Ajax è l’acronimo di Asynchronous JavaScript and XML, un gruppo di tecnologie web che abilitano la creazione di applicazioni web dinamiche e altamente responsive.

Usando Ajax , le applicazioni web possono recuperare contenuti dal server senza interferire con la visualizzazione sul client.

Nella piattaforma JavaEE, la tecnologia JavaServer Faces offre il supporto nativo per Ajax.

Le vecchie applicazioni web erano create principalmente da pagine statiche. Quando una pagina web statica è aggiornata da un client, l’intera pagina viene ricaricata per visualizzare la modifica ottenuta.

Le ricariche ripetute della pagina impattano sulle performance dell’applicazione e causano un eccessivo dispendio di banda.

Tecnologie come Ajax sono state create per superare questi problemi.

Questo capitolo descrive le funzionalità d’uso di Ajax nelle applicazioni web JavaServer Faces.

## 13.1 Panoramica di Ajax

Ajax si riferisce alle tecnologie JavaScript e XML che sono largamente diffuse per la creazione di contenuti web dinamici e asincroni.

Sebbene Ajax non sia limitato alle tecnologie JavaScript e XML, esso viene quasi sempre utilizzato per unire tali tecnologie che vengono usate insieme dalle applicazioni web.

Questo tutorial si concentra sull’utilizzo delle funzionalità di Ajax basate su JavaScript nelle applicazioni web JavaServer Faces.

JavaScript è un linguaggio di scripting dinamico per le applicazioni web che permette agli utenti di aggiungere funzionalità migliorate a interfacce utenti e alle pagine web di interagire con i clients in maniera asincrona. JavaScript viene eseguito principalmente a lato client (nel browser) e in tal modo riduce l’accesso al server da parte dei clients.

Quando una funzione JavaScript spedisce una richiesta asincrona dal client al server, il server manda indietro una risposta che viene usata per aggiornare il Document Object Model (DOM) della pagina.

Tale risposta è spesso presentata come un documento in formato XML.

Il termine Ajax si riferisce a questa interazione tra client e server.

La risposta del server non deve per forza essere in XML, potrebbe essere anche in altri formati quali il JSON che vedremo però più avanti.

Ajax abilita l’update asincrono e parziale delle applicazioni web, questa funzionalità permette una renderizzazione praticamente in tempo reale delle pagine web altamente responsive che crea.

Le applicazioni basate su Ajax possono accedere alle informazioni del server e dei processi e recuperare i dati senza interferire con la visualizzazione e il rendering della pagina web sul client.

Alcuni vantaggi dell’utilizzo di Ajax sono:

* Validazione dei dati del form in tempo reale, con l’eliminazione della necessità di sottomettere il form per la sua verifica
* Funzionalità potenziate per le pagine web, come i prompts (suggerimenti) per user name e password
* Update parziale dei contenuti web, evitando il completo ricaricamento della pagina

## 13.2 Using Ajax Functionality with JavaServer Faces Technology

Una funzionalità Ajax può essere aggiunta a un’ applicazione JavaServer Faces in uno dei seguenti modi:

* Aggiungendo il codice JavaScript necessario all’applicazione
* Utilizzando la resource library Ajax

Nel vecchie versioni della piattaforma Java EE, le applicazioni JavaServer Faces fornivano le funzionalità Ajax aggiungendo il codice JavaScript necessario alla pagina. Nella piattaforma Java EE7, il supporto a Ajax è offerto dalla resource library Javascript nativa.

Con il supporto di questa resource library, i componenti standard JavaServer Faces, come i bottoni, le labels o i campi di testo, possono essere abilitati alle funzionalità Ajax.

Si può inoltre caricare questa libreria e usare i suoi metodi direttamente da dentro il codice del managed bean. Il prossimo capitolo descrive l’utilizzo della libreria.

Dato che il component model della tecnologia JavaServer Faces può essere esteso, possono essere creati componenti personalizzati con funzionalità Ajax.

Gli esempi del tutorial includono una versione Ajax dell’applicazione *guessnumber*. Per maggiori informazioni vedere il capitolo dedicato (13.11).

Il tag specifico di Ajax *f:ajax* e i suoi attributi sono spiegati nei prossimi capitoli.

## 13.3 Using Ajax with Facelets

Come detto nei precedenti capitoli, la tecnologia JavaServer Faces supporta Ajax utilizzando una Javascript resource library che è offerta come parte delle librerie JavaServer Faces core.

