# 11 Using Converters, Listeners, and Validators

Il capitolo precedente descriveva i componenti spiegava come aggiungerli a una pagina web. Questo capitolo fornisce informazioni sull’aggiunta di altre funzionalità ai componenti attraverso convertitori, listeners e validatori.

* I convertitori sono usati per convertire i dati ricevuti dai componenti input. I convertitori permettono ad un’applicazione di portare le caratteristiche di Java nel mondo basato su Stringhe della programmazione HTTP servlet.
* I listeners sono usati per “ascoltare” gli eventi che avvengono nella pagina e eseguire azioni definite.
* I validatori sono usati per convalidare i dati ricevuti dai componenti input. I validatori permettono di esprimere i vincoli sui dati dei form per garantire che i requisiti necessari siano soddisfatti prima di processare i dati input.

## 11.1 Using the Standard Converters

L’implementazione JavaServer Faces fornisce un set di implementazioni Converter che puoi usare per convertire i dati dei componenti. Il fine della conversione è di prendere i dati in formato String provenienti dalla Servlet API e convertirla in oggetti Java adatti per l’ambito di business.

L’implementazione standard Converter è allocata nel package javax.faces.convert. Generalmente, i convertitori sono assegnati implicitamente in base al tipo dell’espressione EL indicata dal valore del componente. Tuttavia, si può anche accedere a questi convertitori da un ID convertitore. La tabella mostra le classi converter e gli ID associato.



Un messaggio di errore standard è associato ad ognuno di questi convertitori. Se hai registrato uno di questi convertitori su un componente della tua pagina e il convertitore non è in grado di convertire il valore del componente, il messaggio d’errore del convertitore sarà mostrato sulla pagina. Per esempio, il seguente messaggio d’errore appare se BigIntegerConverter fallisce la conversione:

{0} must be a number consisting of one or more digits

In questo caso, il parametro sostitutivo {0} sarà rimpiazzato con il nome del componente input su cui è registrato il convertitore.

Due convertitori standard (DateTimeConverter e NumberConverter) hanno i propri tags, che consentono di configurare il formato dei dati del componente usando gli attributi del tag.

### 11.1.1 Converting a Component’s Value

Per usare un particolare convertitore per convertire un valore di un componente, hai bisogno di registrare il convertitore sul componente. Puoi registrare qualsiasi convertitore standard nei seguenti modi:

* Annida uno dei tags dei convertitori standard nel tag del componente. Questi tags sono f:convertDateTime e f:convertNumber.
* Collega il valore del componente a una proprietà di un managed bean dello stesso tipo del convertitore. Questa è la tecnica più usata.
* Crea un riferimento al convertitore dall’attributo converter del tag del componente, specificando l’ID della classe del convertitore.
* Annida un tag f:converter in un tag del componente, e usa o l’attributo converterId del tag f:converter o il suo attributo binding per far riferimento al convertitore.

Come esempio della seconda tecnica, se vuoi che i dati del componente vengano convertiti in un Integer, puoi semplicemente collegare il valore del componente a una proprietà del managed bean. Ecco un esempio:

Integer age = 0;

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

this.age = age

}

I dati dal tag h:inputText nell’esempio saranno convertiti ad un valore java.lang.Integer. il tipo Integer è un tipo supportato da NumberConverter. Se non hai bisogno di specificare istruzioni di formattazione usando l’attributo del tag f:convertNumber, e se uno dei convertitori standard sarà sufficiente, puoi semplicemente far riferimento a quel convertitore usando l’attributo converter del tag del componente.

Puoi inoltre annidare un tag f:converter in un tag del componente e usare o l’attributo converterId del tag del convertitore o il suo attributo binding per far riferimento al convertitore.

L’attributo converterId deve riferirsi all’ID del convertitore. Ecco un esempio che usa uno degli ID della tabella precedente.

