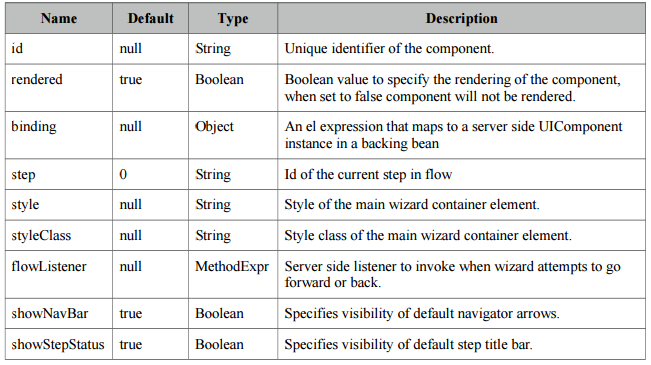
# Paneles

Los componentes de paneles son útiles para agrupar la información que mostremos en nuestra web, así como los formularios, logrando no saturar con demasiado contenido al visitante.

## Wizard



Con este componente podemos agrupar el contenido de largos formularios por categorías. De esta forma nuestro usuario no se agobiará si tiene que rellenar muchos campos. Una de sus ventajas es la validación por partes. Es decir, en la imagen mostrada por ejemplo, no te dejará acceder a rellenar “Address” sin antes haber validado correctamente los campos de “Personal”.

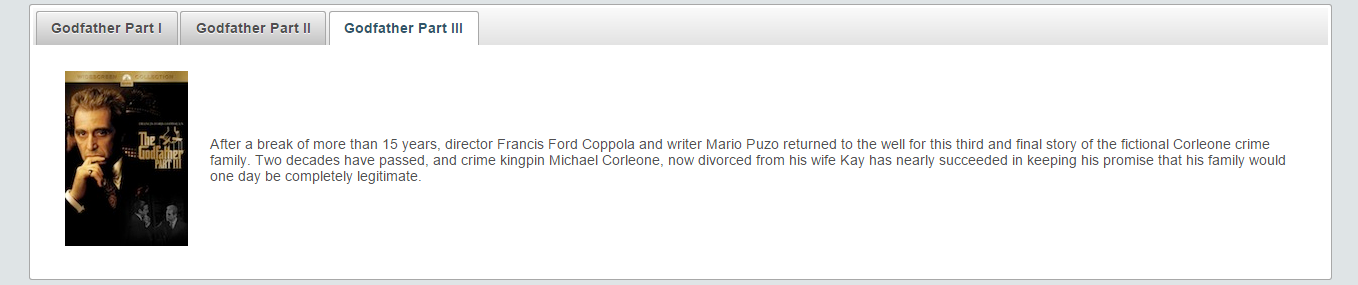


El uso de esta etiqueta es bastante sencillo. Siguiendo el ejemplo mostrado en la imagen, su correspondiente .xhtml sería el siguiente:

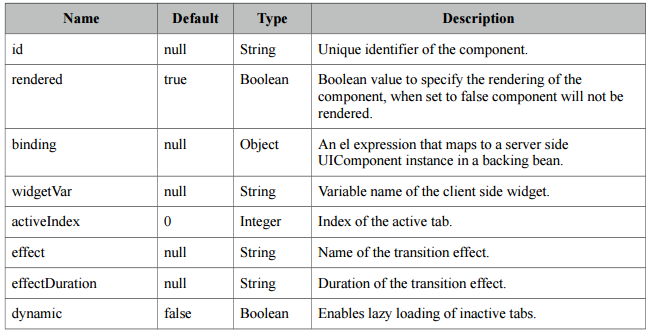


Como podemos observar, tan solo sería necesario especificar las agrupaciones de los campos del formulario, así como sus condiciones de validación, encargándose PrimeFaces de realizar el resto del trabajo.

## TabView



Esta etiqueta agrupa la información que antes mostraríamos como una lista en pestañas. Gracias a ella logramos reducir el contenido mostrado de cada vez al visitante, y dependiendo de los parámetros escogidos, incluso el ancho de banda consumido por la página al cargar las vistas sólo cuando el usuario pinche en ellas.

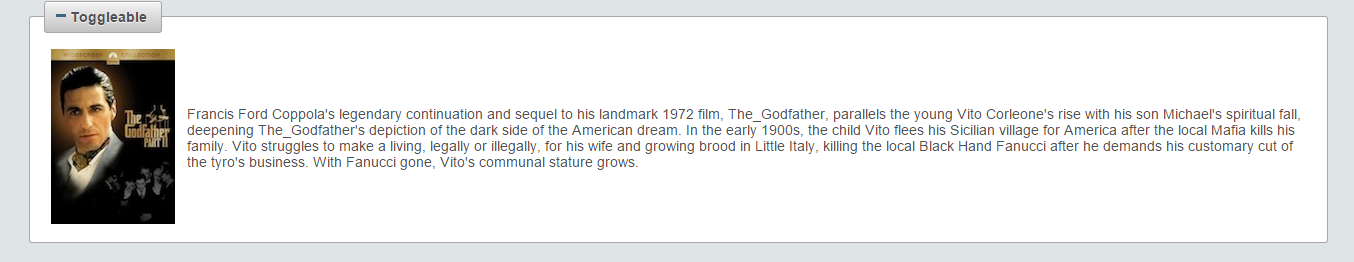


Su funcionamiento al igual que el resto de componentes de este framework es muy sencillo:

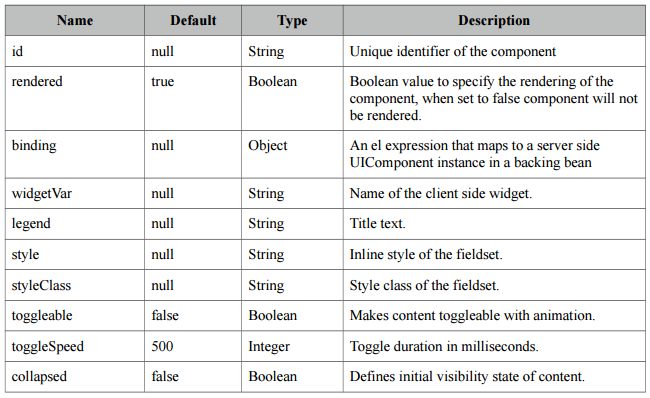


Simplemente, instanciamos el conjunto de pestañas con *<p:tabView>,* y definimos cada una con *<p:tab>*, pudiendo añadir el contenido que queramos dentro de ellas.

