PrimeFaces

Trabajo SDI

Contenido

[Ajax Core 3](#_Toc419113773)

[Listener 3](#_Toc419113774)

[Process 3](#_Toc419113775)

[Dropdown 3](#_Toc419113776)

[Selector 3](#_Toc419113777)

[Poll 3](#_Toc419113778)

[Status 3](#_Toc419113779)

[Atributos 4](#_Toc419113780)

[Ejemplo de código 4](#_Toc419113781)

[Events 5](#_Toc419113782)

[Validation 5](#_Toc419113783)

[Partial Submit 5](#_Toc419113784)

[Fragment 5](#_Toc419113785)

[Input 5](#_Toc419113786)

[Autocompletar 6](#_Toc419113787)

[Atributos 6](#_Toc419113788)

[Ejemplo de código 6](#_Toc419113789)

[BooleanButton 6](#_Toc419113790)

[Calendar 7](#_Toc419113791)

[Atributos 7](#_Toc419113792)

[Ejemplo de código 7](#_Toc419113793)

[SelectOneMenu 8](#_Toc419113794)

[InputText 8](#_Toc419113795)

[SelectOneListbox 8](#_Toc419113796)

[Atributos 8](#_Toc419113797)

[Ejemplo de código 8](#_Toc419113798)

[SelectRadio 9](#_Toc419113799)

[Password 9](#_Toc419113800)

[Atributos 9](#_Toc419113801)

[Ejemplo de código 9](#_Toc419113802)

[Rating 10](#_Toc419113803)

[InputTextArea 10](#_Toc419113804)

[Atributos 10](#_Toc419113805)

[Ejemplo de código 10](#_Toc419113806)

[Editor 11](#_Toc419113807)

[InputMask 11](#_Toc419113808)

[Keyboard 11](#_Toc419113809)

[Atributos 11](#_Toc419113810)

[Ejemplo de código 11](#_Toc419113811)

[Button 12](#_Toc419113812)

[Atributos 12](#_Toc419113813)

[Ejemplo de código 12](#_Toc419113814)

[Data 13](#_Toc419113815)

[Carousel 13](#_Toc419113816)

[Atributos 13](#_Toc419113817)

[DataList 14](#_Toc419113818)

[Atributos 15](#_Toc419113819)

[Ejemplo de código 15](#_Toc419113820)

[DataTable 15](#_Toc419113821)

[Atributos 16](#_Toc419113822)

[Ejemplo de código 16](#_Toc419113823)

[PickList 16](#_Toc419113824)

[GMap 17](#_Toc419113825)

[Atributos 17](#_Toc419113826)

# Ajax Core

Mediante Ajax podemos realizar cambios sobre nuestra página Web sin necesidad de recargarla. Algunos de los componentes sobre los que podemos utilizar Ajax con PrimeFaces son los siguientes:

## Listener

Permite que un método de java sea invocado con Ajax usando la opción “listener”.

## Process

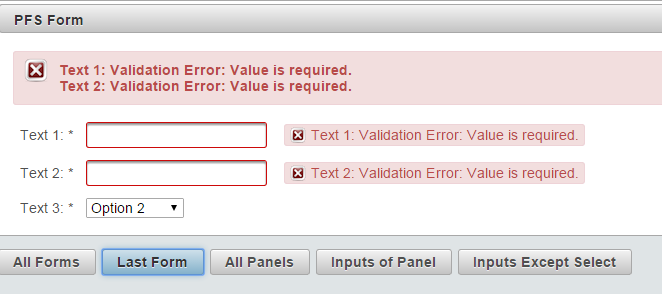
Permite un procesamiento parcial de alguna de los componentes de una Web ignorando otros.

## Dropdown

Con este componente podemos crear una dependencia entre varios desplegables.

## Selector

Este componente permite usar la API jQuery Selector para hacer referencias a componentes.



