

TRABAJO PARA LA ASIGNATURA DE:  
LABORATORIO DE ADMINISTRACIÓN DE SISTEMAS Y SERVICIOS TELEMÁTICOS

# **ZOPE: MUCHO MÁS QUE UN SERVIDOR WEB**

REALIZADO POR:

Pablo López Renaud    DNI: 50860107    [daelon@die.upm.es](mailto:daelon@die.upm.es)  
Guillermo Pérez Pérez    DNI 51981472    [bisho@eurielec.etsit.upm.es](mailto:bisho@eurielec.etsit.upm.es)

# ZOPE: MUCHO MÁS QUE UN SERVIDOR WEB

La primera definición sobre ZOPE que encontramos dice así:

*"Zope es la nueva generación de servidores globales de aplicaciones. Enmarcado en forma de portal web, programado mediante Python, con absoluta independencia de la plataforma en la que se ejecute y con una muy activa comunidad de usuarios detrás que soportan su carácter Opensource, ZOPE es mucho más que un servidor web".*

En el presente trabajo hablaremos sobre ZOPE y lo analizaremos desde diferentes puntos de vista: en primer lugar profundizaremos sobre qué es ZOPE cómo surge y para qué surge, pasado a hablar después de los módulos que lo componen, detallando sus funcionalidades y sus aplicaciones típicas. Eso nos dará una clara idea del potencial que ZOPE es capaz de conceder a las expertas manos de un administrador y su equipo de programadores, diseñadores y demás técnicos, cada uno especializado en un ámbito y coordinados todos a través de ZOPE.

También hablaremos de ZOPE desde el punto de vista del administrador que debe instalarlo, configurarlo y optimizarlo para su entorno concreto y para los requerimientos establecidos por su empresa. En este contexto descubriremos que ZOPE es realmente sencillo de poner a funcionar, a la vez que nos otorga un potencial de configuración y de facilidad de interconexión entre sus módulos sin precedentes.

Por último trataremos de ver ZOPE desde una óptica más imparcial, analizándolo y comparándolo con otros conjuntos de soluciones disponibles en el mercado. Hablaremos en términos de estabilidad, velocidad, seguridad y dimensionado de la solución requerida, para evaluar finalmente así qué es lo que aporta ZOPE, a quién se lo aporta y de que manera lo hace.

## ¿Qué es ZOPE?

Entremos en detalle. Lo que ZOPE nos proporciona es una plataforma de desarrollo web robusta, claramente orientada a objetos, y que soluciona gran parte de los problemas cotidianos de los desarrolladores de aplicaciones web, facilitando enormemente su trabajo. Así, ZOPE garantiza automáticamente, por citar algunos ejemplos: integridad en las bases de datos, persistencia, control de acceso y herramientas de búsqueda integradas.

ZOPE proporciona además una separación bastante clara entre lógica, datos y presentación

-punto sobre el que hablaremos más tarde- a la vez que engloba estas tres capas en un interfaz común de administración y desarrollo.

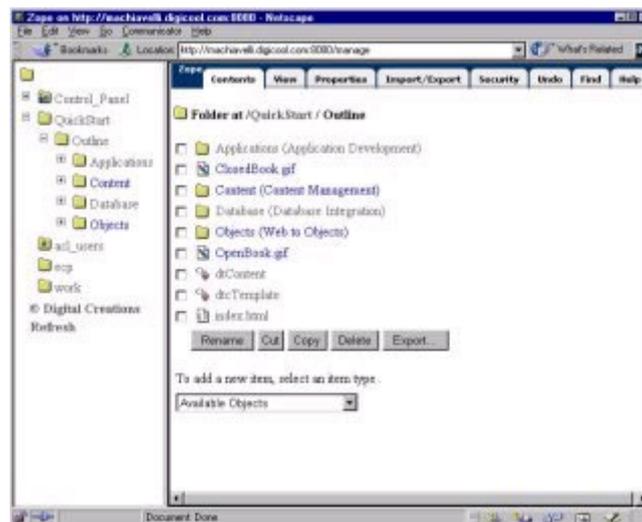
De entre las grandes virtudes de ZOPE cabe destacar su casi inmejorable interfaz global de control y administración, accesible a través de la web y totalmente configurable a gusto del usuario, que almacena en su perfil sus preferencias. Pero la creación de diferentes usuarios no es solo un capricho; nos conduce directos a otro de los puntos fuertes de ZOPE: su seguro y eficiente sistema de delegación de tareas.

