distintos estilos.

**muse-xml-encoding-default**

La codificación predeterminada del búfer de Emacs a emplear en los ficheros publicados.

Se empleará si no se encuentran caracteres especiales.

**muse-xml-charset-default**

El conjunto de caracteres predeterminado de XML a usar si no se encuentra ninguna traducción en `muse-xml-encoding-map`.

## 10 Creación de estilos de publicación propios

### 10.1 Especificación de funciones para marcar texto

#### `muse-publish-markup-functions`

Una a-lista de tipos de estilo para personalizar funciones para este tipo de texto.

Se usa por los estilos de publicación para intentar minimizar la cantidad de expresiones regulares que uno tenga que definir. ‘`muse-publish`’ proporciona reglas para los tipos más comunes de marcado.

Cada miembro de la lista es de la siguiente forma.

(SÍMBOLO FUNCIÓN)

- SÍMBOLO Describe el tipo de texto a asociar a esta regla. `muse-publish-markup-regexps` mapea las expresiones regulares a estos símbolos.
- FUNCIÓN Función a usar para marcar este tipo de regla si no se encuentra una función adecuada en la etiqueta ‘`:functions`’ del estilo en uso.

### 10.2 Reglas de marcado para la publicación

#### `muse-publish-markup-regexps`

Lista de las reglas de marcado para publicar una página con Muse.

Las reglas dadas en esta variable son las primeras que se invocan, seguidas por las especificados por el estilo en uso.

Cada miembro de la lista es una función o una lista de la siguiente forma.

(EXP-REGULAR/SÍMBOLO TXT-INICIAL-GRUPO TXT-REEMPLAZO/FUNCIÓN/SÍMBOLO)

- EXPRESIÓN REGULAR Una expresión regular o símbolo cuyo valor sea una expresión regular, que se usará en `re-search-forward`.
- TEXTO-INICIAL-GRUPO El grupo coincidente dentro de esa expresión regular que denota el comienzo de texto que ha de ser marcado.
- TEXTO-REEMPLAZO Una cadena que se pasará a `replace-match`.

Si no fuera una cadena, sino una función, será llamada para determinar cual sería el texto de reemplazo (debe devolver una cadena). Si fuera un símbolo, el valor de ese símbolo tendría que ser una cadena.

Los reemplazos se hacen en orden, una regla cada vez. La escritura de expresiones regulares puede ser una tarea truculenta. Recuerdese que nunca se ignoran las mayúsculas. `case-fold-search` siempre está asociada a `nil` cuando se procesan las reglas de marcado.

### Orden de publicación

Este es el orden predeterminado en el que se consultan las reglas de marcado. Se puede cambiar personalizando `muse-publish-markup-regexps`.

#### `espacio en blanco de cabecera y de cola`

Remueve el espacio en blanco de cabecera y de cola de un fichero.

**directiva**`'#directiva'`

Sólo se reconoce al comienzo de los ficheros.

**comentario**`' ; una línea comentada'`**etiqueta** `'<etiqueta>'`**ancla** `'#ancla'`**enlaces explícitos**

Previene que los caracteres de énfasis sean marcados en los enlaces explícitos. De hecho no los publica aquí, simplemente añade una propiedad de texto especial sin énfasis.

**palabra** Palabra delimitada con espacio en blanco, posiblemente con caracteres de énfasis.

Esta función es responsable del marcado de los caracteres de énfasis y de escapar algunos de los especiales.

**cabecera** `'** Cabecera'`

Cabeceras al estilo del modo Outline.

**puntos suspensivos finales**`'....'`

Puntos suspensivos con un punto final.

[NdT: Este concepto en sí es incorrecto en castellano, véase

<http://buscon.rae.es/dpdI/SrvltGUIBusDPD?lema=puntos%20suspensivos>

Sin embargo, habría un caso donde la RAE admite cuatro puntos, aquel donde, tras una abreviatura (que termina de por sí con un punto) se añaden puntos suspensivos.]

**puntos suspensivos**`'...'`

Puntos suspensivos.

