EBook Gratis

APRENDIZAJE primefaces

Free unaffiliated eBook created from
Stack Overflow contributors.

#primefaces

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: primefaces

It is an unofficial and free primefaces ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official primefaces.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Primeros pasos con primefaces

Observaciones

PrimeFaces es un framework JSF de código abierto. Sus principales características:

- Más de 100 componentes.
- Ajax incorporado basado en las API JSF Ajax estándar.
- Empuje el soporte a través de Atmosphere Framework.
- Kit de interfaz de usuario móvil para crear aplicaciones web móviles.
- Más de 35 temas incorporados.
- Temas y diseños premium.

Versiones

Versión	Fecha de lanzamiento
0.8.1	2009-02-23
0.8.2	2009-03-26
0.8.3	2009-04-23
0.9.0	2009-06-15
0.9.1	2009-08-04
0.9.2	2009-09-07
0.9.3	2009-10-05
1.0.0 y 2.0.0	2010-02-15
1.0.1 y 2.0.1	2010-04-19
1.0.2 y 2.0.2	2010-05-31
1.1 y 2.1	2010-07-26
2.2	2011-02-07
3.0	2012-01-04
3.1	2012-02-06
3.2	2012-03-12
3.3	2012-05-29

Versión	Fecha de lanzamiento
3.4	2012-09-03
3.5	2013-02-04
4.0	2013-10-03
5.0	2014-05-05
5.1	2014-10-06
5.2	2015-04-08
5.3	2015-10-19
6.0	2016-06-07

Examples

Instalación de PrimeFaces

PrimeFaces se puede usar en todas las aplicaciones web basadas en Java Server Faces (versión 2.x) que se ejecutan en Contenedores de Servlets (por ejemplo, Wildlfy o Tomcat o GlassFish).

Hay varias formas de agregar PrimeFaces a su aplicación.

A mano

Descargue las `primefaces-{version}.jar` y agréguela a su classpath.

Maven

```
<dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>{version}</version>
</dependency>
```

Para versiones anteriores (3.5 y posteriores), adicionalmente debe agregar el repositorio PrimeFaces:

```
<repository>
    <id>prime-repo</id>
    <name>PrimeFaces Maven Repository</name>
    <url>http://repository.primefaces.org</url>
    <layout>default</layout>
</repository>
```

Gradle

```
repositories {
    mavenCentral()
        maven {
            url "http://repository.primefaces.org"
        }
}

dependencies {
    compile "org.primefaces:primefaces:{version}"
}
```

NetBeans

PrimeFaces se incluye con el paquete Java EE de NetBeans . Cuando creas una nueva "Web de Java -> Aplicación Web" puedes seleccionar JavaServer Faces como marco. A continuación, configura JSF para usar los componentes PrimeFaces. Copiará la biblioteca a su proyecto.

Si ha creado una aplicación web de Maven, puede seleccionar las propiedades del proyecto y seleccionar JavaServer Faces como marco y luego seleccionar PrimeFaces como se mencionó anteriormente. Su `pom.xml` se modificará para incluir la dependencia de PrimeFaces.

Hola Mundo

Después de agregar PrimeFaces a su proyecto JSF , puede comenzar a usarlo en sus páginas usando el espacio de nombres:

```
xmlns:p="http://primefaces.org/ui"
```

o, para PrimeFaces Mobile:

```
xmlns:p="http://primefaces.org/mobile"
```

Este ejemplo debería hacer un spinner:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
    </h:head>
    <h:body>
        <p:spinner />
    </h:body>
</html>
```

Lea Primeros pasos con primefaces en línea:
https://riptutorial.com/es/primefaces/topic/1027/primeros-pasos-con-primefaces

Capítulo 2: Cómo utilizar el escaparate de Primefaces

Examples

El ejemplo del componente Primefaces panelGrid en su Showcase.

El escaparate de los componentes de Primefaces que puede encontrar aquí y la documentación está aquí.

Frontend debe ser guardado como un archivo XHTML. Este archivo puede contener JSF, JSTL, JSP, HTML, CSS, jQuery, javaScript y su marco y más tecnologías front-end. Por favor, no mezcle las tecnologías JSF y JSP. No es un buen enfoque.

Tenga en cuenta que debe definir espacios de nombres como c, f, h, p, pe, etc. al principio.

```
<h:form id="form">
    <p:dataGrid var="car" value="#{dataGridView.cars}" columns="3" layout="grid"
        rows="12" paginator="true" id="cars"
        paginatorTemplate="{CurrentPageReport}  {FirstPageLink} {PreviousPageLink} {PageLinks}
{NextPageLink} {LastPageLink} {RowsPerPageDropdown}"
        rowsPerPageTemplate="6,12,16">

        <f:facet name="header">
            Cars for Sale
        </f:facet>

        <p:panel header="#{car.id}" style="text-align:center">
            <h:panelGrid columns="1" style="width:100%">
                <p:graphicImage name="demo/images/car/#{car.brand}.gif"/>