Gracias a la jerarquización de usuarios, el administrador o director de proyecto puede conceder a cada grupo de diseñadores, programadores o “picadores de datos” (a.k.a. administradores de contenidos) los permisos adecuados para trabajar únicamente en su ámbito sin interferir en la labor de los demás. Todo el diseño de ZOPE está orientado hacia este modelo jerárquico.

En cuanto a los componentes de ZOPE, hay que destacar su cuidado desarrollo completamente orientado a objetos. Así, cada funcionalidad que encontramos dentro de su macro-herramienta de gestión es un objeto distinto, perfectamente definido, y accesible según un árbol de navegación como el que nos encontramos en cualquier servidor Web. De este modo, insertar, actualizar o modificar módulos es un juego de niños, convirtiéndose ésta en otra de las ventajas clave del uso de ZOPE.

### Administración centralizada

Ya hemos hablado antes de la potente herramienta de administración de ZOPE, accesible a través de Web, que centraliza todas las tareas de administración. Aparte de contar con un explorador de archivos, podemos explorar también de este modo los objetos instalados en ZOPE, y cambiar sus configuraciones, todo desde dicho interfaz centralizado.



Todos los objetos instalados se almacenan en una base de datos transaccionales ZOPE, que permite cosas tan interesantes como deshacer cambios, o evitar errores durante cualquier tipo de acceso a dichos objetos de la base de datos. En la imagen, una captura de su interfaz central de control desde donde se puede interactuar con todos los objetos que incluyamos en ZOPE.

### Acceso a Bases de Datos

Otro de los módulos de ZOPE se encarga de gestionar las bases de datos, que pueden ser Oracle, MySQL o cualquiera que soporte ODBC. El procedimiento es tan sencillo como insertar en ZOPE un objeto para conectar con el tipo de base de datos que vayamos a usar, y después ir creando diferentes objetos para acceder a los datos (las diferentes QUERIES que necesitemos). De esta manera, los diseñadores de

bases de datos pueden trabajar intensivamente sobre estos objetos de acceso a datos, mientras que los administradores de contenidos pueden simplemente llamar a estos objetos para obtener sus datos deseados, sin interferir un grupo de gente con el otro. Y esto solo es un pequeño ejemplo lo que antes hemos comentado sobre separación de tareas y jerarquización de usuarios.