**regla horizontal**`'____'`

Línea horizontal o separador de sección.

**sin salto de línea**`'~~'`

Previene que las líneas queden separadas antes o después de estos caracteres.

**salto de línea**`'<br>'`

Separa la línea en ese punto.

**separador de notas al pie de página**`'Footnotes:'`

Comienzo de sección de notas al pie de página.

nota al pie de página

‘[1]’

Definición o referencia a nota al pie de página. Si está al principio de la línea, es una definición.

lista

- ‘1.’
- ‘-’
- ‘term ::’

Lista numerada, lista de elementos o lista de definiciones de términos.

table-el

Tabla de estilos ‘table.el’

tabla ‘tabla | celdas’

Tablas al estilo Muse u orgtbl-mode.

texto citado

Espacios antes del comienzo del texto  
Cita en bloque.

guión doble

‘\_\_’

Guión de magnitud doble

verso ‘> texto en verso’

ancla ‘#ancla’

enlace ‘[[enlaces][explícitos]]’

url ‘http://ejemplo.com/’

correo electrónico

‘un-correo@ejemplo.com’

### 10.3 Cadenas específicas de un estilo de publicación

Las *cadenas de marcado* son cadenas empleadas para el marcado de texto para un estilo particular.

Cubren los tipos de marcado más básicos, cuya manipulación difiere poco entre los distintos estilos.

#### Cadenas de marcado disponibles

image-with-desc

Una imagen con una descripción.

Argumento 1: imagen sin extensión. Argumento 2: extensión de la imagen.  
Argumento 3: descripción.

image

Una imagen incrustada.

Argumento 1: imagen sin extensión. Argumento 2: extensión de la imagen.

**image-link**

Imagen con un enlace en torno a ella.

Argumento 1: enlace. Argumento 2: imagen sin extensión. Argumento 3: extensión de la imagen.

**anchor-ref**

Referencia a un ancla en la página en curso.

Argumento 1: nombre del ancla. Argumento 2: descripción en el caso de existir o el enlace original de lo contrario.

**url**

Un URL sin descripción.

Argumento 1: URL.

**link**

Enlace a una página Muse con una descripción.

Argumento 1: enlace. Argumento 2: descripción en caso de existir o el enlace original de lo contrario.

**link-and-anchor**

Enlace a una página Muse con ancla y una descripción.

Argumento 1: enlace. Argumento 2: nombre del ancla. Argumento 3: su descripción en el caso de existir o el enlace original de lo contrario. Argumento 4: enlace sin extensión.

**email-addr**

Enlace a una dirección de correo electrónico.

Argumento 1: dirección de correo. Argumento 2: dirección de correo.

**anchor**

Ancla.

Argumento 1: nombre del ancla.

**emdash**

Guión doble.

Argumento 1: espacio en blanco inicial. Argumento 2: terminación del espacio en blanco.

**comment-begin**

Comienzo de comentario.

**comment-end**

Final de comentario.

**rule**

Línea horizontal o espacio.

**no-break-space**

Espacio que separa dos palabras que no han de ser separadas.

**footnote** Comienzo de nota al pie de página.**footnote-end**

Final de nota al pie de página.

**footnotemark**

Marca una referencia para la nota al pie en curso.

Argumento 1: número de esta nota al pie.