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

                <p:commandLink update=":form:carDetail" oncomplete="PF('carDialog').show()"
title="View Detail">
                    <h:outputText styleClass="ui-icon ui-icon-search" style="margin:0 auto;"
/>
                    <f:setPropertyActionListener value="#{car}"
target="#{dataGridView.selectedCar}" />
                </p:commandLink>
            </h:panelGrid>
        </p:panel>

    </p:dataGrid>

    <p:dialog header="Car Info" widgetVar="carDialog" modal="true" showEffect="fade"
hideEffect="fade" resizable="false">
        <p:outputPanel id="carDetail" style="text-align:center;">
            <p:panelGrid  columns="2" rendered="#{not empty dataGridView.selectedCar}"
columnClasses="label,value">
                <f:facet name="header">
                    <p:graphicImage name="demo/images/car/#{dataGridView.selectedCar.brand}-
big.gif"/>
```

```
                </f:facet>

                <h:outputText value="Id:" />
                <h:outputText value="#{dataGridView.selectedCar.id}" />

                <h:outputText value="Year" />
                <h:outputText value="#{dataGridView.selectedCar.year}" />

                <h:outputText value="Color:" />
                <h:outputText value="#{dataGridView.selectedCar.color}"
style="color:#{dataGridView.selectedCar.color}"/>

                <h:outputText value="Price" />
                <h:outputText value="$#{dataGridView.selectedCar.price}" />
            </p:panelGrid>
        </p:outputPanel>
    </p:dialog>
</h:form>
```

El back-end debe guardarse como un archivo JSF. Este archivo contiene principalmente Java, pero puede haber llamadas de Java a Front-end. La anotación de clase podría ser reemplazada por la configuración de Spring. **La clase de Vista** que se usa para servir datos a Frontend EL (lenguaje de expresión).

```
package org.primefaces.showcase.view.data;

    import java.io.Serializable;
    import java.util.List;
    import javax.annotation.PostConstruct;
    import javax.faces.bean.ManagedBean;
    import javax.faces.bean.ManagedProperty;
    import javax.faces.bean.ViewScoped;
    import org.primefaces.showcase.domain.Car;
    import org.primefaces.showcase.service.CarService;

    @ManagedBean
    @ViewScoped
    public class DataGridView implements Serializable {

        private List<Car> cars;

        private Car selectedCar;

        @ManagedProperty("#{carService}")
        private CarService service;

        @PostConstruct
        public void init() {
            cars = service.createCars(48);
        }

        public List<Car> getCars() {
            return cars;
        }

        public void setService(CarService service) {
            this.service = service;
        }
```

```
        public Car getSelectedCar() {
            return selectedCar;
        }

        public void setSelectedCar(Car selectedCar) {
            this.selectedCar = selectedCar;
        }
    }
```

La clase de servicio sirve datos de DB, pero se usa en la mayoría de los ejemplos en PF Showcase para inicializar y crear datos. Tenga en cuenta que los ejemplos más útiles de código en el escaparate de PF son Frontend.

```
package org.primefaces.showcase.service;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;
import org.primefaces.showcase.domain.Car;

@ManagedBean(name = "carService")
@ApplicationScoped
public class CarService {

    private final static String[] colors;

    private final static String[] brands;

    static {
        colors = new String[10];
        colors[0] = "Black";
        colors[1] = "White";
        colors[2] = "Green";
        colors[3] = "Red";
        colors[4] = "Blue";
        colors[5] = "Orange";
        colors[6] = "Silver";
        colors[7] = "Yellow";
        colors[8] = "Brown";
        colors[9] = "Maroon";

        brands = new String[10];
        brands[0] = "BMW";
        brands[1] = "Mercedes";
        brands[2] = "Volvo";
        brands[3] = "Audi";
        brands[4] = "Renault";
        brands[5] = "Fiat";
        brands[6] = "Volkswagen";
        brands[7] = "Honda";
        brands[8] = "Jaguar";
        brands[9] = "Ford";
    }

    public List<Car> createCars(int size) {
        List<Car> list = new ArrayList<Car>();
        for(int i = 0 ; i < size ; i++) {
```

```
            list.add(new Car(getRandomId(), getRandomBrand(), getRandomYear(),
getRandomColor(), getRandomPrice(), getRandomSoldState()));
        }

        return list;
    }

    private String getRandomId() {
        return UUID.randomUUID().toString().substring(0, 8);
    }

    private int getRandomYear() {
        return (int) (Math.random() * 50 + 1960);
    }

    private String getRandomColor() {
        return colors[(int) (Math.random() * 10)];
    }

    private String getRandomBrand() {
        return brands[(int) (Math.random() * 10)];
    }

    public int getRandomPrice() {
        return (int) (Math.random() * 100000);
    }

    public boolean getRandomSoldState() {
        return (Math.random() > 0.5) ? true: false;
    }

    public List<String> getColors() {
        return Arrays.asList(colors);
    }

    public List<String> getBrands() {
        return Arrays.asList(brands);
    }
}
```

NOTA: Por favor, mejore esta documentación si tiene suficiente experiencia relevante.

Lea Cómo utilizar el escaparate de Primefaces en línea:
https://riptutorial.com/es/primefaces/topic/6623/como-utilizar-el-escaparate-de-primefaces

Capítulo 3: Hola primefaces

Examples

Hola primefaces

Esta es una aplicación simple con primefaces, es una página de inicio de sesión:

1-Configuración de **web.xml** :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

2-Crear **ManagedBean** :