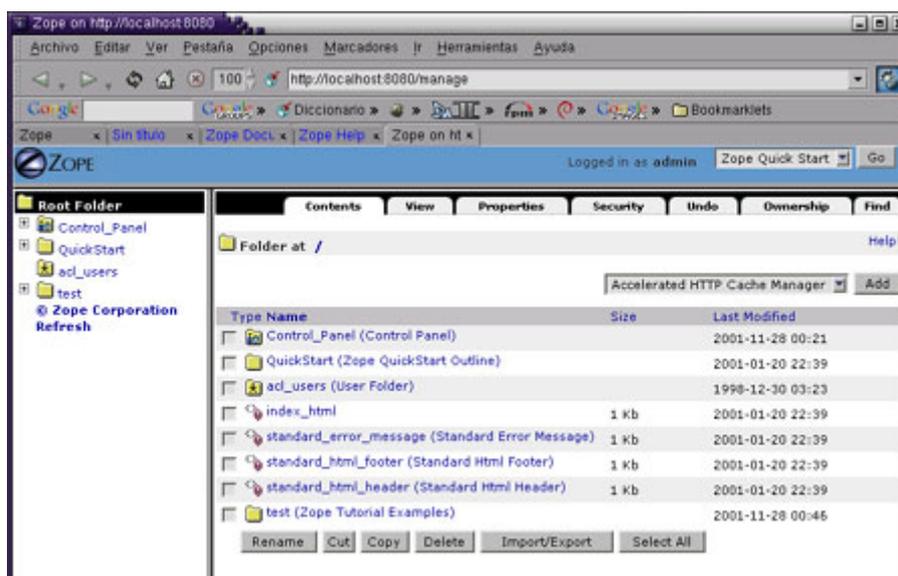
## Búsqueda integrada

Con los objetos de Zope de indexación de datos podemos organizar de manera muy rápida la información que necesitan las potentes herramientas de búsqueda de Zope para catalogar cualquier objeto que incluyamos en nuestro sistema, según su contenido o según unas propiedades que le otorguemos. De esta forma, las búsquedas a lo largo de los módulos creados y mantenidos por diferentes personas, sobre los que no tenemos pleno conocimiento, serán muy sencillas, facilitando aún más el desarrollo conjunto de un proyecto entre grupos de trabajo distintos.

## Soporte para XML

Un último punto a destacar de entre las excepcionales cualidades de Zope es su soporte nativo para XML. Admite protocolos como WebDAV y XML-RPC; pero lo más destacado es su parser XML incorporado, que permite incluso en un momento dado exportar toda la base de datos de objetos de Zope a XML, o bien, en sentido inverso, importar documentos XML y crear objetos Zope a partir de ellos. Esta gran versatilidad termina de abrir las puertas a Zope para ser una gran herramienta de desarrollo de aplicaciones en Internet.

Pasamos ahora a enfocar el análisis de Zope desde el punto de vista del administrador, donde profundizaremos más allá de los datos y las características, para buscar la verdadera utilidad de las cosas y tomar el rendimiento, la fiabilidad, la facilidad de uso, etc. Como parámetros principales a la hora de elegir Zope como nuestro entorno de trabajo.



## Administración y gestión de Zope

Una de las principales ventajas de Zope frente a otros productos de programación orientada a la web es la existencia de un completo y potente sistema de administración y gestión de contenidos via web.

Tras la sencilla instalación se genera un password para el usuario de administración, y desde la interfaz web se puede iniciar las tareas de gestión del servidor de aplicaciones. A continuación veremos alguna de las posibilidades y características de Zope, que por su orientación a objetos y particular sistema de funcionamiento requieren un periodo de adaptación.

Filosofía de trabajo con objetos

En Zope todo son objetos, métodos y propiedades, que se representan en una estructura jerárquica como directorios y ficheros para facilitar la comprensión y gestión de los mismos.

Por ejemplo los directorios son objetos que pueden tener propiedades arbitrarias, y a su vez contener otros objetos como carpetas o documentos.

Zope por defecto trae muchos tipos de objetos, como por ejemplo algunos sencillos como carpetas, ficheros, imágenes, documentos dinámicos, etc, pero otros muy complejos, como aplicaciones enteras. Se puede, por ejemplo, instalar el componente Squishdot, un variante derivada del motor que mueve sitios como Slashdot o Barrapunto, y crear nuevos Squishdots en cualquier directorio sin más que añadir un objeto de ese tipo.

Los objetos se programan en python. No son tan fácilmente programables y personalizables desde la interfaz web. La interfaz de Zope está pensada para la gestión de objetos ya programados previamente, de manera muy fácil y sin tener que volver a programar. No entraremos a estudiar la programación de estos objetos, ya que exceden la profundidad del estudio de Zope realizada en este trabajo.