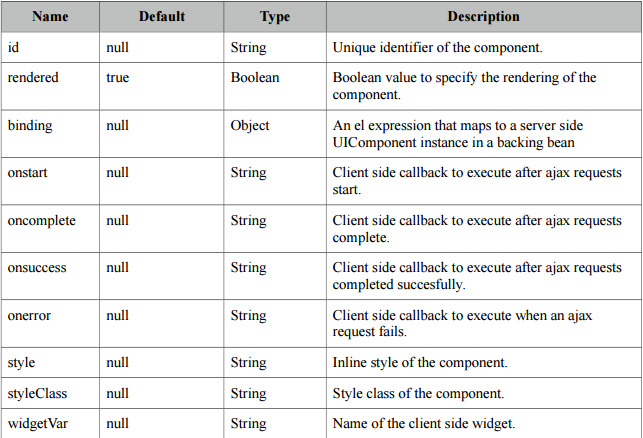
## Poll

Este componente realiza llamadas a métodos periódicamente.

## Status

Mediante AjaxStatus se proporciona información acerca de la solicitud Ajax que se está realizando en ese momento.

### Atributos



### Ejemplo de código

<p:ajaxStatus style="display:block;margin-bottom:2em;height:24px;">

    <f:facet name="default">

        <h:outputText value="Status: StandBy" />

    </f:facet>

    <f:facet name="start">

        <p:graphicImage name="/demo/images/ajaxloadingbar.gif" />

    </f:facet>

    <f:facet name="complete">

        <h:outputText value="Status: Completed" />

    </f:facet>

</p:ajaxStatus>

<p:ajaxStatus onstart="PF('statusDialog').show()" onsuccess="PF('statusDialog').hide()" />

<p:dialog widgetVar="statusDialog" modal="true" draggable="false" closable="false" resizable="false" showHeader="false">

    <p:graphicImage name="/demo/images/ajaxloadingbar.gif" />

</p:dialog>

<h:form>

    <p:commandButton value="Send" icon="ui-icon-refresh" />

</h:form>

## Events

Permite realizar eventos compatibles en los componentes.

## Validation

Este componente permite realizar una comprobación en el servidor y actualizarse con el resultado.



## Partial Submit

Permite reducir el tráfico de la red al cargar una página. En páginas grandes con muchos componentes de entrada, carga en primer lugar los más rápidos.

## Fragment

Permite fragmentar componentes y cargar la parte que se desee.

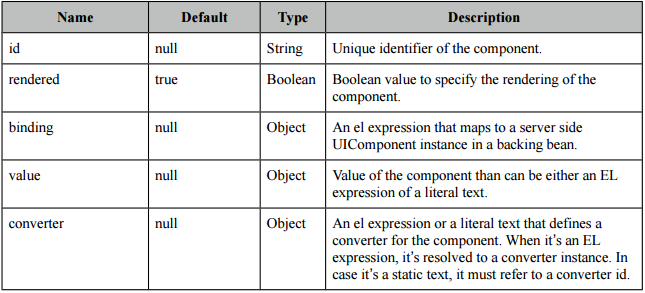
# Input

Mediante los inputs podemos introducir datos en una página Web, con PrimeFaces podemos personalizar los inputs de múltiples formas para mejorar la interacción:

## Autocompletar

Este componente permite mostrar sugerencias para auto completar una palabra que se está escribiendo en una página Web.

