Ejercicio 1 (sin usar jstl)

Página **visor_imagenes.jsp**, que presenta, en principio, el siguiente formulario:

1. El combo carga todas las imágenes que tengamos en una carpeta del proyecto (zona web)
   - **visor_imagenes.jsp** dispone de una constante con el nombre de dicha carpeta
   - La página hará uso de un método que devuelve una lista con las imágenes de una carpeta:
     ArrayList<Imagen> imagenesDeCarpeta(String nomcarpeta)
   - Imagen es un javabean que deberás crear con los siguientes atributos:
     - ruta: es el nombre completo de la imagen (p.e: "img/vacas en el monte.jpg")
     - nombre: es el nombre recortado de la imagen (p.e: vacas en el monte)
     - tamanio: es el tamaño en bytes del fichero correspondiente a la imagen

     Dispondrá del constructor que estimes necesario, getters y un método:
     - tamanioDesglosado, que devuelve un String con el tamaño de la imagen desglosado en Mb, Kb y bytes (p.e, para un tamaño de 8499106 bytes devolvería el string "8 Mb 107 Kb 930 bytes"
   - Para recorrer las imágenes de una carpeta, puedes usar la clase **File** y su método list. Recuerda, al abrir la carpeta para leer su contenido: getServletContext().getRealPath(CARPETA)

2. Al pulsar el submit, la misma página, en su zona inferior, cargará la imagen seleccionada con el tamaño (width) seleccionado. Además, mostrará el tamaño de la imagen desglosado en mb, kb y bytes.

Tamaño 8 Mb 15 Kb 188 bytes

Ejercicio 2 (sin usar jstl)

Nuestro proyecto tiene un archivo de texto con productos en venta: **productos.txt** con líneas (categoría ; producto)

```
100;Monitor 22 pulgadas
100;Teclado wireless
300;Pendrive 32 Gb
200;Laptop 11 pulgadas
200;HDD 2 Tb
300;HDD 3,5pulgadas 2 Tb
100;Alfombrilla
100;Ventilador CPU
….. ; …..
```

Se ejecuta un Servlet **ServletPrepararProductos** al que se le pasa por GET una categoría (ServletPrepararProductos?categ=100)

- El servlet deja preparada en la sesión una colección (decide tú de qué tipo) con los productos del fichero de dicha categoría (haz lo necesario para que, en cada sesión se lea el fichero 1 sola vez)
- Si no se pasa una categoría como parámetro, la colección incluirá todos los productos del fichero
- El nombre del fichero es un parámetro del servlet
- Pasa el control a **compra.jsp**

El script **compra.jsp** muestra una tabla con todos los productos que se guardaron y da la posibilidad de adquirirlos, mediante un botón de submit junto a cada uno.
Cada submit envía el formulario correspondiente al propio script **compra.jsp**
La sesión/navegación actual, tiene que guardar en otra colección, qué productos y en qué cantidad se van adquiriendo.
Piensa qué colección es la más idónea.

PRODUCTO	PEDIR
Monitor 22 pulgadas	Adquirir unidad
Teclado wireless	Adquirir unidad
Alfombrilla	Adquirir unidad
Ventilador CPU	Adquirir unidad

Después de adquirir, por ejemplo, 2 monitores y 1 ventilador, mostrará:

PRODUCTO	PEDIR	
Monitor 22 pulgadas	Adquirir unidad	2 unidades
Teclado wireless	Adquirir unidad	
Alfombrilla	Adquirir unidad	
Ventilador CPU	Adquirir unidad	1 unidades

Finalmente, se incluirá dentro de compra.jsp la salida de muestracarro.jsp, que muestra el contenido del carro actual:

PRODUCTO	PEDIR	
Monitor 22 pulgadas	Adquirir unidad	2 unidades
Teclado wireless	Adquirir unidad	
Alfombrilla	Adquirir unidad	
Ventilador CPU	Adquirir unidad	1 unidades

TU CARRO

- **MONITOR 22 PULGADAS:**2 unidades
- **VENTILADOR CPU:**1 unidades

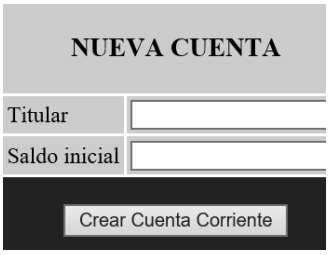
Esto lo muestra otro script **muestracarro.jsp**, que se incluye en compra.jsp mediante una acción jsp

Ejercicio 3 (usar jstl)

Para este ejercicio: Utilizar jstl en las 2 páginas jsp

Crear un bean **Cuenta** que represente una Cuenta bancaria. Tendrá 2 atributos (titular y saldo), Además de lo que requiere un bean, usaremos 2 métodos gastar e ingresar, que actualicen el saldo con la cantidad recibida como parámetro.
Si se intenta gastar más de lo que se tiene no se dejará gastar nada. El método gastar devuelve si se ha podido gastar o no.