Los objetos python de Zope deben tener una serie de métodos fijos. El que a nosotros más nos incumbe es el de impresión. Por ejemplo un objeto del tipo fichero al imprimirse en un documento, imprimirá sus contenidos. Pero un objeto del tipo imagen sin embargo no se imprime en binario, sino que daría lugar en el documento a un código HTML del estilo:

```

```

Los directorios, por ejemplo se evalúan como el contenido del objeto `index_html` que contengan, si es que lo contienen

Además de objetos también existen métodos, cuya principal diferencia con los objetos es que se heredan. Por ejemplo se puede definir el método "header" del tipo documento, y desde cualquier documento de esa carpeta y subcarpetas se podrá invocar a ese método para insertar el contenido de "header". Esto es muy útil para gestionar porciones de documentos sin tenerlos que repetir una y otra vez, y con la ventaja de sólo tener que cambiarlo una vez en caso de que fuera necesario.

## Documentos dinámicos

Quizás el tipo de objeto más importante de Zope es el Documento DTML. Se trata de documentos html normales, con una serie de etiquetas especiales que sirven como pegamento entre los distintos objetos y métodos de Zope.

Por ejemplo la siguiente etiqueta inserta en el documento actual el método "standard\_html\_header":

```
<dtml-var standard_html_header>
```

dtml-var se puede utilizar también para la impresión de propiedades. Si el documento actual tiene la propiedad título, o webmaster, se puede imprimir con esa etiqueta.

Como se puede apreciar, los documentos DTML permiten elaborar de una manera sencilla y controlada documentos dinámicos, sin necesidad de la intervención del programador. Éste se puede limitar a la tarea de programar nuevos objetos, y dejar su uso a los administradores de contenidos sin peligros de seguridad.

Otras etiquetas DTML de uso frecuente son:

**<dtml-in expr="...">**: para ejecutar algo para cada valor devuelto por la expresión. En el bucle está presente la variable sequence-item para saber el elemento actual.

**<dtml-if expr="...">**: para ejecutar una expresión. La expresión puede llamar a rutinas para añadir objetos a un directorio, poner cookies, etc.

**<dtml-let>**: permite definir nuevas variables.

**<dtml-tree>**: permite construir árboles html expandibles de forma sencilla y automática.

En las expresiones y condiciones de las etiquetas DTML se utilizan los métodos de los objetos existentes en Zope. Cada tipo de objeto tiene métodos propios que les permiten ser gestionados desde documentos DTML.

Por ejemplo el siguiente fragmento de código lista los contenidos de una carpeta, evaluando "objectValues()" para obtener los objetos que contiene.

```
<dtml-in expr="objectValues('Folder')">
  <dtml-var icon>
  This is the icon for the: <dtml-var id>
  Folder<br>.
<dtml-else>
  There are no Folders.
</dtml-in>
```

## Interacción con el navegador

La interacción con el navegador desde Zope es completamente transparente. Cuando se envía un formulario via GET o POST, o se envía una cookie, automáticamente se encuentra disponible la variable con el valor enviado para ser tratada desde los documentos DTML.

En el siguiente ejemplo veremos como recoger una imagen enviada desde un formulario, y almacenarla en un directorio. El documento con el formulario sería:

```
<dtml-var standard_html_header>
<h2><dtml-var title></h2>
<p>Subir una imagen:</p>
<form action="photoAction" method="post"
enctype="multipart/form-data">
<p>Title: <input type="text" name="photo_title"></p>
<p>File: <input type="file" name="file"></p>
<input type="submit" value="Enviar">
</form>

<dtml-var standard_html_footer>
Y el documento que recibe la imagen y la almacena:
<dtml-var standard_html_header>
<h2><dtml-var title></h2>

<dtml-call expr="directorio_imgs.manage_addImage(
  id=", file=file, title=photo_title)">
<p>Gracias por su contribución.</p>
<dtml-var standard_html_footer>
```

Para el establecer cookies en el navegador del visitante se utiliza el código `<dtml-call expr="RESPONSE.setCookie('lastVisited',_.DateTime())">`. En este caso se colocaría una cookie en el cliente con la hora actual. Esto puede ser muy útil si por ejemplo queremos ver si el documento fue modificado desde la última visita de cada persona, simplemente con el siguiente código:

```
<dtml-if expr="bobobase_modification_time() > _.DateTime(lastVisited)">
<b>¡Este documento ha sido modificado desde su última visita!</b>
</dtml-if>
```

