Introducción

El presente documento es una guía rápida de instalación de alguna herramienta particular. De seguro existen otras formas de realizar el proceso de instalación, pero esta es la que mejor le ha funcionado al autor a lo largo de los años. Para utilizar el presente documento se requieren conocimientos mínimos de linux, su estructura básica de directorios, el uso de la consola y las instrucciones básicas de shell. Además, se espera que el lector tenga suficiente sentido común para adaptarse a cualquier diferencia existente entre el procedimiento que se describe en este documento y la distribución o versión de linux utilizada.

Tutorial de Echo3

El presente documento no tiene instrucciones reales de como programar en Echo3, sino que sólo muestra como ejecutar el tutorial de echo3 en eclipse galileo JEE.

PASO 1:

Descargar los fuentes del tutorial de Echo3 de la siguiente dirección:

http://sistemas.ing.ula.ve/~demian/B2010/tutoriales_cursos/Echo3Tutorial-20100729-211701.tar.gz

O preferiblemente sincronizar con el siguiente repositorio de google code (abrir para ver el URL del SVN):

http://code.google.com/p/echo-tutorial/

PASO 2:

Si se sincronizó desde el google code entonces se debe omitir este paso (ya el proyecto debería estar importado en el workspace de eclipse).

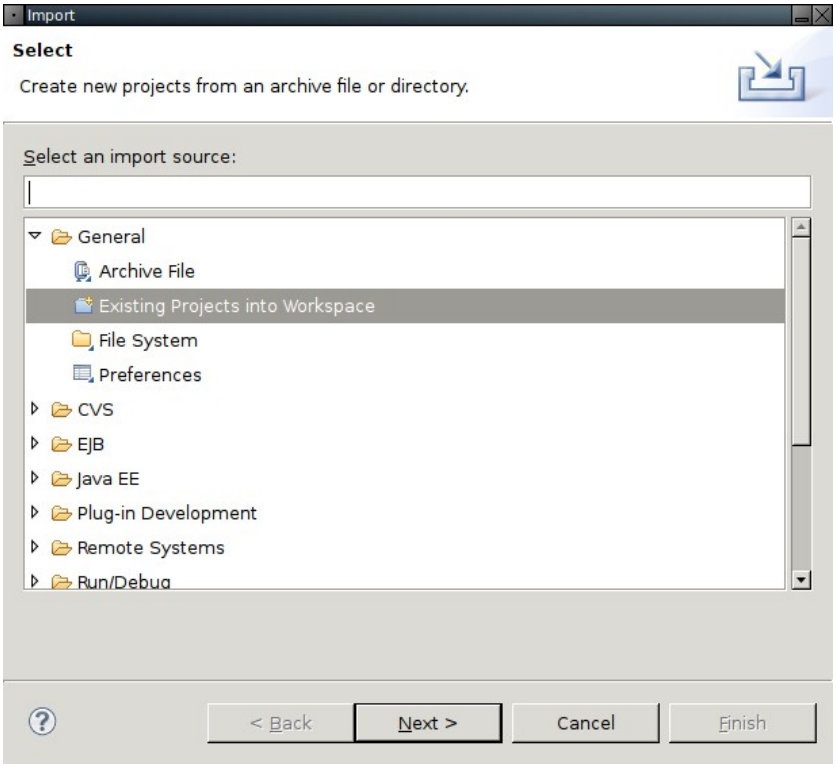
Si se descargó el tutorial (en lugar de sincronizarlo desde el repositorio de google code) es necesario descomprimirlo en el workspace y luego importarlo en eclipse.