<h:inputText value=”#{loginBean.age}”>

<f:converter converterId=”javax.faces.Integer” />

</h:inputText>

Anziché usare l’attributo converterId, il tag f:converter può usare l’attributo binding. L’attributo binding deve trasformarsi in una proprietà di un bean che accetta e restituisce un’appropriata istanza Converter.

Puoi anche creare convertitori personalizzati e registrarli sui componenti usando il tag f:converter.

### 11.1.2 Using DateTimeConverter

Puoi convertire un dato del componente in una java.util.Date annidando il tag convertDateTime nel tag del componente. Il tag convertDateTime ha diversi attributi che permettono di specificare il formato e il tipo del dato. La tabella mostra gli attributi.

Ecco un esempio di un tag convertDateTime:

<h:outputText value=”#cashierBean.shipDate}”>

<f:convertDateTime type=”date” dateStyle=”full” />

</h:outputText>

Quando si collega il DateTimeConverter a un componente, assicurarsi che la proprietà del managed bean a cui il componente è legato sia del tipo java.util.Date.

Il tag dell’esempio può dare il seguente output:

Saturday, Semptember 21, 2013

Puoi anche mostrare la stessa data usando il seguente tag in cui è specificato il formato della data:

<h:outputText value=”#{cashierBean.shipDate}”>

<f:convertDateTime pattern=”EEEEEEEE, MMM dd, yyyy” />

</h:outputText>

Se vuoi mostrare la data dell’esempio in spagnolo, puoi usare l’attributo locale:

<h:outputText value=”#{cashierBean.shipDate}”>

<f:convertDateTime dateStyle=”full”

Locale=”es”

timeStyle=”long” type=”both” />

</h:outputText>

Questo tag darà il seguente output:

jueves 24 de octubre de 2013 15:07:04 GTM



### 11.1.3 Using NumberConverter

Puoi convertire un dato di un componente in un java.lang.Number annidando il tag convertNumber nel tag del componente. Il tag convertNumber ha diversi attributi che permettono di specificare il formato e il tipo del dato. La tabella mostra gli attributi.

Il seguente esempio usa un tag convertNumber per mostrare i prezzi totali dei contenuti di un carrello:

<h:outputText value=”#{cart.total}”>

<f:converterNumber currencySymbol=”$” type=”currency”/>

</h:outputText>

Quando si collega il NumberConverter ad un componente, assicurarsi che la proprietà del managed bean a cui è legato il componente è un tipo primitivo o ha un tipo di java.lang.Number. nel precedente esempio, cart.total è di tipo double.

Ecco un esempio di un numero che questo tag può mostrare:

$934

Il risultato può essere anche mostrato usato il seguente tag in cui è specificato il pattern della valuta:

<h:outputText id=”cartTotal” value=”#{cart.total}”>

<f:convertNumber pattern=”$####”/>

</h:outputText>



## 11.2 Registering Listeners on Components

Lo sviluppatore di un’applicazione può implementare listeners come classi o come metodi di managed bean. Se un listener è un metodo di managed bean, l’autore della pagina fa riferimento al metodo o dall’attributo valueChangeListener del componente o dal suo attributo actionListener. Se il listener è una classe, l’autore della pagina può riferirsi al listener o da un tag f:valueChangeListener o dal tag f:actionListener e annidare il tag nel tag del componente per registrare il listener su questo componente.

11.4.3 Referencing a Method That Handles an Action Event e 11.4.4 Referencing a Method That Handles a Value-Change Event spiegano come l’autore di una pagina usa gli attributi valueChangeListener e actionListener per riferirsi ai metodi del managed bean che gestiscono gli eventi.

### 11.2.1 Registering a Value-Change Listener on a Component

L’autore di una pagina può registrare un’implementazione ValueChangeListener su un componente che implementa EditableValueHolder annidando un tag f:valueChangeListener in un tag del componente sulla pagina. Il tag f:valueChangeListener supporta gli attributi mostrati in tabella, di cui uno dev’essere usato.



