Manual de Productos Plone en la Fundación Cenditel

Versión Última actualización: 14/09/2019 01:28:49 PM

Víctor Terán Herrera, Oswaldo Lopez, Leonardo J. Caballer G.

14 de septiembre de 2019

Índice general

Bienvenidos al manual de Productos Plone en la Fundación Cenditel.

Este documento contiene descripción de API, pedazos de código y enlaces a referencias sobre como instalar los sitios en Plone CMS en la Fundación Cenditel.

CAPÍTULO 1

Tabla de Contenidos

1.1 Introducción

1.1.1 Prefacio

Por definir.

1.1.2 Escribiendo y actualizando este documento

Descripción

Como escribir y enviar mejoras al Manual de Productos Plone en la Fundación Cenditel.

Obtener y compilar la documentación

El almacenamiento de este material está disponible en el servidor de Subversion «Plataforma colaborativa» de los contribuyentes a Plone. Si usted tiene una credenciales en este servidor y desea convertirse en un colaborador ejecute el siguiente comando:

```
$ svn co http://plataforma.cenditel.gob.ve/svn/plataforma/proyectosInstitucionales/
↪renasen/cenditel.documentation/buildout cenditel-docs
```

Si usted no tiene las credenciales de acceso al repositorio SVN «Plataforma colaborativa» de Plone o simplemente solo desea obtener y compilar esta documentación ejecute el siguiente comando:

```
$ svn export http://plataforma.cenditel.gob.ve/svn/plataforma/
↪proyectosInstitucionales/renasen/cenditel.documentation/buildout cenditel-docs
```

Crear entorno virtual de Python para reconstruir este proyecto:

```
# aptitude install python-setuptools subversion
# easy_install virtualenv
# exit
$ cd $HOME ; mkdir $HOME/virtualenv ; cd $HOME/virtualenv
$ virtualenv --no-site-packages --python=/usr/bin/python sphinx
$ cd -
```

Ahora puede generar la documentación de HTML, con los siguiente comandos:

```
$ source virtualenv/sphinx/bin/activate
(sphinx)$ cd cenditel-docs/
(sphinx)$ python bootstrap.py
(sphinx)$ ./bin/buildout -vN
(sphinx)$ ./bin/sphinx
```

Ahora se puede abrir `cenditel-docs/build/html/index.html` desde su navegador Web favorito.

Para obtener la documentación en PDF:

```
$ source virtualenv/sphinx/bin/activate
(sphinx)$ cd ./cenditel-docs/build
(sphinx)$ make latex
(sphinx)$ cd ./latex
(sphinx)$ make all-pdf
```

Ahora se puede abrir `cenditel-docs/sphinx/build/latex/DocumentacionEspanolPlone.pdf` con sus programas de visor de PDF favorito (Evince, Acrobat Reader, . . .)

Reglas de redacción

En primer lugar, debe aprender los fundamentos de Sphinx que es un reStructuredText extendido.

Codificación de caracteres

Su editor debe codificar el texto en **utf-8** si le gusta lo que está leyendo. Si su editor de texto favorito no reconoce esta codificación (en la actualidad, eso es bien extraño), entonces cambie de editor de texto.

Truco

Para `vi`, `emacs` y algunos otros editores de texto soportan utf-8 de forma automática al abrir un archivo de Sphinx, el lugar en primera línea de la siguiente marca (como en este archivo):

```
.. -*- coding: utf-8 -*-
```

Desplazamientos y indentaciones

El uso del carácter de tabulación en el texto fuente para las distintas desplazamientos y indentaciones está **estrictamente prohibido**. Utilice siempre espacios para este fin. Todos los editores de texto ofrecen opciones avanzadas para insertar espacios al pulsar la tecla TAB. No tiene excusa si es necesario.

Estilos de subrayado

Sphinx y ReStructuredText no imponer estilo de subrayado para diferentes niveles de secciones de un documento. Todo se deja a la discreción editores. Para mantener la coherencia nosotros adoptamos la siguiente convención:

```
=============================================
Titulo de capitulo (uno solo por cada archivo)
=============================================
...
Sección del nivel 1
===================
...
Sección del nivel 2
-------------------
...
Sección del nivel 3
...................
...
Sección del nivel 4
~~~~~~~~~~~~~~~~~~~
```

No es necesario ni deseable ir más allá del nivel 4. Cuando la generación del documento allá completado, el nivel de las secciones básicas de un archivo depende del nivel de anidamiento del archivo en la estructura general de documento. Para generar el HTML, no es un problema, pero en LaTeX limita la superposición de las secciones a 6 niveles.

Contribuciones SVN

Wow, estás contento con tu excelente trabajo. Y le gustaría compartirlo con todo el mundo. Al igual que cuando «contribuidor» de código fuente, las pruebas unitarias no deben mostrar ningún error, compruebe en primer lugar:

- Que el comando `make html` no genere ningún error o advertencia.
- Que su redacción no posea ningún error de ortografía.
- Los enlaces de hipertexto que se ha agregado o cambiado (glosario, enlaces externos explícitos, referencias a las secciones, . . .) funcionan correctamente.

Imágenes

Aparte de las capturas de pantalla - ¡Uy, lo siento - las capturas de pantalla!, las imágenes Sphinx se inserta en el documento debe ir acompañada de su versión «Fuente» en un formato público interoperables, y para que el editor pueda abrir el archivo fuente que este disponible. Las imágenes deben estar preferentemente en el formato PNG.

Además, durante cada inserción o cambio de imagen, usted **debe** verificar y ajustar si es necesario la representación PDF, a sabiendas de las limitaciones la imagen a tamaño del papel final.

Ejemplo :

```
.. gs-map.mm: imagen de mapa mental de los servicios de GenericSetup. Creado con␣
↪FreeMind

.. image:: gs-map.png
```

Aplicaciones gráficas recomendadas

Diagramas : Graphviz

Algunas de las herramientas recomendadas

Emacs: usted puede agregar a emacs el módulo rst.el que añade un par de comandos y la sintaxis de la documentación a los escritores simpatizantes de Sphinx y reStructuredText.

FAQ

Pregunta : He añadido una entrada del índice o un nuevo término en el glosario y no se actualiza cuando compilo el documento.

Respuesta : El índice de Sphinx es a veces es desorientado y la gestión de la dependencia a veces, mejor. Por lo tanto, todo se debe reiniciar ejecutando el comando `make clean` dentro del directorio `cenditel-docs/sphinx/build/`.

1.2 Servicios de Zope/Plone

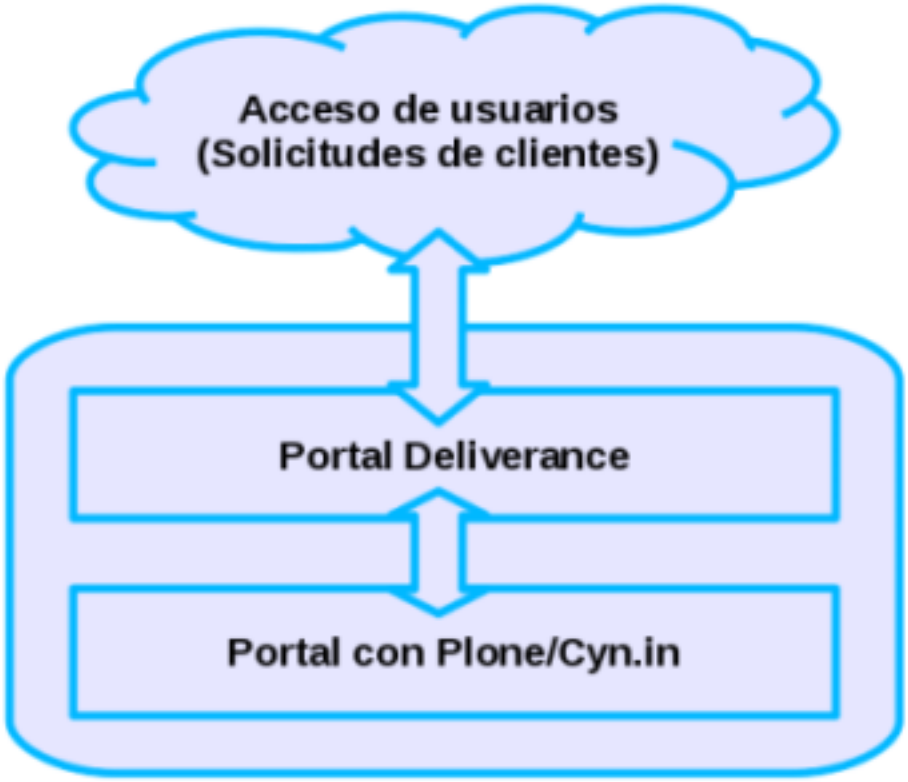
1.2.1 Introducción

Por terminar.

1.2.2 Arquitectura

La aplicación de comunidades de Cenditel esta compuesta de 3 diferentes aplicaciones –

- Cyn.in, es un Herramienta de gestión de conocimiento colaborativo Software Libre - Administración de contenidos Web.
- Aplicación Deliverance como Proxy inverso - Es usado para presentar una interfaz unificada para Cyn.in y cualquier otra potencial sistema de información web que se pueda integrar a esa plataforma - y también para el nivel de aplicación de temas y apariencias.



1.3 Apariencias, Temas, Pieles en Plone

1.3.1 Introducción

Existen diversos enfoques de realizar temas y pieles y sus respectivas herramientas para crear apariencias personalizadas en Plone. Cada una ha resultado como respuesta a momento históricos en el desarrollo Plone y por consiguiente se describirá a continuación:

Deliverance

Deliverance es una herramienta para hacer temas / apariencias HTML, aplicando un estilo consistente a aplicaciones y los archivos que son estáticos, independientemente de cómo se implementan, y separando de todo el sitio del sistema del plantillas y estilos del nivel de aplicación.

Descripción general

Deliverance, es una herramienta para hacer temas de aplicaciones Web la cual reescribe HTML usando selectores CSS (hojas de estilo en cascada) basado en un conjunto de reglas. El único requerimiento es que el diseño Web estático y que la aplicación Web ofrezcan soporte `HTML` y selectores «ID» `CSS` en el código fuente.

Funcionamiento

Deliverance hace transformaciones HTML para diseños estáticos de páginas Web, similar en funcionamiento a la XSLT pero usando un simple lenguaje basado en XML para expresar la transformación. Esta coloca porciones o secciones del código HTML generado por la fuente de datos aquí se llamará **contenido** dentro las secciones del código de los diseños estáticos de páginas Web aquí llamado **tema** el cual sirven de contenedor para emplazar la HTML generado, por ejemplo se pueden añadir desde estructuras de navegación, hojas de estilo desde el **contenido** al **tema**, entre otros.

A diferencia de los típicos sistemas de plantillas Web, los cuales implementan una combinación de técnicas y tecnologías estáticas (HTML(CSS/Javascript) con estructuras de programación dinámicos (ASP, JSP, PHP, Python, entre otros en cuestión) Deliverance le evita no usar ninguna variable de servidor / estructuras de programación para sustituir. Solo se necesitan definir un conjunto simple de reglas que se aplican en el código HTML.

Componentes de Deliverance

Contenidos Es la información a la que quiere aplicar un estilo a través del tema. Este puede ser un sitio Web dinámico o un archivo estático, especificado a través de una dirección URL por medio de reglas de reescritura direcciones URL y proxy inverso.

Tema Este contiene los estilos e información de diagramación/presentación de contenidos que se quiere aplicar al contenido. Este puede ser un sitio Web dinámico o un archivo estático, especificado a través de una dirección URL. El **tema** en si es una pagina HTML con poco código en ella. Es simplemente un ejemplo de lo que debería ser la pagina, lo que lo hace accesible a los diseñadores o a cualquier tipo de herramienta e incluso se pueden generar de forma dinámica.

Reglas Las reglas especifican al servicio de Deliverance como aplicar el tema al contenido. Hay 4 tipos de reglas: replace , append, prepend y drop (reemplazar, agregar, anteponer y quitar). Las reglas son especificadas en un documento XML.

Los atributos en cada regla contienen identificadores CSS o expresiones XPath que describen secciones en el tema y contenido en la cual ejecutara la regla.

¿Por que Deliverance?

- Muchas veces el diseño Web ya existe dentro de la organización.
- Mantiene la separación entre la presentación y el contenido.
- Rendimiento y flexibilidad.
- Porque permite trabajar sin un lenguaje de programación (para usuarios no-programadores).
- Debido a su concepto puede ser usado por diseñadores, integradores, desarrolladores, usuarios.

Conocimiento necesario

Es recomendable poseer conocimiento en las siguientes conceptos/tecnologías/herramientas:

- Lenguaje de marcado de Hipertexto HTML.
- Lenguaje de estilos en Cascada CSS.
- La extensión del navegador Web Mozilla Firefox llamada Firebug.
- Uso de un editor simple de archivos XML.

Beneficios

- Los diseñadores Web no necesitan aprender el sistema administración de contenidos `CMS`, el `Framework` o sistema de temas de un sitio/aplicación Web.
- Se puede unificar el diseño de múltiples aplicaciones Web con un diseño unificado aplicado a wikis, blogs, contenido estático `HTML`, entre otros.

Nota:

Una explicación detallada sobre esta tecnología la puedes encontrar en la conferencia dictada en la Plone Conference 2010 llamada: Easier and faster Plone theming with Deliverance and xdv por Nate Aune de la empresa Jazkarta.

Figura 1: Conferencia Easier and faster Plone theming with Deliverance and xdv por Nate Aune.

Instalación

Para instalar `Deliverance`, usted podría instalar la distribución de Deliverance de PyPI.

Nota: El paquete `Deliverance` sólo se requiere para obtener el comando para manipular el servidor proxy Deliverance.

Usted puede instalar la distribución de Deliverance usando easy_install, pip o zc.buildout. Por ejemplo, usando `easy_install` (sería ideal si se ejecuta dentro de un entorno virtual Python):

```
$ easy_install -U Deliverance
```

Usted también puede instalar con la herramienta `pip` si es su preferencia, puede realizarlo con el siguiente comando:

```
$ pip install Deliverance
```

Modos de instalación

Para este caso se instalara el ejemplo de instalación llamado DeliveranceDemo

Instalando con buildout

Si estas usando zc.buildout, usted puede usar la siguiente configuración dentro de `buildout.cfg` como punto de arranque. Este asegura que el script de consola `deliverance-proxy` esté instalado, lo cual es importante si usted necesita ejecutar manualmente el servidor proxy Deliverance:

```
[buildout]
versions = versions
parts =
   ...
   deliverance-server
   ...
[versions]
Deliverance = 0.5.0
WebOb = 0.9.8
PasteScript = 1.7.5
PasteDeploy = 1.5.0

[deliverance-server]
recipe = zc.recipe.egg
eggs =
    Deliverance
    PasteScript
```

Opcionalmente, en algunos sistemas operativos, en particular, Mac OS X la instalación de un «buen» paquete (Python egg) de `lxml` puede ser problemático, debido a una falta de coincidencia en las versiones del sistema operativo de las librerías `lxml` con respecto a la `libxml2` y `libxslt`. Para resolver esto, se puede compilar un `lxml` estático de paquete egg usando la siguiente receta buildout:

```
[buildout]
versions = versions
# lxml debería estar de primero en la lista ``parts``
parts =
   lxml
   deliverance-server

[versions]
Deliverance = 0.5.0
WebOb = 0.9.8
PasteScript = 1.7.5
PasteDeploy = 1.5.0

[lxml]
recipe = z3c.recipe.staticlxml
egg = lxml

[deliverance-server]
recipe = zc.recipe.egg
```

(continué en la próxima página)

(proviene de la página anterior)

```
eggs =
    Deliverance
    PasteScript
```

Entonces usted tiene que comenzar de arranque:

```
$ python bootstrap.py
```

Luego ejecute la construcción de su configuración zc.buildout, con el siguiente comando:

```
$ ./bin/buildout -vN
```

Nota: Note que el paquete lxml es una dependencia de Deliverance, usted podría necesitar instalar los paquetes de desarrollo de libxml2 y libxslt para poder construir esta configuración zc.buildout. En Debian/Ubuntu Linux usted puede ejecutar:

```
# sudo apt-get install build-essential python-dev libxml2-dev libxslt1-dev
```

Luego vuelva a ejecutar la construcción de su configuración zc.buildout como en paso anterior

Usted debería ver algo como esto:

```
Generated script '/home/user/deliverancedemo/bin/paster'.
Generated script '/home/user/deliverancedemo/bin/deliverance-proxy'.
```

Una ves instalado, usted debería buscar el script `deliverance-proxy` en el directorio `bin`.

Creando una Configuración

Luego de finalizar la intalación correctamente debe tener disponible en el script `bin/paster` el cual tiene disponible dos plantillas PasteScript para construir sitios con configuraciones Deliverance, para comprobar esto ejecute el siguiente comando:

```
$ ./bin/paster create --list-templates
Available templates:
  archetype:          A Plone project that uses Archetypes content types
  basic_buildout:     A basic buildout skeleton
  basic_namespace:    A basic Python project with a namespace package
  basic_package:      A basic setuptools-enabled package
  basic_zope:         A Zope project
  nested_namespace:   A basic Python project with a nested namespace (2 dots in name)
  paste_deploy:       A web application deployed through paste.deploy
  plone_basic:        A project for Plone products
  recipe:             A recipe project for zc.buildout
  deliverance:        Basic template for a deliverance-proxy setup
  deliverance_plone:  Plone-specific template for deliverance-proxy
```

Debería tener disponible la plantilla Paster `deliverance` y `deliverance_plone` la primera le permite crear una configuración básica para la instalación del servidor proxy Deliverance y la segunda permite crear una configuración especifica de Plone con un servidor proxy Deliverance.

A continuación se demostra cada creación de cada una de las plantillas Paster descritas anteriormente, con el siguiente comando:

```
$ ./bin/paster create -t deliverance mi-ejemplo-basico
Selected and implied templates:
  Deliverance#deliverance  Basic template for a deliverance-proxy setup

Variables:
  egg:      mi_ejemplo_basico
  package:  miejemplobasico
  project:  mi-ejemplo-basico
Enter host (The host/port to serve on) ['localhost:8000']: localhost:5000
Enter proxy_url (The main site to connect/proxy to) ['http://localhost:8080']:␣
↪localhost:8000
Enter proxy_rewrite_links (Rewrite links from sub_host?) ['n']: y
Enter password (The password for the deliverance admin console) ['']: secret
Enter theme_url (A URL to pull the initial theme from (optional)) ['']:
Creating template deliverance
Creating directory ./mi-ejemplo-basico
  Recursing into etc
    Creating ./mi-ejemplo-basico/etc/
    Copying deliv-users.htpasswd_tmpl to ./mi-ejemplo-basico/etc/deliv-users.htpasswd
    Copying deliverance.xml_tmpl to ./mi-ejemplo-basico/etc/deliverance.xml
    Recursing into supervisor.d
      Creating ./mi-ejemplo-basico/etc/supervisor.d/
      Copying deliverance.conf_tmpl to ./mi-ejemplo-basico/etc/supervisor.d/
↪deliverance.conf
    Copying supervisord.conf_tmpl to ./mi-ejemplo-basico/etc/supervisord.conf
Creating ./mi-ejemplo-basico/theme
Creating ./mi-ejemplo-basico/theme/theme.html
Creating ./mi-ejemplo-basico/theme/style.css
```

En el caso que requiera aplicar configuraciones Deliverance con sitios web Plone, para hacer esto ejecute el siguiente comando:

```
$ ./bin/paster create -t deliverance_plone mi-ejemplo-plone
Selected and implied templates:
  Deliverance#deliverance         Basic template for a deliverance-proxy setup
  Deliverance#deliverance_plone  Plone-specific template for deliverance-proxy

Variables:
  egg:      mi_ejemplo_plone
  package:  miejemploplone
  project:  mi-ejemplo-plone
Enter site_name (The name of your Plone site (no /'s)) ['']: Plone
Enter host (The host/port to serve on) ['localhost:8000']: localhost:5000
Enter proxy_url (The main site to connect/proxy to) ['http://localhost:8080']:
Enter proxy_rewrite_links (Rewrite links from sub_host?) ['n']: y
Enter password (The password for the deliverance admin console) ['']: secret
Enter theme_url (A URL to pull the initial theme from (optional)) ['']:
Creating template deliverance
Creating directory ./mi-ejemplo-plone
  Recursing into etc
    Creating ./mi-ejemplo-plone/etc/
    Copying deliv-users.htpasswd_tmpl to ./mi-ejemplo-plone/etc/deliv-users.htpasswd
    Copying deliverance.xml_tmpl to ./mi-ejemplo-plone/etc/deliverance.xml
    Recursing into supervisor.d
      Creating ./mi-ejemplo-plone/etc/supervisor.d/
      Copying deliverance.conf_tmpl to ./mi-ejemplo-plone/etc/supervisor.d/
↪deliverance.conf
```

(continué en la próxima página)

(proviene de la página anterior)

