

ZOPE

Z Object Publishing Environment

J. David Ibáñez*
palomar@sg.uji.es
Universitat Jaume I

18 de septiembre de 2001

Resumen

Presentamos Zope (*Z Object Publishing Environment*), un servidor de aplicaciones web. Zope proporciona diferentes facilidades para construir complejos sistemas de información (aplicaciones de comercio electrónico, portales, sistemas de gestión de contenidos, etc..) de forma rápida y sencilla. Zope es software libre, probablemente el servidor de aplicaciones web libre más maduro y extendido.

1 Introducción a Zope

Zope es un servidor de aplicaciones web. Ofrece diversas facilidades para construir complejos sistemas de información que utilicen la web como infraestructura básica, algunos de estos son:

- Separación clara entre lógica, presentación y contenido.
- Base de datos orientada a objetos.
- Fácil conexión a bases de datos externas.
- Un ORB (*Object Request Broker*) integrado.
- Soporte de diversos protocolos: HTTP, WebDAV, XML-RPC, etc..
- Gestión de las aplicaciones mediante la propia web.
- Posibilidad de extender Zope fácilmente.
- Una gran cantidad de productos externos desarrollados por terceros.

*©J. David Ibáñez. Se otorga permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre GNU, Versión 1.1 o cualquier otra versión posterior publicada por la Free Software Foundation. Se considerará como Secciones Invariantes todo el documento, no habiendo Textos de Portada ni Textos de Contra Portada. Puede consultar una copia de la licencia en: <http://www.gnu.org/copyleft/fdl.html>

En el presente artículo veremos una introducción a Zope, el lector tan solo debe tener algunos conocimientos básicos de la web y del lenguaje HTML para comprender este artículo. Aunque conocimientos de programación y de bases de datos le serán también de ayuda.

En la siguiente sección se mostrará cómo instalar Zope y su interfaz web de gestión. A continuación se introducirá el lenguaje DTML. En la cuarta sección se explicará el principio de *adquisición*. En la quinta se mostrará cómo implementar la lógica mediante guiones de código. En la sexta sección se hablará sobre seguridad. En la séptima y penúltima se explicará cómo acceder a bases de datos relacionales. Finalmente se hará un repaso de algunas de las facilidades de Zope que por cuestiones de espacio han quedado fuera del presente artículo, y se darán algunas referencias que permitan al lector profundizar en Zope.

2 Instalación y primeros pasos

2.1 Instalación

Zope funciona en Windows, GNU/Linux, Solaris, *BSD, Macintosh, etc.. La mayoría de distribuciones de GNU/Linux incluyen paquetes de Zope, lo que facilita su instalación. Pero también se puede instalar alguno de los binarios¹ que se distribuyen en Zope.org, para ello basta descomprimir el binario en un directorio, por ejemplo en `/usr/local`, y ejecutar el script `install`.

El script de instalación crea un usuario con nombre **admin** y una clave que hay que recordar porque será con este usuario con el que se accederá por primera vez al sistema.

Una vez instalado, para iniciar Zope hay que ejecutar el script `start`². La ejecución de dicho script arranca ZServer, que es el servidor web que utiliza Zope³. Con Zope en marcha, basta ir a la dirección `http://localhost:8080/` para ver la página de inicio.

2.2 Interfaces de Gestión

Una vez instalado Zope, el primer paso es acceder a las pantallas de administración mediante las que se gestiona, para ello hay que visitar la dirección `http://localhost:8080/manage`, la Figura 1 muestra dicho interfaz.

La pantalla se divide en tres partes. En la parte superior hay, además del logotipo de Zope, un menú que permite acceder a una pequeña ayuda, configurar varios parámetros y salir de los interfaces de gestión.

En la parte izquierda se encuentra un interfaz que nos permite navegar por la base de datos de Zope, la cual tiene una estructura jerárquica similar a un sistema de ficheros.

Finalmente, en el centro de la pantalla, está el *espacio de trabajo* donde se gestionan los diferentes objetos.

¹<http://www.zope.org/Products/Zope>

²Para parar Zope se utiliza el script `stop`.

³También se puede utilizar Apache u otro servidor web.