## Fieldset



Con <p:fieldset> logramos resaltar cierto contenido, del mismo modo que podemos darla opción al usuario de ocultarlo si este lo considerase necesario. Un buen ejemplo aparte del aquí mostrado, podría ser mostrar código al usuario convenientemente formateado dentro de un tutorial.

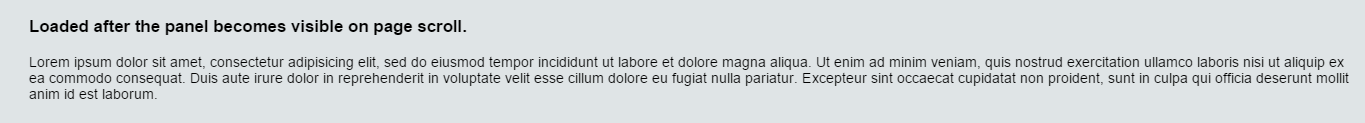


El funcionamiento de esta etiqueta no puede ser más simple:

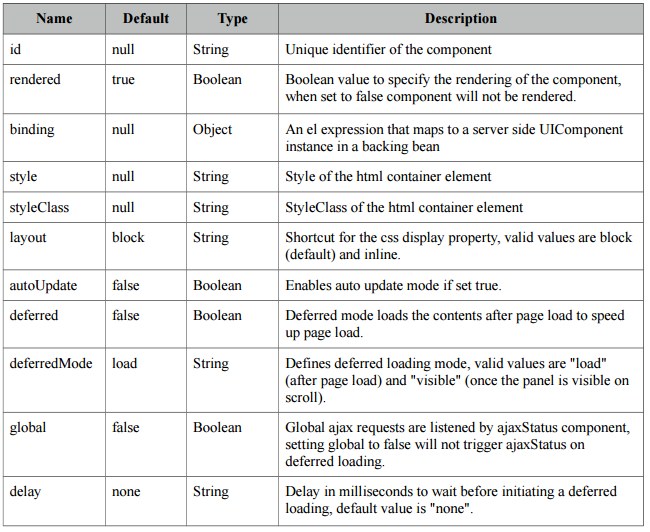


El componente se instancia con <p:fieldset>, y con los atributos toggleable, y toggleSpeed definimos respectivamente si el usuario puede ocultar el contenido de este, y la velocidad de la animación respectivamente.

## OutputPanel



Este componente permite agregar contenido después la descarga de la página. Gracias a él podemos reducir el ancho de banda consumido por el servidor, además de incrementar la velocidad de carga de la web. Es muy interesante, ya que los visitantes no siempre cuentan con una buena conexión, y pueden evitar la carga de partes de la web que en algunas circunstancias no les son relevantes.



Al igual que las anteriores su uso no tiene más complicación:

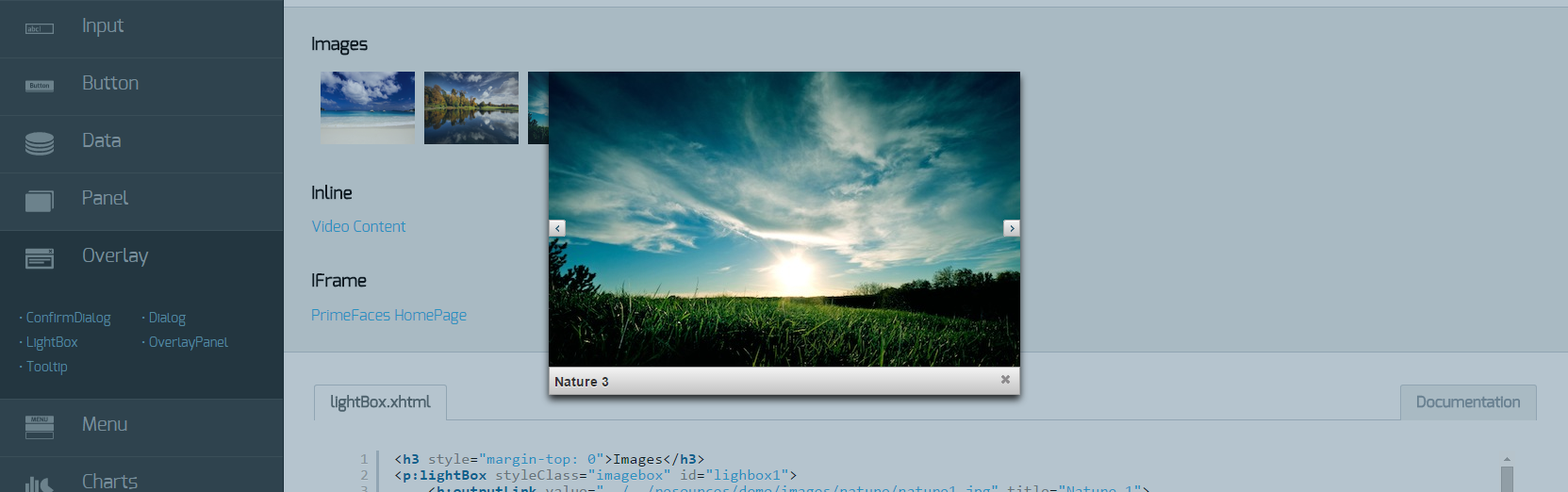


Se instancia con <p:outputPanel>, y con los atributos deferred, y deferredMode, establecemos que se cargue más tarde que el resto de la página, y cuando exactamente ocurrirá esto repectivamente.