```
    Copying supervisord.conf_tmpl to ./mi-ejemplo-plone/etc/supervisord.conf
Creating ./mi-ejemplo-plone/theme
Creating ./mi-ejemplo-plone/theme/theme.html
Creating ./mi-ejemplo-plone/theme/style.css
Creating template deliverance_plone
  Recursing into etc
Replace 1601 bytes with 2062 bytes (3/49 lines changed; 9 lines added)
    Copying deliverance.xml_tmpl to ./mi-ejemplo-plone/etc/deliverance.xml
```

Usted debe iniciar la instancia Zope, con el siguiente comando:

```
$ ./bin/instance start
```

Y para finzalizar, sin importar la plantilla usada para crear la configuración, igualmente debe ejecutar manualmente el servidor proxy Deliverance, puede hacerlo ejecutando el siguiente comando:

```
$ ./bin/deliverance-proxy ./etc/deliverance.xml
To see logging, visit http://localhost:5000/.deliverance/login
    after login go to http://localhost:5000/?deliv_log
serving on http://localhost:5000
```

Como puede ver le esta indicando que Deliverance esta siendo servido por la dirección URL http://localhost:5000/ aplicando su estilo y tema HTML al contenido como se define en la archivo deliverance.xml

Para acceder a la consola depuración de iniciar sesión por la dirección URL http://localhost:5000/.deliverance/login y luego acceder a la dirección URL http://localhost:5000/?deliv_log

Instalación como middleware WSGI

Una de las caracteristicas de `Deliverance` es que soporta ejecución con un middleware WSGI.

Conceptos básicos

WSGI Es una Interfaz de Entrasa de Servidor, del Ingles *Web Server Gateway Interface*. Esto es una especificación para servidores web y servidores de aplicación para comunicarse con aplicaciones web (aunque también se puede utilizar para más que eso). Este es un estandar Python, descrito en detalles en PEP 333.

Entonces si usted quiere usar el filtro middleware WSGI, debe agregar esta configuración zc.buildout a su archivo `buildout.cfg`, a continuación un ejemplo:

```
[buildout]
versions = versions
parts =
    lxml
    deliverance

[versions]
Deliverance = 0.5.0
WebOb = 0.9.8
PasteScript = 1.7.5
PasteDeploy = 1.5.0

[lxml]
recipe = z3c.recipe.staticlxml
```

(continué en la próxima página)

(proviene de la página anterior)

```
egg = lxml

[deliverance]
recipe = zc.recipe.egg
dependent-scripts = true
eggs =
    Deliverance
    PasteScript
    PasteDeploy
```

Además debe crea el archivo `proxy.ini` en el directorio de su proyecto zc.buidlout con el siguiente comando:

```
$ nano ./proxy.ini
```

Y agregar la siguiente configuración:

```
[server:main]
use = egg:Paste#http
host = 0.0.0.0
port = 5000

[composite:main]
use = egg:Paste#urlmap
/static = static
/ = default

[app:static]
use = egg:Paste#static
document_root = %(here)s/theme

[pipeline:default]
pipeline = theme
           content

[filter:deliverance]
use = egg:Deliverance
rule_filename = %(here)s/etc/deliverance.xml
theme_uri = %(here)s/theme/theme.html
execute_pyref = true
debug = true

# Proxy: por ejemplo, Plone, cuyo nombre es Plone en localhost:8080.
[app:content]
use = egg:Paste#proxy
address = http://localhost:8080/VirtualHostBase/http/localhost:5000/Plone
```

Si aun no a comenzado de arranque de proyecto, entonces ejecute el siguiente comando:

```
$ python bootstrap.py
```

Como ha realizado cambios a su configuración zc.buildout, debe iniciar ejecutar la construcción de su configuración zc.buildout, con el siguiente comando:

```
$ ./bin/buildout -vN
```

Al finalizar la construcción de su proyecto más archivos se agregan a los scripts disponibles en el directorio `bin/`, incluyendo `bin/paster`, `bin/deliverance-proxy`.

Una ves terminada la instalación puede iniciar el arranque del mismo con el siguiente comando:

```
$ ./bin/paster serve --reload ./proxy.ini
```

A continuación, puede tener acceso a nuestra página en http://localhost:5000 .

Instalación en diversas plataformas

A continuación se ofrece una serie de recetas de como instalar `Deliverance` en varios sistemas operativos:

Debian GNU/Linux Lenny

Requerimientos previos

Iniciar sesión con el usuario root, con el siguiente comando:

```
$ su -
```

Instalar dependencias de sistema, con el siguiente comando:

```
# aptitude install python2.4 python2.4-dev python2.4-imaging python-profiler python2.
↪4-setuptools libc6-dev subversion git-core
```

Instalar el paquete para entorno virtual Python, con el siguiente comando:

```
# easy_install-2.4 virtualenv
```

Cerrar sesión del usuario root, con el siguiente comando:

```
# exit
```

Entornos virtual Python

Preparar entorno de trabajo, con los siguientes comandos:

```
$ cd $HOME ; mkdir ./virtualenv ; cd ./virtualenv
$ virtualenv --no-site-packages --python=/usr/bin/python2.4 python2.4
```

Activamos el entorno virtual recién creado, para mas información consultar aquí.

```
$ source $HOME/virtualenv/python2.4/bin/activate
```

Descargar código de ejemplo

Para esto debe preparar el directorio destino de la descaga y realizar la descarga propiamente dicha, para esto ejecute los siguientes comandos:

```
$ source $HOME/virtualenv/python2.4/bin/activate
$ cd $HOME ; mkdir ./proyectos ; cd ./proyectos
$ svn co http://svn.plone.org/svn/collective/deliverancedemo/trunk/ deliverancedemo
```

Iniciar construcción del proyecto, con los siguientes comandos:

```
$ cd ./deliverancedemo
$ python bootstrap.py
```

(Nota antes de seguir al siguiente paso se aconseja modificar el `buildout` por el proporcionado acá para evitar errores de instalación, se han proporcionado 2 uno para instalación del servicio de `Deliverance` y otro de instalación de `Deliverance` con `Plone`, modificarlo por el que este acorde a sus preferencias)

```
$ ./bin/buildout -vN
```

Usted debería ver algo como esto:

```
Generated script '/home/user/deliverancedemo/bin/paster'.
Generated script '/home/user/deliverancedemo/bin/deliverance-proxy'.
Generated interpreter '/home/user/deliverancedemo/deliverancedemo/bin/py'.
```

Usted debe iniciar la instancia Zope, con el siguiente comando:

```
$ ./bin/instance start
```

Y por ultimo debe iniciar el servidor proxy Deliverance, con el siguiente comando:

```
$ ./bin/deliverance-proxy ./rules.xml
To see logging, visit http://localhost:5000/.deliverance/login
    after login go to http://localhost:5000/?deliv_log
serving on http://localhost:5000
```

Como puede ver le esta indicando que Deliverance esta siendo servido por la dirección URL http://localhost:5000/ aplicando su estilo y tema HTML al contenido como se define en la archivo deliverance.xml

Para acceder a la consola depuración de iniciar sesión por la dirección URL http://localhost:5000/.deliverance/login y luego acceder a la dirección URL http://localhost:5000/?deliv_log

Entonces la instalación fue realizada correctamente.

Ubuntu 11.04 Natty Narwhal

Instalar dependencias de sistema

```
# sudo apt-get install python python-dev python-imaging python-profiler python-
↪setuptools libc6-dev subversion git-core build-essential
```

Instalar entorno virtual

```
# easy_install-2.6 virtualenv
```

Cerrar sesión del usuario root

```
# exit
```

Preparar entorno de trabajo

```
$ cd $HOME ; mkdir ./virtualenv ; cd ./virtualenv
```

```
$ virtualenv --no-site-packages --python=/home/usr/virtualenv/bin/python2.6 python2.6
```

Nota: /usr se refiere al usuario del sistema el nombre de usuario que usted tenga en el sistema

Activamos el entorno virtual recién creado y seguimos con la instalación, para mas información consultar https://plone-spanish-docs.readthedocs.io/es/latest/python/creacion_entornos_virtuales.html

```
$ source $HOME/virtualenv/python2.6/bin/activate
$ cd $HOME ; mkdir ./proyectos ; cd ./proyectos
$ svn co http://svn.plone.org/svn/collective/deliverancedemo/trunk/ deliverancedemo
```

Iniciar construcción del proyecto

```
$ cd ./deliverancedemo
$ python bootstrap.py
```

(Nota antes de seguir al siguiente paso se aconseja modificar el `buildout` por el proporcionado acá para evitar errores de instalación, se han proporcionado 2 uno para instalación del servicio de `Deliverance` y otro de instalación de `Deliverance` con `Plone`, modificarlo por el que este acorde a sus preferencias)

```
$ ./bin/buildout -vN
```

Usted debería ver algo como esto:

```
Generated script '/home/user/deliverancedemo/bin/paster'.
Generated script '/home/user/deliverancedemo/bin/deliverance-proxy'/.
Generated interpreter '/home/user/deliverancedemo/deliverancedemo/bin/py'.
```

Y por ultimo debe iniciar el servidor proxy Deliverance, con el siguiente comando:

```
$ ./bin/deliverance-proxy ./rules.xml
To see logging, visit http://localhost:5000/.deliverance/login
    after login go to http://localhost:5000/?deliv_log
serving on http://localhost:5000
```

Como puede ver le esta indicando que Deliverance esta siendo servido por la dirección URL http://localhost:5000/ aplicando su estilo y tema HTML al contenido como se define en la archivo deliverance.xml

Para acceder a la consola depuración de iniciar sesión por la dirección URL http://localhost:5000/.deliverance/login y luego acceder a la dirección URL http://localhost:5000/?deliv_log

Entonces la instalación fue realizada correctamente.

Instalación de Python mediante zc.buildout

A continuación se explicara como realizar una instalación de un interprete python2.4 mediante mecanismos de configuración zc.buildout

El primer paso seria crear una carpeta y acceder a ella

```
$ cd $HOME ; mkdir ./python2.4 ; cd ./python2.4
$ svn co http://svn.plone.org/svn/collective/buildout/python/src buildout.python
$ cd buildout.python
```

Iniciar construcción del proyecto

```
$ python bootstrap.py
```

Luego de haber realizado este paso es recomendable hacerle las modificaciones correspondientes al buildout, para este ejemplo se realizo una instalación de un python2.4, si usted necesita hacer una instalación de este tipo verificar que su buildout este igual al que se presenta a continuación:

```
# This is here just for backward compatibility
[buildout]
extends =
    src/base.cfg
    src/readline.cfg
    src/libjpeg.cfg
    src/python24.cfg
#    src/python25.cfg
#    src/python26.cfg
#    src/python27.cfg
#    src/python32.cfg
    src/links.cfg

parts =
    ${buildout:base-parts}
    ${buildout:readline-parts}
    ${buildout:libjpeg-parts}
    ${buildout:python24-parts}
#    ${buildout:python25-parts}
#    ${buildout:python26-parts}
#    ${buildout:python27-parts}
#    ${buildout:python32-parts}
    ${buildout:links-parts}

python-buildout-root = ${buildout:directory}/src

# we want our own eggs directory and nothing shared from a
# ~/.buildout/default.cfg to prevent any errors and interference
eggs-directory = eggs


extra_options_py24 = --with-threads --enable-unicode=ucs4
#extra_options_py24 = --with-threads --enable-unicode=ucs4 --with-readline

[install-links]
prefix = /opt/local
```

Luego de haberle aplicado los cambios respectivos a su buildout ejecutar el siguiente comando

```
$ ./bin/buildout -vN
```

A este punto ya la instalación debe haberse realizado correctamente y se debe poder activar el entorno virtual sin ningún problema.

```
$ source $HOME/python2.4/bin/activate
```

Configuración

Para aplicar las reglas con `Deliverance` debemos hacerlo en el archivo `rules.xml` ubicado en el directorio raíz para este caso particular se ubicaba en `$HOME/proyectos/deliverancedemo` y esto consiste en configurar el puerto que servirá como fuente de contenido, y el puerto donde se verán los cambios y el efecto de las reglas de Deliverance. Para aplicar las reglas hay que seleccionar la ruta y crear una clase.

```
<proxy path="/" class="plone">

<!--Luego llamar a la clase, indicarle la ruta del tema y aplicar las reglas-->

<rule class="plone" suppress-standard="1">

<!-- Tema -->

<theme href="/static/index.html" />
```

Por defecto el `Deliverance` viene configurado para salir por el puerto `5000` pero puede ser cambiado por cualquiera de su preferencia.

Bajo este formato se configuran los proxys, para indicar donde esta la fuente de contenido y en que Puerto serán visualizados los cambios con deliverance.

```
<proxy path="/" class="plone">

<dest href="http://localhost:8080/VirtualHostBase/http/localhost:5000/Plone/
↪VirtualHostRoot/" />

</proxy>
```

Esta configuración previa indica que la fuente de contenido `Plone` sale por el puerto `8080` http://localhost:8080 y los cambios en el `Deliverance` serán vistos por el puerto `5000` http://localhost:5000

Para más información sobre configuración de los Proxy revisar la documentación sobre **How Virtual Host Monster works**: http://web.archive.org/web/20101107014540/http://plone.org/documentation/kb/plone-apache/vhm

Para configurar Deliverance con Plone

El archivo de reglas debe ir así:

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

<server-settings>
<server>localhost:5000</server>
<execute-pyref>true</execute-pyref>
<dev-allow>localhost</dev-allow>
<dev-user username="guest" password="guest" />
</server-settings>

<proxy path="/static" class="static" editable="1">
<dest href="{here}/static/" />
</proxy>

<proxy path="/banjo" class="banjo">
  <dest href="{here}/src/banjo/" />
</proxy>

<proxy path="/" class="plone">
<dest href="http://localhost:8080/VirtualHostBase/http/localhost:5000/Plone/
↪VirtualHostRoot/" />
</proxy>
```

(continué en la próxima página)

(proviene de la página anterior)

```
<rule class="static" />

<rule class="banjo" />

<rule class="plone" suppress-standard="1">


<!-- Theme -->
<theme href="/static/index.html" />

<!-- Rules -->

<replace content='/html/head/title' theme='/html/head/title' />
<append content='/html/head/base' theme='children:/html/head' />

<append content="link[href *= 'authoring']" theme='children:/html/head' />
<append content="link[href *= 'portlets']" theme='children:/html/head' />

<!-- Add in the Plone-created CSS and JS in addition to the static ones -->
<append content='/html/head/script' theme='/html/head' />
<!--><append  content='/html/head/style' theme='/html/head' /> -->

<!-- Append the id/class attributes from the body tag, this is important for Kupu and␣
↪per-section styling -->
<append content="attributes(id,class):/html/body" theme="attributes:/html/body" />

<!-- Copy the logo -->
<replace content='#portal-logo img' theme='#logo h1' />

<!-- Copy the breadcrumbs -->
<!-- <replace content='#portal-breadcrumbs' theme='#pathbar' />
<replace content='#portal-personaltools' theme='#personaltools' /> -->


<!-- Copy the main navigation -->
<replace content='children:#portal-globalnav' theme='children:#links ul' />


<!-- <prepend content='dl.portletLogin' theme='children:#rightbar' /> -->

<!-- <replace content='children:#parent-fieldname-title' theme='children:#leftbar h2'␣
↪/> -->
<!-- Get rid of the user icon and copy the user link -->
<drop content='#user-name img' />
<replace content='#user-name' theme='#user a' />

<!-- Copy the edit bar -->
<replace content='#content-views' theme='children:#edit-menu' />
<replace content='div.contentActions' theme='children:#action-menu' />

<!-- ...but get rid of the content type icons. -->
<drop content='#plone-contentmenu-factories dd ul li a img' />

<!-- <drop content='#link-presentation' />
<drop content='div.documentActions' />
<drop content='div.documentByLine' />
<drop content='span.documentByLine' />
```

(continué en la próxima página)

(proviene de la página anterior)

```
<drop content='#review-history' />
<drop content="attributes(class):a.external-link" />
<drop content="attributes(class):a.plain-link" /> -->

<!-- Copy over the contents of the page body -->
<!-- <replace content='children:#content' theme='children:#leftbar' /> -->

<!-- put the title of the page as the heading -->
<replace content='children:#parent-fieldname-title' theme='children:#heading' />

<!-- remove the history dropdown -->
<drop content='dl#history' />

<!-- put the documentDescription in the first paragraph -->
<replace content='children:#parent-fieldname-description' theme='children:#description
↪' />
<!-- we keep the documentDescription class so we can do some styling later -->

<!-- put the body text in the second paragraph -->
<replace content='children:#parent-fieldname-text' theme='children:#bodytext' />

<!-- drop the more link at the bottom -->
 <drop theme='/html/body/div/div/div[3]/div/a' />

<!-- for news listing page -->
<drop content='div.documentByLine' />
<drop content='attributes(class):h2.tileHeadline a' />
<drop content='attributes(class):h2.tileHeadline' />
<replace content='children:div.tileItem' theme='//*[@id="leftbar"]/p[2]' />

<!-- for event listing page -->
<!-- <replace content='/html/body/div/table/tbody/tr/td/div/div[2]/div[2]/div/dl/dt/
↪span/a' theme='//*[@id="leftbar"]/p[2]' /> -->
<!-- <replace content='span.contenttype-event' theme='//*[@id="leftbar"]/p[2]' /> -->

<!-- <replace content='/html/body/div/table/tbody/tr/td/div/div[2]/div[2]/div/dl'␣
↪theme='//*[@id="leftbar"]/p[2]' /> -->

<!-- stuff to remove from portlet -->
<drop content='dd.portletItem a img' />
<drop content='span.portletItemDetails' />

<!-- <replace content='children:.portletNews span.portletItemDetails' theme=
↪'children:span.orangetext' /> -->

<!-- <replace ifcontent='body.section-events' content='children:dl.portletNews dt.
↪portletHeader a' theme='children:#rightbar h2' />
<replace ifcontent='body.section-events' content='children:dl.portletNews dd.
↪portletItem' theme='children:#rightbar p' />

<replace ifcontent='body.section-news' content='children:dl.portletEvents dt.
↪portletHeader a' theme='children:#rightbar h2' />
<replace ifcontent='body.section-news' content='children:dl.portletEvents dd.
↪portletItem' theme='children:#rightbar p' />     -->

<!-- Bring the portlet columns inside the sidebar -->
<!-- <append content='#portal-column-one'  theme='#rightbar' />
```

(continué en la próxima página)

(proviene de la página anterior)

```
<append content='#portal-column-two'  theme='#rightbar' /> -->

</rule>
</ruleset>
```

Configurar Deliverance con archivos HTML locales

Ideal para cuando no se cuenta con conexión a Internet o no se tiene acceso directo a la fuente de contenido, con esta configuración la fuente de contenido será una pagina `HTML` previamente guardada y colocada dentro de la carpeta correspondiente

Para este caso se crea un Proxy con una clase y se le indica la dirección donde se encontrara el `HTML`, para este ejemplo dentro del directorio raíz `Deliverancedemo` se creo una carpeta llamada local:

```
<proxy path="/" class="plone" rewrite-links="1">
<dest href="{here}/local/" />
</proxy>
```

El archivo de reglas por consiguiente queda de esta manera

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset>
<server-settings>
<server>localhost:5000</server>
<execute-pyref>true</execute-pyref>
<dev-allow>localhost</dev-allow>
<dev-user username="guest" password="guest" />
</server-settings>

<proxy path="/static" class="static" editable="1">
<dest href="{here}/static/" />
</proxy>

<proxy path="/" class="plone" rewrite-links="1">
<dest href="{here}/local/" />
</proxy>

<rule class="static" />
<rule class="plone" suppress-standard="1">

<!-- Tema -->

<theme href="/static/local_pagina_inicio/index.html" />

<!--Reglas-->

</rule>
</ruleset>
```

Guía de Uso

A continuación una guía de sencillos pasos para su uso:

Siete pasos para su uso

1) Ejecutar el servicio de Deliverance.
2) Preparar los archivos HTML/CSS de su tema.
3) Debe colocar los archivos en una carpeta, normalmente dentro del proyecto Deliverance llamada static.
4) Identificar los selectores CSS de los elementos en el sitio / aplicación Web como fuente de datos que se desea mapear al tema estático.
5) Identificar los selectores CSS dentro del tema que servirán como marcadores de posición para los elementos dinámicos traídos desde la fuente de contenidos.
6) Crear un archivo de reglas que integre los elementos de la fuente de contenido en los marcadores de posición del tema.
7) Configurar el servidor host al Proxy inverso de Deliverance.

Se recomienda el navegador Firefox e instalar el complemento llamado Firebug, el cual nos permitirá identificar de manera rápida y simple los identificadores CSS del contenido y del tema.

El complemento firebug se puede descargar desde aquí, una vez instalado podemos acceder a su consola dándole clic a un pequeño insecto que aparecerá en la parte superior o inferior derecha dependiendo de la versión del navegador. Para mas información sobre su uso consultar este tutorial.

Como ya se ha mencionado anteriormente Deliverance utiliza selectores CSS para mapear elementos desde una fuente de contenidos dinámicos que bien pueden ser CMS como (Plone, Joomla, Django entre otros) o archivos HTML locales (estáticos) previamente guardados o como por ejemplo este documento.

Se puede hacer uso de las reglas con los identificadores CSS o bien usando expresiones Xpath en el caso de que el elemento no tenga un identificador CSS.

- Los atributos identificados con id se invocan con el siguiente carácter: #.
- Los atributos identificados con una clase class se invocan con el siguiente carácter: ..

Ejemplos de la aplicación de reglas usando un Plone como fuente de contenidos.

- Para esto debemos activar el entorno virtual Virtualenv.