Questa libreria può essere usata nelle applicazioni web JavaServer Faces in uno dei seguenti modi:

* Utilizzando il tag *f:ajax* insieme a un altro componente standard in un’applicazione Facelets.

Questo metodo aggiunge le funzionalità Ajax a ogni componente UI senza codice e configurazione addizionale.

* Utilizzando il metodo dell’API JavaScript *jsf.ajax.request()* direttamente dall’interno dell’applicazione Facelets.

Questo metofo offre accesso diretto ai metodi Ajax e permette il controllo personalizzato del comportamente del componente.

### 13.3.1 Using the f:ajax Tag

Il tag *f:ajax* è un tag di JavaServer Faces core che fornisce funzionalità Ajax a ogni componente UI quando usato insieme a tale componente.

Nel seguente esempio, il comportamento di Ajax è aggiunto a un componente input includendo il core tag *f:ajax*:

<h:inputText value="#{bean.message}">

<f:ajax />

</h:inputText>

In questo esempio, sebbene Aja sia abilitato, gli altri attributi del tag *f:ajax* non sono definiti.

Se un evento non è definito viene eseguita l’azione di default del componente.

Per il componente *inputText*, se non viene specificato alcun attributo dell’evento, l’evento di default è *valueChange*.

La tabella sotto riportata elenca gli attributi del tag *f:ajax* e le loro azioni di default.





Le keywords elencate nella tabella sottostante possone essere usate con gli attributi *execute* e *render* del tag *f:ajax*.



Notare che quando si usa il tag *f:ajax* in una pagina Facelets, la Javascript resource library viene caricata implicitamente. Può anche essere caricata esplicitamente come descritto in Loading JavaScript as a Resource (13.10).

## 13.4 Sending an Ajax Request

Per attivare le funzionalità Ajax l’applicazione web deve creare e inviare al server, ce poi la processa, una richiesta Ajax.

L’applicazione utilizza gli attributi del tag *f:ajax* elencati nella tabella sopra per creare la richiesta.

I seguenti capitoli spiegano il processo di creazione e spedizione di una richiesta Ajax utilizzando alcuni di questi attributi.

Nota: Il metodo *jsf.ajax.request()* della resource library colleziona i dati forniti dal tag *f:ajax* e pubblica la richiesta al ciclo di vita di JavaServer Faces.

### 13.4.1 Using the event Attribute

L’attributo *event* definisce l’evento che scatena l’azione Ajax.

Alcuni dei possibili valori per quest’attributo sono *click, keyup, mouseover, focus* e *blur*.

Se non specificato, verrà applicato l’evento di default basato sul componente “padre”.

L’evento di default è *action* per i componenti *javax.faces.component.ActionSource* quali ad esempio i bottoni e *valueChange* per i componenti *javax.faces.component.EditableValueHolder*, come *inputText*.

Nel seguente esempio un tag Ajax è associato al componente button e l’evento che scatena l’azione Ajax è il click col mouse:

*<h:commandButton id="submit" value="Submit">*

*<f:ajax event="click" />*

*</h:commandButton>*

*<h:outputText id="result" value="#{userNumberBean.response}" />*

Nota: La lista di eventi è molto simile a quella di JavaScript, solo che non hanno il prefisso *on*.

L’evento di default per un bottone è *click*, quindi non c’è bisogno di specificare *event=”click”* per ottenere il comportamento desiderato.

### 13.4.2 Using the execute Attribute

L’attributo *execute* definisce il componente da eseguire sul server.

Il componente è identificato dal proprio attributo *id*.

E’ possibile specificare più di un componente eseguibile, se se ne vogliono specificare più di uno si devono dichiarare con una lista di componenti delimitati da uno spazio (space limited list).

Quando un componente viene eseguito, esso partecipa a tutte le fasi del ciclo vitale di processamento della richiesta eccetto a quella di Render Response.

Il valore dell’attributo *execute* può anche essere una keyword, come *@all, @none, @this* o *@form*.

Il valore di default è *@this* che si riferisce al componente all’interno del quale è annidato il tag *f:ajax*.