<code>footnotemark-end</code>	Final de la referencia para la nota al pie en curso.
<code>footnotetext</code>	Indica el texto de la nota al pie en curso. Argumento 1: número de esta nota al pie.
<code>footnotetext-end</code>	Línea de final del texto de la nota al pie.
<code>fn-sep</code>	Texto usado para reemplazar la línea “Footnotes:”.
<code>dots</code>	3 puntos.
<code>enddots</code>	4 puntos.
<code>part</code>	Línea indicadora del comienzo de una parte. Se usa para la publicación de libros.
<code>part-end</code>	Línea indicadora del final de una parte. Se usa para la publicación de libros.
<code>chapter</code>	Comienzo de una línea indicadora de capítulo. Se usa para la publicación de libros.
<code>chapter-end</code>	Línea indicadora de final de capítulo. Se usa para la publicación de libros.
<code>section</code>	Línea indicadora de inicio de sección de nivel 1. Argumento 1: nivel de la sección; siempre 1.
<code>section-end</code>	Línea indicadora de final de sección de nivel 1. Argumento 1: nivel de la sección; siempre 1.
<code>subsection</code>	Línea indicadora de comienzo de sección de nivel 2. Argumento 1: nivel de la sección; siempre 2.
<code>subsection-end</code>	Línea indicadora de final de sección de nivel 2. Argumento 1: nivel de la sección; siempre 2.
<code>subsubsection</code>	Línea indicadora de comienzo de sección de nivel 3. Argumento 1: nivel de la sección; siempre 3.
<code>subsubsection-end</code>	Línea indicadora de final de sección de nivel 3. Argumento 1: nivel de la sección; siempre 3.
<code>section-other</code>	Línea indicadora de comienzo de sección, donde el nivel sea superior al 3. Argumento 1: nivel de la sección.

- section-other-end**  
Línea indicadora de final de sección, donde el nivel sea superior al 3.  
Argumento 1: nivel de la sección.
- begin-underline**  
Comienzo de texto subrayado.
- end-underline**  
Final de texto subrayado.
- begin-literal**  
Comienzo de texto literal. Esto incluye las etiquetas `<code>` y `=teletype text=`.
- end-literal**  
Final de texto literal. Esto incluye las etiquetas `<code>` y `=teletype text=`.
- begin-emph**  
Comienzo del primer nivel de texto enfatizado.
- end-emph** Final del primer nivel de texto enfatizado.
- begin-more-emph**  
Comienzo del segundo nivel de texto enfatizado.
- end-more-emph**  
Final del segundo nivel de texto enfatizado.
- begin-most-emph**  
Comienzo del tercer (y último) nivel de texto enfatizado.
- end-most-emph**  
Final del tercer (y último) nivel de texto enfatizado.
- begin-verse**  
Comienzo de texto de verso.
- verse-space**  
Cadena usada para cada espacio que está más sangrado que el comienzo del verso.
- begin-verse-line**  
Comienzo de una línea en verso.
- empty-verse-line**  
Final de una línea en verso.
- begin-last-stanza-line**  
Comienzo de la última línea de una estrofa de versos.
- end-last-stanza-line**  
Final de la última línea de una estrofa de versos.
- end-verse**  
Final de texto en verso.
- begin-example**  
Comienzo de una región de ejemplo. Para hacer uso de esto, hace falta una etiqueta `<example>`.

`end-example`  
Final de una región de ejemplo. Para hacer uso de esto, hace falta una etiqueta ‘`</example>`’.

`begin-center`  
Comienzo de una línea centrada.

`end-center`  
Final de una línea centrada.

`begin-quote`  
Comienzo de una región de texto citado.

`end-quote`  
Final de una región de texto citado.

`begin-quote-item`  
Comienzo de un párrafo de texto citado.

`end-quote-item`  
Final de un párrafo de texto citado.

`begin-uli`  
Comienzo de una lista sin ordenar.

`end-uli` Final de una lista sin ordenar.

`begin-uli-item`  
Comienzo de un elemento de lista sin ordenar.

`end-uli-item`  
Final de un elemento de lista sin ordenar.

`begin-oli`  
Comienzo de lista ordenada.

`end-oli` Final de lista ordenada.

`begin-oli-item`  
Comienzo de elemento de lista ordenada.

`end-oli-item`  
Final de elemento de lista ordenada.

`begin-dl` Comienzo de lista de definiciones.

`end-dl` Final de lista de definiciones.

`begin-dl-item`  
Comienzo de elemento de lista de definiciones.

`end-dl-item`  
Final de elemento de lista de definiciones.

`begin-ddt`  
Comienza un término de lista de definiciones.

`end-ddt` Finaliza un término de lista de definiciones.