Para importar el proyecto en eclipse (esto aplica en general para cualquier proyecto), del menú principal, seleccionar:

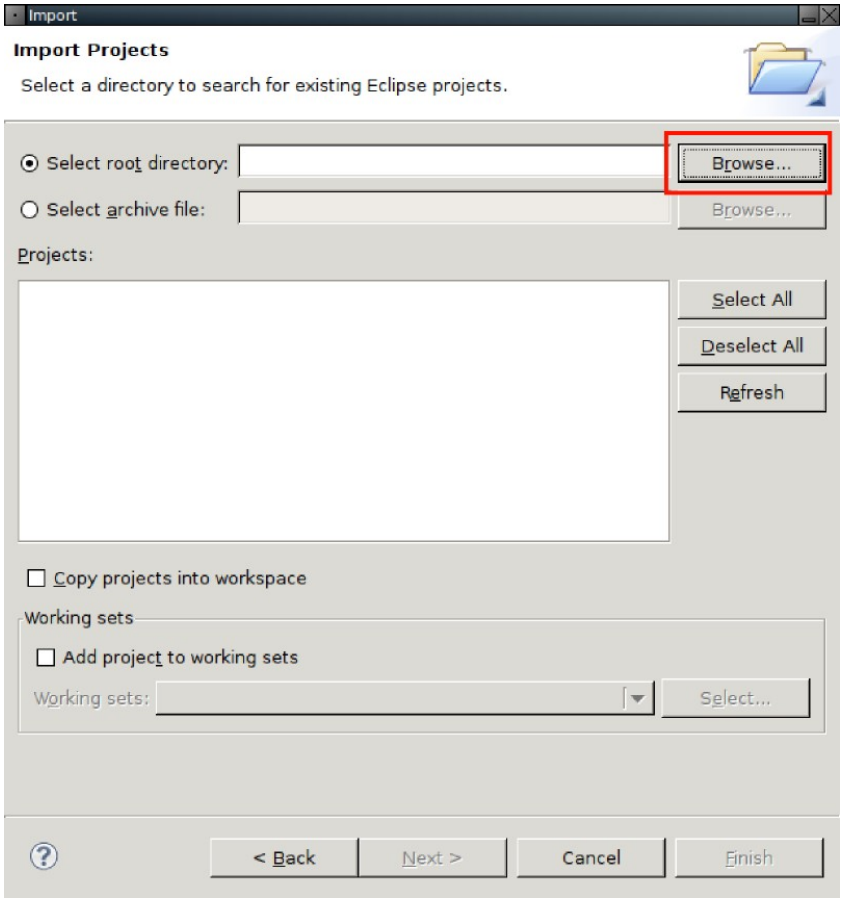
“File->Import”

y luego en el asistente

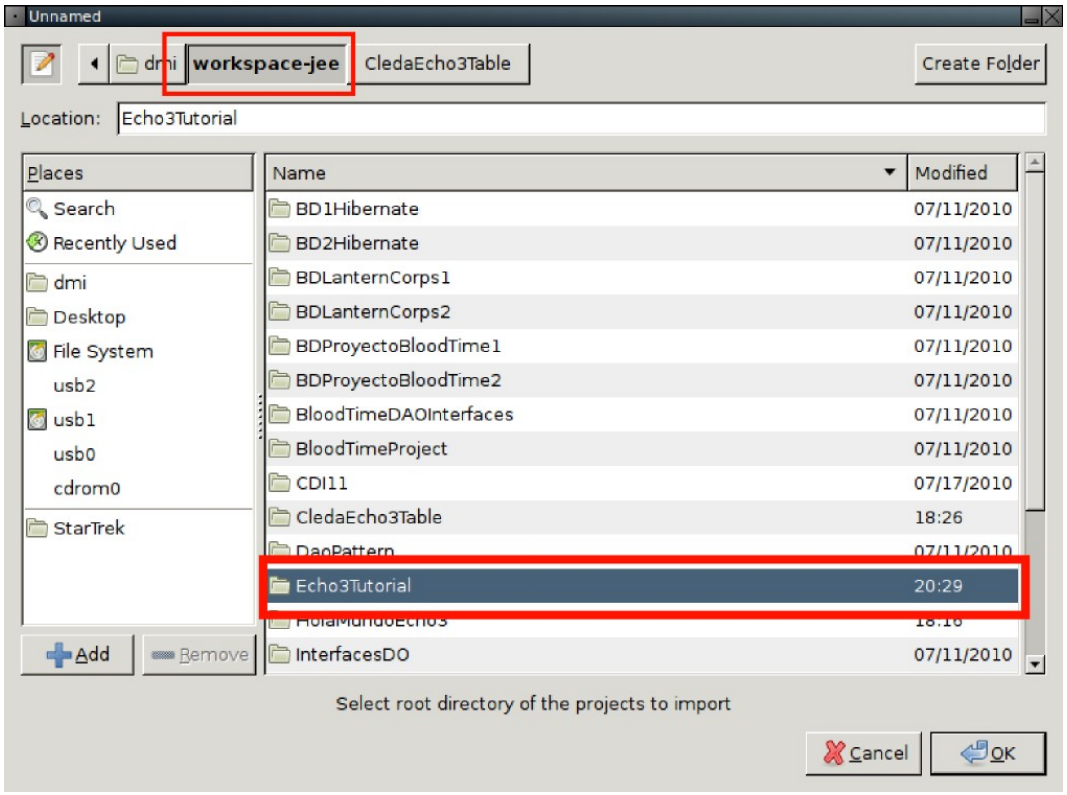
“General->Existing Projects Into workspace”, tal como muestra la siguiente figura:



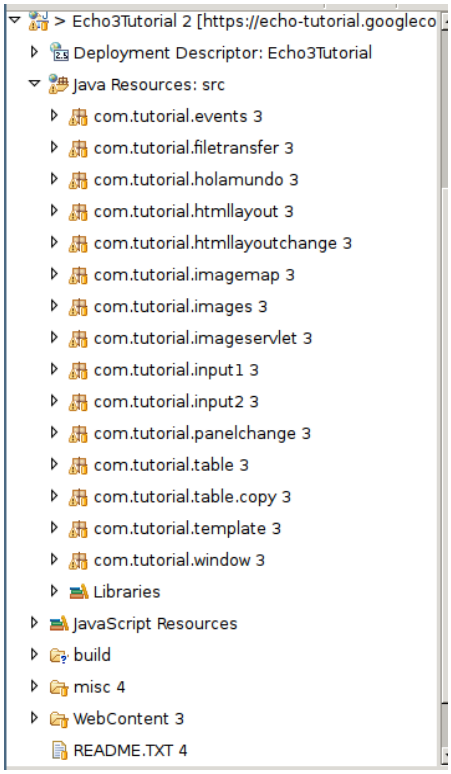
Luego es necesario hacer click en Next y en la siguiente pantalla se selecciona Browse:



Y se selecciona el directorio donde está el proyecto Echo3Tutorial dentro del workspace:



Luego se hace click en finish y el proyecto debería aparecer en el Project Explorer o el Package Explorer tal como se muestra a continuación:



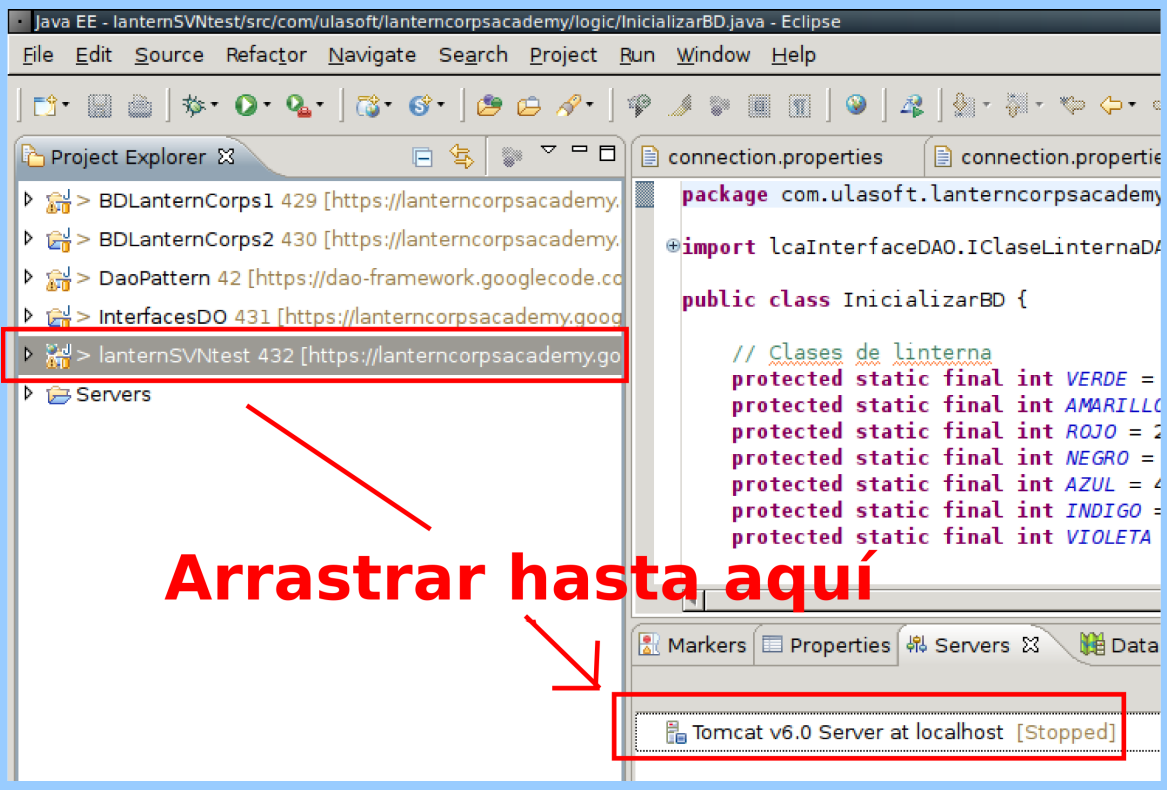
En general, desde este punto en adelante, todo lo que se menciona en el tutorial “tutorial-hola-mundo-echo3” aplica para el proyecto Echo3Tutorial.

PASO 3:

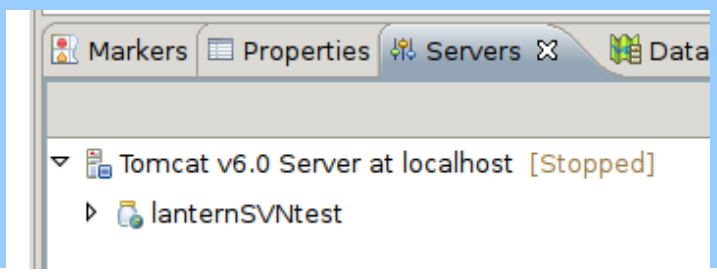
Repetir el Paso 11 y el Paso 12 del tutorial para poner a correr LCA y BloodTime (tutorial-desarrollo-lca-bloodtime.odt) pero para el proyecto Echo3Tutorial en lugar de LCA y BloodTime (Aplicar sentido común aquí). A continuación se reproduce la información de dicho tutorial:

tutorial-desarrollo-lca-bloodtime, PASO 11:

Ahora es necesario asociar el proyecto principal de LCA (lanternSVNtest) con el servidor que acabamos de crear, esto se hace simplemente ***seleccionando y arrastrando*** el proyecto lanternSVNtest desde el Project Explorer hasta el servidor que acabamos de crear:



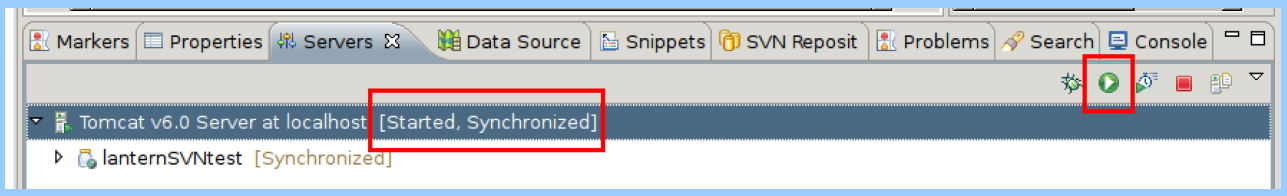
Y la vista Servers debe quedar:



Nota: En el caso de Blood Time es necesario arrastrar el proyecto “BloodTimeProject”

tutorial-desarrollo-lca-bloodtime, PASO 12:

Ahora es necesario arrancar el servidor, para esto hacemos click en el servidor que acabamos de crear en la vista Servers y luego hacemos click en el botón de ejecutar a la derecha:



Si todo sale bien, es posible que se vean algunos mensajes en la consola de eclipse, y luego el estado del servidor pasará de “Stopped” a “Started, Synchronized” tal como se muestra en la figura.

PASO 4:

Finalmente, es posible correr las distintas aplicaciones que trae el tutorial. Por ejemplo, para correr una de las aplicaciones básicas, se abre un navegador y se apunta a la siguiente dirección:

http://127.0.0.1:8080/Echo3Tutorial/window.echo

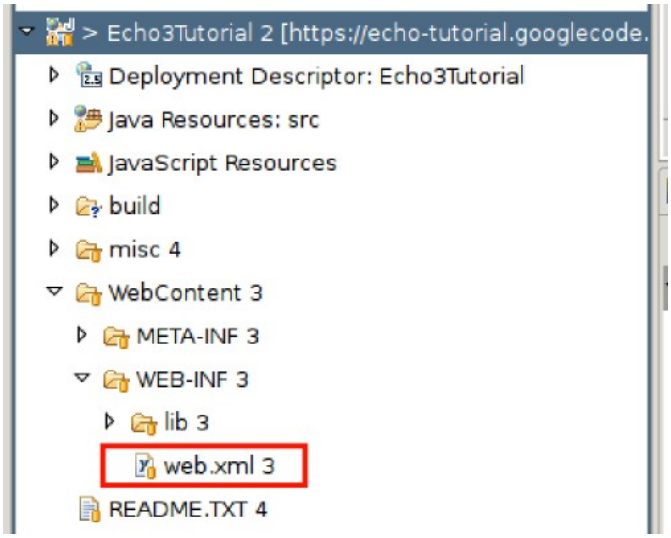
Debería verse lo siguiente:

Nota 1:

El presente tutorial asume que ya existe un “target runtime” definido (si no no se podría hacer el Paso 3). Para ver como definir un “target runtime” vea el tutorial para ejecutar BloodTime o LCA (tutorial-desarrollo-lca-bloodtime).

Nota 2:

Para ver las distintas aplicaciones que están incluidas en el tutorial es necesario revisar el archivo web.xml que se puede encontrar en:



Este archivo tiene pares de etiquetas `servlet` y `servlet-mapping` del tipo:

```
<servlet>
      <servlet-name>WindowServlet</servlet-name>
      <servlet-class>com.tutorial.window.WindowServlet</servlet-class>
</servlet>

<servlet-mapping>
      <servlet-name>WindowServlet</servlet-name>
      <url-pattern>/window.echo</url-pattern>
</servlet-mapping>
```

Y la etiqueta `servlet-mapping` tiene una etiqueta `url-pattern`. En este caso, la etiqueta `url-pattern` tiene un valor de “`/window.echo`”, lo que significa que podemos acceder al URL:

http://el-host-que-sea:8080***/window.echo***

Si reemplazamos “el-host-que-sea” por localhost o por 127.0.0.1 que es lo usual cuando estamos programando y tenemos el servidor instalado de forma local entonces tenemos:

http://127.0.0.1:8080/Echo3Tutorial/window.echo

Si se busca en el archivo web.xml todas las entradas `servlet-mapping` y se aplica esta misma lógica para todas aquellas donde el `url-pattern` es algo como "/lo-que-sea.echo" entonces se pueden encontrar todos las aplicaciones de ejemplo que tiene el tutorial.