Il codice che segue specifica che il componente *h:inputText* con valore di *id* corrispondente a *userNo* deve essere eseguito quando viene cliccato il bottone:

*<h:inputText id="userNo"*

*title="Type a number from 0 to 10:"*

*value="#{userNumberBean.userNumber}">*

*...*

*</h:inputText>*

*<h:commandButton id="submit" value="Submit">*

*<f:ajax event="click" execute="userNo" />*

*</h:commandButton>*

### 13.4.3 Using the immediate Attribute

L’attributo *immediate* indica se gli input dell’utente debbano essere processati prima o dopo nel ciclo di vita dell’applicazione.

Se tale attributo è settato su true, gli eventi generati da questo componente vengono trasmessi durante la fase Apply Request Values, altrimenti durante la fase Invoke Application.

Se non definito il valore di default è false.

### 13.4.4 Using the listener Attribute

L’attributo *listener*  si riferisce a un’espressione di metodo che viene eseguita lato-server in risposta a un’azione Ajax sul client.

Il metodo del *listener javax.faces.event.AjaxBehaviorListener.processAjaxBehavior* viene chiamato una volta durante la fase Invoke Application del ciclo vitale.

Nel seguente codice preso dall’applicazione di esempio *reservation* è definito un attributo *listener* da un tag *f:ajax* che si riferisce a un metodo del bean:

*<f:ajax event="change" render="total"*

*listener="#{reservationBean.calculateTotal}"/>*

Ogni volta che il prezzo o il numero dei biglietti ordinati cambia, il metodo *calculateTotal* di *ReservationBean* ricalcola il costo totale dei biglietti e lo mostra nel componente output chiamato *total*.

## 13.5 Monitoring Events on the Client

Per monitorare richieste Ajax in entrata si utilizza l’attributo *onevent* del tag *f:ajax*.

Il valore di questo attributo è il nome di una funzione JavaScript.

JavaServer Faces chiama la funzione *onevent* a ogni stadio del processamento di una richiesta Ajax: begin, complete e success.

Quando si chiama la funzione JavaScript assegnata alla proprietà *onevent*, JavaServer Faces le passa un data object.

Il data object contiene le proprietà elencate nella tabella sottostante:



Utilizzando la proprietà *status* del data object si può identificare lo stato corrente della richiesta Ajax e monitorarne il progresso.

Nel seguente esempio *monitormyajaxevent* è una funzione JavaScript che monitora la richiesta Ajax inviata dall’evento:

*<f:ajax event="click" render="statusmessage" onevent="monitormyajaxevent"/>*

## 13.6 Handling Errors

JavaServer Faces gestisce gli errori Ajax attraverso l’uso dell’attributo *onerror* del tag *f:ajax*.

Il valore di questo attributo è il nome della funzione JavaScript.

Quando c’è un errore nel processamento di una richiesta Ajax, JavaServer Faces chiama la funzione *onerror* definita e le passa il data object.

Il data object contiene tutte le proprietà disponibili per l’ettributo *onevent* e, in aggiunta, le seguenti proprietà:

* *description*
* *errorName*
* *errorMessage*

Il tipo è *error*, la proprietà *status* del data object contiene uno dei valori di errore elencati nella tabella sotto:



Nel seguente esempio, qualunque errore che si presenti nel processamento della richiesta Ajax viene gestito dalla funzione Javascript *handlemyajaxerror*:

*<f:ajax event="click" render="errormessage" onerror="handlemyajaxerror"/>*

## 13.7 Receiving an Ajax Response

Dopo che l’applicazione invia una richiesta Ajax, essa viene processata a lato server e la risposta viene spedita indietro al client.

Come descritto in precedenza, Ajax da modo di effettuare l’update parziale delle pagine web.

Per abilitare questa funzionalità, la tecnologia JavaServer Faces da modo di processare parzialmente la view.

La gestione della risposta è definita dall’attributo *render* del tag *f:ajax*.

In modo simile all’attributo *execute*, l’attributo *render* definisce quali sezioni della pagina aggiornare.

Il valore dell’attributo *render* può il valore di *id* di uno o più componenti, ona delle keywords *@this, @all, @none*, o *@form* o un’espressione EL.

Nel seguente esempio l’attributo *render* identifica un componente di output che deve essere visualizzato quando viene cliccato il componente button (l’evento di default del comando button):

*<h:commandButton id="submit" value="Submit">*

*<f:ajax execute="userNo" render="result" />*

*</h:commandButton>*

*<h:outputText id="result" value="#{userNumberBean.response}" />*

Nota: Anche questa volta la risposta è gestita dal metodo *jsf.ajax.request()* che registra un callback di gestione della risposta quando viene creata la richiesta originale (“*mettiti il preservativo dei puffi caro*!”).