```
$ source $HOME/virtualenv/python2.4/bin/activate
```

- Luego de esto ubicarse en el directorio raíz del Deliverancedemo y realizar el siguiente comando para levantar una instancia de Plone.

```
$ cd $HOME/proyectos/deliverancedemo/
$ ./bin/instance start
```

Usted debería ver algo como esto

```
. daemon process started, pid=2749
```

- Ahora activaremos el servicio de Deliverance.

```
$ ./bin/deliverance-proxy rules.xml
```

Usted debería ver algo como esto

```
To see logging, visit http://localhost:5000/.deliverance/login
    after login go to http://localhost:5000/?deliv_log
serving on http://localhost:5000
```

- Basándonos en la configuración anterior es recomendable abrir 2 ventanas del navegador una con la fuente de contenido http://localhost:8080 y otra con el `tema` http://localhost:5000
- Luego de realizar estos pasos ya se puede empezar a aplicar las reglas para ello debemos modificar el archivo `rules.xml`, se puede hacer modificándolo directamente y para ver los cambios solo se debe actualizar la pagina http://localhost:5000 presionando `f5` o dando clic en el navegador, también se pueden ver y aplicar las reglas desde la consola de depuración de deliverance http://localhost:5000/?deliv_log o bien reiniciando el servicio para visualizarlos

```
$ ./bin/deliverance-proxy rules.xml
```

- A continuación un ejemplo sobre como reemplazar el logo de `Plone` por el del `tema` para esto identificaremos el atributo mediante selectores `CSS` a través de `firebug` *

Identificando atributos mediante selectores CSS a través de firebug el identificador del logo del tema

Esta regla nos permite reemplazar el logo

```
<replace content='#portal-logo img' theme='#logo' />
```

Ahora se reemplazara el titulo del `tema` por el del contenido dinámico `Plone`. Para realizar esta regla haremos uso de las expresiones `XPath`

```
<replace content='/html/head/title' theme='/html/head/title' />
```

Ahora se reemplazara la barra de edición Para ello utilizaremos de nuevo el `firebug` para identificar los atributos `CSS`

```
<replace content='.content-views' theme='.edit-menu' />
```

Ahora se integrara la columna de contenidos de `Plone` al `tema`

```
<replace content='#portal-column-content' theme='children:#description' />
```

La aplicación de estas reglas dan como resultado que la integración con `Plone` y el `tema` luzca de esta forma:

Como aplicar temas a diferentes secciones del contenido

Aplicar clase de la página por ruta, si tenemos una parte del contenido a la que queremos aplicar otro tema o otras reglas, por ejemplo http://localhost:8080/Plone/applications lo hacemos mediante estas reglas:

```
<match path="/applications" class="applications" />
<rule class="applications" suppress-standard="1">
<theme href="/static/applications.html" />

<replace content='children:p.documentDescription'theme='children:span.SectionSubtitle
↪' />
<replace content='children:#portal-column-two'theme='children:#right-column' />

</rule>
```

Cada sección puede tener su tema y reglas diferentes.

Preguntas frecuentes, consejos y recomendaciones

Preguntas Frecuentes

¿Como cambiar el diseño de esta demostración?.

Debe ubicarse en el directorio $HOME/proyectos/deliverancedemo/static/

¿Donde encuentro el archivo de configuración de reglas XML?.

Para esta demostración esta ubicado en la ruta $HOME/proyectos/deliverancedemo/rules.xml

¿Como activar la consola de Depuración?.

- Para ello debe iniciar sesión, acceda a la dirección http://localhost:5000/.deliverance/login
- las claves de usuario y contraseña si no han sido cambiadas por defecto son: *guest*
- Para activar la consola de Depuración acceda a la dirección http://localhost:5000/?deliv_log
- Se pueden aplicar las reglas desde la consola de depuración y al guardar los cambios inmediatamente se visualizaran es recomendable trabajar de este modo.

¿Como ver el diseño del tema sin cambios?.

Para ver el tema estático acceda a la dirección http://localhost:5000/static

Recomendaciones

Agregar los atributos `id/class` del body del contenido al tema.

Esta regla es beneficiosa debido a que agrega todos los `id/class` de `Plone` al `tema` sin la necesidad de tener que codificarlas una a una, esto representa un gran ahorro de tiempo.

```
<append content="attributes(id,class):/html/body" theme="attributes:/html/body" />
```

Importar los `CSS` y `Javascript` del contenido al tema.

Esto es beneficioso debido a que al realizar estas reglas ya se tendrán los CSS y Javascript de Plone y no habrá necesidad de tener que hacerlos uno por uno en el archivo `CSS` del `tema` representando también un gran ahorro de tiempo.

```
<prepend content="link[href *= 'authoring']" theme="link[href *= 'style']" />

<append content="link[href *= 'public']" theme="children:/html/head" />

<prepend content="/html/head/style" theme="link[href *= 'style']" />

<append content="/html/head/script" theme="children:/html/head" />
```

Consejos

- No importar los estilos a no ser que sea necesario.

```
<prepend content="/html/head/style" theme="link[href *= 'style']" />
```

Debido a que estos podrían cambiar totalmente la apariencia del tema.

- No utilizar directamente el selector `CSS` a no ser que sea realmente necesario, siempre se debe usar un solo elemento, hijos del elemento, atributos de una etiqueta etc. Ejemplo ("children:#heading"). Esto es porque los

selectores CSS y los tipos se superponen en la sintaxis general, cambiando el diseño de nuestro tema lo cual no es recomendable.

Veamos un ejemplo de esto utilizando un Plone como fuente de contenido, se sustituirá la barra de búsqueda de Plone por la lista de usuario del tema.

La barra de búsqueda de Plone identificada con la clase LSBox.

El contenedor cuyo id es user contiene una lista no ordenada identificada como usuario la cual vendría siendo hijo de la clase user.

Si usamos la regla en Deliverance replace usando directamente la clase user tendremos como resultado:

```
<replace content=".LSBox" theme=".user" />
```

Como se podrá ver utilizando directamente la clase user los CSS de la barra de búsqueda de Plone se superponen a los del tema quedando en una ubicación incorrecta.

Si se usa la misma regla de replace pero esta vez invocando el elemento hijo de la clase user veremos que esta vez se sitúa en el sitio correspondiente.

```
<replace content=".LSBox" theme="children:#usuario" />
```

Consejo para el uso de XPath

De ser posible maquetar completamente todos los elementos de nuestro tema para evitar hacer el uso de expresiones XPath, esto es debido a que algunas expresiones XPath al ser cambiadas dañan el DOM (Modelo de Objetos del Documento) de nuestro HTML.

A continuación un caso de practico tomando el ejemplo anterior pero esta vez usando expresiones XPath para reemplazar la barra de búsqueda.

```
<replace content="/html/body/div/div/div/div/form/div" theme="/html/body/div/div/div/
↪div/div" />
```

Nos da como resultado que el DOM de nuestro HTML ha sido cambiado y causa un error como este:

Si estuvo realizando los cambios desde la consola de depuración no podrá solucionarlo volviendo atrás, para volver al estado anterior tendrá que detener el servicio presionando crtl+c, modificar el archivo rules con un editor de texto de su preferencia y borrar la linea de configuración que ocasiono el error, guardar los cambios e iniciar de nuevo el servicio.

```
$ cd deliverancedemo/
$ ./bin/deliverance-proxy rules.xml
```

También puede detener el servicio crtl+c y borrar la linea de configuración que ocasiono el error desde la consola con el comando.

```
$ nano ./rules.xml
```

guardar los cambios si lo hizo con nano ctrl+o y para salir ctrl+x luego iniciar de nuevo el servicio de Deliverance.

```
$ cd deliverancedemo/
$ ./bin/deliverance-proxy rules.xml
```

Integración con cyn.in

Se realizo una integración de `Deliverance` para el portal de comunidades en Cenditel que esta basado en la plataforma `cyn.in` versión 3.1.3 el cual esta basado en una versión de `Plone` 3.3.1

Para el levantamiento de información sobre los cambios que se realizarían se levanto una reunión con algunos de los usuarios del portal de comunidades con los cuales se llegaron a los siguientes acuerdos.

- La creación del botón principios políticos de la fundación Cenditel.
- La creación del botón colectivos comunitarios el cual unificaría los espacios actuales creados como lo son, Medios de comunicación popular liberadora, Renasen y proyectos comunitarios.
- El portal de comunidades en Cenditel esta basado actualmente en la plataforma Cynin, dicha plataforma contiene portlets en la cual se pueden ver las etiquetas de las publicaciones, comentarios recientes, usuarios conectados entre otros, dichos portlets serán quitados.
- Actualmente en el portal de comunidades en Cenditel se pueden agregar muchos contenidos sin embargo a petición de los usuarios se llego al acuerdo de que solo se podrán publicar, imágenes, archivos, wikis, audio y vídeo.
- En la plataforma actual la manera de agregar contenidos ya sean imágenes, archivos, vídeos, audios entre otros, es dando clic al botón Nuevo el cual se encuentra en la parte superior izquierda del portal, agregar contenidos de esta forma causa confusión a los usuarios dificultándole en ocasiones no encontrar la forma de agregar contenidos, por ende la manera de agregar imágenes, vídeos, wikis, audios y archivos serán colocadas en un menú lateral.
- Se pidió la creación de un cintillo institucional de color representativo de la misma en el cual se vea reflejado el logo de la institución.
- Bajo el cintillo institucional se quiere un menú horizontal negra la cual despliegue contenidos de forma vertical, las secciones de este menú desplegable serán: Contenidos y metodologías, Audios y vídeos, Colectivos comunitarios, Articulación y Reflexión.
- Los contenidos del espacio que actualmente es llamado Ayuda, pasaran a ser parte del espacio Contenidos y metodologías.
- Se pidió quitar las vistas de aplicación.
- Se pidió la creación de 4 cajas de texto en la cual se mostrarían las noticias más relevantes.
- Se pidió la creación de un banner o deslizador de imágenes.
- Se desea visualizar imágenes de los colectivos comunitarios que trabajan actualmente con la fundación Cenditel.

También se realizo una encuesta que serviría para evaluar algunos puntos que pudieran haber quedado fuera de discusión en la reunión.

Encuesta Realizada a Juan Lenzo

Experiencias con la Integración

Para el cumplimiento de los requerimientos hablados en la reunión se realizaron 2 diseños, la pagina de inicio, la cual contiene cintillo institucional rojo con el logo de la fundación, menú horizontal negro con contenidos desplegables en forma vertical, menú lateral el cual contiene las formas de ingresar contenidos, los principios políticos de la fundación, un deslizador de imágenes, las 4 cajas de texto con la información mas relevante y una pequeña animación en `javascript` la cual cambia cada 3 segundos, en ella se ven reflejados los distintos colectivos comunitarios que trabajan con la fundación.

Se realizo también un segundo diseño, destinado a mostrar los contenidos, el cual contiene cintillo institucional con logo de la institución y menú horizontal, en este diseño se incorporara por medio del servicio de `Deliverance` la

columna 3 de la plataforma `cyn.in` (la columna3 de la plataforma cyn.in es aquella en la cual se muestran y se crean todos los contenidos).

Columna 3 de la plataforma cyn.in

Este segundo diseño se usara para añadir contenidos y mostrar información de la plataforma de comunidades en Cenditel, se decidió incorporar solo la columna3 de la plataforma actual ya que dicha columna se encuentra presente en todas las secciones de la plataforma de comunidades y se aplicaría a todas las pantallas que fueran necesarias.

Evolución de los diseños

Durante la fase de programación a medida que se empezó a avanzar los diseños, empezaron a evolucionar pasando por tres cambios siendo el tercer cambio el definitivo.

Primer diseño realizado

Segundo diseño realizado

Diseño final

Maquetación en CSS de la pagina de inicio

Fase de pruebas

Durante la fase de prueba surgieron problemas con la pagina de inicio los cuales se mencionan a continuación:

- No incorporaba elementos dinámicos de la plataforma de comunidades en Cenditel.
- Era un `HTML` completamente estático y para su modificación había que ir directamente a modificar el código fuente del archivo lo cual no era la idea principal.
- El menú lateral daba problemas de usabilidad y accesibilidad debido a la gran cantidad de contenidos que desplegaba.
- Las 4 cajas de texto no se modificarían a menos que el administrador de la plataforma modificara el código `HTML` del tema.
- El programa `Javascript` del deslizador de imágenes daba problemas de compatibilidad con los programas de inicio del servicio de `Deliverance` haciendo que este no iniciara correctamente.

Problema de usabilidad y accesibilidad del menú lateral

En la fase de pruebas se decidió descartar el primer diseño y se le aplicaron los cambios pertinentes al segundo diseño y al menú lateral, este menú ya no desplegara contenidos de forma horizontal y sera agregado al segundo diseño. Se decidió utilizar solo el segundo diseño para todo el contenido, como se menciono anteriormente este incorpora la columna 3 de la plataforma de comunidades la cual esta presente en todas las secciones y esta columna incorpora los elementos dinámicos necesarios para su funcionamiento tales como títulos, menús de navegación y contenidos. También incorpora el portal-breadcrumbs el cual es un elemento que permite al usuario orientarse sobre su ubicación dentro de la plataforma de comunidades.

Portal Breadcrumbs

Diseño final

Maquetación en CSS del diseño final

A continuación se explicara un poco sobre la diagramacion en CSS

- logo: Corresponde al logotipo de la institución.
- children:#usuario: Corresponde a la sección para iniciar sesión.
- children:#navegacion: Corresponde a la sección del menú horizontal la cual contendrá los botones, ayuda, colectivos comunitarios entre otros.
- #sidebar: Corresponde al menú lateral en el cual se podrán agregar los distintos contenidos.

- children:#heading: Este identificador sera reemplazado por el portal-breadcrumbs de la plataforma de comunidades, el portal-breadcrumbs es una barra de ubicación.
- children:#bodytext: Este identificador sera reemplazado por la columna3 de la plataforma de comunidades, la columna3 es el elemento en el cual son mostrados y agregados todos los contenidos del portal.
- #rotator: Es una pequeña animación en javascript la cual cambiara cada 3 segundos y en ella se verán visualizados los distintos colectivos comunitarios que trabajan con la fundación.
- #footer: Corresponde al pie de pagina de la plataforma de comunidades.

Menú Horizontal

Div id=»Rotator»

cada imagen puede tener un link que redireccione al portal o Web del colectivo comunitario.

```
<div id="rotator"> <!-- Mini Slide-->
<a href="http://localhost:5000/cynin/home/colectivos-comunitarios/lapiz-    rebelde/
↪mas-de-100000-visitas"><img src="images/02.jpg" alt="" /></a>
<a href="http://localhost:5000/cynin/home/colectivos-comunitarios/    investarte/una-
↪botella-menos-en-nuestro-paisaje"><img src="images/04.jpg"     alt="" /></a>
<!-- Podemos poner todas las imágenes que queramos, siempre metidas entre el div␣
↪rotator--><!--fin rotator--></div>
```

Implementacion

Primer paso

Activar un entorno virtual de python2.4 como se explico anteriormente en la sección de instalación

```
$ source virtualenv/python2.4/./bin/activate
```

Segundo paso

Iniciar la instancia en cyn.in como se explico anteriormente en la sección de instalación

```
(python2.4)$ ./bin/instance fg
```

Tercer Paso

Editar el archivo de reglas. Para este ejemplo la integración de Deliverance con Cynin en base a los requerimientos de los usuarios fueron necesarias las siguientes reglas:

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset>
<server-settings>
<server>localhost:5000</server>
<execute-pyref>true</execute-pyref>
<dev-allow>localhost</dev-allow>
<dev-user username="guest" password="guest" />
</server-settings>

<proxy path="/static" class="static" editable="1">
<dest href="{here}/static/" />
</proxy>

<proxy path="/" class="plone">
<dest href="http://localhost:8081/cynin/" />
```

(continué en la próxima página)

(proviene de la página anterior)

```
<response rewrite-links="1" />
</proxy>

<rule class="static" />
<rule class="plone" suppress-standard="1">

<!-- Theme -->

<theme href="/static/index.html" />

<!-- Reglas -->

<prepend content='/html/head/base' theme='children:/html/head' />

<!-- Agregar los CSS y Javascript de la plataforma al tema -->

<append content='/html/head/meta' theme='/html/head' />

<append content='/html/head/link' theme='/html/head' />

<append content='/html/head/style' theme='/html/head' />

<append content='/html/head/script' theme='/html/head' />

<!--append theme="//head" content="//head/meta" nocontent="ignore" /-->

<!--Reemplazar titulo del contenido al tema estatico-->

<replace content='/html/head/title' theme='/html/head/title' />

<!-- Agregar los Ids y Clases de la plataforma cynin a la sección body del tema -->

<append content="attributes(id,class):/html/body" theme="attributes:/html/body" />

<!-- Reemplazar descripción del contenido  -->

<drop theme='children:#bodytext' />

<replace content='#col3_content' theme='children:#description' />

<!-- Reemplazar titulo del texto por Ubicación -->

<replace content='#portal-breadcrumbs' theme='children:#heading' nocontent="ignore"/>

<!--Reemplazar barra de Usuario-->

<replace content='.myareanotloggedin' theme='children:#usuario'/>

</rule>

</ruleset>
```

Cuarto Paso

Iniciar el Servicio de Deliverance

```
(python2.4)$ ./bin/deliverance-proxy rules.xml
```

A continuación unas capturas sobre como las reglas presentadas anteriormente hacen que el servicio de Deliverance cambie la apariencia por defecto del portal de comunidades.

Portal de comunidades en Cenditel

Portal de comunidades en Cenditel con el servicio de Deliverance

Como se pudo ver en las imágenes anteriores la columna 3 de la plataforma de comunidades y el portal-breadcrumbs se integraron correctamente en el tema, se pueden agregar contenidos, todas las vistas de aplicación funcionan. Sin embargo dicho tema es recomendable realizarle los cambios que se mencionan a continuación:

- El menú horizontal despliega contenidos, pero estos contenidos no son traídos de forma dinámica desde la plataforma de comunidades estos están debidamente enlazados, se realizo de esta manera debido a que la estructura de carpetas actual de la plataforma de comunidades no se encuentra distribuida de la manera como sus usuarios lo desean, para esto hay que aplicar cambios a la estructura de contenidos actual del CMS. Es importante realizar este cambio ya que momentáneamente los usuarios de la plataforma pueden agregar contenidos pero estos no se verán visualizados en los menús hasta que el administrador de la plataforma no modifique el código `HTML` del tema.
- Para agregar contenidos se limito en el diseño del tema a que los usuarios solo puedan agregar wikis, imágenes, archivos y audios, debido a que son estos los contenidos que comúnmente agregan, sin embargo no se debería limitar en esta medida a todos los usuarios, es decir, la plataforma ofrece un panel de configuración dentro de cada espacio de comunidad donde se pueden definir que tipos de contenidos se pueden seleccionar para ser agregados por los usuarios y adicionalmente desde el panel de administración se pueden agregar permisos en los cuales se especifican que clase de contenido se puede agregar a cada sección, haciéndolo de esta forma se podría usar el menú lateral para incorporar otro elemento.
- Se decidió dejar las vistas de aplicación de la plataforma cyn.in ya que remover estas quitarían completamente la usabilidad del portal de comunidades.
- A petición de los usuarios del portal de comunidades no se agrego el portlet de usuario, sin embargo es recomendable agregar este debido a que sin este elemento dificultara a los mismos usuarios cerrar su sesión.
- A petición de los usuarios del portal de comunidades se realizaron los menús desplegables, pero basándose en la experiencia obtenida en la implementación estos menús dan problemas de accesibilidad y usabilidad, en ocasiones no se integran debidamente como corresponde, para futuros cambios es recomendable cambiar al menos el menú lateral por lo que actualmente se conoce como acordeón de botones. Para mas información en esta web se pueden ver algunos ejemplos http://www.celulaweb.net/2008/10/13/30-menús-basados-en-tabs-y-acordeon/

Guía del desarrollo

Por definir.

