prime
teknoloji

PRIMEFACES

USER'S GUIDE

Author

Çağatay Çivici

Covers 2.2
Last Update: 05.02.2011

This guide is dedicated to my wife Nurcan,
without her support PrimeFaces wouldn't exist.

Çağatay Çivici

DISCLAIMER

This guide is an important financial resource for PrimeFaces project, if you have gained this copy without purchasing it, you are seriously damaging PrimeFaces and our enthusiasm for open source development, as a result your own project built with PrimeFaces.

1. Introduction

1.1 What is PrimeFaces?

PrimeFaces is an open source component suite for Java Server Faces featuring 100+ Ajax powered rich set of JSF components. Additional TouchFaces module features a UI kit for developing mobile web applications. Main goal of PrimeFaces is to create the ultimate component suite for JSF.

- 100+ rich set of components (HtmlEditor, Dialog, AutoComplete, Charts and more).
- Built-in Ajax with Lightweight Partial Page Rendering.
- Native Ajax Push/Comet support.
- Mobile UI kit to create mobile web applications for handheld devices with webkit based browsers.(IPhone, Palm, Android Phones, Nokia S60 and more)
- One jar, zero-configuration and no required dependencies.
- Skinning Framework with 30 pre-designed themes.
- Extensive documentation.

1.2 Prime Technology

PrimeFaces is maintained by Prime Technology, a Turkish software development company specialized in Agile and Java EE consulting. Project is led by Çağatay Çivici (aka Optimus Prime), a JSF Expert Group Member.

2. Setup

2.1 Download

PrimeFaces has a single jar called **primefaces-{version}.jar**. There are two ways to download this jar, you can either download from PrimeFaces homepage or if you are a maven user you can define it as a dependency.

Download Manually

Three different artifacts are available for each PrimeFaces version, binary, sources and bundle. Bundle contains binary, sources and javadocs.

http://www.primefaces.org/downloads.html

Download with Maven

Group id of the dependency is *org.primefaces* and artifact id is *primefaces*.

```
<dependency>
      <groupId>org.primefaces</groupId>
      <artifactId>primefaces</artifactId>
      <version>2.2</version>
</dependency>
```

In addition to the configuration above you also need to add Prime Technology maven repository to the repository list so that maven can download it.

```
<repository>
      <id>prime-repo</id>
      <name>Prime Technology Maven Repository</name>
      <url>http://repository.prime.com.tr</url>
      <layout>default</layout>
</repository>
```

2.2 Dependencies

PrimeFaces only requires a JAVA 5+ runtime and a JSF 2.0 implementation as mandatory dependencies. There're some optional libraries for certain features.

Dependency	Version *	Type	Description
JSF runtime	2.0+	Required	Apache MyFaces or Oracle Mojarra
itext	2.1.7	Optional	PDF export support for DataExporter component
apache poi	3.2-FINAL	Optional	Excel export support for DataExporter component
commons-fileupload	1.2.1	Optional	FileUpload
commons-io	1.4	Optional	FileUpload
atmosphere-runtime	0.5.1	Optional	Ajax Push
atmosphere-compat	0.5.1	Optional	Ajax Push

* Listed versions are tested and known to be working with PrimeFaces, other versions of these dependencies may also work but not tested.

2.3 Configuration

PrimeFaces does not require any mandatory configuration.