Quando la risposta è spedita indietro al client viene evocato il callback.Questo callback aggiorna automaticamente il DOM lato-client per riflettere la risposta renderizzata.

## 13.8 Ajax Request Lifecycle

Una richiesta Ajax è diversa dalle altre tipiche richieste JavaServer Pages e il suo processamento è gestito differentemente dal ciclo vitale.

Come già detto, quando una richiesta Ajax viene ricevuta, lo stato associato a tale richiesta è catturato dal *javax.faces.context.PartialViewContext*.

Questo oggetto fornisce accesso a informazioni tipo quali componenti che saranno processati/renderizzati.

Il metodo *processPartial* di *PartialViewContext* utilizza questa informazione per eseguire la processazione e renderizzazione parzial del component tree.

L’attributo *execute* del tag *f:ajax* identifica quale segmente del component tree a lato server deve essere processato.

Dato che i componenti possono essere identificati univocamente nel component tree di JavaServer Faces, è facile identificare e processare un singolo componente, alcuni componenti o l’intero component tree.

Ciò è possibile tramite il metodo *visitTree* della classe *UiComponent*.

I componenti identificati quindi passano attraverso le fasi del ciclo vitale di JavaServer Faces.

Durante la fase di Render viene esaminato l’attributo *render*.

I componenti identificati vengono trovati e gli viene “chiesto” (asked) di renderizzare loro stessi e i loro children.

I componenti vengono quindi inpacchettati e spediti indietro come risposta al client.

## 13.9 Grouping of Components

I capitoli precedenti descrivono come associare un singolo componente UI a una funzionalità Ajax.

Si può anche associare Ajax a più componenti alla volta raggrupandoli insieme in una pagina.

Il seguente esempio mostra come farlo:

*<f:ajax>*

*<h:form>*

*<h:inputText id="input1" value="#{user.name}"/>*

*<h:commandButton id="Submit"/>*

*</h:form>*

*</f:ajax>*

Nell’esempio nessun componente è ancora associato con un attributo Ajax *event* o *render*.

Quindi nessun’azione avrà luogo in caso di input dell’utente.

I componenti sopra citati possono essere associati agli attributi *event* e *render* nel seguente modo:

*<f:ajax event="click" render="@all">*

*<h:form>*

*<h:inputText id="input1" value="#{user.name}"/>*

*<h:commandButton id="Submit"/>*

*</h:form>*

*</f:ajax>*

Nel nuovo codice di esempio, quando un utente clicca uno dei componenti, l’aggiornamento sarà visualizzato per tutti i componenti.

Si puo’ ulteriormente regolare l’azione Ajax aggiungendo eventi specifici a ognuno dei componenti, in tal caso la funzionalità Ajax diventa cumulativa.

Considerare il seguente esempio:

*<f:ajax event="click" render="@all">*

*...*

*<h:commandButton id="Submit">*

*<f:ajax event="mouseover"/>*

*</h:commandButton>*

*...*

*</f:ajax>*

Ora il componente bottone può eseguire un’azione Ajax nel caso di un evento *mouseover* e anche una in caso dell’evento mouse-click.

## 13.10 Loading JavaScript as a Resource

Il file di risorse JavaScript fornito con la tecnologia JavaServer Faces è chiamato *jsf.js* ed è disponibile nella libreria *javax.faces*.

La libreria di risorse in questione supporta le funzionalità Ajax nelle applicazioni JavaServer Faces.

Utilizzando un tag *f:ajax* su una pagina, la risorsa *jsf.js* viene consegnata automaticamente al client.

Non è necessario usare il tag *h:outputScript* per specificare tale risorsa, tale tag può essere usato per specificare altre librerie JavaScript.

Per usare una risorsa JavaScript (libreria) direttamente con *UIComponent* bisogna caricare esplicitamente la risorsa in uno dei seguenti modi:

* Utilizzando il tag *h:outputScript* direttamente da una pagina Facelets
* Utilizzando l’annotazione *javax.faces.application.ResourceDependency* su una classe Java *UIComponent*.

### 13.10.1 Using JavaScript API in a Facelets Application

Per usare l’API di risorse JavaScript direttamente in un’applicazione web, ad esempio una pagina Facelets, c’è bisogno di identificare la risorsa JavaScript di default per la pagina con l’aiuto del tag *h:outputScript*.