Adicionalmente, si se desea saber que clases son las que implementan cada aplicación de ejemplo se puede detallar la etiqueta `servlet-class` hija de la etiqueta `servlet` correspondiente, luego sólo se debe buscar la clase y paquete especificados en el directorio `src` del proyecto.

Por ejemplo, y a modo de resumen, si se tiene:

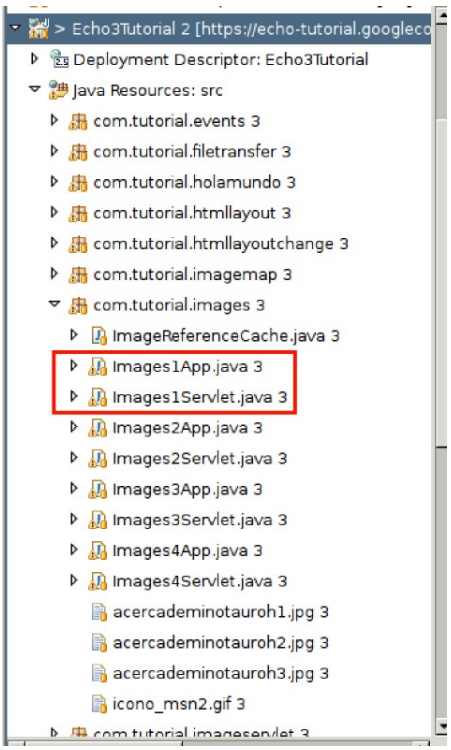
```
<servlet>
	<servlet-name>Images1Servlet</servlet-name>
	<servlet-class>com.tutorial.images.Images1Servlet</servlet-class>
</servlet>

<servlet-mapping>
	<servlet-name>Images1Servlet</servlet-name>
	<url-pattern>/images1.echo</url-pattern>
</servlet-mapping>
```

Entonces, lo que está marcado en rojo nos permite deducir que el URL de este ejemplo será:

http://127.0.0.1:8080/Echo3Tutorial/images1.echo

Y lo que está marcado en azul nos permite determinar que el servlet que sirve de punto de entrada para la aplicación es `com.tutorial.images.Images1Servlet` y que se puede encontrar en el `src` del proyecto tal como muestra la siguiente figura:



En general, el cuadro siguiente resume los ejemplos existentes (Ojo, lo estoy haciendo de memoria muy rápido así que me puedo equivocar):

Nombre / Servlet	**Dificultad**	**URL**	**Descripción**
`HolaMundoServlet`	`Elemental`	`/holamundo.echo`	Hola Mundo en Echo3
`WindowServlet`	Básico	`/window.echo`	¿Cómo hacer una ventana interna en Echo3?
`InputTextServlet1`	Básico	`/inputtext1.echo`	Uso de controles básicos (Cajas de texto, botones, etc). Este tutorial explica el uso básico de controles pero muestra la forma INCORRECTA de como se organiza una aplicación (incorrecta en cuanto a la falta de arquitectura y de orientación a objetos). La forma correcta se muestra en el siguiente tutorial.
`InputTextServlet2`	Básico	`/inputtext2.echo`	Uso de controles básicos (Cajas de texto, botones, etc). El valor de este tutorial es que muestra la forma correcta en que se debería organizar el código (una clase por formulario, herencia de window frame, etc). Si mal no recuerdo es la misma implementación del tutorial anterior pero con la adecuada orientación a objetos.
`TemplateServlet`	Medio	`/template.echo`	No está funcionando en Echo3, es un residuo de Echo2
`HtmlLayoutServlet`	Medio	`/htmllayout.echo`	Muestra la forma de hacer “layout” (organizar los componentes, paneles, cajas de texto, botones, etc) usando un layout escrito manualmente en Html, esto es necesario para casi cualquier aplicación que quiera hacer una organización visual compleja de los componentes
`PanelChangeServlet`	Medio-Avanzado	`/panelchange.echo`	Muestra como usar SplitPane, pero fundamentalmente como tener una estructura de código que permita tener una ventana principal y distintos paneles que cambian dependiendo de lo que el usuario seleccione en un “menu” o en un ”toolbar”. Esto es bastante trivial de hacer si se hace lo