### Atributos



### Ejemplo de código

<p:outputLabel value="Simple:" for="acSimple" />

        <p:autoComplete id="acSimple" value="#{autoCompleteView.txt1}" completeMethod="#{autoCompleteView.completeText}" />

        <p:outputLabel value="Min Length (3):" for="acMinLength" />

        <p:autoComplete id="acMinLength" minQueryLength="3" value="#{autoCompleteView.txt2}" completeMethod="#{autoCompleteView.completeText}" effect="fade" />

        <p:outputLabel value="Delay(1000):" for="acDelay" />

        <p:autoComplete id="acDelay" queryDelay="1000" value="#{autoCompleteView.txt3}" completeMethod="#{autoCompleteView.completeText}" effect="blind" />

        <p:outputLabel value="Max Results(5):" for="acMaxResults" />

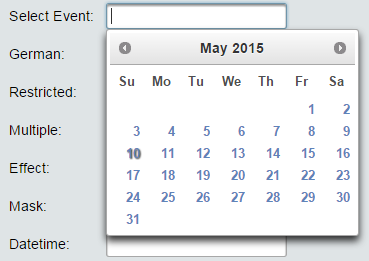
        <p:autoComplete id="acMaxResults" maxResults="5" value="#{autoCompleteView.txt4}" completeMethod="#{autoCompleteView.completeText}" />

## BooleanButton

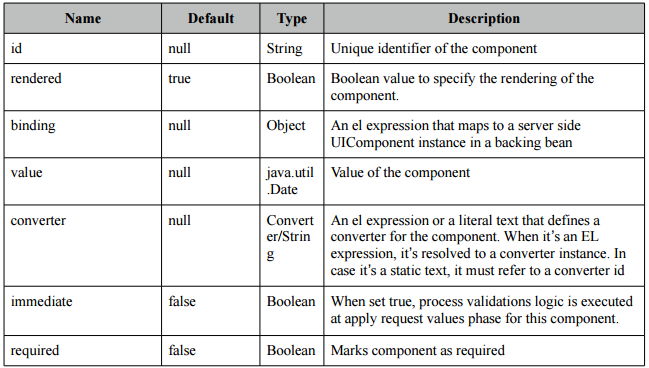
Permite crear un botón con un valor boolean que cambie al pulsarlo.

## Calendar

Este componente es un input que, al seleccionarlo, nos muestra un calendario y nos permite seleccionar una fecha en él para no tener que escribirla a mano.



### Atributos



### Ejemplo de código

<p:outputLabel for="inline" value="Inline:" />

        <p:calendar id="inline" value="#{calendarView.date1}" mode="inline" />

## SelectOneMenu

Muestra un menú con diferentes opciones para seleccionar, permite elegir el tipo de menú que deseemos: Basic, Grouping, Editable, Advanced…

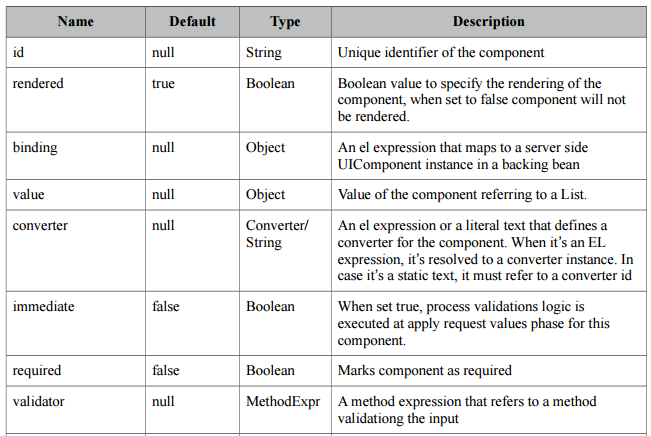
## InputText

Campo de texto que se puede personalizar el estilo.

## SelectOneListbox

Componente que muestra una lista de opciones que se pueden seleccionar.