<code>begin-dde</code>	Comienza una entrada de lista de definiciones.
<code>end-dde</code>	Finaliza una entrada de lista de definiciones.
<code>begin-table</code>	Comienzo de tabla.
<code>end-table</code>	Final de tabla.
<code>begin-table-group</code>	Comienza un agrupamiento de tabla.
<code>end-table-group</code>	Finaliza un agrupamiento de tabla.
<code>begin-table-row</code>	Comienzo de fila de tabla.
<code>end-table-row</code>	Final de fila de tabla.
<code>begin-table-entry</code>	Comienzo de entrada de tabla.
<code>end-table-entry</code>	Final de entrada de tabla.

## 10.4 Especificaciones de etiquetas para marcados especiales

### `muse-publish-markup-tags`

Lista de especificaciones de etiquetas, para marcado especial del texto.

Las etiquetas al estilo XML son la mejor vía para añadir marcado personalizado a Muse. Se consigue fácilmente personalizando esta lista de etiquetas de marcado.

Para cada entrada, se da el nombre de la etiqueta, si espera una etiqueta de cierre y/o un conjunto optativo de atributos, si es anidable y una función que realice la acción que se desee dentro de la región delimitada.

Las etiquetas en sí se eliminan durante la publicación, antes de que sea llamada la función. Se llama a la función con tres argumentos, el comienzo y el final de la región rodeada por las etiquetas. Si se permiten las propiedades, se le pasan como tercer argumento en la forma de una *a*-lista. El argumento *end* siempre es un marcador.

El punto siempre está al principio de la región dentro de las etiquetas, cuando se llama a la función. Allá donde el punto esté cuando la función finaliza es donde continuará el marcado de etiquetas.

Estas reglas de etiquetado se procesan de una vez al comienzo del marcado y otra vez al final, para capturar cualquier etiqueta que pudiera haberse insertado mientras tanto.

## 10.5 Parámetros empleados para la definición de estilos

Los elementos de estilo son etiquetas que definen un estilo. Se usa o bien `muse-define-style` o `muse-derive-style` (véase [Sección 10.6 \[Estilos derivados\]](#), página 58) para crear estilos nuevos.

`muse-define-style` *name* &rest *elements* [Función]

### Elementos utilizables

- ‘:suffix’ Extensión de fichero a usar para la publicación de ficheros con este estilo.
- ‘:link-suffix’  
Extensión de fichero a usar para la publicación de enlaces a ficheros Muse con este estilo.
- ‘:osuffix’  
Extensión de fichero a usar para la publicación de ficheros de segunda etapa con este estilo.  
Por ejemplo, la publicación en PDF genera primero un fichero LaTeX y después un PDF a partir del fichero LaTeX.
- ‘:regexps’  
Listado de reglas de marcado para la publicación de páginas con Muse. Véase [\[muse-publish-markup-regexps\]](#), página 49.
- ‘:functions’  
Una a-lista de tipos de estilos para personalizar funciones para ese tipo de texto. Véase [\[muse-publish-markup-functions\]](#), página 49.
- ‘:strings’  
Cadenas usadas para el marcado de texto con este estilo.  
Cubren los tipos de marcado más básicos, cuya manipulación difiere poco entre los distintos estilos.
- ‘:tags’  
Lista de especificaciones de etiquetas, usada para manejar etiquetas extra. Véase [\[muse-publish-markup-tags\]](#), página 56.
- ‘:specials’  
Tabla de los caracteres que deban ser representados de manera especial.
- ‘:before’  
Función a ejecutar en el búfer de publicación recién creado (o en la región activa) antes de que ocurra ningún trabajo de publicación.  
Se usa para establecer parámetros extra que dirijan el proceso de publicación.
- ‘:before-end’  
Función a ejecutar en el búfer de publicación (o en la región activa) inmediatamente después de aplicar todas las expresiones regulares de marcado.  
Se usa para modificar el orden de los elementos de la tabla (cabecera, pie, cuerpo) en los estilos XML-ados.
- ‘:after’  
Función a ejecutar en el búfer de publicación después de `:before-end` e inmediatamente después de haber insertado la cabecera y el pie.

Se usa para generar el índice de contenidos así como para establecer el sistema de codificación del fichero.