Il seguente esempio mostra un value-change listener registrato sul componente:

<h:inputText id=”name”

size=”30”

value=”#{cashierBean.name}”

required=”true”

requiredMessage=”#{bundle.ReqCustomerName}”>

<f:valueChangeListener

type=”javaeetutorial.dukesbookstore.listeners.NameChanged”/>

<h:inputText>

Nell’esempio, l’attributo type del tag core specifica il listener personalizzato NameChanged come l’implementazione ValueChangeListener registrata sul componente name.

Dopo che questo tag viene processato e il valore locale è stato convalidato, l’istanza del componente corrispondente metterà in coda il ValueChangeEvent associato allo specifico ValueChangeListener al componente.

L’attributo binding è usato per collegare un’implementazione ValueChangeListener a una proprietà del managed bean. Questo attributo lavora in modo simile all’attributo binding supportato dagli standard converter tags.

### 11.2.2 Registering an Action Listener on a Component

L’autore di una pagina può registrar un’implementazione ActionListener su un componente di comando annidando un tag f:actionListener in un tag del componente sulla pagina. Come il tag f:valueChangeListener, il tag f:actionListener supporta entrambi gli attributi type e binding. Uno di questi attributi dev’essere usato per riferirsi all’action listener.

Ecco un esempio di un tag h:commandLink che fa riferimento all’implementazione ActionListener:

<h:commandLink id=”Duke” action=”bookstore”>

<f:actionListener

type=”javaeetutorial.dukesbookstore.listeners.LinkBookChangeListene r” />

<h:outputText value=”#{bundle.Book201}/>

</h:commandLink>

L’attributo type del tag f:actionListener specifica il nome qualificatore completo della classe dell’implementazione ActionListener. Nello stesso modo del tag f:valueChangeListener, il tag f:actionListener supporta anche l’attributo binding.

In aggiunta al tag actionListener che permette di registrare un listener personalizzato su un componente, la libreria di tag core include il tag f:setPropertyActionListener. Si usa questo tag per registrare uno speciale action listener sull’istanza ActionSource associata al componente. Quando il componente viene attivato, il listener memorizzerà l’oggetto referenziato dall’attributo value del tag in un’oggetto referenziato dall’attributo target del tag.

La pagina bookcatalog.xhtml dell’applicazione Duke’s Bookstore usa f:setPropertyActionListener con due componenti: il componente h:commandLink usato per linkare la pagina bookdetails.xhtml e il componente h:commandButton usato per aggiungere un libro al carrello:

<h:dataTable id=”books”

value=”#{store.books}”

var=”book”

headerClass=”list-header”

styleClass=”list-background”

rowClasses=”list-row-even, list-row-odd”

border=”1”

summary=”#{bundle.BookCatalog}”>

...

<h:column>

<f:facet name=”header”>

<h:outputText value=”#{bundle.ItemTitle}”/>

</f:facet>

<h:commandLink action=”#{catalog.details}”

Value=”#{book.title}”>

<f:setPropertyActionListener target=”#{requestScope.book}”

value=”#{book}”/>

</h:commandLink>

</h:column>

...

<h:column>

<f:facet name=”header”>

<h:outputText value=”#{bundle.CartAdd}”/>

<f:facet>

<h:commandButton id=”add”

action=”#{catalog.add}

value=”#{bundle.CartAdd}”>

<f:setPropertyActionListener target=”#{requestScope.book}”

Value=”#{book}”/>

</h:commandButton>

</h:column>

I tags h:commandLink e h:commandButton sono interni al tag h:dataTable, che itera per la lista di libri. L’attributo var si riferisce a un singolo libro della lista di libri.

L’oggetto referenziato dall’attributo var di un tag h:dataTable è nell’ambito della pagina. Tuttavia, in questo caso hai bisogno di inserire questo oggetto nell’ambito richiesto cosicché quando l’utente attiva il componente commandLink per andare a bookdetails.xhtml o attiva il componente commandButton per andare a bookcatalog.xhtml, i dati del libro sono disponibili in quelle pagine. Perciò, il tag f:setPropertyActivationListener è usato per impostare l’oggetto del libro corrente nell’ambito richiesto quando il componente commandLink o commandButton è attivato.