### Atributos



### Ejemplo de código

<p:selectOneListbox id="basic" value="#{selectOneView.option}">

            <f:selectItem itemLabel="Option 1" itemValue="1" />

            <f:selectItem itemLabel="Option 2" itemValue="2" />

            <f:selectItem itemLabel="Option 3" itemValue="3" />

        </p:selectOneListbox>

## SelectRadio

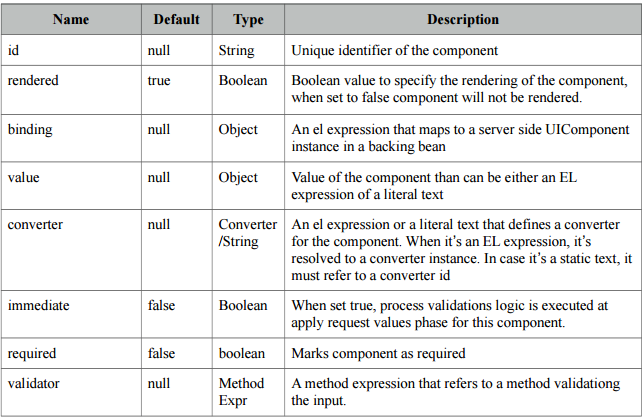
Permite elegir un elemento de una colección.

## Password

Es un campo para introducir contraseñas al que se le puede añadir un indicador de seguridad, para saber cómo de segura es la contraseña que hemos introducido.



### Atributos

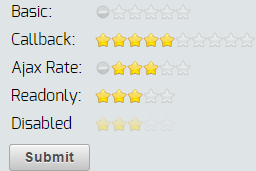


### Ejemplo de código

<p:password id="inlineFeedback" value="#{passwordView.password4}" feedback="true" inline="true" />

## Rating

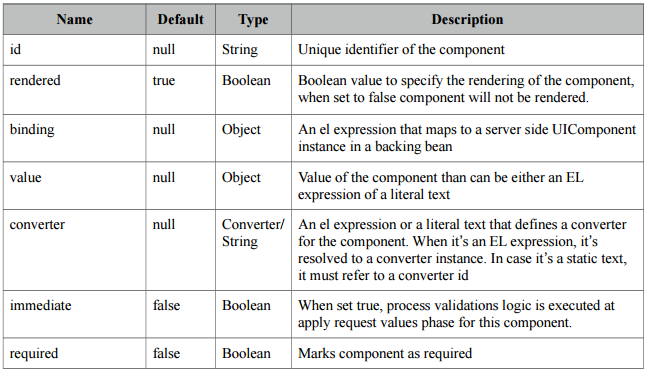
Es un componente para introducir valoraciones mediante estrellas.



## InputTextArea

Permite crear componentes textarea en los que se pueden añadir opciones como auto completado, cambiar el tamaño, contadores, temas…

### Atributos



### Ejemplo de código

 <p:inputTextarea rows="10" cols="50" completeMethod="#{inputTextareaView.completeArea}" queryDelay="750" minQueryLength="4" />

## Editor

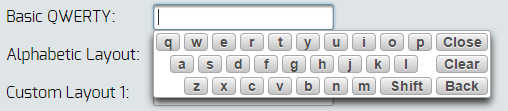
Podemos introducir en nuestra Web un editor de texto con múltiples opciones para editar el texto, con barra de herramientas y posibilidad de configurar los controles.

## InputMask

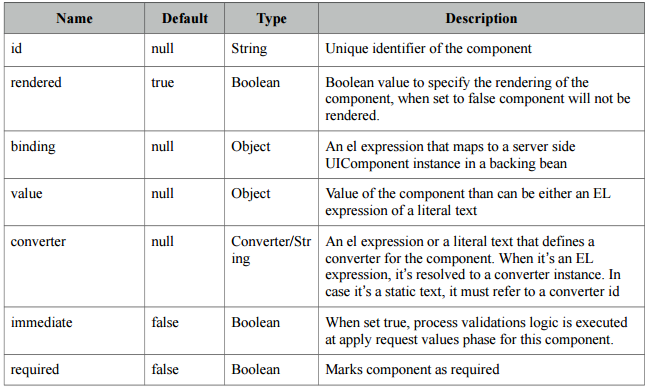
Permite mostrar el formato en el que debemos introducir el texto.

