Questo capitolo descrive le funzionalità avanzate dei componenti compositi nella tecnologia JavaServer Faces.

Un componente composito è un tipo speciale di template JavaServer Faces che agisce come un componente (vedere il capitolo dedicato precedentemente a tale argomento per rinfrescarsi la memoria).

## 14.1 Attributi di un Componente Composito

Si definisce un attributo per un componente composito usando il tag *composite:attribute*.

La lista sottostante elenca gli attributi di questo tag più comunemente utilizzati:





Il seguente frammento di codice definisce un attributo di un componente composito e gli assegna un valore di default:

*<composite:attribute name=”username” default=”admin”/>*

Il seguente invece utilizza un elemento *method-signature*:

*<composite:attribute name="myaction"*

*method-signature="java.lang.String action()"/>*

E il seguente un elemento *type*:

*<composite:attribute name="dateofjoining" type="java.util.Date"/>*

## 14.2 Invocare un managed bean

Per abilitare un componente composito a gestire dati lato-server, si può invocare un managed bean in uno dei seguendo modi:

* Passare il riferimento del managed bean al componente composito
* Usare direttamente proprietà del managed bean

L’applicazione di esempio mostrata al capitolo 14.4 mostra come usare un managed bean con un componente composito passando al componente il riferimento al managed bean.

## 14.3 Validare i valori di un componente composito

JavaServer Faces offre i seguenti tags per validare i valori di componenti di input.

Questi tags possono essere usati con i tag *composite:valueHolder* o *composite:editableValueHolder*.

La tabella sottostante elenca i tags di validazione (vedere 11.3 per la lista completa e dettagli):



## 14.4 L’applicazione di esempio compositecomponentexample

uesta applicazione crea un componente composito che accetta un nome (o qualsiasi stringa).Questa applicazione crea un componente composito che accetta un nome (o una qualunque stringa).

Il componente interagisce con un managed bean che calcola se le lettere nel nome, se convertite in valori numerici, equivalgono a un numero primo.

Il componente mostra la somma dei valori delle lettere e fornisce la risposta.

L’applicazione ha un file di un componente composito, una using page e un managed bean.

### 14.4.1 Il file del componente composito

Il file del componente composito è un file XHTML chiamato *PrimePanel* che ha una sezione *composite:interface* che dichiara le labels per il il nome e il bottone di comando.

Inoltre dichiara un managed bean che definisce delle proprietà per il nome.

*<composite:interface>*

*<composite:attribute name="namePrompt"*

*default="Name, word, or phrase: "/>*

*<composite:attribute name="calcButtonText" default="Calculate"/>*

*<composite:attribute name="calcAction"*

*method-signature="java.lang.String action()"/>*

*<composite:attribute name="primeBean"/>*

*<composite:editableValueHolder name="nameVal" targets="form:name"/>*

*</composite:interface>*

L’implementazione del componente composito accetta il valore di input per la proprietà *name* del managed bean.

Il tag *h:outputStylesheet* specifica lo stylesheet come una risorsa riallocabile.

L’implementazioni quindi specifica il formato dell’output, usando proprietà del managed bean, e il formato dei messaggi di errore.

Il valore *sum* è renderizzato solo dopo essere stato calcolato e il report che dice se la somma è un numero primo o no è renderizzato solo se il valore di input è stato validato.

Non riporto tutto il resto della pagina, vederla sul programma.

### 14.4.2 La Using Page

La using page in questo esempio è un file XHTML (*index.xhtml*) che invoca il componente composito *PrimePanel.xhtml* insieme al managed bean. Serve a validare l’input:

*<div id="compositecomponent">*

*<ez:PrimePanel primeBean="#{primeBean}"*

*calcAction="#{primeBean.calculate}">*

*</ez:PrimePanel>*

*</div>*

### 1.4.4.3 Il managed bean

Il managed bean *PrimeBean.java* definisce il metodo chiamato *calculate* che esegue il calcolo della stringa di input e setta le proprietà.

Il bean per prima cosa crea un array di numeri primi.

Poi calcola la somma delle lettere nella stringa, con “a” che equivale a 1 e “z” che equivale a 26, e determina se il valore è contenuto nel’array di numeri primi.

Il valore delle lettere maiuscole e minuscole è lo stesso.

Il bean specifica il *size* minimo e massimo della stringa che è forzato dal bean di validazione *@Size*.

Il bean utilizza l’annotazione *@Model*  che unisce le annotazioni *@Named* e *@RequestScoped*.

*@Model*

*public class PrimeBean implements Serializable {*

*...*

*@Size(min=1, max=45)*

*private String name;*

*...*

*public String calculate() {*

*...*

*}*

*}*