‘:final’ Función a ejecutar después de guardar el fichero publicado, pero mientras aún está en su búfer.

Se usa para generar la segunda etapa de los documentos, como los ficheros PDF a partir de los ficheros LaTeX recién publicados.

La función debe aceptar tres argumentos: el nombre del fichero fuente muse, el nombre del fichero recién publicado y el nombre del fichero destino de segunda etapa. Éste último es el mismo que el recién publicado si no se requieren segundas etapas en la publicación.

‘:header’ Cabecera usada para la publicación de los ficheros de este estilo.

Puede ser una variable, un texto o el nombre de un fichero. Se inserta al comienzo del fichero, después de evaluar el marcado de publicación.

‘:footer’ Pie usado para la publicación de los ficheros de este estilo.

Puede ser una variable, un texto o el nombre de un fichero. Se inserta al final del fichero, después de evaluar el marcado de publicación.

‘:style-sheet’

Hoja de estilos usada para la publicación de los ficheros de este estilo.

Puede ser una variable o un texto. Se usa en la cabecera de los estilos de publicación basados en HTML y XHTML.

‘:browser’

Función usada para navegar el resultado publicado de los ficheros de este estilo.

## 10.6 Derivación de estilos nuevos a partir de otro existente

Para crear un estilo nuevo a partir de otro existente se usa `muse-derive-style` como sigue. Esta es una buena vía para modificar algo que no guste en un estilo de publicación particular o para personalizarlo.

`muse-derive-style` *nombre-nuevo nombre-base* **&rest** *elementos* [Función]

El nombre derivado es una cadena que define el estilo nuevo, como "mi-html". El *nombre-base* debe identificar un estilo existente, tal como "html", si se ha cargado ‘`muse-html`’. Los parámetros de estilo son los mismos que aquellos que se usan para crear un estilo, excepto que sobre-escriben cualquier definición que exista en el estilo base. Sin embargo, algunas definiciones sólo se sobre-escriben parcialmente. Los parámetros siguientes admiten la sobre-escritura parcial.

Véase [Sección 10.5 \[Elementos de estilo\]](#), página 57, para la lista completa de todos los parámetros.

‘:functions’

Si no se encuentra una función de marcado en la lista de funciones del estilo derivado, se consultará a la lista de funciones del estilo base.

‘:regexps’

Se usarán todas las expresiones regulares del estilo en uso y del base.

`‘:strings’`

Si una cadena de marcado no se encuentra en la lista de cadenas del estilo derivado, se consultará a la lista de cadenas del estilo base.

# 11 Complementos misceláneos, por ejemplo un modo menor

## 11.1 Edición fácil de listas en otros modos mayores

`muse-list-edit-minor-mode` está concebido para usarse con otros modos mayores, tales como Message (para la redacción de correo) y `debian-changelog-mode` (para la edición de los ficheros changelog de Debian ).

Implementa un soporte prácticamente perfecto para la edición y ajuste de listas. Puede manejar incluso listas anidadas. Además de los elementos de listas específicos de Muse ("-", números, listas de definiciones, notas al pie de página), también puede manejar elementos que comiencen con "\*" o "+". El ajuste de los elementos de las listas se comporta de la misma manera a como se hace en Muse, sin considerar si `filladapt` también está activado, que es la razón principal para usar esta herramienta.

### Instalación

Para usarlo se añade (`require 'muse-mode`) al fichero de personalización de Emacs y se añade la función `turn-on-muse-list-edit-minor-mode` a los ganchos de cualquier modo donde se quiera activar este modo menor.

### Asociaciones de teclas

`muse-list-edit-minor-mode` emplea las siguientes asociaciones de teclas.

`M-RET` (`'muse-l-e-m-m-insert-list-item'`)

Inserta un elemento nuevo de lista en el punto, empleando el nivel de sangrado del elemento en curso de la lista.

`C-<` (`'muse-l-e-m-m-decrease-list-item-indent'`)

Aminora la sangría del elemento en curso de la lista.