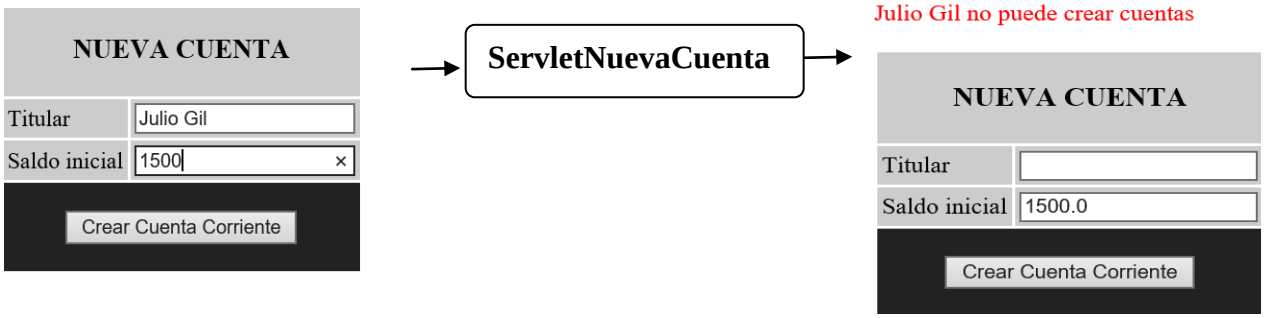
La primera página que se cargará (**nuevacuenta.jsp**) contiene este formulario



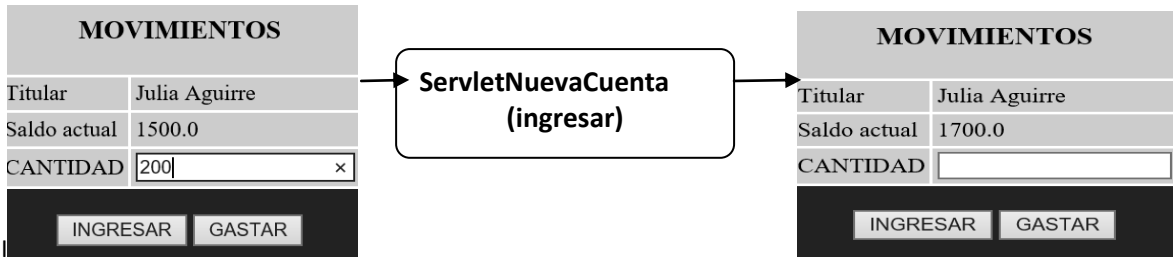
El formulario se envía al servlet **ServletNuevaCuenta**, que realizará lo siguiente:

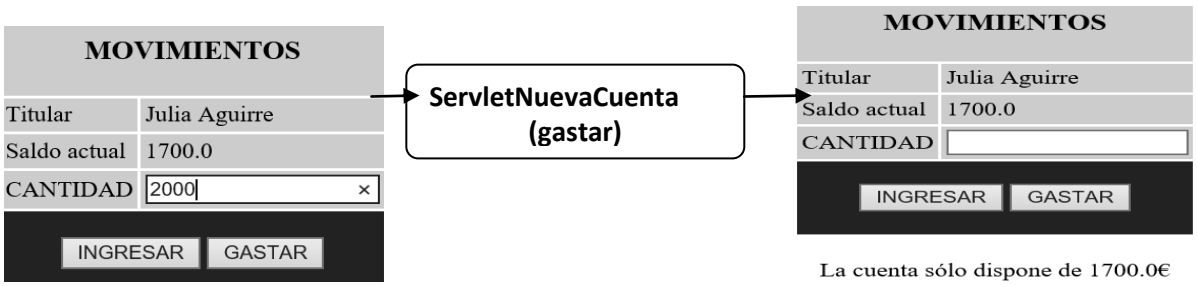
1. Si detecta algún error para crear una nueva cuenta, redirige la petición a **nuevacuenta.jsp**, preparando previamente:
   - Una lista con el texto de los errores: titular vacío, saldo negativo o titular prohibido (La aplicación web tiene como atributo una lista usuarios (nombres) no autorizados para crear cuentas)
   - Una "Cuenta" en la que sí se hayan cargado los atributos "no erróneos", de cara a repoblar el formulario

   Modifica **nuevacuenta.jsp** para que tenga en cuenta el reenvío por error.



2. Si todo va bien, se crea una cuenta en la sesión y se pide la página **movimientos.jsp**
   movimientos.jsp refleja en una tabla la cuenta de la sesión y permite realizar en ella ingresos/gastos
   El formulario se envia a un nuevo servlet **ServletMovimientos** que:
   - Realizará el ingreso/gasto correspondiente y dirigirá la petición de nuevo a **movimientos.jsp**
   - Si se ha dado algún error (no se ha podido gastar o cantidad errónea) se guarda antes del reenvío.