## Conexión a Bases de Datos

Cualquier servidor de aplicaciones que se precie debe poder acceder y consultar bases de datos para construir páginas. Zope por supuesto dispone de esta posibilidad. Zope utiliza por defecto su propia base de datos, orientada a objetos con transacciones, que viene dada por la propia forma de python para almacenar variables y objetos en memoria. Sin embargo puede conectarse a muchas otras bases de datos. La arquitectura empleada por Zope es muy flexible y permite mucha seguridad. Se gestionan independientemente conexiones, queries y las páginas que utilizan dichos queries. De esa manera es posible dividir las tareas, y evitar que los editores de contenidos tengan que preocuparse de conexiones a bases de datos o del lenguaje SQL utilizado para los queries. Permiten además tener bien independizados los queries, de manera que sean fáciles de modificar si se producen cambios en la Base de datos, sin tener que retocar todos los documentos que usan un query en particular. Como veremos más adelante el esquema de permisos de Zope es muy completo y permite asignar a cada persona o perfil diversos privilegios, ya sea para objetos concretos o para añadir tipos de objetos, haciendo así posible la subdivisión de tareas. Para hacer un listado de una tabla de una base de datos es necesario por tanto:

**Objeto conexión:** Especifica la Base de datos y las características de la conexión.

**Objeto SQL Method:** Es el query que se puede ejecutar desde los documentos. Contiene código SQL con posibles variables para personalizar la consulta, y necesita especificar que conexión se va a usar. Puede probarse y ejecutarse en la interfaz al efecto y configurar algunos parámetros como el número máximo de valores devueltos.

**Documento DTML:** Recorre la consulta con un simple `<dtml-in nombre_consulta_sql>`. No necesita por tanto saber si el listado proviene de una consulta a una base de datos, de un listado de objetos de un directorio o de una simple variable.

Si la consulta fuese una inserción, eliminación o modificación, que no devuelven resultados, la ejecución de la consulta se realizaría con un `<dtml-call nombre_consulta_sql>`

Sistema de permisos y transacciones

El principal objetivo de Zope con su interfaz web es posibilitar la gestión distribuida y dividida en tareas de una web. Para ello divide claramente varios aspectos típicamente unidos de la programación web:

**Estilo:** Aunque no es uno de los puntos fuertes de Zope, ya que siguen siendo necesarios conocimientos de html para gran parte de la creación de contenidos, si que facilita la separación de los elementos que son puramente estilo, como la cabecera y pie de las páginas, que pueden mantenerse por separado.

**Contenidos:** Una página de noticias puede simplemente listar los objetos contenidos en un subdirectorío, con su fecha de modificación. La persona encargada de los contenidos tendría sólo permisos para crear documentos en ese subdirectorío, con las noticias de cada día.

**Otras fuentes de datos:** Las aplicaciones y objetos extra, como los de conexión a bases de datos se integran de tal manera que puedan ejecutarse y llamarse desde las páginas de contenidos, sin necesidad siquiera de ver la base de datos o la consulta que se está realizando.

El avanzado esquema de permisos de Zope permite configurar de manera muy sencilla y potente las posibilidades de modificación de cada usuario, según el tipo de objetos que puede crear, las zonas que puede modificar, los eventos que puede ejercer sobre cada objeto (borrado, adición, modificación...). También es posible el tener distintos listados de usuarios para distintas zonas de la web, de manera totalmente independiente.

A esto se le suma que todos los objetos de Zope se encuentran almacenados en su base de datos con soporte de transacciones. Es por tanto posible en cualquier momento ver el historial de modificaciones de un fichero y deshacer cambios realizados sin ningún problema, y sin perder en ningún instante la coherencia.