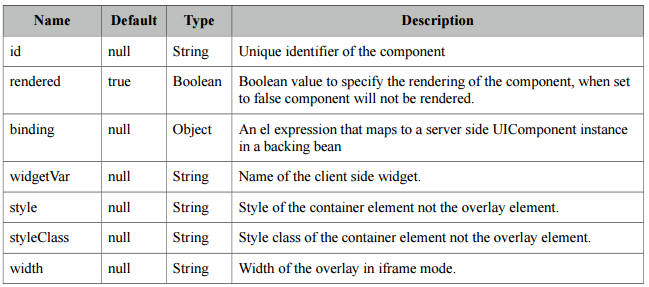
# Overlay

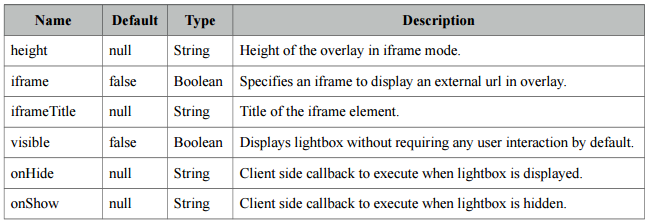
En esta categoría nos encontramos componentes con los que generar contenido extra sin romper la estructura de la página, y que pueden ser útiles para hacer la web más amigable de cara al usuario, así como para mejorar su rendimiento y aspecto.

## LightBox



Este elemento nos permite integrar contenido multimedia en un modal. Es una buena forma de mostrar contenido que ocupará un gran espacio en la pantalla, y que concentrará nuestra atención totalmente mientras es visualizado, como por ejemplo videos o imágenes en alta resolución.



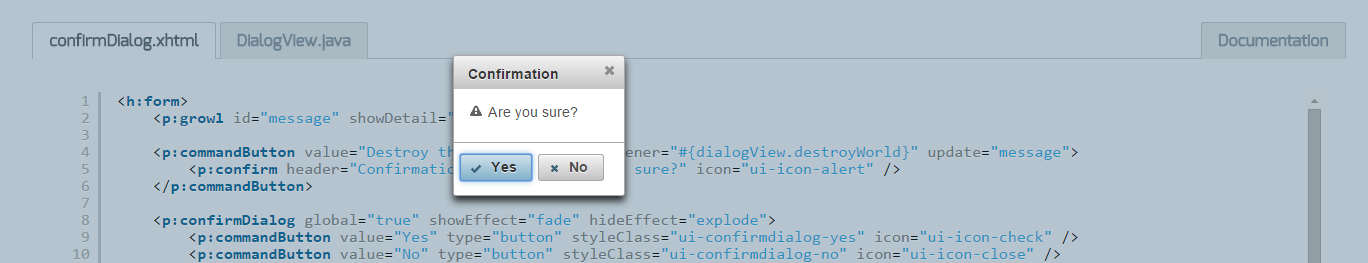


En el siguiente ejemplo vemos su funcionamiento con una galería de imágenes:

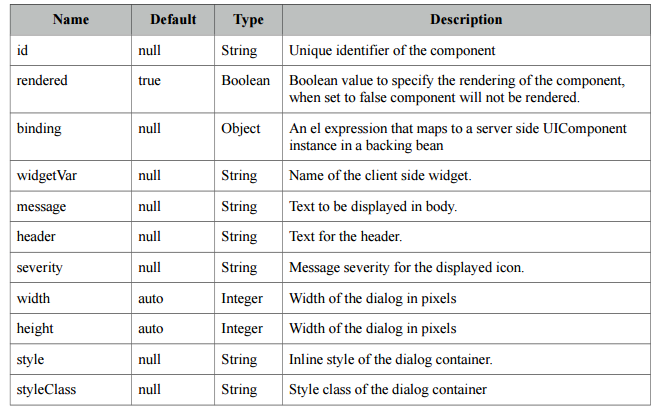


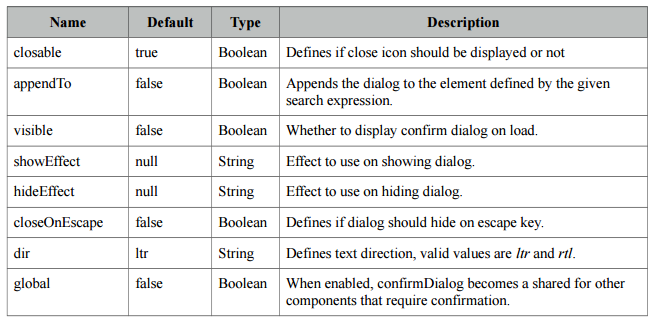
Para usarlo, como vemos sólo debemos instanciar el elemento con <p:lightBox>, y a continuación añadir los <h:outputLink> con el contenido a mostrar dentro de ellos.

## Confirm Dialog



Con este componente, mostraremos al usuario un panel de confirmación al realizar alguna acción “peligrosa”, como podría ser borrar su usuario de la web. Con él, evitaremos clicks accidentales en botones, así como dar una oportunidad a que el usuario recapacite al realizar acciones que no tengan vuelta atrás.