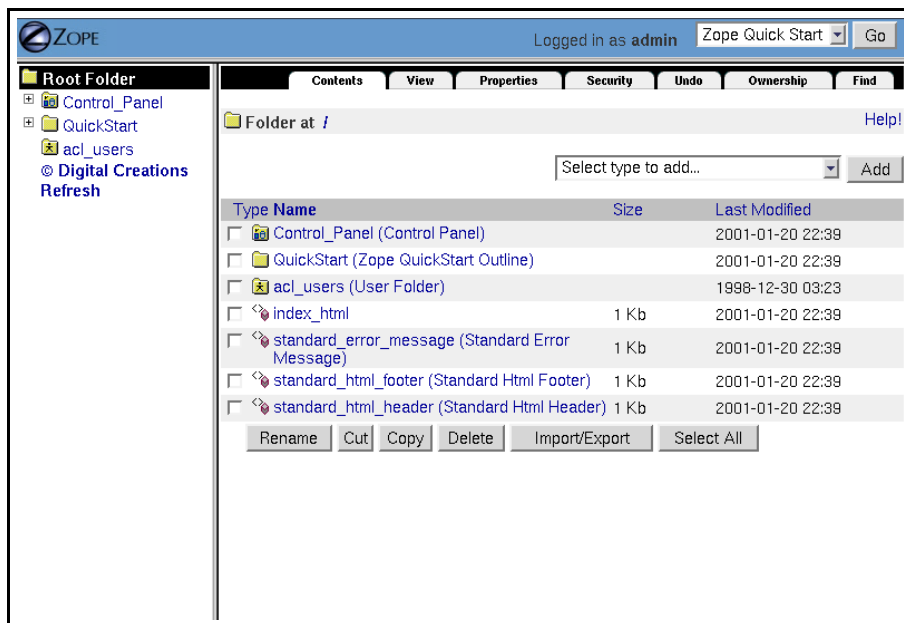


Figura 1: Interfaz de gestión de Zope

El espacio de trabajo

Normalmente, en la parte superior del espacio de trabajo se encuentran una serie de pestañas. Cada pestaña proporciona una vista del objeto que se está editando; así, diferentes clases de objetos pueden tener diferentes vistas.

Debajo de las pestañas hay una barra, en su parte izquierda se muestra la clase de objeto que se está editando seguida por la ruta del objeto desde la raíz. En la parte derecha suele encontrarse el acceso a la ayuda *online*.

A continuación aparece el cuerpo del interfaz, el cual depende del objeto que estemos editando y de la pestaña que se haya seleccionado.

2.3 El sistema de ayuda *online*

Zope dispone de un sistema de ayuda al que se puede acceder desde los interfaces de gestión, cada pantalla dispone de un enlace situado en la parte superior derecha, cuando se pulsa en él se abre una nueva ventana como la de la Figura 2. Dicha ventana se divide en dos partes, a la izquierda un menú permite navegar por todo el sistema de ayuda y a la derecha se presenta la página seleccionada.

2.4 Carpetas

Las carpetas (*Folder*) son similares a los directorios de un sistema de ficheros: permiten almacenar jerárquicamente colecciones de objetos. A continuación vamos a ver las diferentes vistas de las carpetas.