MEJORAS:

- Si no lo has hecho, controla los accesos "no lícitos" a recursos, redirigiendo en cada caso, al punto más adecuado.
  Ejemplos de accesos ilegales: acceder a los 2 servlets por GET, acceder a movimientos.jsp o a ServletMovimientos sin que exista una cuenta en la sesión, etc
  → Si se accede al script movimientos.jsp sin una cuenta en la sesión, el propio script redirige a **nuevacuenta.jsp**

- Crea una página ilegales.jsp, con un formulario para que el administrador añada los nombres de usuarios no autorizados a crear cuentas bancarias. Con la ayuda de un servlet ServletIlegales, se encargarán de:
  - Mostrar un formulario para añadir nuevos nombres
  - Mostrar los nombres prohibidos actuales
  - Mostrar un enlace al lado de cada nombre para permitir eliminarlo de la lista de ilegales

Ejercicio 4: Primitiva (Utilizar jstl)

Para este ejercicio: Utilizar jstl en las 2 páginas jsp

Bean Apuesta, que representa una apuesta de un jugador: 6 número entre 1 y 49 y 1 reintegro entre 0 y 9

- Con atributos: nombre del apostante, nº de administración, array de 6 números (los jugados) y reintegro (un número)
- Constructor: recibe apostante, administración, array de números jugados y reintegro.
  Si se da algún error (array recibido no tiene tamaño 6, o contiene números repetidos, o algún número no está entre 1 y 49, o reintegro no está entre 0 y 9…), genera una excepción personalizada de tipo BadBetException.
- Método toString

  ** Cuando se necesiten más métodos, se incluirán*

Bean Fecha:

- Con atributos: dia, mes y año
- Constructor
- Método toString
- boolean correcta(): que devuelve si la fecha es válida

  ** Cuando se necesiten más métodos, se incluirán*

Bean Primitiva: representa la combinación de números+reintegro ganadores del sorteo de la primitiva en una fecha concreta:

- Atributos: una fecha, un array con los 6 números ganadores y un reintegro
- Constructor: recibe la fecha y crea y carga de modo aleatorio tanto el array de números como el reintegro.
  - ➔ No puede haber repetidos en el array de números (ayúdate, si quieres, de otro método que detecte repetidos)
  - ➔ Después de generar el array con los números premiados, llama a otro método que lo deje ordenado ascendentemente
- Método privado numTocado, que recibe como parámetro un número y devuelve si es uno de los agraciados de la primitiva actual
- Método resultApuesta, que recibe una Apuesta y devuelve:
  - una cadena con los números acertados, + si se ha acertado el reintegro o no. Por ejemplo: “Aciertos:12 38. Sin reintegro”

  ** Cuando se necesiten más métodos, se incluirán*

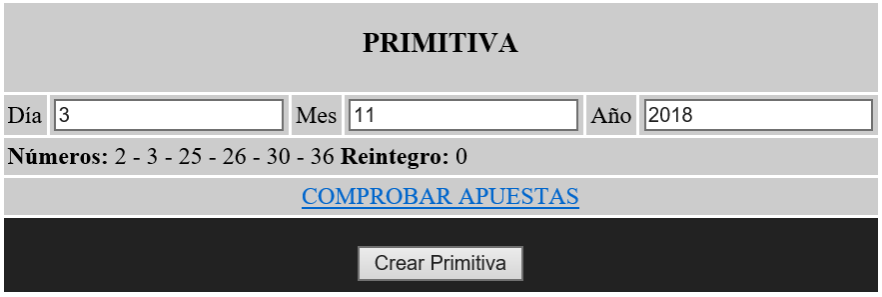
crear_primitiva.jsp

Contiene el siguiente formulario que se envía a **ServletCheckFecha**

ServletCheckFecha

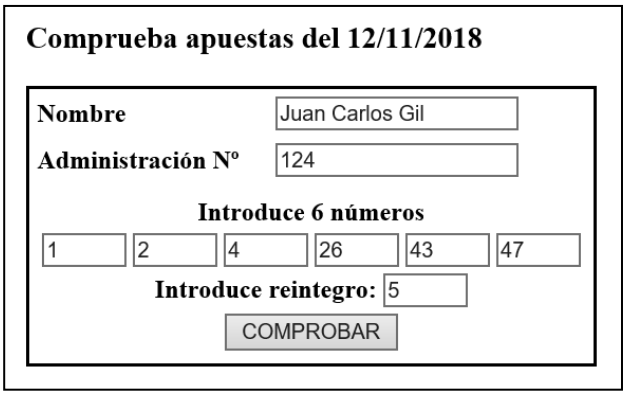
Comprueba que la fecha introducida en los cuadros de texto es correcta. Crea un bean Fecha y:

- Si es errónea, se redirige de nuevo a **crear_primitiva.jsp** (que mostrará además un mensaje de error)
- Si es correcta, crea una primitiva (bean) en dicha fecha y se redirige, de nuevo, a **crear_primitiva.jsp** (que muestra los datos completos de la primitiva creada y un enlace a **check_apuestas.jsp** para comprobar apuestas.



check_apuestas.jsp

Muestra el siguiente formulario que se envía al servlet **ServletCheckApuesta**



ServletCheckApuesta chequea la apuesta recibida y se redirige de nuevo a **check_apuestas.jsp**, dejando preparados los datos que consideres necesarios en los ámbitos que consideres necesarios.
check_apuestas.jsp, al volver del servlet, visualizará el resultado de la apuesta (si es válida) o un error en caso contrario.