`C->` (`'muse-l-e-m-m-increase-list-item-indent'`)

Incrementa la sangría del elemento en curso de la lista.

### Funciones

`muse-list-edit-minor-mode` [Función]

Es un modo menor global para la edición de ficheros con listas. Está pensado para su uso con otros modos mayores y no con el modo Muse.

De manera interactiva, sin argumento prefijo, conmuta el modo. Con el prefijo universal `arg` activa el modo. Con un `arg` cero o negativo lo desactiva.

Este modo menor proporciona las asociaciones de teclas de Muse para la edición de listas y le da soporte al ajustes apropiado de las listas.

No sólo reconoce las listas al estilo de Muse, que emplean el carácter "-" o números, sino también listas que empleen asteriscos o signos de más. Esto debería convertir a este modo menor en algo de utilidad general.

También se reconocen las definiciones de listas y las notas al pie de página.

Téngase presente que los elementos de las listas pueden omitir los espacios antepuestos, por compatibilidad con los modos que establecen `left-margin`, tal como `debian-changelog-mode`.



`turn-on-muse-list-edit-minor-mode` [Función]

Activa incondicionalmente el modo menor Muse List Edit.

`turn-off-muse-list-edit-minor-mode` [Función]

Desactiva incondicionalmente el modo menor Muse List Edit.

## 12 Obtención de ayuda y reporte de fallos

Después de haber leído esta guía, si aún se tienen preguntas acerca de Muse o si se va a informar de fallos, existen varios sitios donde es posible hacerlo.

- <http://www.emacswiki.org/cgi-bin/wiki/EmacsMuse> es la página en emacswiki.org donde cualquiera puede añadir consejos, trucos o descripciones de fallos.
- <http://www.gnu.org/software/emacs-muse> es la página web oficial del proyecto en GNU.
- Muse tiene varias listas de correo.

`'muse-el-announce'`

Lista de tráfico bajo para los anuncios relacionados con Muse.

Es posible suscribirse a esta lista de correo ([muse-el-announce@gna.org](mailto:muse-el-announce@gna.org)) empleando el formulario de

<http://mail.gna.org/listinfo/muse-el-announce/>.

Esta lista de correo también está disponible por medio de Gmane (<http://gmane.org/>). El grupo se llama `'gmane.emacs.muse.announce'`.

`'muse-el-discuss'`

Discusiones, correcciones de fallos, sugerencias, consejos y cosas similares para Muse. Esta lista de correo incluye también el contenido de `muse-el-announce`.

Es posible suscribirse a esta lista de correo ([muse-el-discuss@gna.org](mailto:muse-el-discuss@gna.org)) empleando el formulario de

<http://mail.gna.org/listinfo/muse-el-discuss/>.

Esta lista de correo también está disponible por medio de Gmane con el identificador `'gmane.emacs.muse.general'`.

`'muse-el-logs'`

Mensajes de registro para los cambios compulsados en Muse.

Es posible suscribirse a esta lista de correo ([muse-el-logs@gna.org](mailto:muse-el-logs@gna.org)) empleando el formulario de

<http://mail.gna.org/listinfo/muse-el-logs/>.

Esta lista de correo también está disponible por medio de Gmane con el identificador `'gmane.emacs.muse.scm'`.

`'muse-el-commits'`

Informes de fallos generados para Emacs Muse. Al emplear el registrador de fallos de <https://gna.org/bugs/?group=muse-el>, los informes de fallos se enviarán a esta lista automáticamente.

Es posible suscribirse a la lista de correo ([muse-el-commits@gna.org](mailto:muse-el-commits@gna.org)) empleando el formulario de

<http://mail.gna.org/listinfo/muse-el-commits/>.

Esta lista de correo también está disponible por medio de Gmane con el identificador `'gmane.emacs.muse.cvs'`.

`'muse-el-internationalization'`

Debates acerca de la traducción a muchos idiomas del sitio web de Muse y de la documentación.

Es posible suscribirse a esta lista de correo

(`muse-el-internationalization@gna.org`)

empleando el formulario de

<http://mail.gna.org/listinfo/internationalization/>.