XDV / diazo

diazo es una herramienta para hacer temas/apariencias HTML, aplicando un estilo consistente a aplicaciones y los archivos estáticos, independientemente de cómo se implementan, y separando de todo el sitio de estilo del nivel de aplicación de plantillas.

XDV / Diazo

El paquete diazo implementa un lenguaje como Deliverance usando un puro motor XSLT. Con Diazo, usted «compila» el tema y el conjunto de reglas (ruleset) en un solo paso, entonces usa una súper / simple transformación en

cada solicitud adicional. Por otra parte, usted puede compilar el tema durante el desarrollo, protegerlo dentro de un repositorio de control de versiones Subversion o Git, y no tocar Diazo durante la implementación.

Este le permite aplicar un tema incluido en una página web HTML estática a un sitio web creado de manera dinámica utilizando cualquier tecnología del lado del servidor. Con Diazo, usted puede tomar una diagramación de un diseño HTML creado por un diseñador de páginas web y convertirlo en un tema para su favorito de la CMS, el rediseño de la interfaz de usuario de una aplicación web operativa sin ni siquiera tener acceso al código fuente original, o crear una experiencia de usuario unificada a través de múltiples sistemas dispares, todo en cuestión de horas, no semanas.

Cuando se utiliza Diazo, tendrá que trabajar con la sintaxis y conceptos familiares al trabajar con HTML y CSS. Y por lo que le permite integrar sin problemas los archivos XSLT en su regla, Diazo hace que en los casos más simple y complejos comunes las exigencias sean posibles.

Para obtener documentación detallada, consulte diazo.org.

Instalación

Para instalar Diazo, usted podría instalar la distribución de diazo de PyPI

Nota: El paquete Diazo sólo se requiere para obtener el compilador Diazo y herramientas de desarrollo. Si se implementa el tema Diazo en un servidor web, no es necesario la distribución diazo en ese servidor.

Usted puede instalar la distribución de diazo usando easy_install, pip o zc.buildout. Por ejemplo, usando easy_install (sería ideal si se ejecuta dentro de un entorno virtual Python):

```
$ easy_install -U diazo
```

Opcionalmente, si estas usando zc.buildout, usted puede usar la siguiente configuración buildout.cfg como punto de arranque. Este asegura que los scripts de consola estén instalados, lo cual es importante si usted necesita ejecutar manualmente el compilador Diazo:

```
[buildout]
parts =
   diazo

[diazo]
recipe = zc.recipe.egg
eggs = diazo
```

En algunos sistemas operativos, en particular, Mac OS X la instalación de un «buen» paquete (Python egg) de `lxml` puede ser problemático, debido a una falta de coincidencia en las versiones del sistema operativo de las librerías `lxml` con respecto a la `libxml2` y `libxslt`. Para resolver esto, se puede compilar un `lxml` estático de paquete egg usando la siguiente receta buildout:

```
[buildout]
# lxml debería estar de primero en la lista ``parts``
parts =
   lxml
   diazo

[lxml]
recipe = z3c.recipe.staticlxml
egg = lxml

[diazo]
```

(continué en la próxima página)

(proviene de la página anterior)

```
recipe = zc.recipe.egg
eggs = diazo
```

Entonces usted tiene que comenzar de arranque:

```
$ python bootstrap.py
```

Luego ejecute la construcción de su configuración zc.buildout, con el siguiente comando:

```
$ ./bin/buildout -vN
```

Nota: Note que el paquete lxml es una dependencia de diazo, usted podría necesitar instalar los paquetes de desarrollo de libxml2 y libxslt para poder construir esta configuración zc.buildout. En Debian/Ubuntu Linux usted puede ejecutar:

```
# sudo apt-get install build-essential python2.6-dev libxml2-dev libxslt1-dev
```

Una ves instalado, usted debería buscar los scripts `diazocompiler` y `diazorun` en su directorio `bin`.

Si usted quiere usar el filtro middleware WSGI, usted debería usar el parámetro extra [wsgi] cuando se instale el paquete egg Diazo, a continuación un ejemplo:

```
[buildout]
extends = http://good-py.appspot.com/release/diazo/1.0b1
versions = versions
parts =
    diazo

[diazo]
recipe = zc.recipe.egg
eggs =
    diazo [wsgi]
    PasteScript

[lxml]
recipe = z3c.recipe.staticlxml
egg = lxml
```

Entonces usted tiene que comenzar de arranque:

```
$ python bootstrap.py
```

Luego ejecute la construcción de su configuración zc.buildout, con el siguiente comando:

```
$ ./bin/buildout -vN
```

Al finalizar la construcción zc.buildout más archivos se añaden a la lista scripts disponibles en el directorio `bin/`, incluyendo `bin/paster`, `bin/diazocompiler` o `bin/diazorun`.

Ahora puede crear una carpeta que contiene diversos recursos para nuestro tema.

```
$ mkdir mitema
```

A continuación, crea el archivo `proxy.ini` en el directorio de su proyecto zc.buidlout:

```
[server:main]
use = egg:Paste#http
host = 0.0.0.0
port = 5000

[composite:main]
use = egg:Paste#urlmap
/static = static
/ = default

[app:static]
use = egg:Paste#static
document_root = %(here)s/theme

[pipeline:default]
pipeline = theme
           content

[filter:theme]
use = egg:diazo
rules = %(here)s/rules.xml
prefix = /static
debug = true

# Proxy: por ejemplo, Plone, cuyo nombre es MiSitio en 127.0.0.1:8080.
[app:content]
use = egg:Paste#proxy
address = http://127.0.0.1:8080/VirtualHostBase/http/127.0.0.1:5000/MiSitio
```

Uno sólo tiene que lanzar el proxy con el siguiente comando:

```
$ bin/paster serve --reload proxy.ini
```

A continuación, puede tener acceso a nuestra página en http://127.0.0.1:5000 .

Configuración

Por Terminar.

Guía de Uso

Por Terminar.

Preguntas Frecuentes

¿Qué diferencia hay entre XDV y Diazo? XDV es ahora Diazo

¿Qué es Diazo/XDV? Diazo/XDV es una extensión de la tecnología Deliverance.

¿Por que el uso de XSLT? POR DEFINIR

¿Que diferencia hay entre XDV y Deliverance? Realmente no sé, pero intuyo que tiene que ver con el hecho de con XSLT se pueden hacer transformadas no solamente para HTML, se pueden manipular XML de todo tipo como RSS, ATOM. Deliverance te deja usar CSS3, pero sus procesos internos son un misterio para la mayoría,

al menos sabiendo que Diazo/XDV usa XSLT, y de esta forma el programador/integrador puedo intuir que está haciendo, ya que XSLT es estándar.

1.4 Project Portfolio Management Framework NG

1.4.1 Introducción

Esta es la documentación del producto `cenditel.ppm` para el sistema administración de contenidos Plone.

PPM lo cual sus siglas en ingles significa Project Portfolio Management (`Gestion de cartera de Proyectos`) es una herramienta la cual ayuda a describir los métodos para el análisis y la gestión colectiva de un grupo de propuestas o proyectos, sobre la base de numerosas características clave. Los objetivos fundamentales de PPM son:

- Mantener la transparencia, control, monitoreo y evaluación de las propuestas de proyectos.
- Uso de una herramienta de incidencias para reportes en los proyectos.
- Tener un repositorio de proyectos y propuestas de estos.

En las siguientes paginas encontrará material referente al su proceso de instalación usando la herramienta `buildout`, sus dependencias tanto en el sistema administración de contenidos Plone y la suite de participación Web cyn.in.

1.4.2 Instalación

PPM (Portafolio Project Manamegent), es un producto para Plone el cual ayuda a determinar la combinación óptima y la secuencia de los proyectos propuestos para lograr los mejores objetivos y así describir los métodos para el análisis y la gestión colectiva de un grupo de proyectos

Para instalar el producto es necesario Un sitio Plone 3

A continuación veremos los pasos para replicar el ambiente de desarrollo:

Primer Paso:

Descargar de la pagina Web del Proyecto el script de instalación buildout que le permitirá configurar un sitio.

Para realizar la descarga proceda en un terminal con los siguientes comandos:

```
$ mkdir buildouts ; cd buildouts
$ svn co http://plataforma.cenditel.gob.ve/svn/plataforma/proyectosInstitucionales/
↪renasen/cenditel.ppm/buildouts/plone3/ ppm.buildout
$ cd ppm.buildout ; ls -p
00-varibles.cfg        02-mrdeveloper.cfg  bootstrap.py  cenditel.ppm.cfg     etc/    ␣
↪       products/  templates/
01-dumpedversions.cfg  03-mountpoint.cfg   buildout.cfg  dumped-versions.cfg ␣
↪plonesite.cfg  src/       versions.cfg
```

El primer comando es el encargado de crear un directorio para nuestros proyectos buildout, el segundo comando abre dicho directorio , el tercero es el encargado de descargar del repositorio lo necesario para instalar nuestro producto

Para saber un poco más acerca de scripts listados, serán explicados a continuación.

bootstrap.py:

Es un script python, encargado de realizar la descarga y construcción de un ambiente de desarrollo buildout para sitios Web basados en Zope/Plone

buildout.cfg:

Archivo de configuración que posee las características necesarias para la construcción de un sitio Web basado básico en el sistema manejador de contenidos Plone.

cenditel.ppm.cfg:

Contiene las configuraciones del Producto y productos adicionales que necesita PPM para poder instalarlo

Instale una jaula de python2.4 en su sistema para evitar daños a su sistema operativo. Proceda como se señala a continuación.

```
$ sudo aptitude install python2.4 python2.4-minimal python2.4-dev python-virtualenv␣
↪python-setuptools
$ virtualenv -p python2.4 python2.4/
$ cd python2.4/
$ source bin/activate
(python2.4)$ cd $HOME/buildouts/ppm.buildout
(python2.4)$ python bootstrap.py
```

El primer comando, instala las dependencias python en el sistema operativo. Si usted se encuentra bajo el sistema operativo Debian Lenny o Ubuntu Karmic Koala, no tendrá problemas de dependencias. El segundo comando, crea una jaula virtual de python en su directorio de usuario llamada python2.4, con el tercer comando entramos a ella, para activarla usamos el cuarto comando, los siguientes comandos nos llevan al entorno de desarrollo allí llamamos al interprete de python para que ejecute al archivo bootstrap.py; el cual nos dará una salida como:

Segundo Paso:

A continuación, podemos proceder a realizar la instalación del sitio. Para eso, ejecutaremos el siguiente comando.

```
(python2.4)$ ./bin/buildout -vNc cenditel.ppm.cfg
```

Este comando descarga automáticamente todas las dependencias de PPM y las pone en los lugares correspondientes e instala Plone 3.3

Tercer Paso:

Ya con Plone instalado y descargado los productos necesarios solo queda subir la instancia de plone, ir a agregar Productos e instalar Project Portfolio Management Framework NG y sus dependencias

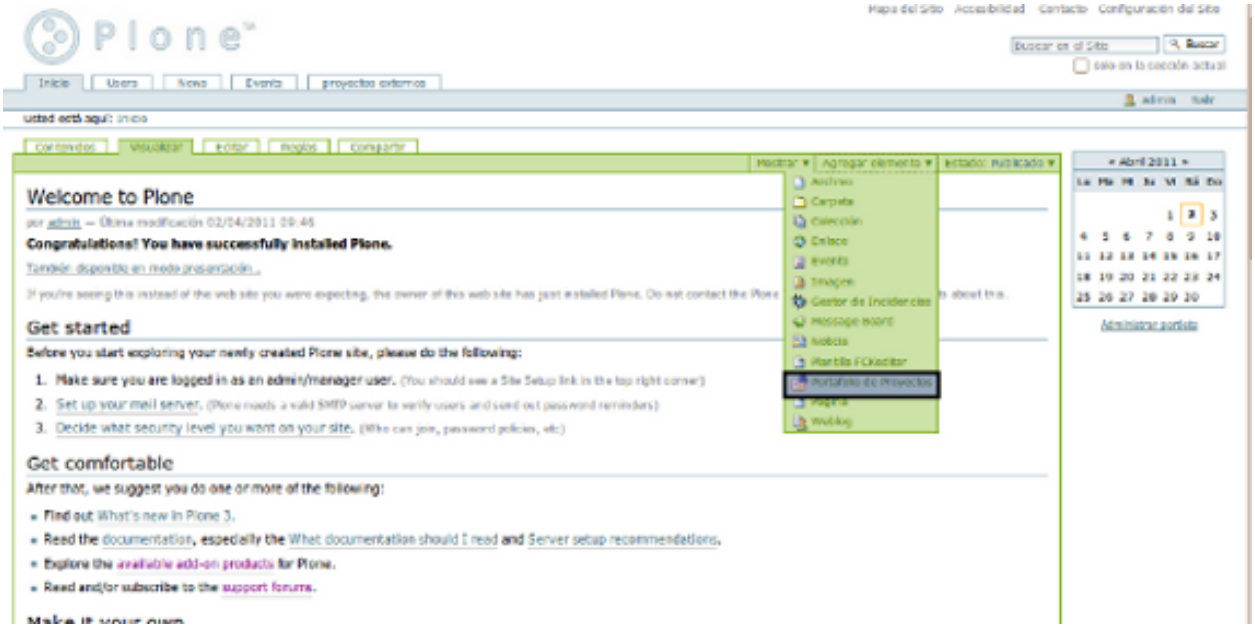
1.4.3 Dependencias

- **DataGridField**, es un componente de entrada de la tabla para Plone. Utiliza Javascript para hacer la introducción de datos tabulares más de proceso de usuario amigable - no hay ida y vuelta peticiones HTTP al servidor al insertar o eliminar filas. Más información consulte la pagina de proyecto de DataGridField.

- **Poi**, es un producto de seguimiento de incidencias para Plone. su objetivo es ser sencillo y atractivo, proporcionando la cuestiones más necesarias para el seguimiento de funcionalidades. Más información consulte la pagina de proyecto de Poi.
- **Quills:** es un weblog para Plone. Se ha diseñado desde cero para trabajar bien y ofrecer funciones especializadas para una multi-blog, el medio ambiente multi-usuario. Más información consulte la pagina de proyecto de Quills.
- **Ploneboard:** es un producto para Plone su objetivo es poner el comportamiento de un foro de debate en un sitio Plone. Más información consulte la pagina de proyecto de Ploneboard.
- **CPFCKTemplates:** es un producto con el que usted puede agregar una plantilla para el editor FCKeditor como un contenido de Plone. Cada usuario verá el «habilitado » las plantillas que se encuentran en el catálogo, es decir, sólo las plantillas que el usuario tiene permiso para ver. Más información consulte su manual de uso del producto.

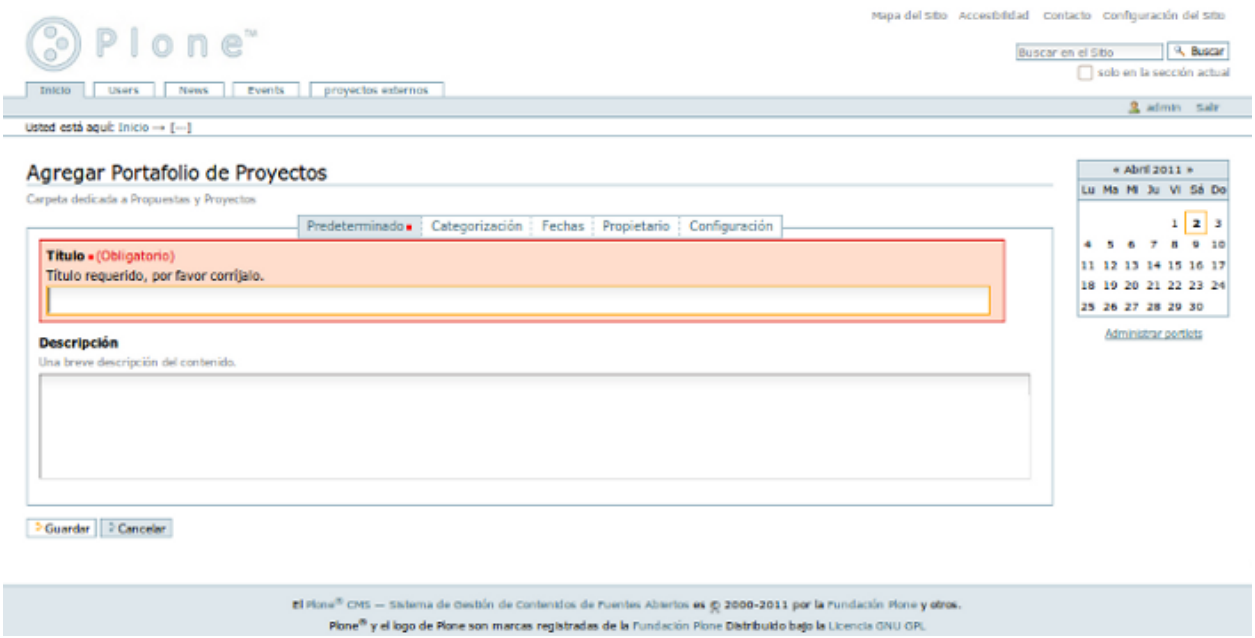
1.4.4 Guía de Uso

Para agregar contenido en su sitio Plone, vuelva a la página principal http://localhost:8080/demo.



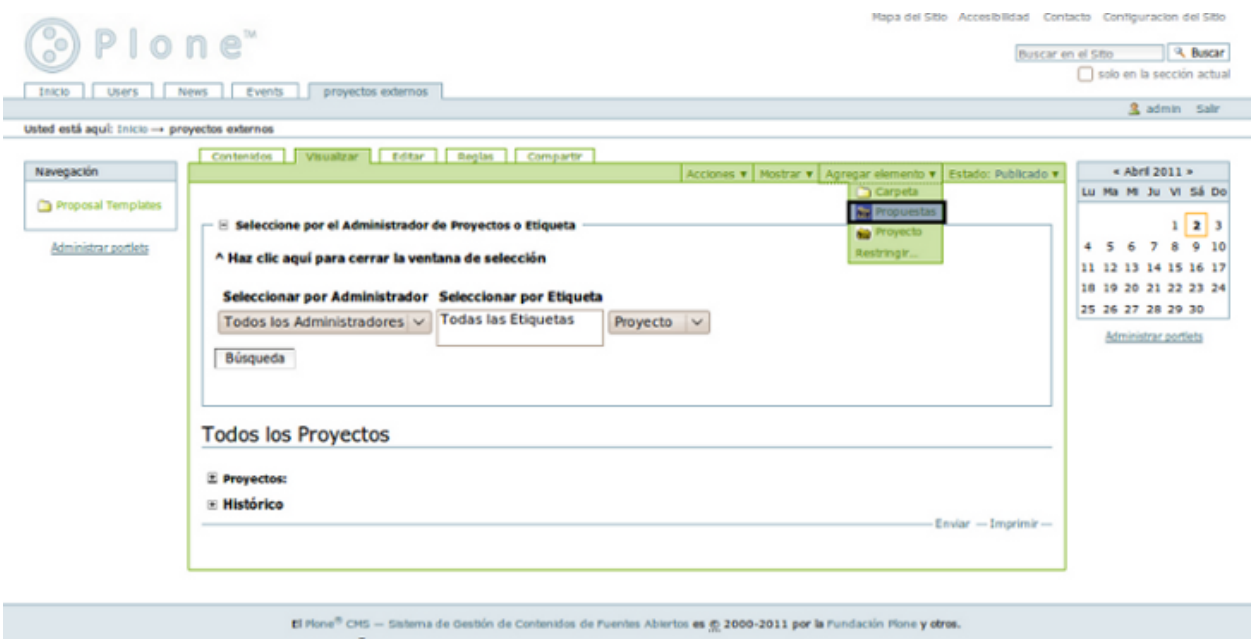
Como se puede ver en la imagen anterior, al hacer clic en el botón agregar elemento se despliega una serie opciones de tipos de contenido que pueden ser agregados, para agregar una cartera de proyectos hacemos Clic en Portafolio de Proyectos.