```
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

@ManagedBean
@RequestScoped
public class LoginBean {

    private String username;
    private String password;

    public LoginBean() {
    }
```

```
    public void login() {
        //Simple login
        if (!this.username.equals("") && this.username.equals(password)) {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(
                    FacesMessage.SEVERITY_INFO, "Success", "Login with success"));
        } else {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(
                    FacesMessage.SEVERITY_ERROR, "Error", "Username or password not
correct"));
        }
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Página 3-Create * **.xhtml** :

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Hello Primefaces</title>
    </h:head>
    <h:body>
        <h:form>
            <p:messages id="my_message" showDetail="true" autoUpdate="true" closable="true" />
            <h:panelGrid columns="2">
                <p:outputLabel value="UserName :" for="username"/>
                <p:inputText id="username" value="#{loginBean.username}" required="true"
                             requiredMessage="Username is required"/>
                <p:outputLabel value="Password :" for="password"/>
                <p:password id="password" value="#{loginBean.password}" required="true"
                            requiredMessage="Password is required"/>
                <span/>
                <p:commandButton value="Login" actionListener="#{loginBean.login}"/>
            </h:panelGrid>
        </h:form>
    </h:body>
</html>
```

4-Implementar su aplicación.

5-abra su navegador y escriba: `http://localhost:8080/HelloPrimefaces/`

Lea Hola primefaces en línea: https://riptutorial.com/es/primefaces/topic/3309/hola-primefaces

Capítulo 4: widgetVar

Introducción

widgetVar es el nombre de las variables del lado del cliente que contiene todos los widgets PF de javascript en la página. Hay un gran intro / tutorial para usar el componente widgetVar escrito por Hatem Alimam llamado Intro To PrimeFaces widgetVar

Examples

Uso básico de widgetVar

```
<h:form>
    <p:dialog widgetVar="myDialog"></p:dialog>
    <p:commandButton onclick="PF('myDialog').show();" />
</h:form>
```

Tabla de datos

`datatable.js` en GitHub Reporitory

Función	Detalles
`bindPaginator: function()`	Enlaza el detector de eventos de cambio y renderiza el paginador
`loadLiveRows: function()`	Carga filas sobre la marcha cuando se desplaza en vivo
`paginate: function(newState)`	Paginación del Ajax
`fetchNextPage: function(newState)`	Carga la página siguiente de forma asíncrona para mantenerla en viewstate y Actualiza viewstate
`sort: function(columnHeader, order, multi)`	Ajax sort
`filter: function()`	Filtro ajax
`onRowClick: function(event, rowElement, silent)`	
`onRowDblclick: function(event, row)`	
`highlightRow: function(row)`	Resalta la fila como seleccionado
`unhighlightRow: function(row)`	Borra las imágenes seleccionadas

Función	Detalles
`fireRowSelectEvent: function(rowKey, behaviorEvent)`	Envía un rowSelectEvent en el lado del servidor para invocar un rowSelectListener si está definido
`fireRowUnselectEvent: function(rowKey, behaviorEvent)`	Envía un rowUnselectEvent en el lado del servidor para invocar un rowUnselectListener si está definido
`selectRowWithRadio: function(radio)`	Selecciona la fila correspondiente de una selección de columna basada en radio
`unselectAllRows: function()`	
`selectAllRowsOnPage: function()`	
`unselectAllRowsOnPage: function()`	
`selectAllRows: function()`	
`toggleExpansion: function(toggler)`	Expande una fila para mostrar contenido detallado
`collapseRow: function(row)`	
`collapseAllRows: function()`	
`getExpandedRows: function()`	
`switchToRowEdit: function(row)`	
`showRowEditors: function(row)`	
`saveRowEdit: function(rowEditor)`	Guarda la fila editada
`cancelRowEdit: function(rowEditor)`	
`updateRow: function(row, content)`	Fila de actualizaciones con contenido dado
`clearSelection: function()`	Borra el estado de selección.
`clearFilters: function()`	Borra los filtros de mesa
`removeSelection: function(rowIndex)`	Eliminar el rowIndex dado de la selección
`addSelection: function(rowKey)`	Agrega un rowKey dado a la selección si no existe ya
`isSelected: function(rowKey)`	Encuentra si el rowKey dado está en la selección
`saveColumnOrder: function()`	
`isEmpty: function()`	Devuelve si hay algún dato mostrado
`getSelectedRowsCount: function()`	

Lea widgetVar en línea: https://riptutorial.com/es/primefaces/topic/9841/widgetvar

Creditos

S. No	Capítulos	Contributors
1	Primeros pasos con primefaces	Abaddon666, ArgaPK, Community, Jasper de Vries, Onur Senture, Xtreme Biker, YCF_L
2	Cómo utilizar el escaparate de Primefaces	Skyware
3	Hola primefaces	ArgaPK, Xtreme Biker, YCF_L
4	widgetVar	Eric B., Paolo Forgia