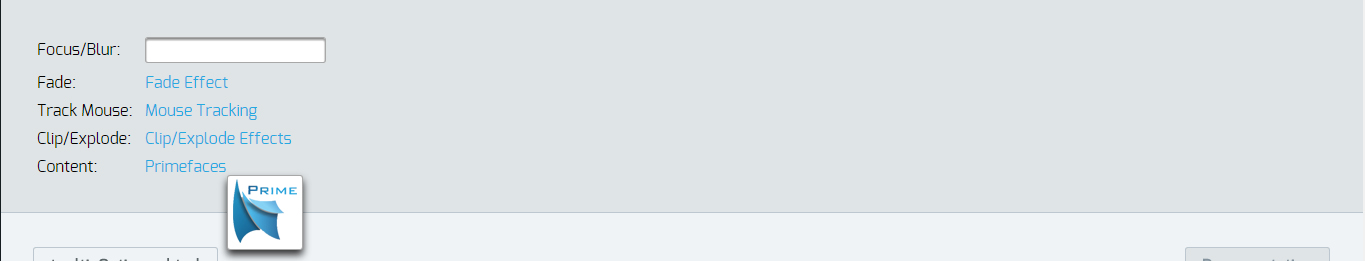


En este ejemplo se muestra un ejemplo de botón que activaría este diálogo:

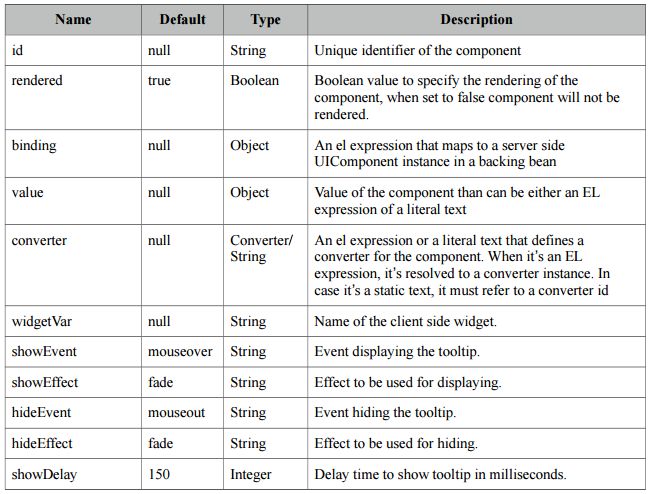


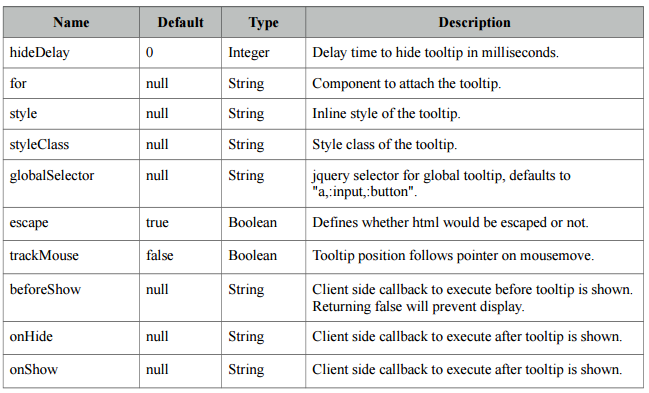
Si nos fijamos en el código necesario para agregarlo a nuestra web, necesitaremos un commandButton que al pulsarlo activará el diálogo de confirmación. Dentro este agregaremos el mensaje a mostrar al usuario con <p:confirm>, y finalmente un <p:confirmDialog> junto a este botón en el que agregaremos las distintas opciones a ofrecer al usuario, así como efectos o animaciones que queramos que este tenga. Este <p:confirmDialog> puede ser global y ser usado en toda la página, activado por distintos botones.

## Tooltip



Este componente agregará funcionalidad a los ya conocidos tooltips, permitiéndonos agregar imágenes, efectos, o que se muevan junto con el cursor del ratón.



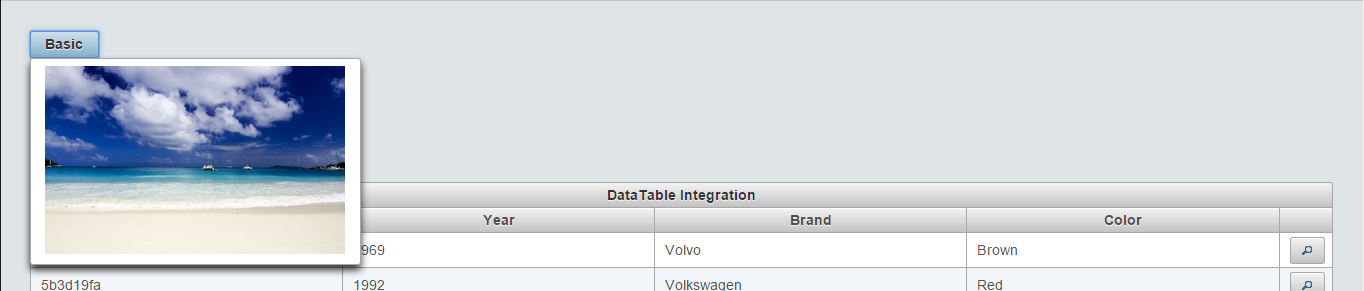


Una muestra del código es la siguiente:

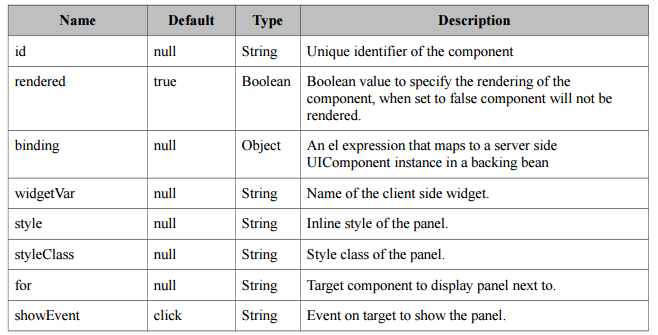


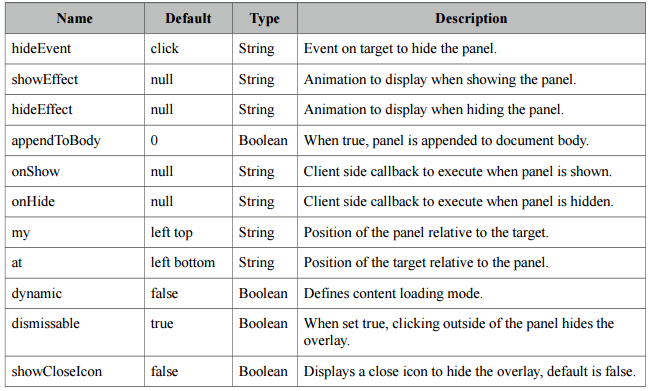
Agregarlo es muy fácil, y ayudará a hacer nuestra web más fácil de usar por el usuario. Con <p:tooltip> instanciamos el elemento, y con el atributo “for” le indicamos a que otro objeto pertenece.