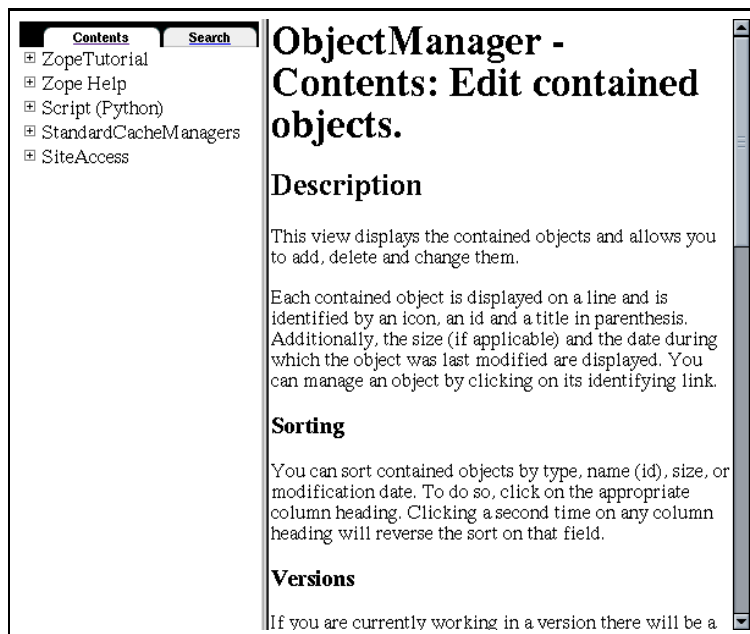


Figura 2: Sistema de ayuda *online*

Contents

Es la vista principal. En la parte superior a la derecha hay un menú que permite realizar diferentes acciones, como añadir nuevos objetos a la base de datos. Las acciones disponibles dependerán de los productos que tengamos instalados. Más adelante veremos las acciones disponibles en una instalación simple de Zope.

A continuación se muestran todos los objetos que contiene esta carpeta, con información diversa sobre los mismos: tipo, identificador, título, tamaño y fecha de la última modificación. El interfaz permite ordenar los objetos por diferentes criterios, basta con pinchar en la cabecera de la columna por la cual se quiera ordenar. Para editar un objeto en particular basta con pinchar sobre él.

En la parte inferior hay una serie de botones que permiten realizar diferentes operaciones sobre los objetos: renombrar, cortar, copiar, pegar, exportar e importar y seleccionar o deseleccionar todos los objetos.

View

Esta vista muestra cómo se verá la carpeta. Al intentar visualizar una carpeta Zope busca un objeto con identificador *index.html*⁴. Si lo encuentra, utiliza su vista como vista de la carpeta.

Si la carpeta no contiene ningún objeto *index.html* este se busca mediante adquisición (Sección 4).

⁴Este comportamiento es parecido al de otros servidores web, como Apache, que al intentar visualizar un directorio suelen mostrar el contenido de un fichero llamado *index.html*, si existe. Normalmente dicha forma de proceder es configurable.

Properties

Esta vista permite gestionar (añadir, editar y borrar) las *propiedades* de la carpeta. Las propiedades son como los atributos de un objeto. Pueden haberlos de varios tipos: *boolean*, *date*, *float*, *int*, *lines*, *long*, *string*, *text*, *tokens*, *selection* y *multiple selection*. El que una propiedad sea de un tipo u otro determina dos cosas, la primera es el tipo de datos de Python interno que se utilice y la segunda el elemento HTML para editar la propiedad.

Por ejemplo, una propiedad de tipo *text* se almacena como una cadena de caracteres y se edita mediante un área de texto.

Security

Esta vista permite gestionar los parámetros de seguridad de la carpeta: crear nuevos roles, asignar permisos a roles y otorgar roles locales a usuarios. En la Sección 6 se estudiarán estos temas con mayor detalle.

Undo

Esta vista permite deshacer transacciones. La base de datos de Zope (ZODB) guarda las modificaciones que se realizan, de tal modo que es posible deshacer (y rehacer) transacciones.

Ownership

Esta vista muestra quien es el propietario del objeto. Esto afecta a la seguridad que se verá en la Sección 6.

Find

Esta vista permite buscar objetos en la base de datos según diferentes criterios.

2.5 El panel de control

En la raíz de la base de datos se encuentra un objeto especial, el panel de control, que permite realizar algunas operaciones básicas de administración tales como reiniciar y parar el servidor y otras un poco más complejas:

- Gestión de la base de datos. Permite compactar la base de datos y ajustar varios parámetros de la caché.
- Gestión de versiones. Desde aquí se pueden guardar o descartar los cambios de las diferentes versiones.
- Gestión de los productos. Los productos permiten extender Zope con nuevas funcionalidades. Desde estas pantallas se puede acceder a ellos y manipularlos.
- Información de depuración. Aquí se muestra información útil para depurar y optimizar Zope.