Esta lista de correo también está disponible por medio de Gmane con el identificador `'gmane.emacs.muse.internationalization'`.

- Se puede visitar el canal IRC Freenode `'#emacs'`. Muchos de los contribuidores están por ahí con frecuencia deseosos de responder preguntas. El canal `'#muse'` también está disponible para ayuda específica en cuanto a Muse y su actual mantenedor siempre está colgado por ahí.

## 13 Historia de este documento

- 2004 John Wiegley comenzó Muse después de percatarse de que EmacsWiki tenía algunas limitaciones serias. En torno a Febrero de 2004, comenzó a hacer la "versión 3.00 APLHA de emacs-wiki", que en su momento pasó a conocerse como Muse.

La mayoría de aquellos que frecuentaban la lista de correo de emacs-wiki continuaron usando emacs-wiki, principalmente a causa de que Planner no había sido portado para Muse.

A partir de 2004-12-01, Michael Olson se convirtió en el mantenedor de Muse, tras solicitud de John Wiegley.

- 2005 Michael Olson expandió este documento y le añadió muchas secciones nuevas preparando la primera entrega de Muse (3.01).
- 2010 [A principios de 2010, después de que Olson publicara la versión 3.20 de Muse, Alex Ott se hizo cargo del mantenimiento.]

## 14 Contribuidores a esta documentación

El primer borrador de este documento se tomó del manual texinfo de emacs-wiki. Michael Olson lo adaptó para Muse y le añadió la mayor parte de su contenido.

John Sullivan hizo la mayoría del trabajo del manual texinfo de emacs-wiki.

Mientras Sacha Chua mantuvo emacs-wiki, trabajó en el manual texinfo de emacs-wiki.

# Apéndice A Licencia de Documentación Libre de GNU

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.  
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft,” which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document,” below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you.” You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque.”

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements,” “Dedications,” “Endorsements,” or “History.”) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:



- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History," Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications," Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements." Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at

your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements," provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements," and any sections Entitled "Dedications." You must delete all sections Entitled "Endorsements."

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted

document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements,” “Dedications,” or “History,” the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## **ADDENDUM: How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ‘‘GNU
Free Documentation License.’’
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being  list their titles, with the
Front-Cover Texts being  list, and with the Back-Cover Texts being
list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# Índice

## #

#author .....	16
#date .....	16
#desc .....	16
#title .....	16

## A

actualización de Muse con git .....	4
anclas .....	22
ayuda, obtención de .....	62

## B

bitácoras .....	38
blog, estilo de diario .....	38
blog, estilo un fichero por entrada .....	28
bugs, reportes de .....	62

## C

cabeceras .....	16
citas .....	15, 22
comentarios .....	23
compilación de Muse .....	6
configuración .....	7
configuración, fichero init .....	7
contribuidores .....	65
Correo electrónico, direcciones .....	20

## D

Debian, paquete de Muse para .....	3
desarrollo .....	3
desarrollo, cómo colaborar .....	4
diarios .....	38
directivas .....	16

## E

edición de ficheros Muse .....	7, 8
ejemplos .....	15
ELPA, paquete Muse para .....	6
enlaces con destino en la misma página .....	22
enlaces con imágenes .....	21
enlaces explícitos .....	20
enlaces implícitos .....	20
enlaces inter-proyectos .....	20
enlaces, en bruto .....	20
entregas, fuentes .....	3
entregas, paquete para Debian .....	3
entregas, paquete para Ubuntu .....	3
etiquetas .....	24
etiquetas, <cite> .....	22

extensiones de ficheros, especificación .....	8
-----------------------------------------------	---

## F

fallos, reportes de .....	62
---------------------------	----

## G

git, sistema de control de versiones .....	3
guiones .....	22

## H

historia de Muse .....	64
HTML, inserción de bloques en bruto .....	15
HTML, procesado de bloques en monoespacio ..	15

## I

imágenes .....	21
imágenes, insertas .....	21
imágenes, locales .....	21
imágenes, sin descripciones .....	21
imágenes, títulos .....	21
imágenes, visualización .....	21
inserción de ficheros a la hora de publicar .....	24
instalación de Muse .....	6
InterWiki, enlaces .....	20