## Overlay Panel



Este componente nos permite mostrar contenido como imágenes, tablas, o simplemente texto al activar un “trigger” como un botón. Este contenido puede cargarse sólo al activar el correspondiente “trigger”, reduciendo el tráfico consumido por la web, y acelerando la carga de la página en conexiones lentas.





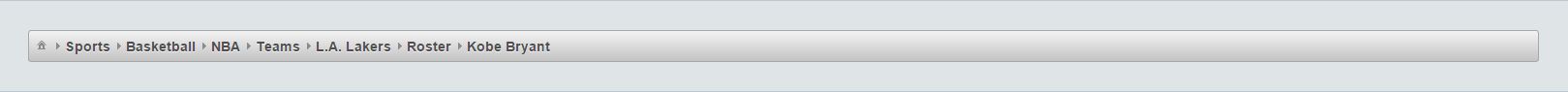


Como podemos observar, el panel se instancia con una etiqueta <p:overlayPanel>, que contendrá el contenido a cargar, y en el atributo for el valor del correspondiente “trigger”.

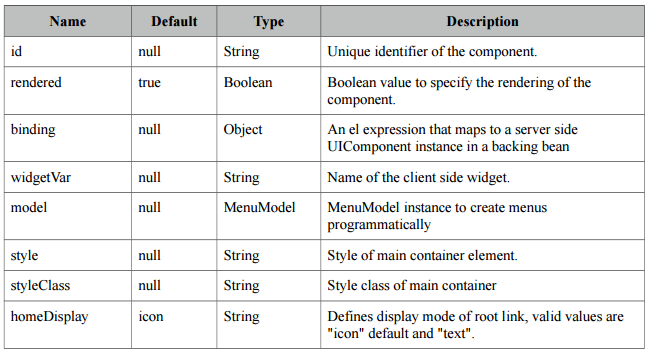
# Menu

En esta categoría de elementos nos encontramos básicamente con distintos estilos de agrupaciones de links y/o botones, que nos ayudarán a indicar de manera más clara al usuario donde se encuentra y sus distintas opciones de navegación.

## BreadCrumb



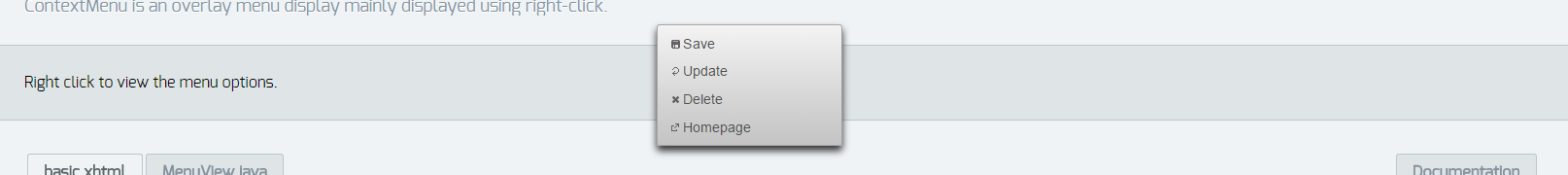
Este componente muestra al usuario en que sección de la web se encuentra, así como sus secciones padre en forma de links. Con esto logramos dar una mejor navegabilidad a la web.



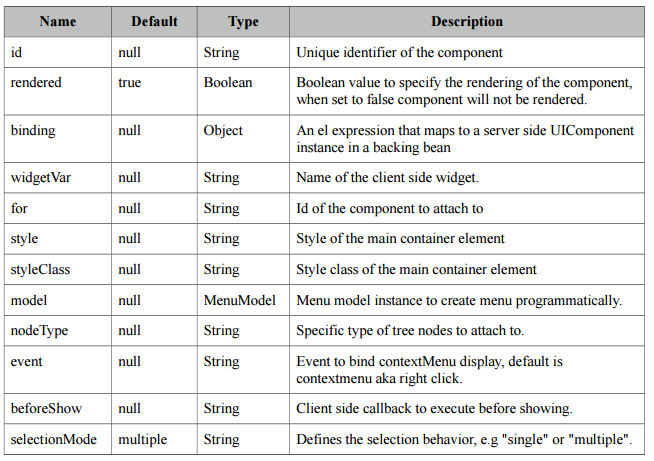


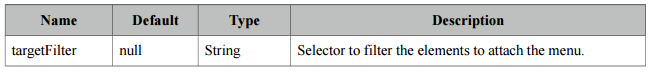
Como vemos el código no tiene más complicación, simplemente es una etiqueta <p:breadCumb> que contendrá dentro un listado de enlaces ordenados de mayor a menor jerarquía. Sería interesante generar estos enlaces siempre de manera dinámica y no a mano.

## Context Menu



Gracias a este elemento podremos sustituir el menú abierto por defecto al clicar con el botón derecho en cualquier parte de la web, mostrando un listado de acciones personalizadas.