2.6 Objetos básicos

Como hemos visto al estudiar las carpetas, en su pantalla principal hay un menú que permite realizar diferentes acciones, como añadir nuevos objetos. A continuación se enumeran y describen brevemente las acciones disponibles en una instalación simple de Zope, agrupadas en unidades lógicas:

- *Folder*.
Permiten almacenar otros objetos, son equivalentes a los directorios de un sistema de ficheros.
- *DTML Document* y *DTML Method*.
Son los objetos responsables de la presentación en la web, se estudiarán en la Sección 3.
- *Image* y *File*.
Objetos para almacenar imágenes o ficheros binarios arbitrarios.
- *Script (Python)* y *External Method*.
Son los objetos responsables de la lógica de la aplicación, se estudiarán en la Sección 5.
- *Accelerated HTTP Cache Manager* y *RAM Cache Manager*.
Permiten incrementar el rendimiento de un sitio web mediante caches.
- *Set Access Rule*, *Site Root* y *Virtual Host Monster*.
Permiten la manipulación de las urls, lo cual, por ejemplo, sirve para servir diferentes dominios desde una sola máquina, lo que se conoce como *virtual hosting*.
- *Vocabulary* y *ZCatalog*.
Se utilizan para indexar y buscar objetos en la base de datos.
- *Z Gadfly Database Connection*, *Z SQL Method* y *Z Search Interface*.
El primero permite establecer una conexión con la base de datos relacional Gadfly, el segundo permite hacer consultas SQL a bases de datos relacionales y el último ayuda en la creación de formularios que consulten una base de datos. Se estudiarán en la Sección 7.
- *Mail Host*.
Sirve para enviar mensajes de correo electrónico mediante el protocolo SMTP.
- *User Folder*.
Permite almacenar usuarios, se estudiará en la Sección 6.
- *Version*.
Permite crear versiones, esto es, modificaciones en el sitio web que no son visibles mientras se desee.
- *Zope Tutorial*.
Es un tutorial que permite aprender Zope paso a paso.

3 Document Template Markup Language

El lenguaje DTML proporciona una serie de marcas, con una sintaxis similar a la de HTML, que son procesadas en el servidor para generar páginas web.

A continuación veremos las tres marcas básicas más utilizadas de DTML: **dtml-var**, **dtml-if** y **dtml-in**.

3.1 Inserción de variables

La marca DTML más sencilla es la que permite insertar el contenido de una variable en un documento, se llama **dtml-var** y su sintaxis es:

```
<dtml-var variable>
```

Por ejemplo, el siguiente código DTML:

```
<html>
<head>
  <title><dtml-var title></title>
</head>
<body bgcolor="#ffffff">
  <h3><dtml-var title></h3>

  <dtml-var body>
</body>
</html>
```

podría generar la siguiente página web:

```
<html>
<head>
  <title>Mi página web</title>
</head>
<body bgcolor="#ffffff">
  <h3>Mi página web</h3>

  ¡Hola mundo!
</body>
</html>
```

si las variables `title` y `body` contienen, respectivamente, “Mi página web” y “¡Hola mundo!”.

Expresiones python

Además de mostrar variables también se pueden utilizar expresiones y mostrar su resultado, entonces la sintaxis del tag **dtml-var** es:

```
<dtml-var variable | "expresión">
```

Las expresiones se distinguen de las variables porque se encierran en comillas dobles. Estas expresiones de DTML son en realidad expresiones del lenguaje Python. Veamos un ejemplo:

```

<tr>
  <td><dtml-var precio></td>
  <td><dtml-var "precio + precio*0.16"></td>
</tr>

```

En este ejemplo se utiliza la etiqueta **dtml-var** primero con una variable y a continuación con una expresión.

Atributos

Las etiquetas DTML modifican su comportamiento mediante atributos. Como ejemplo **dtml-var** admite, entre otros, los siguientes atributos:

html_quote Convierte los caracteres con un significado especial en HTML a entidades.

```
<dtml-var ejemplo_de_codigo_html html_quote>
```

fmt Aplica el formato especificado, por ejemplo:

```
<dtml-var fecha fmt=dd>
```

si la variable `fecha` es un objeto de tipo fecha esta sentencia mostraria tan solo el día.

Finalmente, la sintaxis de la etiqueta **dtml-var** es en realidad:

```
<dtml-var variable|"expresión" atributos>
```

3.2 Procesamiento condicional

En ocasiones queremos mostrar algunas partes de un página web solo si se cumplen ciertas condiciones, para ello disponemos de la etiqueta **dtml-if**, su sintaxis es:

```

<dtml-if variable|"expresión">
  ...
<dtml-elif variable|"expresión">
  ...
<dtml-else>
  ...
</dtml-if>

```

Vemos que las etiquetas DTML pueden clasificarse en dos tipos segun su sintaxis. Algunas como **dtml-var** no necesitan ser cerradas, mientras que otras como **dtml-if** encierran un bloque utilizando dos marcas e incluso pueden tener otras marcas internas seperando otros bloques.