			suficientemente organizado, pero puede ser un verdadero dolor de cabeza si no se estructura el código de forma adecuada
`HtmlLayoutChangeServlet`	Medio-Avanzado	`/htmllayoutchange.echo`	Similar al ejemplo anterior, sólo que utilizando un html layout. Además muestra como cambiar dinámicamente el html layout en respuesta a un evento de usuario. Este ejemplo es fundamental para hacer aplicaciones complejas que constan de múltiples paneles y vistas (de hecho es la base de BloodTime y LCA).
`EventsServlet`	???	`/events.echo`	Se desconoce el estado de este ejemplo, de hecho los fuentes están perdidos...
`TableServlet`	Avanzado (Bastante)	`/table.echo`	Muestra como hacer una tabla en Echo3 utilizando un pequeño framework desarrollado para tal fin. El jar que contiene el framework se llama cleda-echo-table-0.9.0.jar y los fuentes de dicho framework están en misc. En general, la tabla está implementada en base al modelo MVC y es muy similar a la de Echo2 y a la de Swing.
`Images1Servlet`	Básico	`/images1.echo`	Mánejo básico de imagenes en Echo3
`Images2Servlet`	Básico	`/images2.echo`	Manejo de múltiples imágenes. Muestra los inconvenientes de crear un ImageReference para imágenes que se reutilizan o se muestran mucho
`Images3Servlet`	Medio	`/images3.echo`	Resuelve el problema del ejemplo anterior (`Images2Servlet`) utilizando un cache de imagenes en forma de un singleton (`ImageReferenceCache`)
`Images4Servlet`	Básico	`/images4.echo`	No funciona al 100% pero sirve para el propósito planteado, se estrella al darle next. El objetivo es mostrar que el navegador maneja eficientemente las imágenes si se especifica un URL externo
`ImageMapServlet`	Medio-Avanzado	`/imagemap.echo`	Muestra como se puede utilizar un “ImageMap” en Echo3. Un image map permite mostrar una imagen y definir áreas particulares de la imagen

			sobre las que los usuarios pueden hacer click y enviar eventos (tal cual como si fuera un botón). El mapa de planetas de LCA está hecho con este principio.
`ImagesAppServlet`	Avanzado	`/imagesapp.echo`	Muestra como mostrar desde Echo3 una imagen dinámica (por ejemplo que viene de una Base de Datos). En el ejemplo las imágenes se leen de disco, pero el mismo principio se puede aplicar como si vinieran de un Blob de una BD. El ejemplo coopera con el servlet `ImagesDataServlet` que es el que se encarga de generar de forma dinámica las imágenes según un ID proporcionado. Para ver el resultado introducir los números 1, 2, o 3 en la caja de texto y hacer click en go. Cualquir otra cosa en la caja de texto muestra la imagen de error (la x roja)
`FileTransferServlet`	Avanzado	`/filetransfer.echo`	Mustra como hacer uploads y downloads desde Echo3 utilizando la biblioteca “FileTransfer”. Esta biblioteca cambió un poco desde Echo2 y fue necesario ajustar un poco el ejemplo.

Ejemplos pendientes por hacer:

1.- Un image map en el que la imágen y los objetos (botones / opciones) mostrados sean completamente dinámicos (Ej, vengan de una BD y que la imagen se construya al vuelo con Buffered Image).

2.- Un ejemplo que muestre las habilidades de “Server Push” de Echo3.