## L

líneas, rupturas .....	15
lisp y el comando insert .....	22
lisp, integrado .....	22
listas .....	17
listas anidadas .....	18
listas enumeradas .....	18
listas, definiciones de .....	18
listas, redondeles .....	17
listas, saltos de líneas .....	18

## M

mercado .....	15
monoespacio, procesado de bloques .....	15
monoespacio, procesado de palabras .....	16
<code>muse-define-style</code> .....	57
<code>muse-derive-style</code> .....	58
<code>muse-list-edit-minor-mode</code> .....	60
<code>muse-project-alist</code> , referencia .....	10
<code>muse-xml-encoding-map</code> .....	47

**N**

NombresWiki .....	20
notas al pie de página .....	17

**P**

párrafos .....	15
párrafos centrados .....	15
párrafos citados .....	15
poesía .....	17
proyectos .....	9
proyectos, ejemplo en solitario .....	9
proyectos, múltiples .....	9
proyectos, opciones .....	10
proyectos, subdirectorios .....	10
publicación de estilos .....	28
publicación de estilos, bloxom-html .....	30
publicación de estilos, bloxom-xhtml .....	30
publicación de estilos, book-latex .....	31
publicación de estilos, book-pdf .....	31
publicación de estilos, chapbook-latex .....	45
publicación de estilos, chapbook-pdf .....	45
publicación de estilos, ConTeXt .....	32
publicación de estilos, ConTeXt-pdf .....	32
publicación de estilos, context-slides .....	32
publicación de estilos, context-slides-pdf .....	32
publicación de estilos, derivados .....	58
publicación de estilos, docbook .....	34
publicación de estilos, html .....	35
publicación de estilos, ikiwiki .....	37
publicación de estilos, info-pdf .....	46
publicación de estilos, journal-book-latex .....	39
publicación de estilos, journal-book-pdf .....	39
publicación de estilos, journal-html .....	39
publicación de estilos, journal-latex .....	39
publicación de estilos, journal-pdf .....	39
publicación de estilos, journal-rdf .....	39
publicación de estilos, journal-rss .....	39
publicación de estilos, journal-rss-entry .....	39
publicación de estilos, journal-xhtml .....	39
publicación de estilos, latex .....	42
publicación de estilos, latexcjk .....	42
publicación de estilos, lecture-notes .....	42
publicación de estilos, lecture-notes-pdf .....	42
publicación de estilos, pdf .....	42
publicación de estilos, pdfcjk .....	42
publicación de estilos, poem-latex .....	45
publicación de estilos, poem-pdf .....	45
publicación de estilos, RSS 1.0 .....	39

publicación de estilos, RSS 2.0 .....	39
publicación de estilos, slides .....	42
publicación de estilos, slides-pdf .....	42
publicación de estilos, texi .....	46
publicación de estilos, xml .....	47
publicación, cadenas de marcado .....	51
publicación, elementos de estilo .....	57
publicación, etiquetas de marcado .....	56
publicación, expresiones regulares de marcado ..	49
publicación, funciones de marcado .....	49
publicación, inclusión de marcado en cabeceras y pies .....	24
publicación, inserción de ficheros .....	24
publicación, omisión de líneas .....	23
publicación, reglas .....	49

**R**

reglas horizontales .....	22
---------------------------	----

**T**

tablas .....	19
tablas al estilo orgtbl-mode .....	19
tablas al estilo table.el .....	19
tablas simples .....	19
teclas .....	13
texto “verbatim” .....	16
texto destacado .....	16
texto en cursivas .....	16
texto literal .....	15
texto subrayado .....	16
turn-off-muse-list-edit-minor-mode .....	61
turn-on-muse-list-edit-minor-mode .....	61

**U**

Ubuntu, paquete de Muse para .....	3
URL .....	20

**V**

versos .....	17
versos, estrofas múltiples .....	17

**W**

WYSIWYG .....	17
---------------	----