Ad esempio, considerare il seguente pezzo di una pagina Facelets:

*<h:form>*

*<h:outputScript name="jsf.js" library="javax.faces" target="head"/>*

*</h:form>*

Specificando il target come head si fa sì che la risorsa sia renderizzata all’interno dell’elemento head della pagina HTML.

Il prossimo passo è di identificare il componente a cui si vuole unire la funzionalità Ajax.

Tale funzionalità si aggiunge utilizzando la JavaScript API.

Per esempio:

*<h:form>*

*<h:outputScript name="jsf.js" library="javax.faces" target="head">*

*<h:inputText id="inputname" value="#{userBean.name}"/>*

*<h:outputText id="outputname" value="#{userBean.name}"/>*

*<h:commandButton id="submit" value="Submit"*

*onclick="jsf.ajax.request(this, event,*

*{execute:'inputname',render:'outputname'});*

*return false;" />*

*</h:form>*

Il metodo *jsf.ajax.request* prende tre parametri che specificano la fonte, l’evento e le opzioni.

Il parametro source (fonte) identifica l’elemento DOM che ha generato la richiesta Ajax, in genere *this*.

Il parametro *event* è opzionale e identifica l’evento DOM che ha scatenato la richiesta.

Il parametro opzionale options contiene un set di coppie nome/valore, i cui valori ammissibili sono indicati nella tabella sotto:





Se nessun identificatore è specificato, la parola chiave che viene associata di default all’attributo *execute* è *@this* e all’attributo *render* è *@none*.

Si può anche posizionare il metodo JavaScript in un file e includerlo come risorsa.

### 13.10.2 Using the @ResourceDependency Annotation in a Bean Class

Si utilizza l’annotazione *javax.faces.application.ResourceDependency* per far sì che la classe bean carichi la libreria di default *jsf.js*.

Per caricare la risorsa Ajax dal lato server si utilizza il metodo *jsf.ajax.request* all’interno della classe bean.

Questo metodo è usato solitamente quando si crea un componente personalizzato o un rendered personalizzato per un componente.

L’esempio seguente mostra come caricare la risorsa nella classe bean:

*@ResourceDependency(name="jsf.js" library="javax.faces" target="head")*

## 13.11 The ajaxguessnumber Example Application

Per dimostrare i vantaggi dell’utilizzo di Ajax andiamo a modificare l’esempio *guessnumber* del capitolo 8.

Con le modifiche apportate la risposta non necessita di essere visualizzata sulla pagina *response.xhtml*.

Invece viene effettuata una chiamata asincrona al bean lato server e la risposta viene visualizzata sulla pagina di origine eseguendo solo il componente di input invece che sottomettendo il form.

### 13.11.1 The ajaxguessnumber Source Files

I cambiamenti all’applicazione *guessnumber* sono in due source files.

#### 13.11.1.1 The ajaxgreeting.xhtml Facelets Page

La pagina Facelets *ajaxgreeting.xhtml* è praticamente uguale alla vecchia pagina *greeting.xhtml*.

La modifica più importante è nel tag *h:commandButton* in cuiè stato rimosso l’attributo *action* sostituendolo con il tag *f:ajax*.

Questo tag specifica che quando il bottone viene cliccato, il componente *h:insputText* con valore di *id userNo* deve essere eseguito.

I componenti all’interno del panel *outputGroup* sono quindi renderizzati.

Se avviene un errore di validazione il managed bean non viene eseguito e appare il messaggio di errore.

In caso contrario il risultato viene visualizzato nel componente *result*.

#### 13.11.1.2 The UserNumberBean Backing Bean

E’ stato fatto un piccolo cambiamento anche nel codice di *UserNumberBean* in modo che il componente di output non mostri alcun messaggio per il valore di default (null) della proprietà *response*.

#### 13.11.1.3 The DukesNumberBean CDI Managed Bean

Il menaged bean session- scoped *DukesNumberBean* conserva la gamma di possibili numeri e sceglie casualmente un numero da tale gamma.

E’ iniettato in *UserNumberBean* con l’annotazione CDI *@Inject* in modo che il valore del numero casuale possa essere comparato con il numero inserito dall’utente:

*@Inject*

*DukesNumberBean dukesNumberBean;*

Le CDI (Context and Dependency Injections) saranno spiegate in dettaglio al capitolo 23.