El cual lleva a otra pantalla para llenar los suguientes campos:

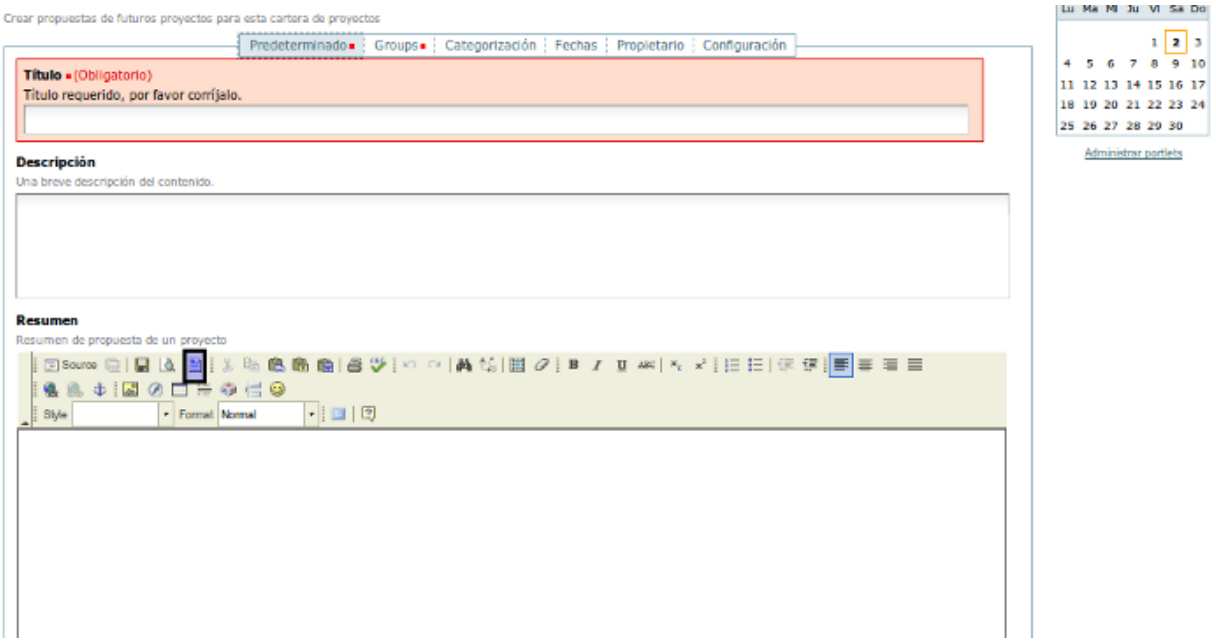


Así queda la vista de Portafolio de Proyectos una vez creado

Una vez creado un Portafolio de Proyectos nos da la opcion de agregar Propuestas y Proyectos ahora vamos a agregar una Propuesta, nos dirigimos a agregar elemento y hacemos Clic en Propuestas



Nos llevara a la siguiente pantalla donde llenaremos los siguientes campos



1.4.5 Flujos de Trabajos

Por definir

1.4.6 Integración con cyn.in

Por definir

1.5 Productos Multimedia de Cenditel

1.5.1 Introducción

Esta es la documentación de los productos multimedia de la fundación Cenditel para el sistema administración de contenidos Plone.

En las siguientes páginas encontrará material referente al su proceso de instalación usando la herramienta `buildout`, la instalación y configuración de sus dependencias tanto en el sistema administración de contenidos Plone y la herramienta cyn.in. En todos los casos, presentados en el siguiente manual, se entiende que usted esta corriendo una versión de Python 2.4 en un sistema operativo Debian Lenny, en su defecto en una distribución basada en esta.

Componentes

Los productos multimedia de Cenditel se componente en los siguientes módulos:

cenditel.audio

Este producto proporciona un tipo de contenido para que archivos de audio puedan ser agregados al sistema administración de contenidos Plone permitiendo su Streaming a través de la Web previa conversión a formatos libres.

cenditel.video

Este producto proporciona un tipo de contenido para que archivos de vídeo puedan ser agregados al sistema administración de contenidos Plone permitiendo su Streaming a través de la Web previa conversión a formatos libres.

cenditel.transcodedaemon

Este producto proporciona un panel de configuración de servicio de conversión de formatos multimedia para los productos `cenditel.video` y `cenditel.audio`, en el cual se permite definir los parámetros de conversión que usa la herramienta FFMPEG. Prestando el servicio de conversión mediante una cola que sigue la disciplina primero en entrar, primero en salir.

cenditel.multimediaplayertheme

Este producto proporciona un piel para el reproductor HTML5 por defecto de los navegadores permitiendo una vista entandar para cada uno de ellos. Se basa en la librería javascript jMediaElement de *Alexader Farkas*.

1.5.2 Instalación

Para instalar el producto cenditel.multimedia usted necesita solventar las siguientes dependencias en su instancia Plone/Zope

- Un sitio Plone 3.3.5 configurado con el producto File System Storage usando la estrategia de almacenamiento «site2».
- Un servidor web nginx, con una configuración especifica en mimetypes, que permitirá el streaming a partir de este.

A continuación veremos los pasos necesarios para replicar el ambiente de desarrollo:

Primer Paso

Descargar de la pagina Web del Proyecto el script de instalación buildout que le permitirá configurar un sitio demostrativo.

Para realizar la descarga proceda en un terminal con los siguientes comandos:

```
$ mkdir buildouts
$ cd buildouts
$ svn co http://plataforma.cenditel.gob.ve/svn/plataforma
/proyectosInstitucionales/renasen/cenditel.multimedia/buildout/plone/3.3/␣
↪cenditelmultimedia
$ cd cenditelmultimedia
$ ls -p
00-varibles.cfg           06-contenttypes.cfg     plonesite.cfg
01-dumpedversions.cfg     bootstrap.py            products/
02-mrdeveloper.cfg        buildout.cfg            src/
03-prerequemients.cfg     cenditelmultimedia.cfg  templates/
04-mountpoint.cfg         dumped-versions.cfg     versions.cfg
05-mediafilesstorage.cfg  etc/
```

El primer comando es el encargado de crear un directorio para nuestros proyectos buildout, el segundo comando abre dicho directorio pero la magia realmente se encuentra en el tercero, el cual es el encargado de descargar del repositorio subversion los scripts listados por el cuarto comando.

Para saber un poco más acerca de scripts listados, serán explicados a continuación.

bootstrap.py

Es un script python, encargado de realizar la descarga y construcción de un ambiente de desarrollo buildout para sitios Web basados en Zope/Plone

buildout.cfg

Archivo de configuración que posee las características necesarias para la construcción de un sitio Web basado básico en el sistema administración de contenidos Plone.

00-varibles.cfg

Como su nombre lo indica, contiene las variables a ser utilizadas por el sistema de instalación buildout para la reconstrucción del sitio de pruebas.

Se divide en cuatro partes:

- **buildout: Posee la definición de las variables de extensión del producto al** momento de ser ejecutado el script, es decir indica el orden de ejecución, para eso usa la variable `extends=`, además declara la variable `site-id` con un nombre de sitio que va a ser usado más adelante.
- **zopeserver: Contiene configuraciones referentes al servidor de aplicaciones Zope.**
  - **La primera variable de esta parte es la variable `user` que indica el** nombre de usuario y contraseña del usuario `admin` del servidor con las configuraciones por defecto.
  - La segunda variable `effective-user` corresponde al usuario Unix del servidor.
  - **La tercera variable `host` es una variable correspondiente a la dirección IP** del servidor Web.
  - **La cuarta variable `debug-mode` indica si el servidor se encuentra en** modo de depuración, por defecto se encuentra en off.
  - **La quinta variable, `verbose-security` indica si se presentan informes** detallados de la seguridad, por defecto se encuentra en off.
- hosts: Contiene las distintas variables de conexión al servidor de producción.
  - Variable `http-address`, corresponde a la dirección del Servidor Web Zope.
  - Variable `ftp-address`, corresponde a la dirección del Servidor FTP de Zope.
  - Variable `webdav-address`, corresponde a la dirección del Servidor WebDav de Zope
  - Variable `dns`, nombre del dominio producción o la dirección IP
  - Variable `user`, corresponde al usuario de servicios ssh que se conectara al Hosting del servidor.
  - **Variable `password`, corresponde a la contraseña del servicio ssh para** conectarse al Hosting del servidor.
- ports: Contiene las variable de los distintos puertos para los diferentes servicios.
  - Variable `http-address`, contiene el puerto de entrada a la aplicación http.

- Variable `ftp-address`, contiene el puerto de entrada a la aplicación mediante ftp.
- Variable `webdav-address`, contiene el puerto de entrada a la aplicación mediante servicios Webdav.

01-dumpedversions.cfg

Contiene el receta buildout.dumppickedversions el cual crea al archivo dumped-versions.cfg que contiene la lista de productos instalados y sus respectivas versiones. Se actualiza cada vez que se ejecuta buildout.

02-mrdeveloper.cfg

Consta de la sección `buildout` y la sección `sources`, en la primera es declarada la variable `extends` que permite la extensión de las configuraciones a partir del archivo 01-dumpedversions.cfg. Por otro lado agrega la extensión para buildout mr.developer

El récipe mr.developer admite las siguientes variables de configuración:

- sources-dir: Indica el directorio donde serán descargados los distintos productos, por defecto es `src`.
- sources: Indica el nombre de la sección donde serán indicados los paquetes a descargar.
- always-check: Especifica el nombre de los archivos a los cuales siempre que buildout se ejecute se le realizará check out.
- auto-checkout: Especifica el nombre de los archivos a los cuales siempre que buildout se ejecute se le realizará check out.

Por otro lado, la sección `sources` se encuentra vacía porque aún no es necesaria su utilización.

03-prerequemients.cfg

Este Script cuenta de las siguientes partes: pre-requemients, make-fss-directory, vhost-nginx, mime-types-nginx, config-nginx.

- **pre-requemients: Usa el recipe plone.recipe.command** el cual es utilizado para lanzar el comando de instalación necesario para instalar nginx y ffmpeg, mediante la variable `command`.
- **make-fss-directory: Usa el recipe plone.recipe.command , con el cual se crean** los directorios necesarios para el producto File System Storage y para la creación de archivos de configuración del servidor nginx. Además de los comandos lanzados con `command` utiliza las siguientes variables:
  - **update-command: Esta variable, es utilizada cuando buildout es ejecutado** pero la parte no ha sido alterada.
  - **stop-on-error: Cuando el valor es yes, no o true. Buildout detiene su ejecución** si un comando recibe un valor de salida cero.
- **vhost-nginx: Usa el recipe collective.recipe.template,** mediante este, se crea una template de ejemplo que va a ser utilizada por el servidor nginx para realizar el servicio de streaming.
- **mime-types-nginx: Usa el recipe collective.recipe.template** para crear un archivo de configuración de mimetypes para el servidor web nginx.
- **config-nginx: Usa el recipe plone.recipe.command** y con los comandos, crea enlaces simbólicos, verifica la configuración del servidor web nginx y además recarga la configuración.
- **update-command: Esta variable, es utilizada cuando buildout es ejecutado pero** la parte no ha sido alterada.

- **stop-on-error: Cuando el valor es yes, no o true. Buildout detiene su ejecución** si un comando recibe un valor de salida cero.

04-mountpoint.cfg

Este archivo de configuración, crea punto de montaje en la para un sitio web basado en Plone de manera tal, que se permitan Bases de Datos separadas para cada sitio Plone. Para mayor información puede visitar Multiple Plone sites per zope instance – using separate Data.fs files for each one.

05-mediafilestorage.cfg

Este script tiene las configuraciones necesarias para el manejo de los archivos de audio y vídeo, a nivel del disco duro. Consta de cuatro secciones:

- buildout: Se declara la variable extends, y se indica que este script continua con las configuraciones a partir del archivo 04-mountpoint.cfg. Y se declara la adición de la parte fss.
- instance: agrega eggs python necesarios para la configuración del servidor Zope de manera que este use el sistema de archivos.
- fss: Utiliza el recipe iw.recipe.ffs, el récipe consta de las siguientes variables:
  - zope-instances: Por defecto tiene asignado el valor `${instance:location}`
  - storage: En esta variable se indica los lugares donde serán colocados los distintos archivos, consiste en tres configuraciones:
    - global: Explica el tipo de almacenamiento global para todos los sitios.
    - Almacenamiento específico para cada sitio: Después de la linea global se pueden declarar estrategias de almacenamiento específicas para cada sitio.

      Para ello se sigue la sintaxis: `plone_flat /sitename site2 path/to/storage`

      donde:

      `plone_flat`, es un alias para la configuración;

      `sitename`, es el nombre de un sitio que se encuentra en el root de la ZMI;

      `site2`: es la configuración de almacenamiento para el sitio;

      `path/to/storage`: Es el sitio en el disco duro donde `iw.fss` colocará los archivos que vienen de la ZODB.
- versions: Especifica versiones especificas que son necesarias para la instalación del sistema.

06-contenttypes.cfg

Extiende del archivo de configuración 05-mediafilestorage.cfg, además en este archivo es declarada una parte llamada `contenttypes-conf` que utiliza el recipe plone.recipe.atcontenttypes en esta configuración la variable `max-file-size` especifica el tamaño máximo que los tipos de contenido pueden tener dentro de los sitios Plone, la variable `max-image-dimension` específica la resolución máxima en pixeles, para las imágenes de los contenidos de noticias y para las imágenes. Por ultimo, la variable `pil-quality` señala, la calidad con que serán guardadas las imágenes.

cenditelmultimedia.cfg

Extiende del archivo de configuración 06-contenttypes.cfg, posee las siguientes variables:

- auto-checkout: Declara los eggs a los cuales el recipe mr.developer mencionado previamente realizará un check out.
- eggs: Indica al script buildout cuales paquetes de tipo huevo python debe descargar para instalación.
- zcml: Indica a buildout, cuales paquetes de tipo huevo python deben ser configurados en base a archivos de configuración zcml.

Además contiene la parte de la declaración de los paquetes a los cuales se les realizará un check out para la instalación de los mismos en la instancia Zope, es decir la parte sources que fue previamente mencionada en el archivo 02-mrdeveloper.cfg.

plonesite.cfg

Extiende del archivo de configuración cenditelmultimedia.cfg. Utiliza el récipe collective.recipe.plonesite aceptando las siguientes variables de configuración:

- site-id: Nombre del sitio de ejemplo creado con el récipe.
- intance: Corresponde al nombre de la instancia que esta corriendo el script, por defecto `instance`.
- profiles: Corresponde a una lista de perfiles de GenericSetup que se ejecutaran cada vez que se ejecute el script buildout.

templates

Este directorio contiene modelos de archivos de configuración que son modificados en base a las variables declaradas en el archivo 00-variables.cfg, permitiendo replicar configuraciones para el servidor nginx.

products

Corresponde al directorio **products** creado por bootstrap.py.

src

Es el directorio de instalación donde serán colocados los archivos en desarrollo. En este caso, el récipe mr.developer coloca aquí dichos archivos.

etc

En este directorio, son colocados los archivos de salida generados a partir del récipe de collective.recipe.template

Segundo paso

Instale una jaula de python2.4 en su sistema para evitar daños a su sistema operativo. Proceda como se señala a continuación.

```
# apt-get install python2.4 python2.4-minimal python2.4-dev
python-virtualenv python-setuptools
$ virtualenv -p python2.4 python2.4/
$ cd python2.4/
$ source bin/activate
(python2.4)$ cd $HOME/buildouts/cenditelmultimedia
(python2.4)$ python bootstrap.py
```

El primer comando, instala las dependencias python en el sistema operativo. Si usted se encuentra bajo el sistema operativo Debian Lenny o Ubuntu Karmic Koala, no tendrá problemas de dependencias. El segundo comando, crea una jaula virtual de python en su directorio de usuario llamada python2.4, con el tercer comando entramos a ella, para activarla usamos el cuarto comando, los siguientes comandos nos llevan al entorno de desarrollo allí llamamos al interprete de python para que ejecute al archivo bootstrap.py; el cual nos dará una salida como:

```
Downloading http://pypi.python.org/packages/source/d/distribute/distribute-0.6.14.tar.
↪gz
Extracting in /tmp/tmpIUY_yz
Now working in /tmp/tmpIUY_yz/distribute-0.6.14
Building a Distribute egg in /tmp/tmptWrUVV
/tmp/tmptWrUVV/distribute-0.6.14-py2.4.egg
Creating directory '/home/victor/buildouts/tutorial/bin'.
Creating directory '/home/victor/buildouts/tutorial/parts'.
Creating directory '/home/victor/buildouts/tutorial/eggs'.
Creating directory '/home/victor/buildouts/tutorial/develop-eggs'.
Getting distribution for 'zc.buildout==1.4.3'.
Got zc.buildout 1.4.3.
Generated script '/home/victor/buildouts/tutorial/bin/buildout'.
```

Tercer Paso

```
(python2.4)$ ./bin/buildout -vNc plonesite.cfg
```

Al realizar esto, buildout ejecutará las configuraciones necesarias en el sitio para instalar los productos. A continuación vamos a ver como configurar el resto de la aplicación.

1.5.3 Configuración

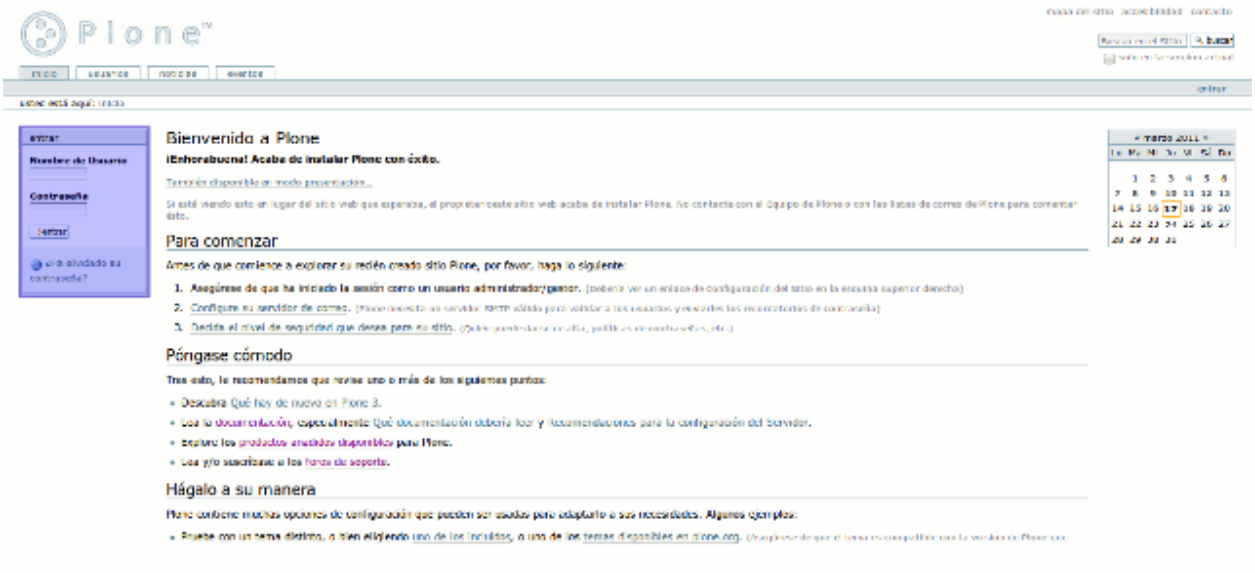
Iniciando la Instancia en Plone 3.3.5

Para poder visualizar su sitio Plone, usted debe proceder a iniciar la instancia del mismo:

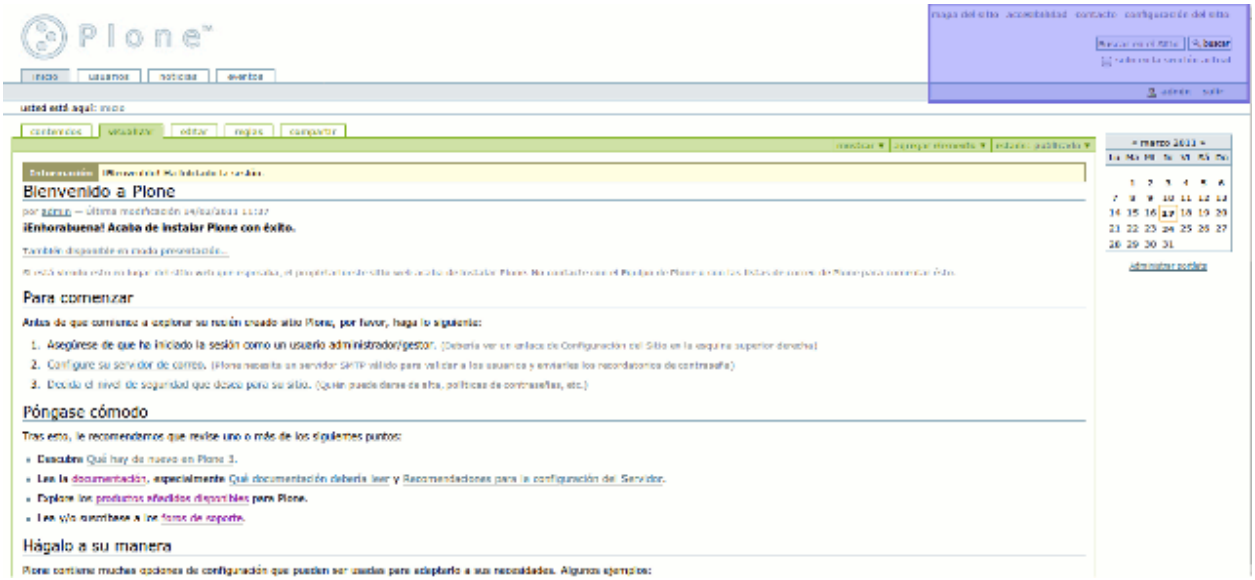
```
(python2.4)$ ./bin/instance fg
```

Una vez creado su sitio Plone mediante el uso de archivos de configuración buildout, usted podrá acceder a él mediante la siguiente dirección en su navegador web favorito (Todos digan conmigo, Firefox!!!) http://localhost:8080/demo.