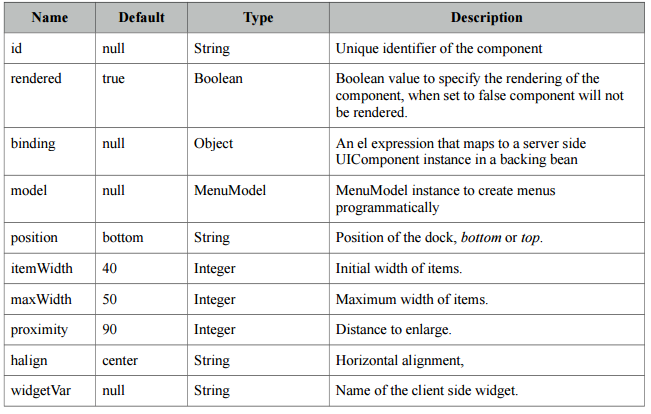


En el código podemos observar como tan sólo tenemos que instanciar un <p:contextMenu> con un listado de <p:menuitem> que serán los enlaces a nuestras acciones.

## Dock



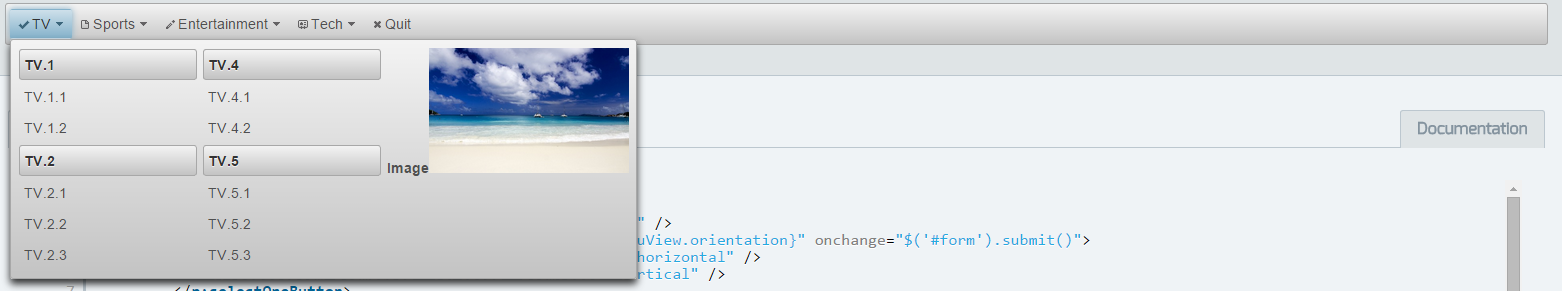
Este componente inspirado claramente en los mismos docks vistos en algunos sistemas Unix (Mac OSX por ejemplo), nos permite mostrar de una manera llamativa enlaces en nuestra web.



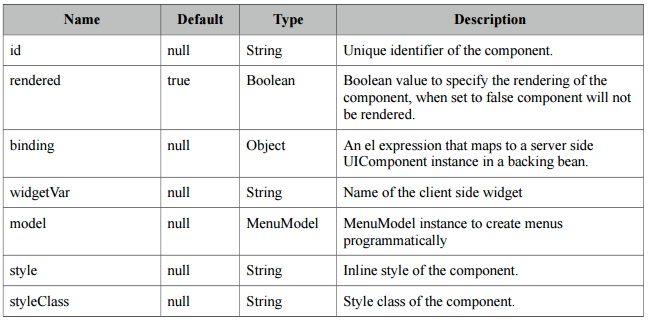


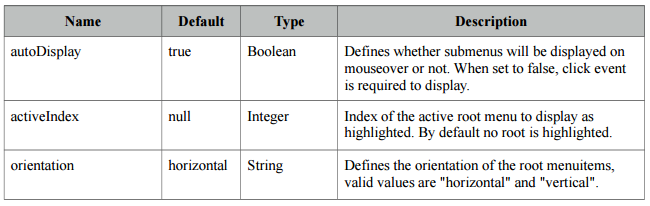
Como ya estamos acostumbrados con este framework el código es muy intuitivo, tan solo teniendo que crear una etiqueta <p:dock> que contenga los <p:menuItem> con sus iconos y enlaces correspondientes.

## MegaMenu



Con él podremos mostrar un completo menú con opciones de todo tipo en nuestra web, ya sea horizontal o vertical.

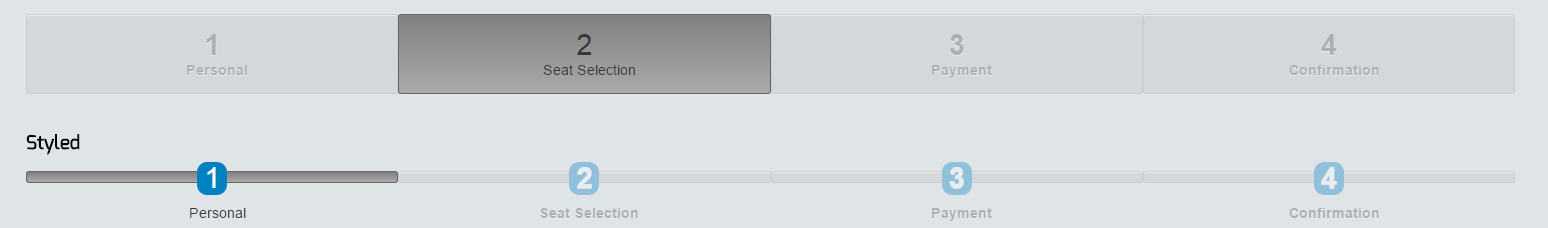




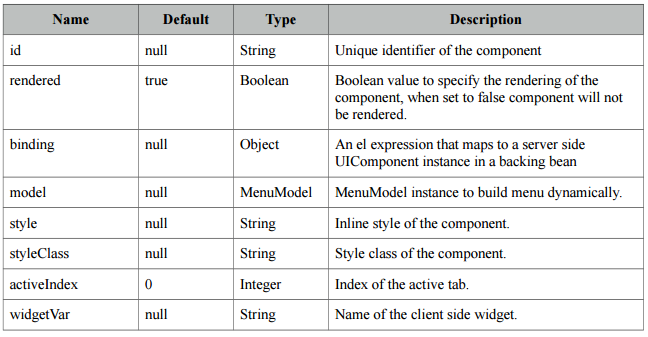


El código esta vez es algo más complejo, dada la dimensión del elemento que estamos creando, pero sigue siendo muy legible y fácil de manejar. El procedimiento básico para crear este menú será agregar una etiqueta <p:megaMenu> con una serie de <p:submenu> dentro en los que podremos agregar más contenido y menús o enlaces.