Terminemos con un ejemplo de la etiqueta **dtml-if**:

```

<dtml-if "precio <= 5000">
  ¡Qué barato!
<dtml-elif "precio > 5000 and precio <= 10000">
  Ni barato ni caro.
<dtml-else>
  ¡Qué caro!
</dtml-if>

```

3.3 Mostrar listas de elementos

La última etiqueta DTML que estudiaremos en esta sección es la que nos permite iterar sobre listas de elementos: **dtml-in**, su sintaxis es:

```

<dtml-in variable|"expresión">
  ...
</dtml-in>

```

Como ejemplo veamos lo que podría ser una parte de un formulario:

```

<select name="color">
  <dtml-in "['rojo', 'verde', 'azul']">
    <option value="<dtml-var sequence-index>">
      <dtml-var sequence-item></option>
  </dtml-in>
</select>

```

En este ejemplo se aprecia la utilización de dos variables, `sequence-index` y `sequence-item`, estas variables adquieren un valor diferente en cada iteración, en este caso:

<code>sequence-index</code>	<code>sequence-item</code>
0	rojo
1	verde
2	azul

3.4 Conclusiones

Hemos visto brevemente las tres etiquetas más básicas del total de 17 marcas estándar⁵ de DTML. En las siguientes secciones veremos más ejemplos de código DTML.

Métodos y documentos DTML

Hay dos clases de objetos para utilizar DTML y generar así las páginas web, son los métodos y los documentos. La diferencia es que los documentos DTML tienen propiedades (al igual que las carpetas) mientras que los métodos carecen de ellas.

¿Cuándo utilizar los documentos y cuándo los métodos? Si tan solo se quiere hacer una plantilla para mostrar un contenido que se almacena en alguna otra

⁵Es posible extender DTML definiendo nuevas marcas.

parte, por ejemplo en una base de datos relacional, es preferible utilizar métodos DTML. Pero si se quiere guardar el contenido en el propio objeto utilizando propiedades habrá que usar documentos DTML.

4 Adquisición

La adquisición es un mecanismo que permite a un objeto *adquirir* propiedades de su entorno, lo veremos mejor con un ejemplo. Sea la siguiente jerarquía de objetos:

```
carpeta1/  
  carpeta2/  
    objeto
```

Supongamos que `carpeta2` tiene una propiedad `color`, entonces, si `objeto` carece de dicha propiedad, la expresión `objeto.color` devolverá la propiedad `color` de `carpeta2`.

Es decir, cuando buscamos una propiedad o atributo en un objeto, si este no tiene dicho atributo, se buscará en la carpeta que contiene al objeto, si esta tampoco tiene el atributo buscado se seguirá buscando hasta llegar a la raíz.

Veamos ahora un ejemplo más concreto que muestre mejor la utilidad de la adquisición. Normalmente un sitio web tiene la misma apariencia en todo el sitio, con páginas estáticas mantener dicha coherencia es hartos costoso, incluso cuando se trata de un sitio web pequeño, con sólo un puñado de páginas. La forma de mantener la coherencia con Zope implica el uso de la adquisición, normalmente tendremos dos métodos DTML en la raíz del sitio web, que por convenio suelen llamarse `standard_html_header` y `standard_html_footer`, nuestras páginas web harán referencia a dichos métodos, por ejemplo:

- `standard_html_header`

```
<html>  
<head>  
  <title><dtml-var title></title>  
</head>  
<body bgcolor="#ffffff">
```

- `standard_html_footer`

```
</body>  
</html>
```

- `index.html`

```
<dtml-var standard_html_header>  
  
<h3><dtml-var title></h3>  
  
<dtml-var body>
```

```
<dtml-var standard_html_footer>
```

De este modo, para cambiar la imagen de todo el sitio web tan solo hay que editar dos ficheros.

5 Guiones

Si la presentación de un sitio Zope se realiza mediante plantillas DTML, la lógica se implementa mediante guiones de código, normalmente escritos en Python.