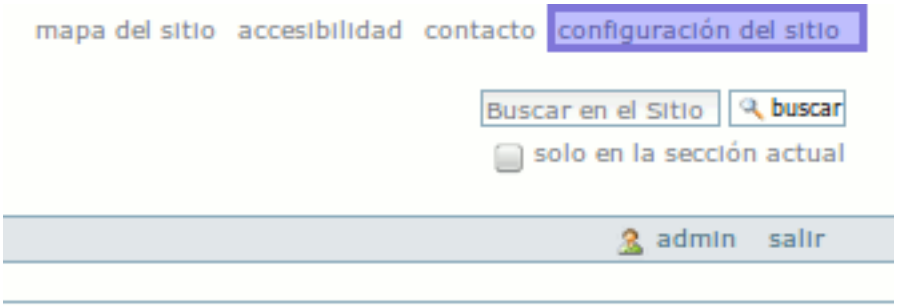
A continuación verá una página web como la que se muestra a continuación.



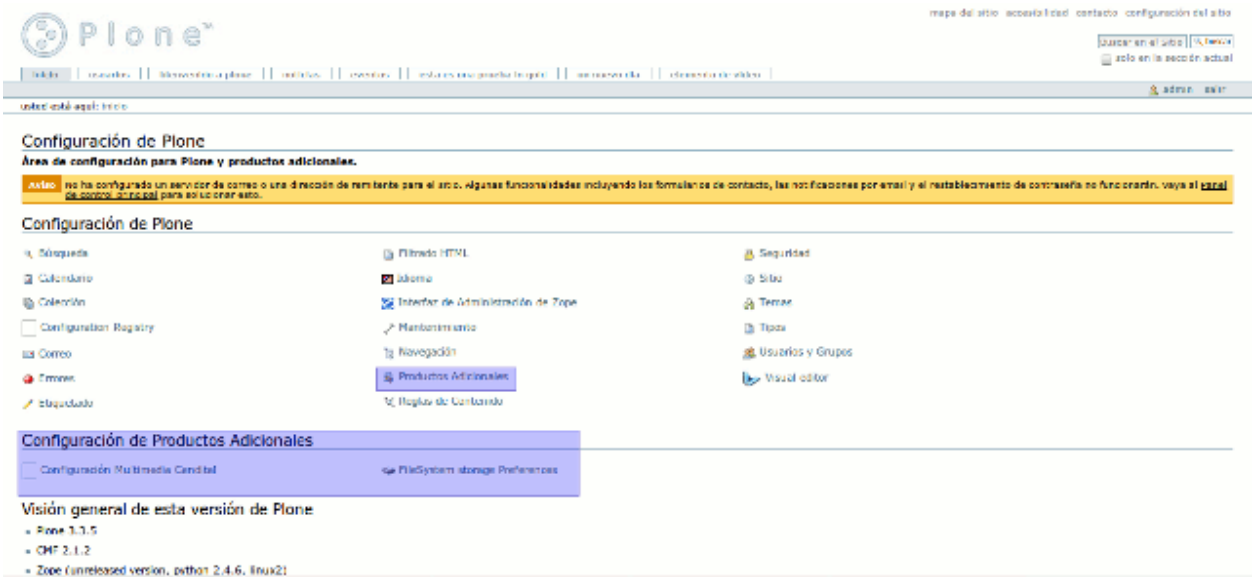
En la parte izquierda verá un ventana de identificación, por defecto tras instalar el nombre de usuario para el sitio será `admin` y la contraseña `admin`. Luego de llenar las casillas, haga clic en log in y verá la siguiente pantalla.



En la parte superior izquierda busque el botón que dice site setup, que se puede ver en la siguiente imagen.



Esto lo trasladará a la pantalla general de configuración del sitio Plone.



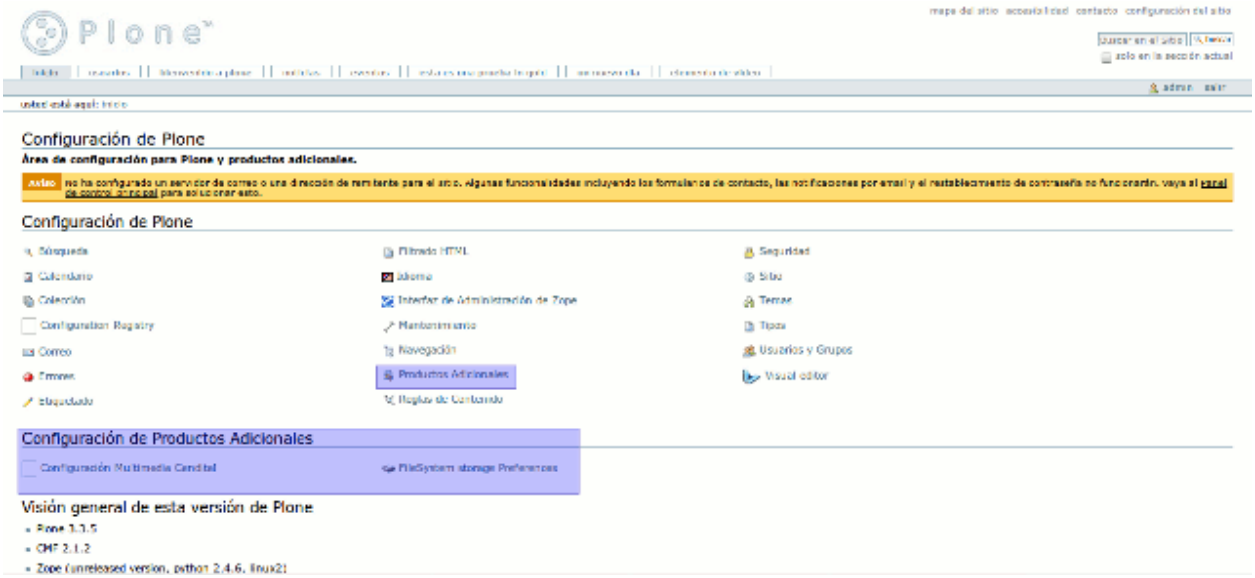
En la parte inferior, se puede observar que existen dos paneles de uno de ellos se obtendrá información que es de importancia para nosotros al momento de configurar nuestra instancia. En la parte superior de la imagen, se muestra otro enlace, que dice Productos Adicionales, en este panel podremos ver el resultado del uso del recipe Plonesite y los perfiles de instalación que este disparó durante la ejecución del script buildout.

Veamos primero el panel de productos adicionales.

Productos adicionales

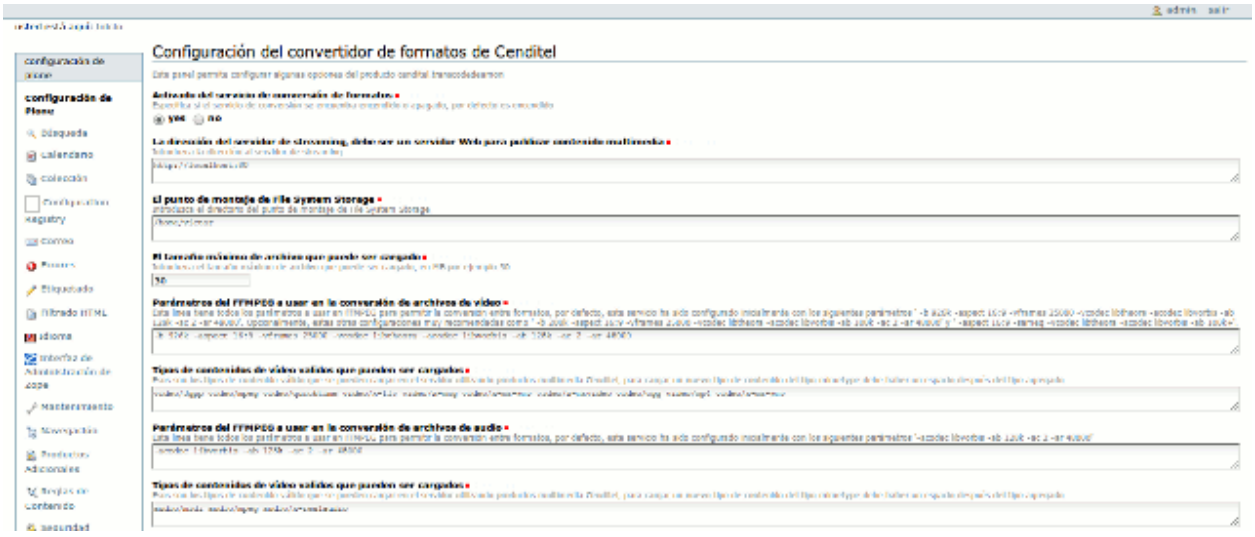
Como se puede observar en la imagen siguiente, los productos correspondientes a los tipos de contenido de audio y vídeo, el programa demonio encargado de la recodificación y el tema específico utilizado por el reproductor html5 se encuentran ya instalados en el sitio Plone.

Ahora vuelva al menú superior y haga clic en el panel de control del menú inferior que dice Configuración Multimedia Cenditel.



Paneles de configuración de los productos

Cenditel Transcode Deamon Panel



Como se puede observar en la imagen anterior, el panel consta de distintos elementos de configuración. A continuación vamos a mencionar cada uno de ellos:

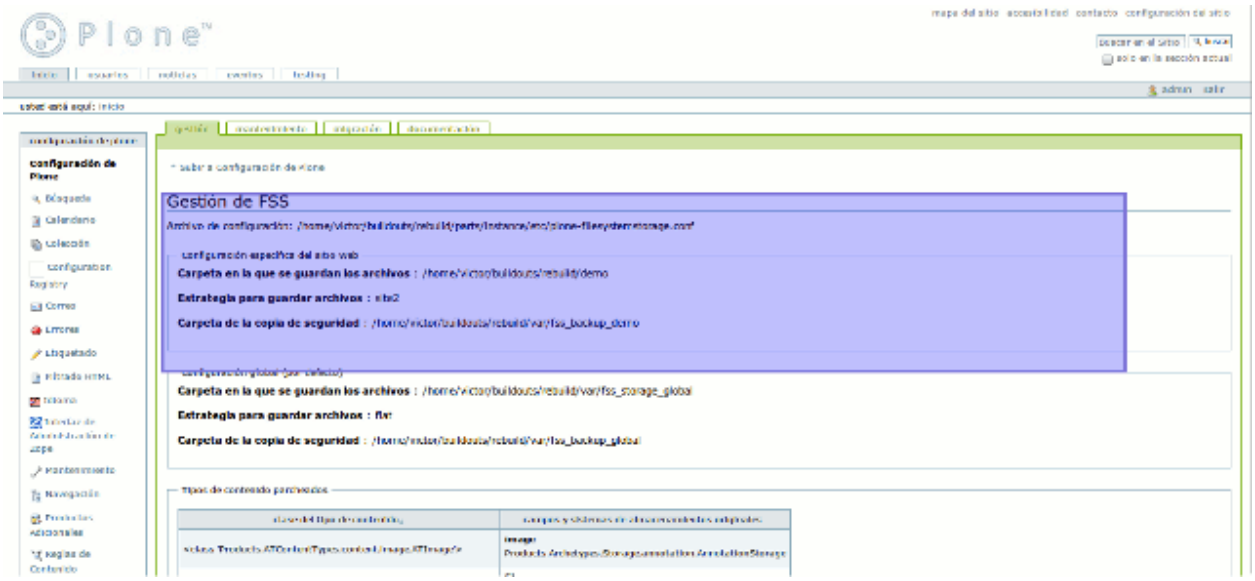
- Encendido del Convertidor de Archivos: Esta opción indica si el convertidor de archivos se encuentra activado, y por defecto se encuentra en encendido. Al estar apagado, un validador se encarga de controlar que los usuarios solo puedan cargar archivos de tipo de contenido correspondientes a formatos de audio y vídeo libres. Entiéndase vídeo vorbis theora o audio vorbis con extensión ogg.
- Dirección del Servidor Web que presta el servicio: En este caso, se hace referencia al servidor web que presta el servicio de streaming. Por defecto se encuentra configurado en http://localhost:80, puede ser cambiado por un dominio local o de Internet.
- Punto de Montaje de File System Storage: Corresponde al directorio donde File System Storage coloca los archivos del sitio. En este caso, este valor corresponde al nivel de directorios superior del cuarto parámetro, de

la variable storage, en la parte fss del archivo 05-mediafilestorage.cfg de buildout o en otras palabras dado el caso particular al directorio donde se encuentra el archivo buildout.cfg. En este campo, evite dejar espacios en blanco.

- Tamaño máximo de archivo a ser cargado: Especifica el tamaño máximo para archivos de audio o vídeo que pueden ser cargados. Por defecto, 30 MegaBytes
- Parámetros de FFMPEG a usar en la conversión de archivos de vídeo: Para mayor información visite documentación oficial de ffmpeg
- Tipos de contenidos de vídeo validos que pueden ser cargados: Corresponde a la salida del comando file -i «Archivo.old» de sistemas Unix. Si el resultado del archivo no corresponde con alguno de los siguientes, el archivo no es codificado como archivo de vídeo e incluso no puede ser subido al servidor. Estos son los admitidos por defecto:
  - video/3gpp: Información de video/3gpp en Alegsa.
  - video/mpeg: Información de video/mpeg en Alegsa.
  - video/quicktime: Información de video/quicktime en Alegsa.
  - video/x-flv: Información de video/x-flv en Alegsa.
  - video/x-mng: Información en wikipedia, soporte por remover.
  - video/x-ms-wmv: Información de video/x-ms-wmv en wikipedia.
  - video/x-msvideo: Información de video/x-msvideo en Alegsa.
  - video/ogg: Información de video/ogg en Alegsa.
  - video/mp4: Información de video/mp4 en Alegsa.
- Parámetros de FFMPEG a usar en la conversión de archivos de audio: Para mayor información visite documentación oficial de ffmpeg
- Tipos de contenidos de audio validos que pueden ser cargados: Corresponde a la salida del comando file -i «Archivo.old» de sistemas Unix. Si el resultado del archivo no corresponde con alguno de los siguientes, el archivo no es codificado como archivo de audio e incluso no puede ser subido al servidor. Estos son los admitidos por defecto:
  - audio/midi: Información de audio/midi en Alegsa
  - audio/mpeg: Información de audio/mpeg en Alegsa
  - audio/x-realaudio: Información de audio/x-realaudio en wikipedia
  - application/octec-stream: Referente a algunos archivos mp3 de baja calidad.

Como se mencionó anteriormente, para llenar el campo del punto de montaje de File System Storage es necesario tener cierta información, veamos entonces el otro panel correspondiente a `FileSystem Storage Preferences`.

File System Storage Preferences



Como se puede observar en la imagen, el campo `Carpeta donde se guardan los archivos` corresponde a un directorio dependiente del directorio donde se encuentra el script buildout, el siguiente punto corresponde a la estrategia de almacenamiento que tiene el valor `site2` del cual se puede obtener más información en la página oficial de FSS

A continuación veamos el archivo de salida generado para el servidor nginx durante la ejecución de buildout. Este archivo, contiene la configuración necesaria para permitir el acceso a nuestros vídeos en un directorio demo. Usando la normativa location de nginx que apunta a nuestro directorio buildout y que debe coincidir con el directorio donde esta apuntando la configuración de File System Storage.

En este caso:

```
$ gedit etc/nginx/sites-enabled/demo
```

Abrirá el siguiente archivo:

```
server {
        # DNS/IP y Puerto en que escucha la aplicación
        listen   * :80;

        # Nombre del servidor
        server_name  192.168.12.215;

        # Tamaña máximo de subida de archivos
        client_max_body_size 24M;

        # Tamaña máximo de buffer de archivos
        client_body_buffer_size 128K;

        # Archivo de registro de acceso del sitio web
        access_log  /var/log/nginx/demo.access.log;

        # Archivo de registro de error del sitio web
        error_log  /var/log/nginx/demo.error.log error;
```

(continué en la próxima página)

(proviene de la página anterior)

```
        # Interfaz Administrativa de Zope
        location /manage {
                proxy_pass       http://192.168.12.215:8080/VirtualHostBase/
                http/192.168.12.215:80/manage_main/VirtualHostRoot/;

                proxy_set_header Host $host;
        }

        # Sitio Proyecto Canaima
        location / {
                proxy_pass       http://192.168.12.215:8080/VirtualHostBase/
                http/192.168.12.215:80/demo/VirtualHostRoot/;

                proxy_set_header Host $host;
        }

        # Sitio de publicación de archivos para Streaming
        location /demo {
                root /home/victor/buildouts/cenditelmultimedia;
                autoindex on;
        }

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
                root   /var/www/nginx-default;
        }

}
```

Para habilitar la carpeta, sencillamente se necesita realizar un enlace simbólico desde `etc/nginx/sites-enabled/demo` a `/etc/nginx/sites-enabled` de la siguiente manera.

```
# ln -s etc/nginx/sites-enabled/demo /etc/nginx/sites-enabled
```

Configuración de permisos

Se recomienda dar una configuración de permisos que permita la lectura y escritura de archivos en la carpeta que prestará el servicio de streaming a través de nginx. Por tanto, para la carpeta `/home/usuario/buildout/directory/demo` debe permitir la lectura, escritura y ejecución por parte del dueño y la lectura por parte de cualquier otro. Para lo cual se recomienda aplicar el siguiente comando:

```
$ chmod -R 755 /home/usuario/buildout/directory/demo
```

Para asignar los siguientes permisos:

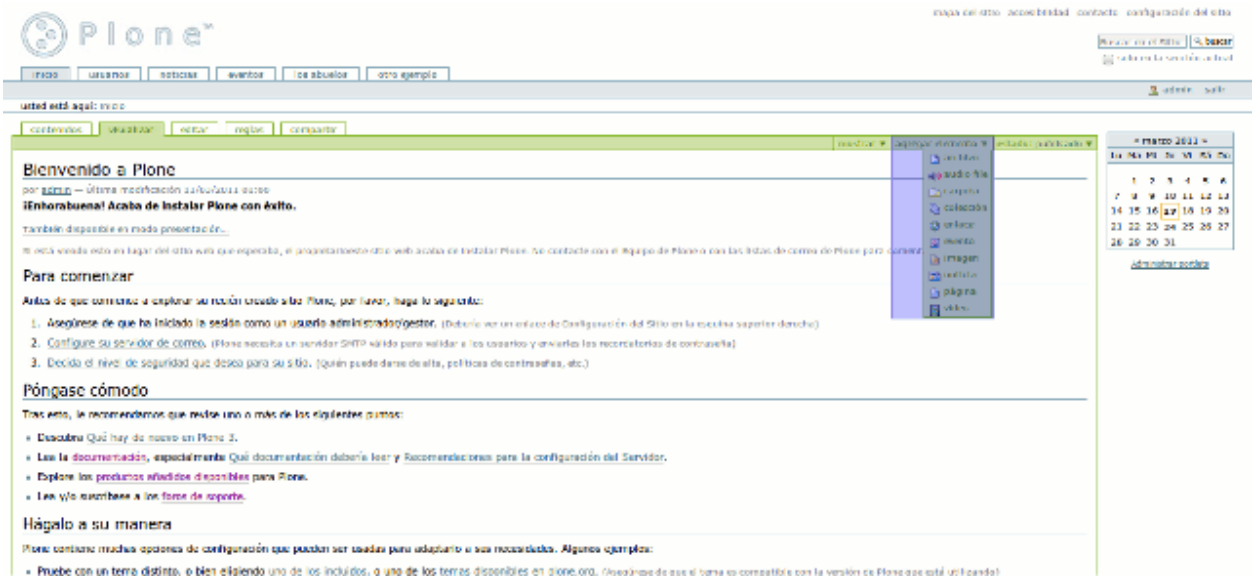
- Dueño: Lectura, Escritura, Ejecución.
- Grupo: Lectura y Ejecución.
- Otros: Lectura y Ejecución.

De lo contrario, las solicitudes realizadas al servidor nginx devolverán un error de acceso denegado a nivel de log en `/var/log/nginx/error.log`.