## 11.3 Using the Standard Validators

La tecnologia JavaServer Faces fornisce un set di classi standard e i tags associati che gli autori della pagina e gli sviluppatori dell’applicazione possono usare per convalidare i dati di un componente. La tabella mostra tutte le standard validator class e i tag che consentono di usare i validatori dalla pagina.



Tutte queste validator class implementano l’interfaccia Validator. I component writers e gli sviluppatori dell’applicazione possono inoltre implementare questa interfaccia per definire un proprio set di limiti per il valore di un componente.

Come gli standard converters, ognuno di questi validatori ha uno o più messaggi di errore standard associati. Se hai registrato uno di questi validatori su un componente sulla tua pagina e il validatore non è in grado di convalidare il valore del componente, sarà visualizzato il messaggio d’errore sulla pagina. Per esempio, il messaggio d’errore che viene visualizzato quando il valore di un componente supera il valore massimo consentito da LongRangeValidator è:

{1}: Validation Error: Value is greater than allowable maximum of “{0}”

In questo caso, il parametro sostitutivo {1} sarà rimpiazzato dall’etichetta del component o il suo id, e il parametro sostitutivo {0} sarà rimpiazzato con il valore massimo consentito dal validatore.

Anziché utilizzare i validatori standard, puoi usare Bean Validation per convalidare i dati. Se specifichi i vincoli di validazione del bean sulle proprietà del tuo managed bean, i vincoli saranno automaticamente allocati nei campi corrispondenti sulle pagine dell’applicazione. Non hai bisogno di specificare il tag validateBean per usare Bean Validation, ma il tag permette di usare caratteristiche più avanzate di Bean Validation. Per esempio, puoi usare l’attributo validationGroups per specificare gruppi vincolati.

Puoi inoltre creare e registrare validatori personalizzati, sebbene Bean Validation abbia reso questa caratteristica meno utile.

### 11.3.1 Validating a Component’s Value

Per convalidare il valore di un componente usando un particolare validatore, hai bisogno di registrare questo validatore sul componente. Puoi fare ciò in uno dei seguenti modi.

* Annidando il validator tag corrispondente (in tabella) ndl tag del componente. Puoi usare altre tags standard nello stesso modo.
* Creare un riferimento a un metodo che esegue la convalida dall’attributo validator del tag del componente.
* Annidando un validator tag nel tag del componente, e usando o l’attributo validatorId del validator tag o il suo attributo binding per far riferimento al validatore.

Ricorda che questo validatore può essere eseguito solo sui componenti che implementano EditableValueHolder, perché questi componenti accettano valori che possono essere convalidati.

### 11.3.2 Using Validator Tags

L’esempio seguente mostra come usare il validator tag f:validateLongRange sul componente input chiamato quantity:

<h:inputText id=”quantity” size=”4” value=”#{item.quantity}”>

<f:validateLongRange minimum=”1”/>

</h:inputText>

<h:message for=”quantity”/>

Questo tag richiede all’utente di inserire un numero che sia almeno 1. Il tag validateLongRange ha anche un attributo maximum, che imposta il valore massimo per l’input.

Gli attributi di tutti i validatori standard accettano i valori delle espressioni EL. Questo significa che gli attributi possono far riferimento a proprietà di managed bean piuttosto che specificare valori letterali. Per esempio, il tag f:validateLongRange dell’esempio precedente può far riferimento a proprietà di un managed bean chiamate minimum e maximum per prendere i valori minimo e massimo accettabili per l’implementazione del validatore, come mostrato in questo frammento dall’esempio guessnumber-jsf:

<h:inputText id=”userNo”

title=”Type a number from 0 to 10:”

value=#{userNumberBean.userNumber}”>

<f:validateLongRange minimum=”#{userNumberBean.minimum}”

maximum=”#{userNumberBean.maximum}”/>

</h:inputText>

Il seguente tag f:validateRegex mostra come puoi garantire che la password sia lunga da 4 a 10 caratteri e contenga almeno una cifra, almeno una lettera minuscola e almeno una lettera maiuscola:

<f:validateRegex pattern=”((?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{4,10})”

For=”passwordVal”/>

## 11.4 Referencing a Managed Bean Method

Un tag di un componente ha un set di attributi per fare riferimento ai metodi di un managed bean che può eseguire determinate azioni per il componente associato con il tag. Questi attributi sono riassunti nella tabella.