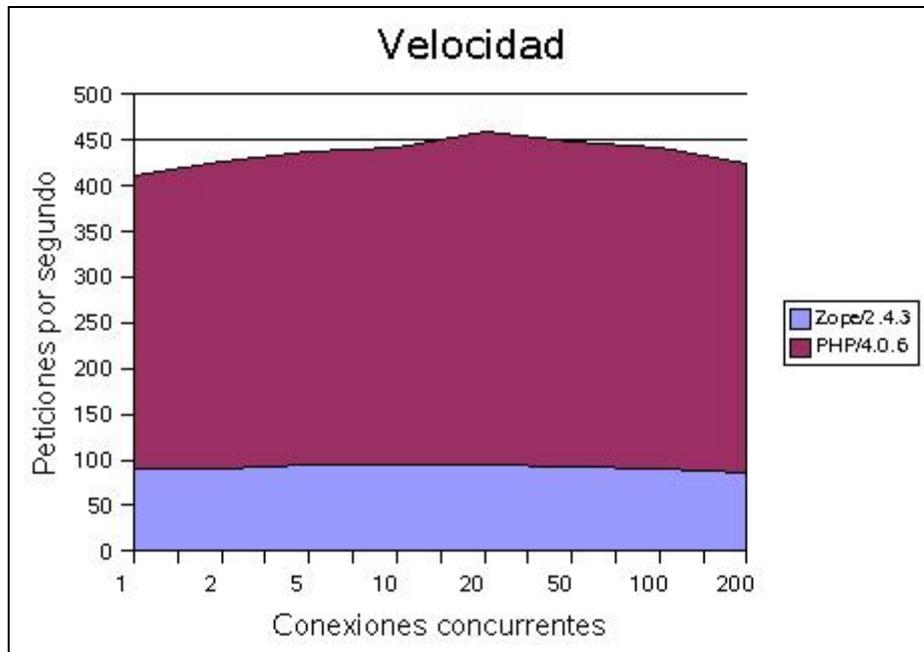
Gracias a este sistema de permisos y control de versiones, Zope ofrece un entorno cooperativo para el desarrollo de páginas web muy potente y seguro.

Comparación con otras alternativas

Realmente en el mundo del software libre tiene pocas alternativas, pues no existen demasiados servidores de aplicaciones completos, y si muchos lenguajes, como perl o php.

Elemento	Zope	PHP	Perl
<b>Lenguaje de programación</b>	Python Dificultad media	PHP Sencillo	Perl Complejo
<b>Desarrollo de nuevas aplicaciones</b>	requiere programar nuevos objetos en python	Rápido	Rápido
<b>Reutilización del código</b>	Muy alta Desde DTML se usan los objetos creados	En función del programador	En función del programador
<b>Componentes y aplicaciones listos para usar</b>	Muchos. Homogéneos No requiere programación	Muchos. Heterogéneos Suelen necesitar adaptación via programación.	Muchos. Homogéneos (CPAN) Requieren programación para usar los objetos
<b>Facilidad de uso</b>	Alta	No está orientado a no-programadores	No está orientado a no-programadores
<b>Gestión via web</b>	SI	N/A	N/A
<b>División de tareas y permisos por perfiles</b>	Integrado	No implementado. Depende de lo desarrollado por el programador	No implementado. Depende de lo desarrollado por el programador
<b>Soporte de Bases de datos</b>	Buena. Lleva integrada una lista para su uso. No requiere programación.	Muy buena. Heterogénea. Requiere programación.	Muy buena. Homogénea. Requiere programación.
<b>Facilidad de instalación</b>	Alta	Media-complicada (si se necesitan algunas funcionalidades puede ser necesario recompilar)	Media
<b>Velocidad</b>	Acceso rápido a los datos (cargados en memoria) Lenguaje DTML lento	Alta-Muy alta	Alta-Muy alta

## Pruebas de Rendimiento



---

Como se puede observar en esta comparación PHP resulta mucho más rápido que Zope. La prueba se ha realizado con un documentos sencillo, que insertaba un header, un footer y una imagen, tratando que el script en PHP también insertara el mismo número de ficheros externos. Seguramente en el manejo de bucles se verían aún más las diferencias de velocidad.