Una vez realizadas las configuraciones, se puede proceder a agregar contenido de ejemplo.

1.5.4 Guía de Uso

Para agregar contenido en su sitio Plone, vuelva a la página principal http://localhost:8080/demo.

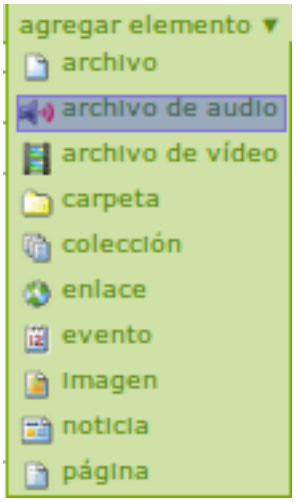


Como se puede ver en la imagen anterior, al hacer clic en el botón agregar elemento se despliega una serie opciones de tipos de contenido que pueden ser agregados, pero para este sitio demostrativo y el presente manual, solo dos son de interés.

Archivos de Audio

A continuación se observa una serie de pasos para agregar tipos de contenido «Archivos de Audio»:

- Uno: desplace su mouse hasta el elemento que se señala en la imagen:



- Dos: haga clic en él, e inmediatamente será enviado a la siguiente pantalla.

Allí debe llenar un total de tres campos, de los cuales dos son obligatorios:

- Campo Titulo: El campo es obligatorio.
- Campo Descripción: El campo es opcional, pero se recomienda su llenado con una breve descripción del contenido.
- Campo Audio: Este es el campo donde será cargado el archivo. Aplican ciertas restricciones que se presentan a continuación:
  - Tamaño máximo de archivo: El administrador del sitio configura un tamaño máximo en Megabytes para los archivos que pueden ser subidos al servidor, en caso de que el archivo sea demasiado grande el usuario será advertido de esto, mediante un mensaje como el que sigue:

    Audio ▪ (Obligatorio)
    El archivo de audio ha sido cargado
    ¡Fallo validación! (data subida es demasiado grande: su tamaño actual es 15.609MB (máximo permitido 10.000MB)
    Seleccionar archivo No se eligi...gún archivo

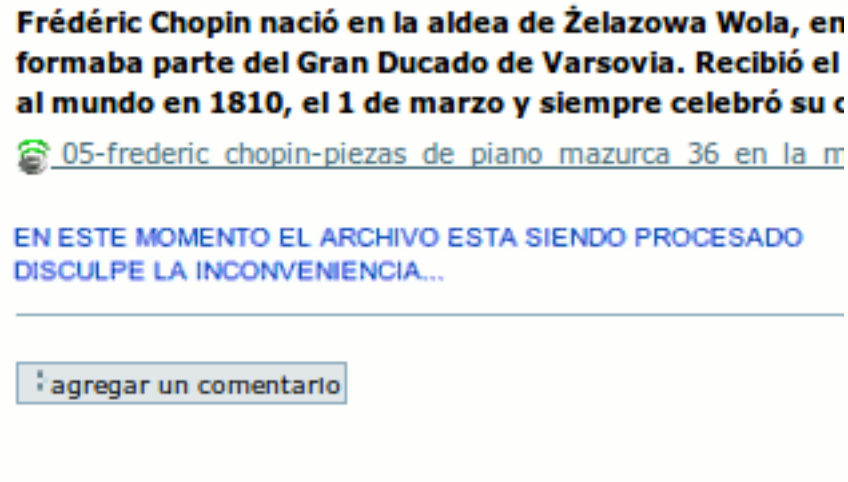
  - Tipo de archivo cargado: El administrador define los tipos de archivos que pueden ser subidos al servidor, en caso de que el archivo cargado no coincida con ninguna de las restricciones el usuario verá un mensaje como el que se presenta a continuación:

    Audio ▪ (Obligatorio)
    El archivo de audio ha sido cargado
    El archivo tiene que ser uno de los siguientes formatos de audio 'audio/midi, audio/mpeg, audio/x-realaudio'.
    Seleccionar archivo No se eligi...gún archivo

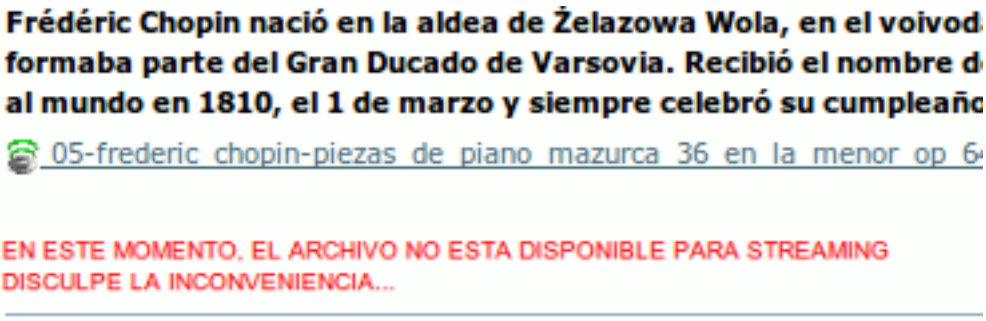
  - Estado del convertidor de archivos: En este caso, si el administrador desactiva el convertidor de archivos se activa una restricción para los tipos de contenidos que pueden ser cargados. En caso de que eso suceda, la notación de archivos corresponden a archivos en formatos libres ogg vorbis y solo los siguientes tipos de archivos pueden ser cargados al servidor:
    - audio/ogg
    - audio/x-theora+ogg
    - application/ogg
  - Campo vacío: en caso del campo Audio encontrarse vacío se le notificará al usuario acerca del evento.

Una vez llenados los tres campos, se debe presionar el botón guardar. Luego, el archivo será subido al servidor y se pueden presentar cuatro vistas distintas según lo que suceda:

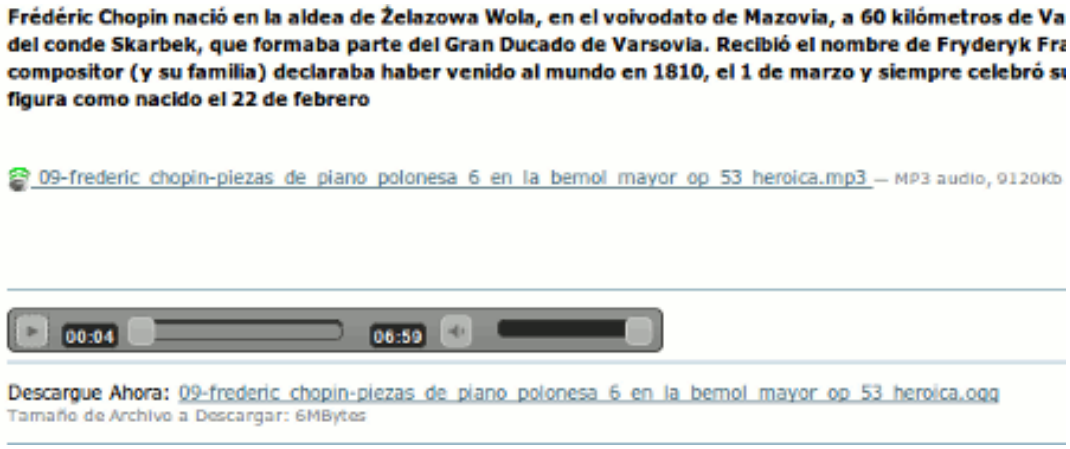
- Vista de Espera: Cuando un archivo esta en cola de espera, o al momento de ser convertido, al usuario le será mostrado un mensaje como el que sigue:



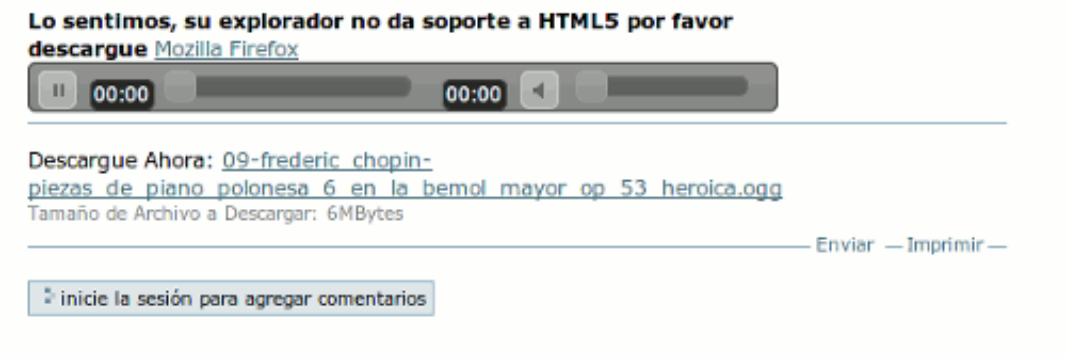
- Vista de Error: Esta vista se presenta en casos en que no se realice la conversión del archivo, se realice parcialmente generando un archivo corrupto o el archivo al que se le va a realizar streaming no exista.



- Vista de Streaming: Presenta un visor para HTML5 que permitirá la reproducción del contenido y el control de volumen del mismo.



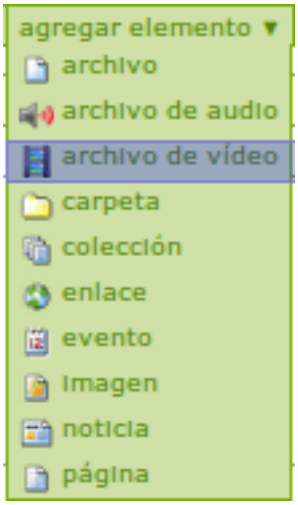
- Vista de Navegador Invalido: Se muestra en caso de que el navegador no de soporte al lenguaje HTML5.



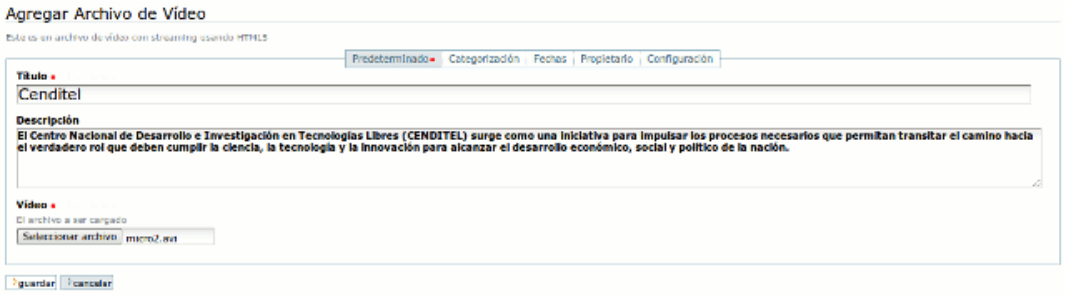
Archivos de Vídeo

A continuación se observa una serie de pasos para agregar tipos de contenido «Archivos de Vídeo»:

- Uno: desplace el cursor del mouse hasta el elemento que se señala en la imagen:



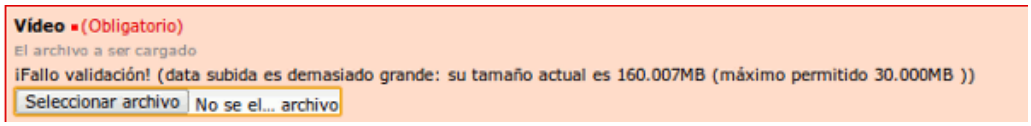
- Dos: haga clic en él, e inmediatamente será enviado a la siguiente pantalla.



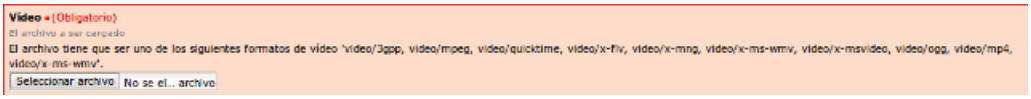
Allí debe llenar un total de tres campos, de los cuales dos son obligatorios:

- Campo Titulo: El campo es obligatorio.
- Campo Descripción: El campo es opcional, pero se recomienda su llenado con una breve descripción del contenido.
- Campo Vídeo: Este es el campo donde será cargado el archivo. Aplican ciertas restricciones que se presentan a continuación:

- Tamaño máximo de archivo: El administrador del sitio configura un tamaño máximo en Megabytes para los archivos que pueden ser subidos al servidor, en caso de que el archivo sea demasiado grande el usuario será advertido de esto, mediante un mensaje como el que sigue:



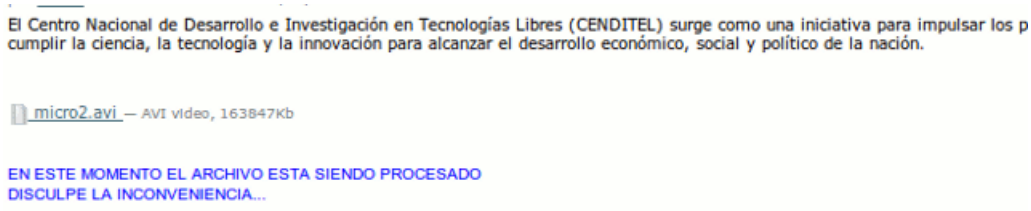
- Tipo de archivo cargado: El administrador define los tipos de archivos que pueden ser subidos al servidor, en caso de que el archivo cargado no coincida con ninguna de las restricciones el usuario verá un mensaje como el que se presenta a continuación:



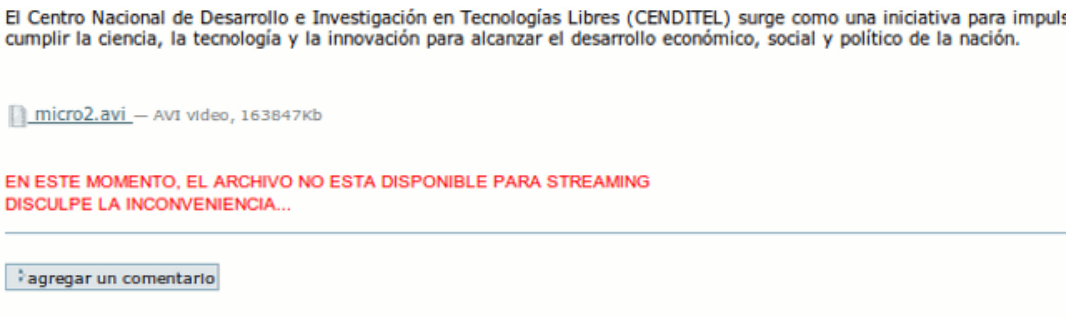
- Estado del convertidor de archivos: En este caso, si el administrador desactiva el convertidor de archivos se activa una restricción para los tipos de contenidos que pueden ser cargados. En caso de que eso suceda, la notación de archivos corresponden a archivos en formatos libres ogg vorbis y solo los siguientes tipos de archivos pueden ser cargados al servidor:
  - video/ogg
  - video/x-theora+ogg
  - application/ogg
- Campo vacío: en caso del campo Vídeo encontrarse vacío se le notificará al usuario acerca del evento.

Una vez llenados los tres campos, se debe presionar el botón guardar. Luego, el archivo será subido al servidor y se pueden presentar cuatro vistas distintas según lo que suceda:

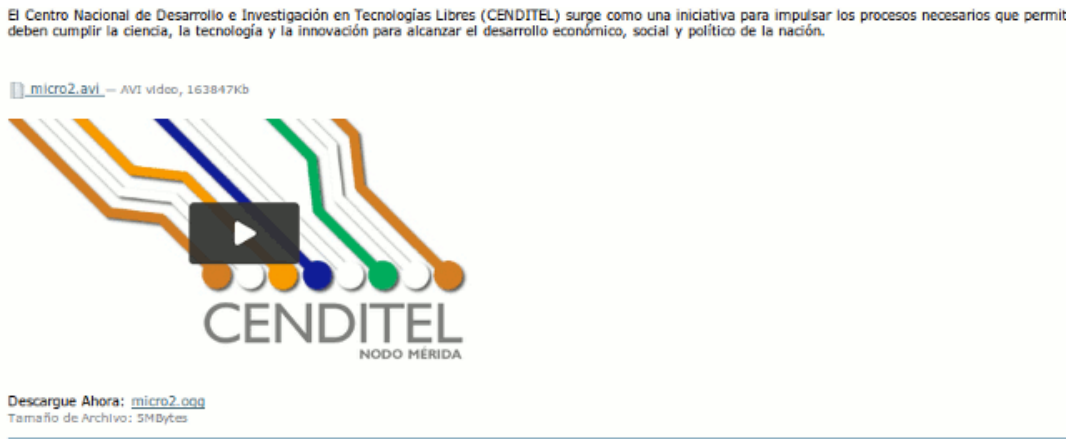
- **Vista de Espera: Cuando un archivo esta en cola de espera, o al momento de** ser convertido, al usuario le será mostrado un mensaje como el que sigue:



- Vista de Error: Esta vista se presenta en casos en que no se realice la conversión del archivo, se realice parcialmente generando un archivo corrupto o el archivo al que se le va a realizar streaming no exista.



- Vista de Streaming: Presenta un visor para HTML5 que permitirá la reproducción del contenido y el control de volumen del mismo.



- Vista de Navegador Invalido: Se muestra en caso de que el navegador no de soporte al lenguaje HTML5.

1.5.5 Flujos de Trabajos

Por defecto, los productos de multimedia de la fundación Cenditel no posee su propio flujo de trabajo al estilo de Plone.

cenditel.transcodedaemon

Este producto posee su propio flujo de trabajo que se basa en teoría de colas, usa la disciplina primero en entrar, primero en salir (FIFO). El flujo de trabajo, se encuentra estrechamente relacionado a los productos `cenditel.audio` y `cenditel.video` en los métodos de PlayingAudioType and PlayingVideoType de los módulos audioview.py y videoview.py de los paquetes browser.

De manera general, los métodos mencionados anteriormente obtienen una serie de configuraciones del panel de control de la aplicación los modifica parcialmente en caso de ser necesario eliminando los simbolos / sobrantes, realizando concatenaciones entre las configuraciones, URL del sitio, y la porción de URL que sera utilizada por los nuevos archivos codificados en formatos libres.