2.4 Hello World

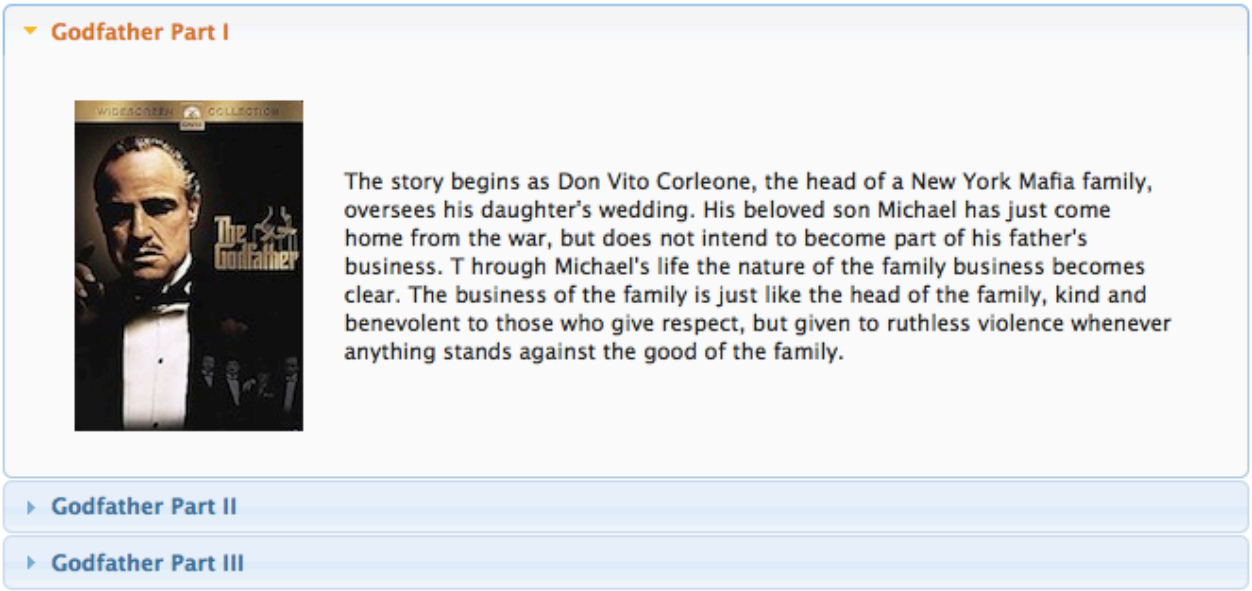
Once you have added the downloaded jar to your classpath, you need to add the PrimeFaces namespace to your page to begin using the components. Here is a simple page;

```
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:p="http://primefaces.prime.com.tr/ui">

<h:head>
</h:head>

<h:body>

      <p:spinner />

</h:body>

</html>
```

3. Component Suite

3.1 AccordionPanel

AccordionPanel is a container component that displays content in stacked format.



Info

Tag	**accordionPanel**
Component Class	**org.primefaces.component.accordionpanel.Accordionpanel**
Component Type	**org.primefaces.component.AccordionPanel**
Component Family	**org.primefaces.component**
Renderer Type	**org.primefaces.component.AccordionPanelRenderer**
Renderer Class	**org.primefaces.component.accordionpanel.AccordionPanelRenderer**

Attributes

Name	Default	Type	Description
id	null	String	Unique identifier of the component
rendered	TRUE	boolean	Boolean value to specify the rendering of the component.
binding	null	Object	An EL expression that maps to a server side UIComponent instance in a backing bean.

Name	Default	Type	Description
activeIndex	0	Integer	Index of the active tab.
style	null	String	Inline style of the container element.
styleClass	null	String	Style class of the container element.
disabled	FALSE	Boolean	Disables or enables the accordion panel.
effect	slide	String	Effect to use when toggling the tabs.
autoHeight	TRUE	Boolean	When enabled, tab with highest content is used to calculate the height.
collapsible	FALSE	Boolean	Defines if accordion panel can be collapsed all together.
fillSpace	FALSE	Boolean	When enabled, accordion panel fills the height of it's parent container.
event	click	String	Client side event to toggle the tabs.
widgetVar	null	String	Name of the widget to access client side api.
tabChangeListener	null	MethodExpr	Server side listener to invoke when active tab changes
onTabChangeUpdate	null	String	Component(s) to update with ajax after dynamic tab change.
onTabChange	null	String	Client side callback to invoke when active tab changes.
dynamic	FALSE	Boolean	Defines the toggle mode.
cache	FALSE	Boolean	Defines if activating a dynamic tab should load the contents from server again.

Getting Started with Accordion Panel

Accordion panel consists of one or more tabs and each tab can group any other jsf components.

```
<p:accordionPanel>
      <p:tab title="First Tab Title">
            <h:outputText value= "Lorem"/>
            ...More content for first tab
      </p:tab>
      <p:tab title="Second Tab Title">
            <h:outputText value="Ipsum" />
      </p:tab>
      //any number of tabs
</p:accordionPanel>
```

Toggle Event

By default toggling happens when a tab header is clicked, you can also specify a custom event. For example below, toggling happens when mouse is over the tab headers.

```
<p:accordionPanel effect="hover">
      //..tabs
</p:accordionPanel>
```

Dynamic Content Loading

AccordionPanel supports lazy loading of tab content, when dynamic option is set true, only active tab contents will be rendered to the client side and clicking an inactive tab header will do an ajax request to load the tab contents. This feature is useful to reduce bandwidth and speed up page loading time. By default activating a previously loaded dynamic tab does not initiate a request to load the contents again as tab is cached. To control this behavior use *cache* option.

```
<p:accordionPanel dynamic="true">
      //..tabs
</p:accordionPanel>
```

onTabChange

You can use client/server side callbacks to get notified when active tab changes. On client side use *onTabChange* option.

```
<p:accordionPanel onTabChange="handleChange(event, ui)">
      //..tabs
</p:accordionPanel>

<script type="text/javascript">
      function handleChange(event, ui) {
            //Execute custom logic
      }
</script>
```

ui object will be passed to your callback containing information about the tab change event.