## Keyboard

Permite introducir un teclado en la pantalla al seleccionar un elemento de introducción de texto.



### Atributos



### Ejemplo de código

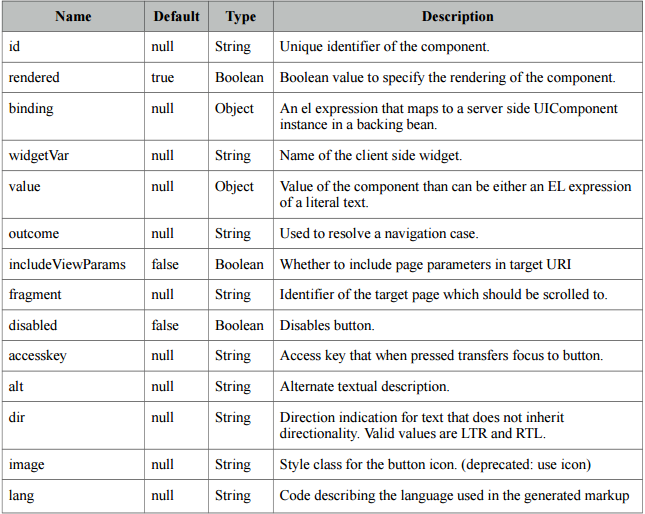
 <p:keyboard id="default" value="#{keyboardView.value1}" />

# Button

Este componente es una extensión del componente de JSF “h:button”. Con esta opción podemos modificar múltiples características del botón que de la otra manera no podíamos, ponerle un icono, cambiar el skin…



### Atributos



### Ejemplo de código

<p:button outcome="productDetail" value="Bookmark" icon="ui-icon-star" disabled="true">

    <f:param name="productId" value="40" />

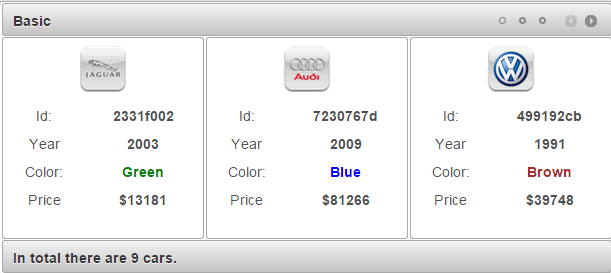
</p:button>

# Data

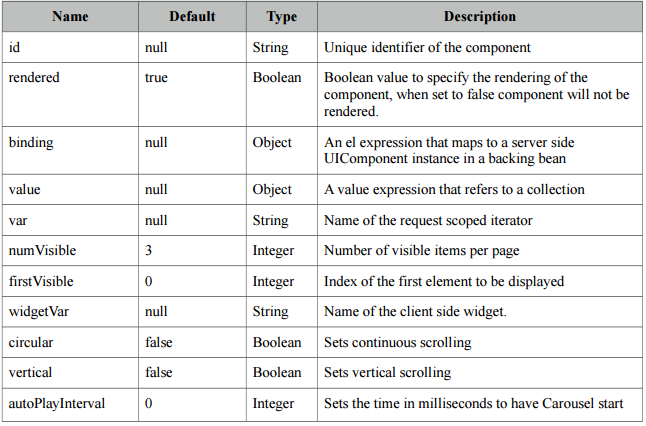
Con primefaces tenemos múltiples opciones de mostrar datos en una página Web:

## Carousel

Con este componente podemos mostrar datos usando un slide al que le podemos dar la forma que queramos (básico, tabla, avanzado).