## Steps



Al igual que el componente wizard, los steps nos permiten subdividir un gran formulario en unos más pequeños. Muy útil para webs con carritos de la compra para indicar al usuario los pasos que le quedan hasta completar su pedido, e ir comprobando poco a poco que los datos introducidos son correctos.



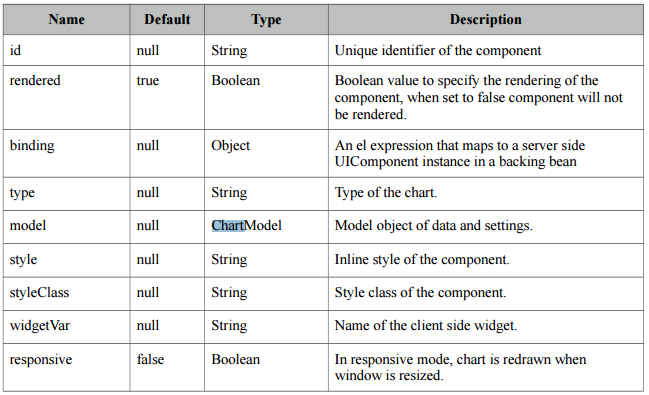


Para agregarlo en nuestra web tendremos que crear un <p:steps> con una serie de <p:menuitem> dentro, y un atributo activeIndex que dirán en que parte nos encontramos.

# Chart

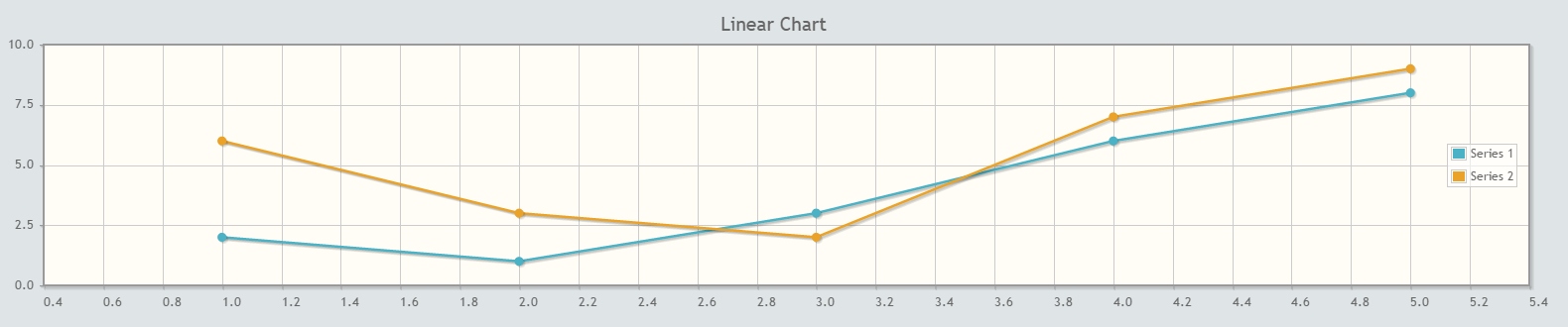
Aquí nos encontraremos básicamente gráficos de todo tipo, con los que mostrar información de manera vistosa, y más fácil de interpretar y comparar para nuestros visitantes.

En el caso de los charts los atributos son comunes para todos.



Veremos ahora algunos ejemplos de los gráficos que nos ofrece este framework con varias de sus opciones más interesantes.

## Line



Clásico gráfico lineal, que será generado dentro de un elemento de tipo <canvas> con los datos suministrados por la aplicación. De necesitar otro tipo de gráfico, tan solo tendríamos que cambiar el atributo type por otro modelo que se ajustase más a nuestras necesidades.

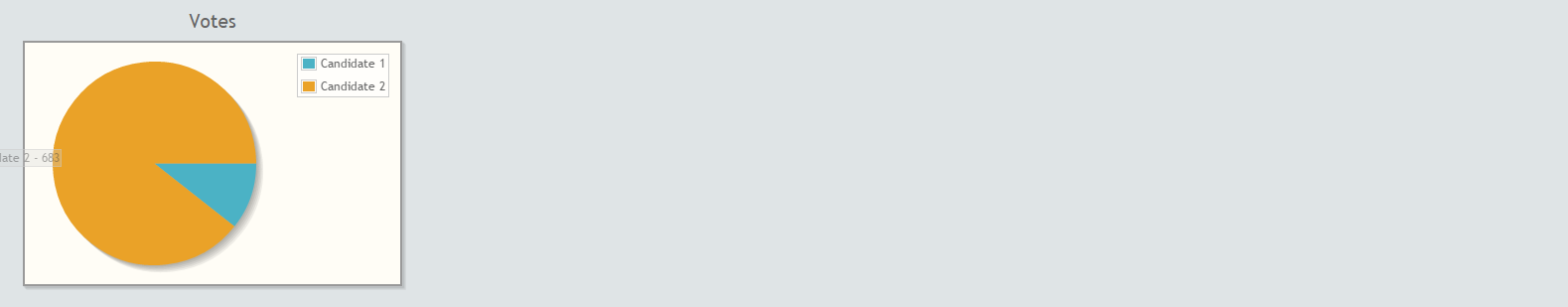


Esta vez ya no será suficiente con añadir contenido tan sólo en el .xhtml que necesitemos. Los gráficos necesitan tomar los datos de algún ManagedBean como veremos a continuación



En nuestro ManagedBean correspondiente debemos añadir un objeto de tipo LineChartModel, e inicializarlo con los datos correspondientes, ya sea manual o automáticamente, así como configurar la escala y leyenda si fuese necesario.