Se puede observar sin embargo como el rendimiento de Zope es muy estable con el número de peticiones, gracias a su servidor web basado en hebras que trae integrado. Apenas se nota el descenso de rendimiento con el aumento de peticiones. PHP se ve más afectado, fundamentalmente debido al esquema basado en procesos que sigue el servidor web Apache, pero su mayor rendimiento le convierte en claro vencedor en el aspecto de rendimiento.

Las pruebas se realizaron en un Pentium III 866, 256Mb de RAM, y 3 discos duros SCSI en RAID 5 de gran rendimiento.

## **Conclusiones**

Zope constituye una estupenda plataforma de desarrollo de aplicaciones web, si bien está más orientada al uso de objetos ya existentes que al desarrollo de nuevos.

En cualquier caso los objetos incluidos por defecto y los productos y módulos extra que se pueden descargar de la web son más que suficientes, y permiten la instalación de aplicaciones complejas como Squisdot con unos pocos clics de ratón. Requiere, eso sí, un periodo de adaptación al lenguaje DTML y a la manera de trabajar, como la mayoría de los servidores de aplicaciones.

Zope resulta ser un servidor de aplicaciones web muy potente, y capaz de competir con soluciones comerciales, como Websphere de IBM, con la ventaja de que se trata de un producto GPL, pero que cuenta con el soporte y respaldo de una empresa si necesitamos consultoría o soporte técnico.

Quizás la única pega sea su velocidad, que puede resultar un tanto lenta en algunas aplicaciones.

El proyecto ZOPE consta de una enorme cantidad de información y documentación OnLine, y existen soluciones preparadas para asegurar su escalabilidad, como sistemas para servidores distribuidos, con balanceo de carga.

## Ejemplos de pantallas de ZOPE

### Administración de permisos

Permission	Roles				
	Anonymous	Authenticated	Manager	Owner	
<b>Acquire permission settings?</b>					
<input checked="" type="checkbox"/>	Access contents information	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Change Database Methods	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Change permissions	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Delete objects	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Manage WebDAV Locks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Manage properties	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Take ownership	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Undo changes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Use Database Methods	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	View	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	View management screens	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	WebDAV Lock items	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	WebDAV Unlock items	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	WebDAV access	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Save Changes