Origen de los datos

En el caso del producto `cenditel.audio`, el método `PlayingAudioType` que pertenece a la clase `audioView` del modulo `audioview.py` en el paquete browser, es el encargado de obtener los datos que serán manipulados por `cenditel.transcodedaemon`. A continuación se desglosa dicho método.

```
def PlayingAudioType(self):

        """
        Primero vamos  a obtener la configuración desde el panel de control:
        para eso, aplicamos la documentación de Plone.app.registry
        que puede ser vista en https://pypi.org/project/plone.app.registry

            >>> registry = getUtility(IRegistry)
            >>> settings = registry.forInterface(ITranscodeSetings)
            >>> self.SERVER = self.RemoveSlash(settings.adress_of_streaming_server)
            >>> VIDEO_PARAMETRES_TRANSCODE = settings.ffmpeg_parameters_video_line
            >>> AUDIO_PARAMETRES_TRANSCODE = settings.ffmpeg_parameters_audio_line
            >>> audio_content_types=settings.audio_valid_content_types
            >>> video_content_types=settings.video_valid_content_types

        De esta manera, hacemos un llamado a los distintos registros almacenados
        por el panel de configuración que nos serán útiles en este caso.
        Ahora, se declaran variables que son de interés para entender el contexto
        de la aplicación:

           >>> self.MyTitle = self.context.Title()
           >>> idvideo=self.context.getId()
           >>> url = self.context.absolute_url()
           >>> self.PathOfFile = MFNI.ReturnPathOfFile(url)
           >>>virtualobject=self.context.getVideo()
           >>>self.filenamesaved=virtualobject.filename
           >>> self.extension=MTDI.CheckExtension(self.filenamesaved)

        Para entender de mejor manera el código. Verifique la documentación de
        los métodos en el los script asociados a ellos.
        Las importaciones e instancias se encuentran en el modulo videoview.py.
        A continuación, se verifica si el archivo que fue cargado por el usuario
        tiene extensión ogg la cual corresponde a formatos de archivo basados en
        estándares libres.

            >>> if self.extension=="ogg":
                ... self.folderfileOGG=self.PathOfFile+'/' + quote(self.
↪filenamesaved)
                ... self.prefiletranscoded=self.STORAGE+self.PathOfFile+'/'+self.
↪filenamesaved

        En caso de cumplirse la condición, se crea una nueva variable llamada
        FolderFile que posee parte de la URL del servidor de streaming, y otra
        variable que por su nombre indica que el archivo ha sido precodificado
        por el usuario desde antes de ser cargado al servidor.
        A continuación se verifica si ese archivo existe, y en caso de ser verdad,
        revisa si no existe en la lista de archivos disponibles, de ser así
        se agrega a dicha lista, en caso contrario se completa la ejecución
        del método y por tanto el flujo del transcode ya que no fue necesario
        convertir el archivo y puede ser publicado inmediatamente.
```

(continué en la próxima página)

(proviene de la página anterior)

```
            >>> if path.isfile(self.prefiletranscoded)==True:
                    ... self.StatusOfFile=ServiceList.available(idvideo,self.
↪prefiletranscoded)
                    ... if self.StatusOfFile == False:
                    ....... ServiceList.AddReadyElement(idaudio,self.
↪prefiletranscoded)
                    ....... self.StatusOfFile=True
                    ....... ServiceList.SaveInZODB()
                    ....... self.AbsoluteServerPath = self.SERVER + self.folderfileOGG
                    ... else:
                    ....... self.AbsoluteServerPath = self.SERVER + self.folderfileOGG

        En caso contrario a que la extensión del archivo subido por el usuario
        sea ogg, se dispara la ejecución de un método que se encargará de
        registrar el archivo en lista de espera y almacenar en cenditelmultimedia.fs
        datos que no son accesibles desde el panel de control que pueden ser
        usados por el convertidor de formatos. También se declaran otras
        variables que son utilizadas a nivel de vista de usuario para la
        presentación del contenido, para información acerca de los métodos
        revise la documentación de los mismos en el código correspondiente.

        >>> else:
            newtrans_init_(self.STORAGE,
                           self.PathOfFile,
                           self.filenamesaved,
                           idvideo,
                           VIDEO_PARAMETRES_TRANSCODE,
                           AUDIO_PARAMETRES_TRANSCODE,
                           audio_content_types,
                           video_content_types)
            self.folderfileOGG=MTDI.newname(self.PathOfFile+'/' + self.filenamesaved)
            self.AbsoluteServerPath = self.SERVER + MTDI.nginxpath(self.folderfileOGG)
            self.newfiletranscoded=MTDI.nginxpath(self.STORAGE+self.folderfileOGG)
            self.StatusOfFile = ServiceList.available(idvideo, self.newfiletranscoded)

        La ultima sección del método, verifica el valor de una variable bandera
        que revisa si el archivo se encuentra disponible. En caso contrario devuelve
        un error.
            >>> if self.StatusOfFile == True:
                ... self.newfilename=MTDI.newname(self.filenamesaved)
            ... else:
                ... self.newfilename=_('The file is not ready yet, please contact
                site administration')
        """

        registry = getUtility(IRegistry)
        settings = registry.forInterface(ITranscodeSetings)
        self.SERVER = self.RemoveSlash(settings.adress_of_streaming_server)
        VIDEO_PARAMETRES_TRANSCODE = settings.ffmpeg_parameters_video_line
        AUDIO_PARAMETRES_TRANSCODE = settings.ffmpeg_parameters_audio_line
        audio_content_types=settings.audio_valid_content_types
        video_content_types=settings.video_valid_content_types
        self.STORAGE = self.RemoveSlash(settings.mount_point_fss)
        self.MyTitle = self.context.Title()
        idvideo=self.context.getId()
        url = self.context.absolute_url()
        self.PathOfFile = MFNI.ReturnPathOfFile(url)
```

(continué en la próxima página)

(proviene de la página anterior)

```
        virtualobject=self.context.getVideo()
        self.filenamesaved=virtualobject.filename
        self.extension=MTDI.CheckExtension(self.filenamesaved)
        if self.extension=="ogg":
            self.folderfileOGG=self.PathOfFile+'/' + quote(self.filenamesaved)
            self.prefiletranscoded=self.STORAGE+self.PathOfFile+'/'+self.filenamesaved
            if path.isfile(self.prefiletranscoded)==True:
                self.StatusOfFile=ServiceList.available(idvideo,self.
↪prefiletranscoded)
                if self.StatusOfFile == False:
                    ServiceList.AddReadyElement(idaudio,self.prefiletranscoded)
                    ServiceList.SaveInZODB()
                    self.AbsoluteServerPath = self.SERVER + self.folderfileOGG
                else:
                    self.AbsoluteServerPath = self.SERVER + self.folderfileOGG
            else:
                print _("File not found "+self.prefiletranscoded)
                self.Error=True
                self.ErrorSituation()
        else:
            newtrans_init_(self.STORAGE,
                           self.PathOfFile,
                           self.filenamesaved,
                           idvideo,
                           VIDEO_PARAMETRES_TRANSCODE,
                           AUDIO_PARAMETRES_TRANSCODE,
                           audio_content_types,
                           video_content_types)
            self.folderfileOGG=MTDI.newname(self.PathOfFile+'/' + self.filenamesaved)
            self.AbsoluteServerPath = self.SERVER + MTDI.nginxpath(self.folderfileOGG)
            self.newfiletranscoded=MTDI.nginxpath(self.STORAGE+self.folderfileOGG)
            self.StatusOfFile = ServiceList.available(idvideo, self.newfiletranscoded)
            if self.StatusOfFile == True:
                self.newfilename=MTDI.newname(self.filenamesaved)
            else:
                self.newfilename=_('The file is not ready yet, please contact site
                administration')
        return
```

Registro en espera

Como se mencionó en la sección anterior, cuando un archivo subido no corresponde a un archivo ogg dicho archivo es registrado en espera para posteriormente ser codificado según la posición en la cola. En otras palabras, imagine la cola de un banco, donde usted entra y espera su turno, luego es atendido, y posteriormente sale del banco. El sistema de conversión funciona de igual manera.

Ahora se va a analizar, el método `newtrans_init_` que es el encargado de registrar los elementos en la lista de espera.

```
"""
Como se puede observar a continuación el método recibe los siguientes parámetros:

* STORAGE: Es la URL al directorio raíz del elemento.
* path: Corresponde a la URL del elemento y es donde se guarda el archivo
original en el disco duro del servidor.
```

(continué en la próxima página)

(proviene de la página anterior)

```
* filenamesaved: Nombre del archivo guardado originalmente.
* idfile: Identificador del Elemento en el sitio Plone
* VIDEO_PARAMETRES_TRANSCODE: Los datos de configuración del elemento vídeo
en el panel de control.
* AUDIO_PARAMETRES_TRANSCODE: Los datos de configuración del elemento vídeo
el panel de control.
* audio_content_types: Corresponde a los mimetypes para archivos de audio
validos en el panel de control.
* video_content_types: Corresponde a los mimetypes para archivos de vídeo
validos en el panel de control.
La primera linea, crea una variable que concatena los valores especificados
anteriormente luego, esta es modificada por un método
para entender el funcionamiento de este, vea la documentación respectiva en
el código fuente del respectivo paquete.
>>> PathToOriginalFile = STORAGE + path +'/'+ filenamesaved
>>> newfolderfile=MTD.nginxpath(PathToOriginalFile)

Las siguientes lineas, verifican si el valor de las variables en el panel de
control corresponden a las variables almacenadas en la Base de Datos orientada
a objetos del producto, y en caso de no ser de esa manera, cambian el valor␣
↪almacenado.

La siguiente linea, verifica si el archivo no esta en la lista de espera
(resultado del método uploaded) y no se encuentra tampoco en la lista de archivos
disponibles (resultado del método available) y no se encuentra siendo codificado en␣
↪el momento
(resultado del método transcoding)y en el caso de ser negativas todas las condiciones
se el archivo se registrá en la lista de espera.

>>> if ServiceList.uploaded(idfile, PathToOriginalFile)== False and\
ServiceList.available(idfile, newfolderfile)== False\
and ServiceList.transcoding(PathToOriginalFile)== False:
    ... ServiceList.RegisterWaitingFile(idfile, PathToOriginalFile)

La ultima seccion de codigo de este metodo, se encarga de disparar un sub
proceso basado en programación multihilos que se ejecutará siempre que no se
este convirtiendo ningun archivo en el momento.

>>> import threading
>>> if ServiceList.CurrentTranscoding()=="":
    ... class MyThread(threading.Thread):
    ... def run(self):
            transcodedaemon()
            MyThread().start()
>>> return
"""
def newtrans_init_(STORAGE, path, filenamesaved,\
                   idfile, VIDEO_PARAMETRES_TRANSCODE,\
                   AUDIO_PARAMETRES_TRANSCODE,\
                   audio_content_types, video_content_types):
        PathToOriginalFile = STORAGE + path +'/'+ filenamesaved
        newfolderfile=MTD.nginxpath(PathToOriginalFile)
        if ServiceList.CheckItemZODB('waiting')==False:
                ServiceList.AddObjectZODB('waiting',[])
        if ServiceList.CheckItemZODB('current')==False:
                ServiceList.AddObjectZODB('current','')
                ServiceList.SaveInZODB()
```

(continué en la próxima página)

(proviene de la página anterior)

```
        if ServiceList.CheckItemZODB('ready')==False:
                ServiceList.AddObjectZODB('ready',[])
                ServiceList.SaveInZODB()
        if ServiceList.CheckItemZODB('Video_Parameters')==False:
                ServiceList.AddObjectZODB('Video_Parameters', VIDEO_PARAMETRES_
↪TRANSCODE)
                ServiceList.SaveInZODB()
        if ServiceList.CheckItemZODB('Audio_Parameters')==False:
                ServiceList.AddObjectZODB('Audio_Parameters', AUDIO_PARAMETRES_
↪TRANSCODE)
                ServiceList.SaveInZODB()

        if ServiceList.CheckItemZODB('Video_ContentTypes')==False:
                ServiceList.AddObjectZODB('Video_ContentTypes', video_content_types)
                ServiceList.SaveInZODB()
        if ServiceList.CheckItemZODB('Audio_ContentTypes')==False:
                ServiceList.AddObjectZODB('Audio_ContentTypes', audio_content_types)
                ServiceList.SaveInZODB()

        if ServiceList.root['Audio_Parameters']!=AUDIO_PARAMETRES_TRANSCODE:
                ServiceList.root['Audio_Parameters']=AUDIO_PARAMETRES_TRANSCODE
                ServiceList.SaveInZODB()
        if ServiceList.root['Video_Parameters']!=VIDEO_PARAMETRES_TRANSCODE:
                ServiceList.root['Video_Parameters']=VIDEO_PARAMETRES_TRANSCODE
                ServiceList.SaveInZODB()

        if ServiceList.root['Video_ContentTypes']!=video_content_types:
                ServiceList.root['Video_ContentTypes']=video_content_types
                ServiceList.SaveInZODB()

        if ServiceList.root['Audio_ContentTypes']!=audio_content_types:
                ServiceList.root['Audio_ContentTypes']=audio_content_types
                ServiceList.SaveInZODB()

        if ServiceList.uploaded(idfile, PathToOriginalFile)== False and \
        ServiceList.available(idfile, newfolderfile)== False and \
        ServiceList.transcoding(PathToOriginalFile)== False:
                ServiceList.RegisterWaitingFile(idfile, PathToOriginalFile)
        import threading
        if ServiceList.CurrentTranscoding()=="":
                class MyThread(threading.Thread):
                        def run(self):
                                transcodedaemon()
                MyThread().start()
        return
```

Prestación del servicio

Una vez que el archivo cliente ha sido registrado en la lista de espera, y no existe ningún elemento siendo codificado en el momento, se dispara el siguiente método que es el encargado de extraer los archivos en espera y pasar la URL en el disco duro a una instancia de `ffmpeg` que se encargará de crear un archivo de salida con un nuevo formato libre que permita su utilización usando html5.

```
def transcodedaemon():
    """
```

(continué en la próxima página)

(proviene de la página anterior)

```
    La primera linea, obtiene la cantidad de registros en espera.
    Si la cantidad es mayor a 0.

    >>> while(ServiceList.FileWaitings()>0):

    Las siguientes lineas extraen la dirección del primer archivo

        ... element=ServiceList.WaitingElement()
        ... listpath=element.values()
        ... PathToOriginalFile=listpath[0]

    Por otro lado, reseteando la variable se extrae el identificador del archivo.

        ... listpath=''
        ... listpath=element.keys()
        ... idfile=listpath[0]

    A continuación se verifica si el archivo existe, de ser así se elimina el elemento
    de la lista de espera, se agrega la url del a una variable que controla
    el flujo en el codificador se dispara el método ``transcode`` que recibe
    una serie de parámetros que serán explicados a continuación.

    >>> if os.path.isfile(PathToOriginalFile):

                ... ServiceList.DeleteElement(idfile, PathToOriginalFile)
                ... ServiceList.AddActiveTranscoding(PathToOriginalFile)
                ... PathToTranscodedFile = MTD.transcode(PathToOriginalFile,
                                                         ServiceList.root['Video_Parameters'],
                                                         ServiceList.root['Audio_Parameters'],
                                                         ServiceList.root['Video_ContentTypes
↪'],
                                                         ServiceList.root['Audio_ContentTypes
↪'],)

    * ServiceList.root['Video_Parameters']: Parámetros de codificación de archivos
        de vídeo.
    * ServiceList.root['Audio_Parameters']: Parámetros de codificación de archivos
        de audio.
    * ServiceList.root['Video_ContentTypes']: Tipos de contenido de vídeo validos.
    * ServiceList.root['Audio_ContentTypes']: Tipos de contenido de audio validos.

   Luego, es eliminado el elemento de la variable de control de codificación y luego
   es pasada a una lista de elementos guardados y se guarda la información en
   la base de datos del producto.

    >>> ServiceList.RemoveActiveTranscoding()
    >>> ServiceList.AddReadyElement(idfile, PathToTranscodedFile)
    >>> ServiceList.SaveInZODB()

    En caso de que la condición de existencia del archivo no se cumpla,
    Se elimina el elemento de la lista de espera y se manda un log de elemento
    no encontrado.

    ... else:
        ... ServiceList.DeleteElement(idfile, PathToOriginalFile)
        ... print "NOT FOUND "+ PathToOriginalFile
```

(continué en la próxima página)

(proviene de la página anterior)

```
        ... ServiceList.SaveInZODB()
    """
    while(ServiceList.FileWaitings()>0):
            element=ServiceList.WaitingElement()
            listpath=element.values()
            PathToOriginalFile=listpath[0]
            listpath=''
            listpath=element.keys()
            idfile=listpath[0]
            if os.path.isfile(PathToOriginalFile):
                    ServiceList.DeleteElement(idfile, PathToOriginalFile)
                    ServiceList.AddActiveTranscoding(PathToOriginalFile)
                    PathToTranscodedFile = MTD.transcode(PathToOriginalFile,
                                                         ServiceList.root['Video_Parameters'],
                                                         ServiceList.root['Audio_Parameters'],
                                                         ServiceList.root['Video_ContentTypes
↪'],
                                                         ServiceList.root['Audio_ContentTypes
↪'],)
                    ServiceList.RemoveActiveTranscoding()
                    ServiceList.AddReadyElement(idfile, PathToTranscodedFile)
                    ServiceList.SaveInZODB()
            else:
                    ServiceList.DeleteElement(idfile, PathToOriginalFile)
                    print "NOT FOUND "+ PathToOriginalFile
                    ServiceList.SaveInZODB()
    print "Daemon is waiting for File"
    return
```

Una vez finalizado los procesos, el archivo queda disponible para streaming. Ya sea en el caso de los archivos de audio o de los archivos de vídeo.

1.5.6 Integración con cyn.in

Primer Paso

Descargar de la pagina Web del Proyecto el script de instalación buildout que le permitirá configurar un sitio demostrativo. Para realizar la descarga proceda en un terminal con los siguientes comandos:

```
$ mkdir buildouts
$ cd buildouts
$ svn co http://plataforma.cenditel.gob.ve/svn/plataforma/proyectosInstitucionales/
renasen/cenditel.multimedia/buildout/cynin313 cenditelmultimedia
$ cd cenditelmultimedia
$ ls -p
00-varibles.cfg           06-contenttypes.cfg     plonesite.cfg
01-dumpedversions.cfg     bootstrap.py            products/
02-mrdeveloper.cfg        buildout.cfg            src/
03-prerequemients.cfg     cenditelmultimedia.cfg  templates/
04-mountpoint.cfg         dumped-versions.cfg     versions.cfg
05-mediafilesstorage.cfg  etc/                    user.cfg
```

Segundo paso

Instale una jaula de `python2.4` en su sistema para evitar daños a su sistema operativo. Proceda como se señala a continuación.

```
$ sudo aptitude install python2.4 python2.4-minimal python2.4-dev
python-virtualenv python-setuptools
$ virtualenv -p python2.4 python2.4/
$ cd python2.4/
$ source bin/activate
(python2.4)$ cd $HOME/buildouts/cenditelmultimedia
(python2.4)$ python bootstrap.py
(python2.4)$ pip install ZopeSkel lxml python-ldap
(python2.4)$ easy_install-2.4 -i http://dist.serverzen.com/pypi/simple PILwoTk
```

Tercer Paso

Edite el archivo `user.cfg` y cambie la variable `effective-user` de manera que el valor corresponda con el nombre de usuario de su sesión Unix.

```
(python2.4)$ ./bin/buildout -vNc cenditelmultimedia.cfg
```

Al realizar esto, buildout ejecutará las configuraciones necesarias en la instancia para instalar los productos. Una vez finalizado, se debe iniciar la instancia para crear un sitio nuevo de demostración.

Iniciando la Instancia en Cyn.in 3.1.3:

A continuación usted debe proceder a iniciar la instancia del mismo:

```
(python2.4)$ ./bin/instance fg
```

Dirijase a la interfaz administrativa de Zope a través de http://localhost:8080/manage El nombre de usuario por defecto es `admin` y la contraseña es `secret`. En la parte superior derecha, se observa un menú desplegable. Haga clic en él y seleccione la opción que dice: `Plone site`

En la siguiente ventana, agregue un identificador en minúsculas para el sitio, dadas la configuraciones que realizo buildout en la etapa anterior, debe colocar en esta casilla el valor `demo`. Luego haga clic en aceptar. Como podrá observar ahora posee un sitio Plone básico, para continuar proceda a volver a la interfaz administrativa http://localhost:8080/manage. Haga clic en elemento demo a mano derecha, la ventana se actualizará. Luego debe ir a la opción `portal_quick_instaler` y seleccionar para instalación el producto ubify.sitepolicy. Haga clic en install y espere unos segundos.

Tras esto, ya debe de poseer un sitio basico con el sistema cyn.in. Vaya a http://localhost:8080/demo

Luego, detenga su instancia usando `Ctrl+C` en la ventana donde ejecuto.

```
(python2.4)$ ./bin/instance fg
```

Proceda a editar el archivo: src/ubify.policy/ubify/policy/config.py

Cambie las variables `spacesdefaultaddablenonfolderishtypes` y `PRODUCT_DEPENDENCIES` como se muestra a continuación:

```
spacesdefaultaddablenonfolderishtypes = ('Document',
                                         'Event',
                                         'File',
                                         'Image',
                                         'Link',
                                         'Blog Entry',
                                         'Discussion',
                                         'audio',
                                         'video',
                                        )

PRODUCT_DEPENDENCIES = ('Calendaring',
                        'plone.app.iterate',
                        'Marshall',
                        'CMFPlacefulWorkflow',
                        'CMFNotification',
                        'ZipFileTransport',
                        'Scrawl',
                        'ubify.coretypes',
                        'ubify.smartview',
                        'ubify.spaces',
                        'ubify.viewlets',
                        'ubify.cyninv2theme',
                        'ubify.recyclebin',
                        'ubify.xmlrpc',
                        'Products.OpenXml',
                        'ATRatings',
                        'ubify.ffxmpp',
                        'cenditel.audio',
                        'cenditel.video',
                        )
```

A continuación modifique los siguientes valores a los archivos de configuración xml en `ubify.coretypes/ubify/coretypes/profiles/default/ContentSpace` y `ubify.coretypes/ubify/coretypes/profiles/default/ContentRoot`.

```
<property name="allowed_content_types">
  <element value="Document"/>
  <element value="Event"/>
  <element value="File"/>
  <element value="Image"/>
  <element value="Link"/>
  <element value="Blog Entry"/>
  <element value="ContentSpace"/>
  <element value="Video"/>
  <element value="Discussion"/>
  <element value="Audio"/>
  <element value="audio"/>
  <element value="video"/>
 </property>
```

Vuelva a la zona donde esta el script buildout y ejecute el mismo de nuevo.

```
(python2.4)$ cd $HOME/buildouts/cenditelmultimedia
(python2.4)$ ./bin/buildout -vNc cenditelmultimedia.cfg
```

Al finalizar ejecute nuevamente la instancia.

```
(python2.4)$ ./bin/instance fg
```

Luego, diríjase a http://localhost:8080/manage diríjase nuevamente a `portal_quick_installer` y marque la casilla ubisite.policy presione reinstall.

Al terminar vuelva a http://localhost:8080/demo y presione el botón nuevo. Debería de poder agregar tipos de contenido de audio y vídeo usando html5.

1.5.7 Código Fuente

Por definir

CAPÍTULO 2

Índices y tablas

- genindex
- search

Índice