Solo i componenti che implementano ActionSource possono usare gli attributi action e actionListener. Solo i componenti che implementano EditableValueHolder possono usare gli attributi validator e valueChangeListener.

Il tag del componente fa riferimento a un metodo di un managed bean usando un’espressione di metodo come valore di uno degli attributi. Questo metodo riferito da un attributo deve seguire un particolare segno distintivo, definito dalla definizione dell’attributo del tag nella documentazione JavaServer Faces Facelets Tag Library. Per esempio, la definizione dell’attributo validator del tag inputText è la seguente:

void validate(javax.faces.context.FacesContext,

javax.faces.component.UIComponent, java.lang.Object)

### 11.4.1 Referencing a Method That Performs Navigation

Se la tua pagina include un component, come un bottone o un link, che provoca che l’applicazione navighi verso un’altra pagina quando il componente viene attivato, il tag corrispondente al componente deve includere un attributo action. Questo attributo fa una delle seguenti azioni:

* Specifica un risultatoogico di tipo String che dice all’applicazione quale pagina accedere dopo
* Fa riferimento a un metodo di un managed bean che esegue alcuni processi e restituisce un risultato logico String

Il seguente esempio mostra come far riferimento a un metodo di navigazione:

<h:commandButton value=”#{bundle.Submit}”

action=”#{cahierBean.submit}” />

### 11.4.2 Referencing a Method That Handles an Action Event

Se un componente sulla tua pagina genera un evento action, e se questo evento viene gestito da un metodo di un managed bean, fai riferimento al metodo usando l’attributo actionListener del componente.

Il seguente esempio mostra come tale metodo può essere referenziato:

<h:commandButton id=”Duke” action=”bookstore”

actionListener=”#{actionBean.chooseBookFromLink}”>

l’attributo actionListener di questo tag del component fa riferimento al metodo chooseBookFromLink usando un’espressione di metodo. Il metodo chooseBookFromLink gestisce l’evento quando l’utente clicca il link interpretato da questo componente.

### 11.4.3 Referencing a Method That Performs Validation

Se l’input di uno dei componenti sulla tua pagina è convalidato da un metodo di un managed bean, fai riferimento al metodo dal tag del componente usando l’attributo validator.

Il seguente esempio mostra come referenziare un metodo che esegue la validazione su inputGuess, un componente di input:

<h:inputText id=”inputGuess”

value=”#{userNumberBean.userNumber}”

required=”true” size=”3”

disabled=”#{userNumberBean.number eq userNumberBean.userNumber ...}”

validator=”#{userNumberBean.validateNumberrange}”>

</h:inputText>

il metodo validateNumberRange del managed bean verifica che il valore di input sia compreso nel range, che cambia ogni volta che viene fatta un’altra stima.

### 11.4.4 Referencing a Method That Handles a Value-Change Event

Se vuoi che il componente sulla tua pagina generi un evento value-change e vuoi che l’evento sia gestito da un metodo di un managed bean anzichè l’implementazione ValueChangeListener, fai riferimento al metodo usando l’attributo valueChangeListener del componente:

<h:inputText id=”name”

size=”30”

value=”#{cashierBean.name}”

required=”true”

valueChangeListener=”#{cashierBean.processValueChange}” />

<h:inputText>

l’attributo valueChangeListener di questo tag del componente fa riferimento al metodo processValueChange di CashierBean usando un’espressione di metodo. Il metodo processValueChange gestisce l’evento dell’inserimento del nome da parte dell’utente nel campo di testo tradotto da questo componente.