### Atributos



Ejemplo de código

 <p:carousel value="#{carouselView.cars}" headerText="Basic" var="car" itemStyleClass="carItem">

        <h:panelGrid columns="2" style="width:100%" cellpadding="5" columnClasses="label,value">

            <f:facet name="header">

                <p:graphicImage name="demo/images/car/#{car.brand}.gif"/>

            </f:facet>

            <h:outputText value="Id:" />

            <h:outputText value="#{car.id}" />

            <h:outputText value="Year" />

            <h:outputText value="#{car.year}" />

            <h:outputText value="Color:" />

            <h:outputText value="#{car.color}" style="color:#{car.color}"/>

            <h:outputText value="Price" />

            <h:outputText value="$#{car.price}" />

        </h:panelGrid>

        <f:facet name="footer">

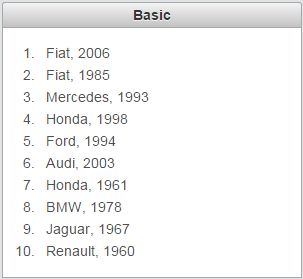
                In total there are #{fn:length(carouselView.cars)} cars.

        </f:facet>

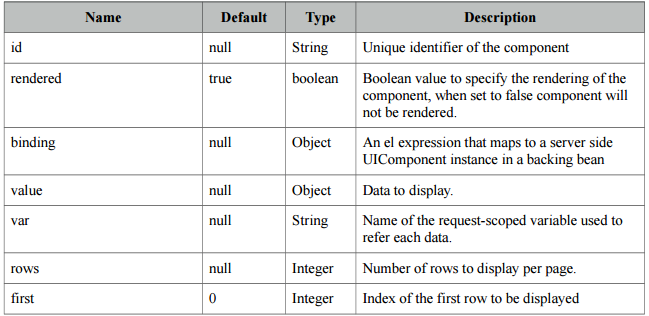
    </p:carousel>

## DataList

Con este componente podemos mostrar los datos en un layout con forma de lista.



### Atributos



### Ejemplo de código

<p:dataList value="#{dataListView.cars1}" var="car" type="ordered">

        <f:facet name="header">

            Basic

        </f:facet>

        #{car.brand}, #{car.year}

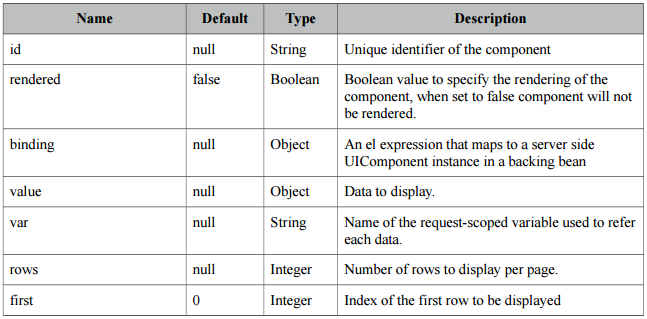
    </p:dataList>

## DataTable

Este componente permite mostrar los datos en una tabla tabulada.



### Atributos



### Ejemplo de código

<p:dataTable var="car" value="#{dtBasicView.cars}">

    <p:column headerText="Id">

        <h:outputText value="#{car.id}" />

    </p:column>

    <p:column headerText="Year">

        <h:outputText value="#{car.year}" />

    </p:column>

    <p:column headerText="Brand">

        <h:outputText value="#{car.brand}" />

    </p:column>

    <p:column headerText="Color">

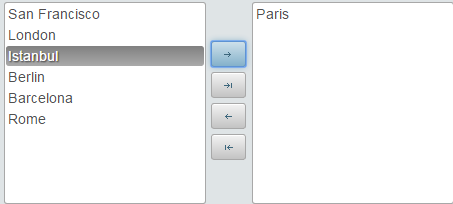
        <h:outputText value="#{car.color}" />