5.1 Python

Los guiones de Python son objetos que implementan una función en dicho lenguaje, pero con algunas restricciones por cuestiones de seguridad.

Código restringido

En Zope es habitual ceder la gestión de una carpeta a algunos usuarios de tal modo que estos tengan capacidad para realizar algunas tareas, entre las cuales puede estar la utilización de guiones Python. Para poder hacer esto de forma segura se deben aplicar ciertas restricciones, por ejemplo, no es posible acceder desde un guión Python al sistema de ficheros.

Esto es lo que en Zope se conoce como código restringido, otros ejemplos de código restringido son los métodos DTML y, en general, cualquier código que se pueda editar a través de la web.

Bindings

Los *bindings* són una serie de variables accesibles desde en los guiones Python, se pueden modificar en la pestaña del mismo nombre que hay en los interfaces de gestión. Los tres *bindings* más usados son:

- **script**
Hace referencia al propio objeto, al guión python.
- **container**
Es la carpeta que realmente contiene al guión, independientemente de la ruta mediante la que se haya accedido al mismo.
- **context**
Es el objeto a través del cual se accede al guión, puede variar dependiendo de la ruta a través de la cual se acceda al guión.

Por ejemplo, si tenemos la siguiente jerarquía de objetos:

```
/carpeta1/  
  guion  
  carpeta2/
```

y accedemos a la dirección `http://localhost/carpeta1/carpeta2/guion`, entonces el valor de cada *binding* será:

script	guion
container	carpeta1
context	carpeta2

Ejemplo

Vamos a ver un ejemplo que combine DTML y Python: un conversor de pesetas a euros. Para ello utilizaremos un método DTML que mostrará un formulario donde se introducirá la cantidad de pesetas y se mostrará su equivalente en euros. Además habrá un guión Python que realizará los cálculos.

El formulario DTML (calculadora) presenta este aspecto:

```
<dtml-var standard_html_header>

<form action="calculadora" method="get">
  <input type="text" name="pesetas" size="9"
        value="<dtml-var pesetas missing">"> pts
  <input type="submit" value=" = ">
  <dtml-if pesetas>
    <dtml-var "pesetas2euros(pesetas)"> euros
  </dtml-if>
</form>

<dtml-var standard_html_footer>
```

El guión Python (`pesetas2euros`) tendrá un solo parámetro: `pesetas`. El cuerpo del guión será:

```
return round(int(pesetas)/166.386, 2)
```

Este es un ejemplo muy sencillo, pero aun así muestra la separación entre presentación (DTML) y lógica (Python).

5.2 Perl

La empresa ActiveState ha desarrollado un producto que añade una nueva clase de objetos Zope, los guiones de Perl⁶. Así, es posible implementar la lógica de la aplicación con el lenguaje de programación Perl. Se puede obtener más información al respecto en Zope.org⁷.

5.3 Métodos externos

Los métodos externos son funciones Python, pero al contrario que los guiones Python residen en el sistema de ficheros, en el directorio `Extensions`. No se pueden editar a través de la web.

Su ventaja reside en que no tienen restricciones por cuestiones de seguridad ya que para acceder a ellos se necesita acceso al sistema de ficheros, es código no restringido.