- ui.newHeader = jQuery object representing the header of new tab
- ui.oldHeader = jQuery object representing the header of previous tab
- ui.newContent = jQuery object representing the content of new tab
- ui.oldContent = jQuery object representing the content of previous tab

TabChangeListener

onTabChange is used on the client side, in case you need to execute logic on server side, use *tabChangeListener* option.

```
<p:accordionPanel tabChangeListener="#{bean.onChange}">
      //..tabs
</p:accordionPanel>
```

```
public void onChange(TabChangeEvent event) {
      //Tab activeTab = event.getTab();
      //...
 }
```

Your listener will be invoked with an *org.primefaces.event.TabChangeEvent* instance that contains a reference to the new active tab and the accordion panel itself.

If you'd like to update some parts of your page after your tabChangeListener is invoked, use *onTabChangeUpdate* option. Following example, adds a FacesMessage at listener and displays it using a growl component.

```
<p:growl id="messages />

<p:accordionPanel tabChangeListener="#{bean.onChange}"
                        onTabChangeUpdate="messages">
      //..tabs
</p:accordionPanel>
```

```
public void onChange(TabChangeEvent event) {
      FacesMessage msg = new FacesMessage("Tab Changed",
                        "Active Tab:" + event.getTab().getId());
      FacesContext.getCurrentInstance().addMessage(null, msg);
}
```

Note

```
For both dynamic loading and tabChangeListener features to work, at least one form
needs to present on page, location of the form does not matter.
```

Client Side API

Widget: *PrimeFaces.widget.AccordionPanel*

Method	Params	Return Type	Description
select(index)	index: Index of tab to display	void	Activates tab with given index
collapseAll()	-	void	Collapses all tabs.

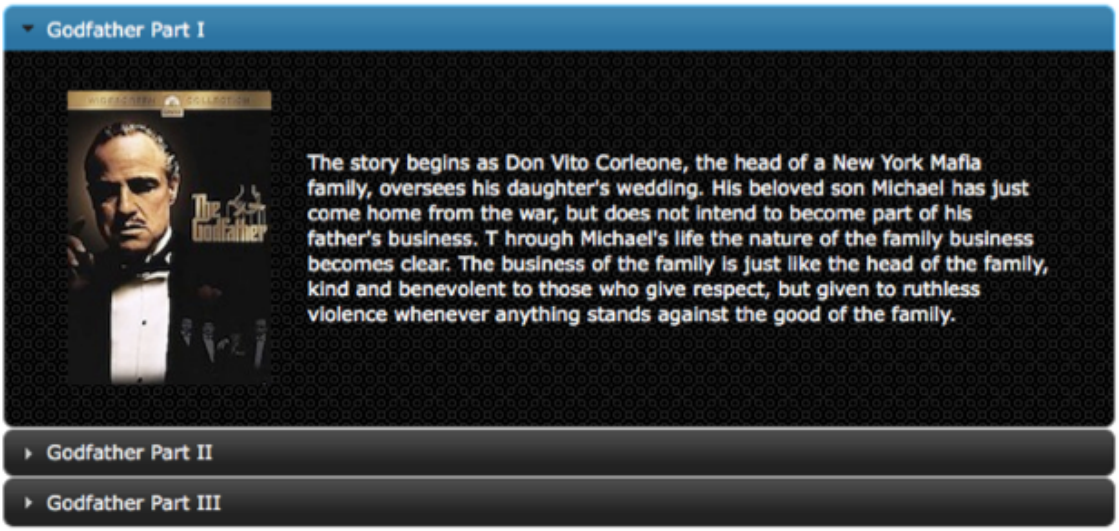
Skinning

AccordionPanel resides in a main container element which *style* and *styleClass* options apply.

Following is the list of structural style classes;

Class	Applies
.ui-accordion	Main container element
.ui-accordion-header	Tab header
.ui-accordion-content	Tab content
.ui-accordion-content-active	Content of active tab.

As skinning style classes are global, see the main Skinning section for more information. Here is an example based on a different theme;



Tips

- autoHeight option provides more consistent animations when enabled.
- Use c:forEach to create tabs on the fly, ui:repeat will not work as p:tab has no Renderer.