## Live



La peculiaridad de este tipo de gráfico es que se actualiza automáticamente cada cierto intervalo de tiempo, sin necesidad de recargar la página. Si los datos cambian rápidamente y es importante tenerlos siempre actualizados, este puede ser un gran componente en nuestra web.

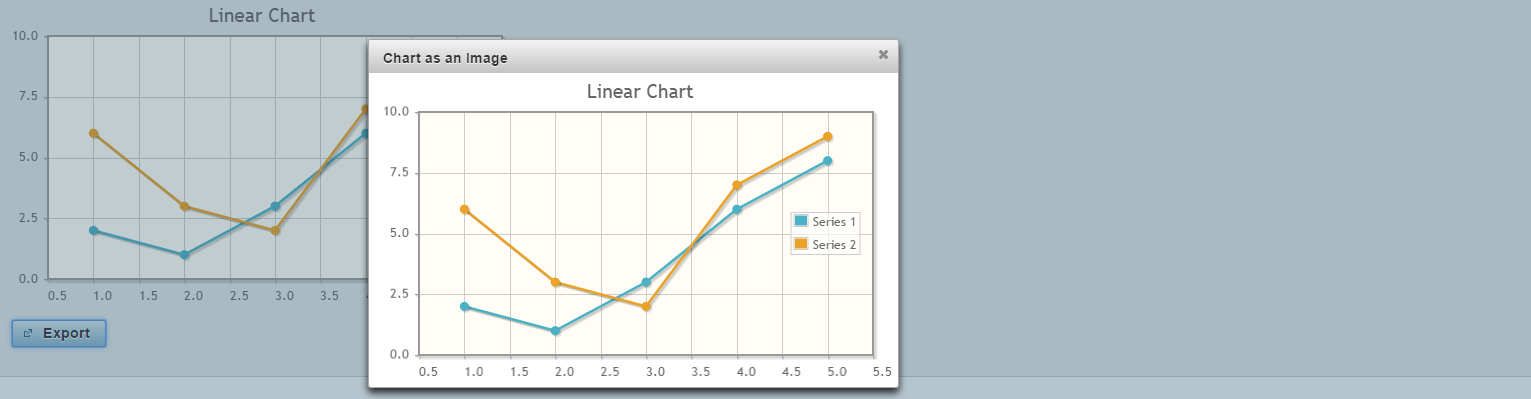


Para que el gráfico se actualice automáticamente deberemos añadir junto a la etiqueta <p:chart>, una <p:poll>, que referencie en el atributo update al gráfico, y en interval el tiempo en segundos para cada petición de datos nueva.



En el ejemplo, los datos simplemente son dos enteros aleatorios, que se generan en el ManagedBean correspondiente como podemos observar.

## Export

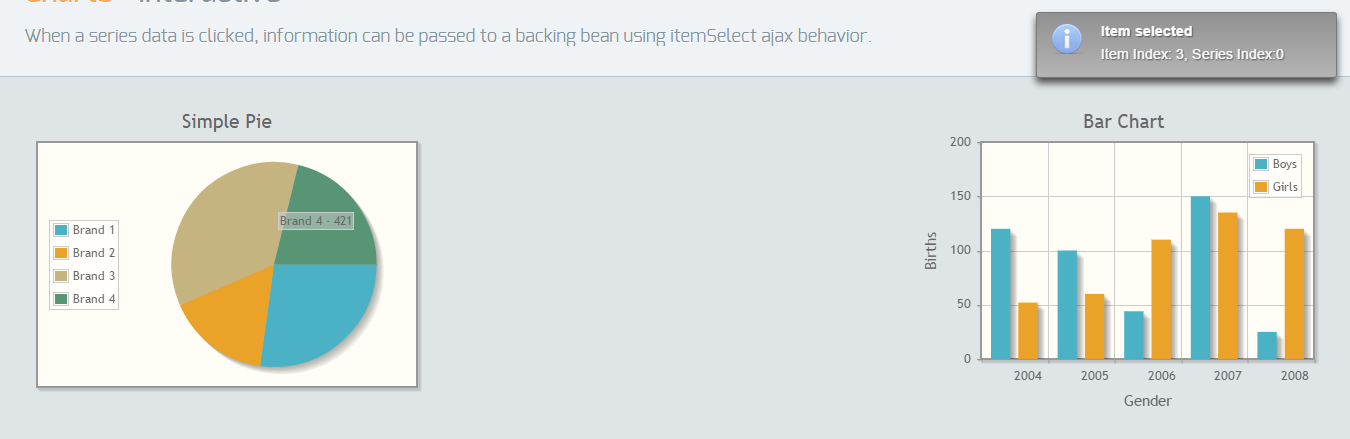


Utilizando un poco de javascript podemos permitir al usuario descargar el gráfico que está visualizando en forma de imagen.



Como se puede ver en el código, aparte del básico <p:chart>, tendremos también que añadir un script, que exporte el gráfico a una imagen.

## Interactive



Podemos permitir a nuestros visitantes interactuar con los gráficos clicando sobre ellos gracias a AJAX, actualizando otros componentes de la página o mostrando una información más detallada por ejemplo.



En nuestro .xhtml tendremos que añadir dentro del chart una etiqueta de tipo <p:Ajax>, a la cual le diremos el evento al que tiene que estar esperando, el método que se encargará de procesar los datos del evento, y finalmente si tiene que actualizar algún otro componente de la web. En el ejemplo mostrado se actualiza un componente de tipo <p:growl>.



En el ManagedBean que se encargue de gestionar la petición AJAX, tendremos que crear un método que procese los datos recibidos y realice alguna acción con ellos. En este caso enviar un mensaje al usuario de que datos ha seleccionado.