⁶<http://downloads.activestate.com/Zope-Perl>

⁷<http://www.zope.org/Members/andym/wiki>

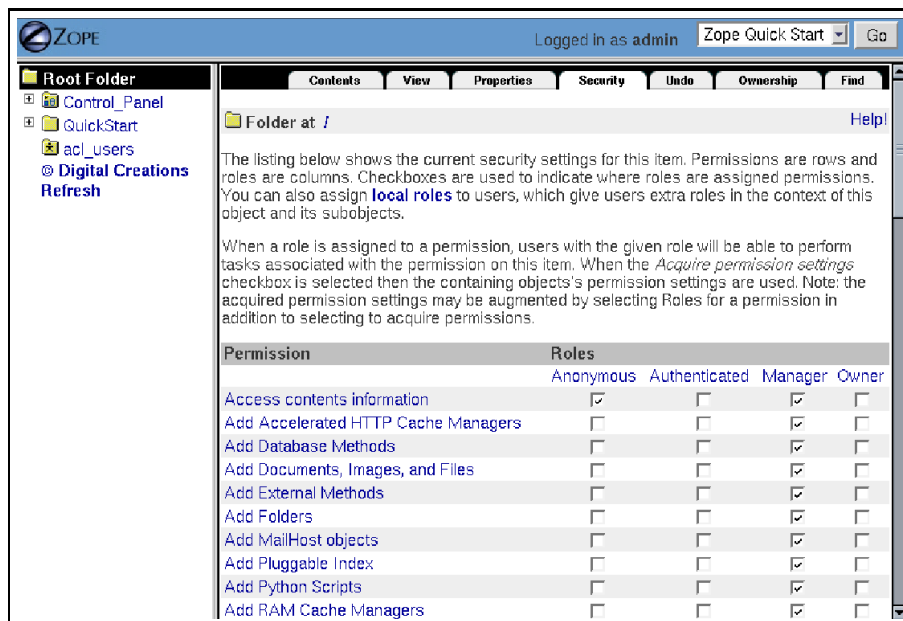


Figura 3: Interfaz para gestionar la seguridad

6 Seguridad

En esta sección vamos a ver tres aspectos sobre seguridad: cómo se controla qué usuarios pueden hacer qué cosas, la identificación de los usuarios y el soporte de SSL.

6.1 Permisos, roles y usuarios

Los permisos definen qué cosas se pueden hacer, por ejemplo editar documentos DTML, acceder a los interfaces de gestión, etc.. Los permisos disponibles dependen de los productos que haya instalados.

Los permisos se asocian a roles y los roles a usuarios. En principio hay cuatro roles definidos (se pueden crear más):

Anonymous Todos los usuarios tienen este rol, se utiliza para determinar qué permisos tienen usuarios no identificados.

Authenticated Todos los usuarios identificados tienen este rol.

Manager Este rol se utiliza para los administradores del sitio, suele tener todos los permisos.

Owner Un usuario tiene este rol en los objetos por él creados.

Casi todos los objetos Zope tienen una pestaña en los interfaces de gestión con la etiqueta *Security*. En la Figura 3 se puede ver dicho interfaz, mediante el cual se gestiona la seguridad relativa al objeto.

Si se tuviera que definir la seguridad para cada objeto el trabajo sería enorme, una vez más la adquisición nos resuelve el problema. Normalmente se definirá

la seguridad tan sólo en la raíz de la aplicación, y el resto de objetos adquirirán la misma configuración.

Finalmente, cada usuario en el sistema tiene asociados una serie de roles y es capaz de hacer todo lo que sus roles le permitan.

6.2 Autenticación

Los usuarios se definen en objetos con el identificador `acl_users`, un usuario existe en la carpeta en la que es definido y en todas sus subcarpetas.

Zope viene con una clase de objetos llamada *User Folder*, se trata de una carpeta que contiene usuarios: su identificador, clave, roles asociados y dominios desde los que puede acceder. Se trata de una *contenedor* de usuarios muy sencillo, que sólomente suele ser útil en aplicaciones sencillas.

Para aplicaciones más complejas se suelen requerir otros métodos más sofisticados que permitan, por ejemplo, identificar mediante LDAP. Para ello existen varios productos que se pueden obtener de Zope.org⁸.

6.3 SSL

Otro aspecto importante, sobre todo en aplicaciones de comercio electrónico, es la encriptación del canal de comunicación para impedir que información sensible sea interceptada. Es decir, SSL.

Zope, a día de hoy no soporta nativamente SSL, para resolver esto hay dos soluciones:

- Aplicar unos parches desarrollados por Ng Pheng Siong que se pueden obtener en <http://www.post1.com/home/ngps/zope/zssl> y que proporcionan soporte SSL a Zope.
- Acceder a Zope a través de otro servidor web que sí soporte SSL, como por ejemplo Apache.

7 Bases de datos relacionales

Desde Zope es posible acceder a bases de datos relacionales. Lo primero es establecer la conexión con la base de datos, para ello se precisa un adaptador. Existen numerosos adaptadores para bases de datos propietarias (Oracle, Informix, etc..) y libres (PostgreSQL, MySQL, etc..). La mayoría se pueden obtener de Zope.org⁹.

En la distribución estándar de Zope se incluye la base de datos Gadfly y un adaptador para la misma. Se trata de una base de datos muy sencilla que normalmente solo se utilizará para hacer pruebas o para hacer pequeñas aplicaciones.