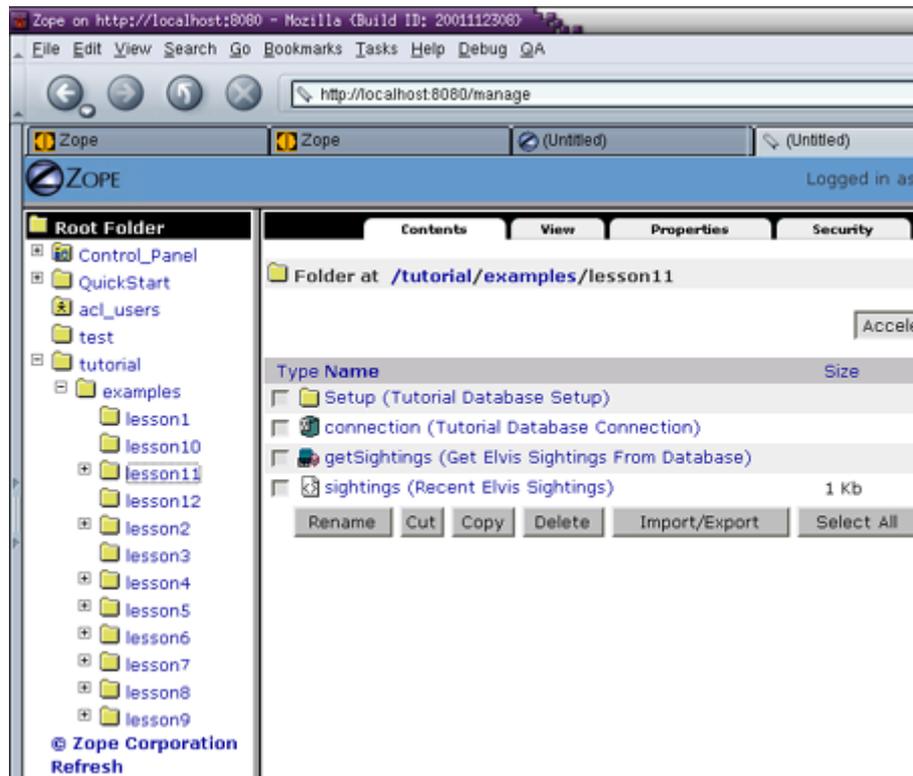
### Editando documento DTML

The screenshot shows the ZOPE web interface. The top bar indicates the user is logged in as 'admin' and is viewing 'Zope Quick Start'. The left sidebar shows a tree view of the site structure, including folders like 'Control\_Panel', 'Products', 'QuickStart', and 'test'. The main content area is titled 'DTML Method at /QuickStart/index\_html' and contains a text editor with the following DTML code:

```
management screens</H4>
<!--#tree Outline leaves=dtcTemplate-->
<SPAN CLASS="tree">
  <!--#if tree-item-expanded-->
    <!--#if expr="objectValues(['Folder'])"-->
      <IMG SRC="<!--#var BASE2-->/Outline/OpenBook.gif"
        ALIGN="TOP" BORDER="0">
    <!--#/if-->
    <STRONG>
      <I><!--#var title--></I>
    </STRONG>
  <!--#else-->
    <!--#if expr="objectValues(['Folder'])"-->
      <IMG SRC="<!--#var BASE2-->/Outline/ClosedBook.gif"
        ALIGN="TOP" BORDER="0">
    <!--#/if-->
    <!--#var title-->
  <!--#/if-->
</SPAN>
<!--#/tree-->
<!--#var standard_html_footer-->
```

At the bottom of the editor, there are buttons for 'Save Changes', 'Taller', 'Shorter', 'Wider', and 'Narrower'.

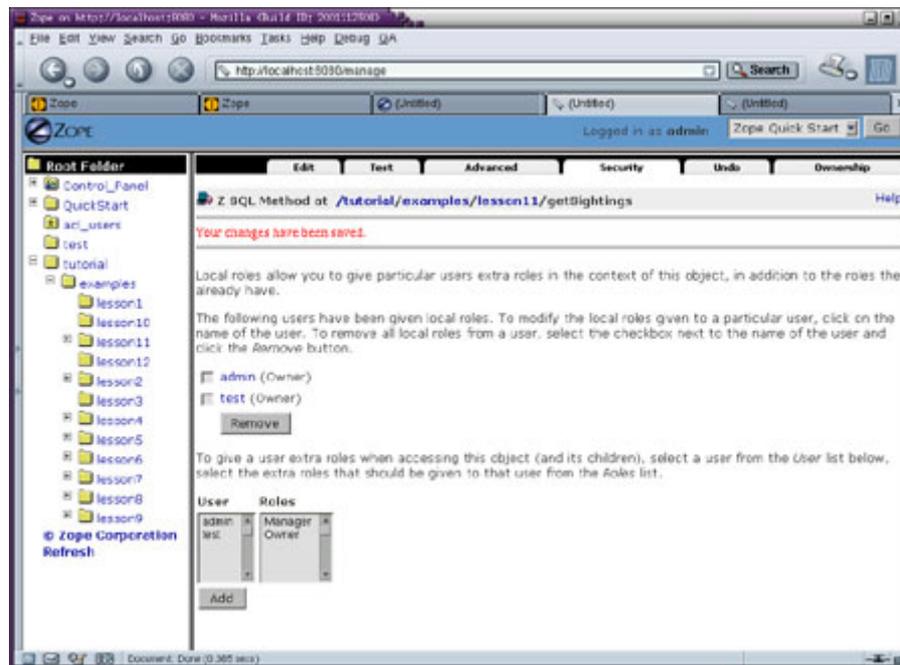
## Ejemplo de conexión a DB



## Sistema de Undo



## Gestión de roles de usuarios en un objeto



## Administración de versiones de un documento