    </p:column>

</p:dataTable>

## PickList

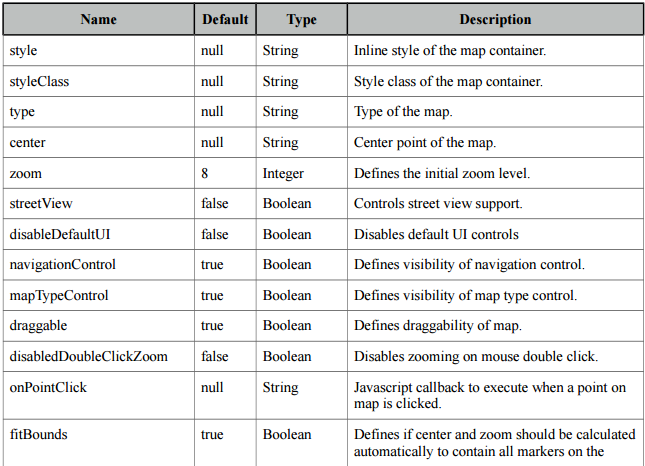
Este componente permite seleccionar datos de una lista inicial y añadirlos a otra.



## GMap

Con este componente podemos mostrar en nuestra Web un mapa con una posición.

### Atributos



# Client side validation

PrimeFaces permite validación del lado cliente, mediante patrones establecidos en el propio xhtml como definidos en las clases java anexas. Estos elementos actúan sobre otros elementos, en la mayoría de los casos input o output de texto simples y pueden ser de varios tipos:

## Basico

El patrón se define dentro del xhtml en etiquetas de validación, o de conversión.

## Custom

El patrón se define dentro del xhtml pero separado de la estructura y anidado en una etiquetal javascript.

## Bean

Los patrones se definen sobre los atributos java dentro del Bean asociado, gracias a anotaciones con las que acotamos el rango o valores asociados.

## Event

Son comprobados dinámicamente sobre el input , y se definen en el xhtml con atributos event dentro de etiquetas de input o output.

### Ejemplo de código

Este ejemplo se corresponde con una validación de tipo básico, en este caso formada por una comprobación por patrón y una conversión.

<h:form>

    <p:panel header="Validate">

        <p:messages autoUpdate="true"/>

<h:outputLabel for="money" value="Currency ($):" style="font-weight:bold"/>

            <p:inputText id="money" value="#{validationView.money}" label="Currency">

                <f:convertNumber type="currency" currencySymbol="$"/>

            </p:inputText>

            <p:message for="money" />

            <h:outputText value="#{validationView.money}">

                <f:convertNumber  type="currency" currencySymbol="$" />

            </h:outputText>

            <h:outputLabel for="regex" value="Regex (^[a-zA-Z]+$):" style="font-weight:bold"/>

            <p:inputText id="regex" value="#{validationView.regexText}" validatorMessage="Value does not match pattern.">

                <f:validateRegex pattern="^[a-zA-Z]+$" />

            </p:inputText>

            <p:message for="regex" />

            <h:outputText value="#{validationView.regexText}" />

### Visualización



# Dialog framework

Permite generar cuadros de dialogo generados dinámicamente en tiempo de ejecución. Pueden ser tanto tablas de datos a visualizar, como mensajes emergentes, incluso elementos con los que se permita interactuar.

### Ejemplo de código

Menú de opciones:

<h:form>

    <p:commandButton value="Show" icon="ui-icon-script" actionListener="#{dfView.showMessage}" />

</h:form>

Dialog message (Definido en el bean):

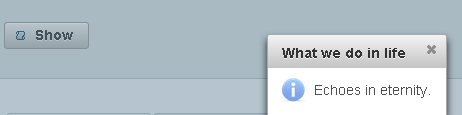
public void showMessage() {

        FacesMessage message = new FacesMessage(FacesMessage.SEVERITY\_INFO, "What we do in life", "Echoes in eternity.");

        RequestContext.getCurrentInstance().showMessageInDialog(message);

    }

### Visualizacion



# Misc

Este apartado se comentan elementos añadidos de primeFaces que no se pueden encuadrar en ninguna de las otras agrupaciones.

## FontAwesome

Se trata de una librería con más de 479 iconos vectoriales escalables que podemos utilizar para dar una visualización más interesante a nuestro proyecto.