Una vez establecida la conexión con la base de datos se pueden crear métodos SQL, los cuales pueden recibir parámetros y de este modo implementar consultas dinámicas. Con estos métodos accederemos a los datos almacenados en la base de datos relacional.

⁸http://www.zope.org/Products/user_management

⁹http://www.zope.org/Products/external_access

Supongamos que tenemos una base de datos con una tabla llamada `libros`, dicha tabla tiene sólo dos columnas: `titulo` y `autor`. Para obtener un listado con todos los libros bastará con crear un método SQL que contenga el código:

```
select * from libros
```

Para generar consultas dinámicas se deben especificar los argumentos que se esperan recibir y utilizar código DTML para insertar las variables. Existen tres etiquetas específicas de DTML para los métodos SQL: `dtml-sqlvar`, `dtml-sqltest` y `dtml-sqlgroup`, aquí tan solo estudiaremos la primera.

Por ejemplo, el siguiente método SQL (llamado `consulta_libros`) con un parámetro `nombre`:

```
select titulo from libros
      where autor=<dtml-sqlvar nombre type="string">
```

Este código se procesa primero como código DTML y el resultado es la consulta SQL que se ejecuta. Se podría hacer un sencillo interfaz de búsqueda con dos métodos DTML¹⁰:

- `buscaLibrosForm`

```
<dtml-var standard_html_header>

<form action="buscaLibros" method="post">
  <input type="text" name="nombre">
  <input type="submit" value=" Buscar ">
</form>

<dtml-var standard_html_footer>
```

- `buscaLibros`

```
<dtml-var standard_html_header>

<ul>
  <dtml-in consulta_libros>
    <li><dtml-var titulo>
  </dtml-in>
</ul>

<dtml-var standard_html_footer>
```

8 Conclusiones

En este artículo hemos visto tan solo algunas de las características de Zope, suficientes para empezar y crear sitios web sencillos. Pero hemos omitido muchos otros aspectos importantes. A continuación relacionamos algunos de ellos:

¹⁰Una forma rápida de crear interfaces de búsqueda es mediante un *Z Search Interface* (accesible desde la pantalla principal de las carpetas).

- Extendiendo Zope, ZClasses y productos Python.

En ocasiones necesitaremos crear nuestras propias clases de objetos o añadir nueva funcionalidad a Zope. Para ello hay dos métodos, las ZClasses y los productos Python.

Las ZClasses se crean y gestionan a través de la web, permiten crear rápidamente nuevas clases de objetos, pero presentan serias limitaciones.

Los productos Python se desarrollan en el sistema de ficheros, son muy potentes pero un poco más complejos, requieren conocer el lenguaje de programación Python.

- Indexación y búsqueda.

Zope incluye un producto llamado ZCatalog que permite indexar los objetos Zope y realizar búsquedas.

- Zope Page Templates.

Se trata de un nuevo lenguaje, alternativo a DTML, para generar las páginas web. Acentua la separación entre presentación, lógica y contenido.

- Gestión de sesiones.

El producto *CoreSessionTracking* permite gestionar sesiones fácilmente.

- Otros protocolos.

Zope soporta FTP, WebDAV y XML-RPC, cada uno tiene un uso diferente, los dos primeros permiten gestionar Zope desde clientes que soporten dichos protocolos, como alternativa a utilizar un navegador web. El soporte de XML-RPC facilita la construcción de sistemas heterogéneos.

- Rendimiento, fiabilidad y escalabilidad.

La base de datos de Zope (ZODB, *Zope Object Database*): arquitectura, rendimiento, escalabilidad, etc..

ZEO (*Zope Enterprise Objects*), permite construir clusters de servidores Zope.

¿Por donde seguir ahora? A continuación mostramos una lista de referencias:

- La comunidad Zope: <http://www.zope.org>.
- La comunidad Zope hispana: <http://sf.net/projects/hispapyzope>. En particular es recomendable suscribirse a la lista de correo.
- El libro de Zope, publicado por NewsRider, debería ser la siguiente lectura. Por ahora tan solo está disponible en inglés, se puede obtener en Zope.org: <http://www.zope.org/Members/michel/ZB>.