### Ejemplo de código

< h3 > Menú </ h3 >

< p: Menú >

    < p: submenú etiqueta = "Documento" >

        < p: menuitem valor = "Nuevo" url = "#" icon = "fa fa-plus" />

        < p: menuitem valor = "Refresh" url = "#" icon = "fa fa-refresh" />

    </ p: submenú >

    < p: submenú etiqueta = "Navegaciones" >

        < p: menuitem valor = "Inicio" url = " [http://www.primefaces.org](http://www.primefaces.org/) " icon = "fa fa-casa" />

        < p: menuitem valor = "Usuario" url = "#" icon = "fa fa-usuario" />

    </ p: submenú >

</ p: Menú >

### Visualización



## IdleMonitor

Consiste en un detector de inactividad que nos da mensajes según nuestra interacción con la página.

### Ejemplo de código

Los mensajes concretos se definen en el Bean asociado.

<h:form>

    <p:growl id="messages" showDetail="true" sticky="true" />

    <p:idleMonitor timeout="5000">

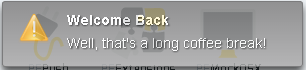
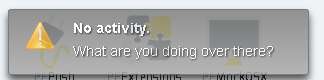
        <p:ajax event="idle" listener="#{idleMonitorView.onIdle}" update="messages" />

        <p:ajax event="active" listener="#{idleMonitorView.onActive}" update="messages" />

    </p:idleMonitor>

</h:form>

### Visualización



## ThemeSwitcher

Es un elemento de lista desplegable que nos permite cambiar el tema de la interfaz de forma dinámica y fácil de implementar. Los temas se definen como una clase java y se importan directamente de primeFaces.

### Ejemplo de código

<p:themeSwitcher id="basic" style="width:165px">

            <f:selectItem itemLabel="Choose Theme" itemValue="" />

            <f:selectItems value="[afterdark, afternoon, afterwork, aristo, black-tie, blitzer, bluesky, bootstrap, casablanca, cupertino, cruze, dark-hive, delta, dot-luv, eggplant, excite-bike, flick, glass-x, home, hot-sneaks, humanity, le-frog, midnight, mint-choc, overcast, pepper-grinder, redmond, rocket, sam, smoothness, south-street, start, sunny, swanky-purse, trontastic, ui-darkness, ui-lightness, vader]" var="theme" itemLabel="" itemValue=""/>

        </p:themeSwitcher>

        <p:outputLabel for="advanced" value="Advanced:" />

        <p:themeSwitcher id="advanced" style="width:165px" effect="fade" var="t">

            <f:selectItem itemLabel="Choose Theme" itemValue="" />

            <f:selectItems value="[afterdark, afternoon, afterwork, aristo, black-tie, blitzer, bluesky, bootstrap, casablanca, cupertino, cruze, dark-hive, delta, dot-luv, eggplant, excite-bike, flick, glass-x, home, hot-sneaks, humanity, le-frog, midnight, mint-choc, overcast, pepper-grinder, redmond, rocket, sam, smoothness, south-street, start, sunny, swanky-purse, trontastic, ui-darkness, ui-lightness, vader]" var="theme" itemLabel="" itemValue="" />

            <p:column>

                <h:outputText styleClass="ui-theme ui-theme-" style="display-block" />

            </p:column>

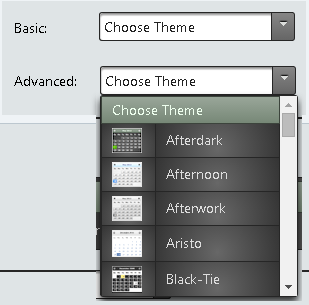
            <p:column>

            </p:column>

        </p:themeSwitcher>

### Visualización

Este sería el selector.



Y así como actuaría sobre algunos elementos con diferentes temas.

