JAVA + Spring

Introduzione a JavaEE + Spring

Presented by

Gianluca Avizzano/Donata Uricchio

IBM Client Innovation Center - Bari

Napoli, 12/02/2022

IBM Client Innovation Center
Italy

Java Server Page

JSP

Linguaggio flessibile e multipiattaforma in grado di generare pagine dinamiche lato server.

una pagina JSP è costituita da markup (X)HTML frammentato da sezioni di codice Java.

Si basano su tecnologia JAVA e quindi object oriented.

Può essere invocata usando due diversi metodi:

- la richiesta venie effettuata direttamente ad una pagina JSP, che grazie alle risorse messe a disposizione lato server, è in grado di elaborare i dati di ingresso per ottenere e restituire l'output voluto
- la richiesta può essere filtrata da una servlet che, dopo l'elaborazione dei dati, incapsula adeguatamente i risultati e richiama la pagina JSP, che produrrà l'output

JSP esempio

```
<html>
        <head>
                <title>Test</title>
        </head>
        <body>
                <p>Data attuale: <%= (new java.util.Date()).toLocaleString()%></p>
        </body>
</html>
```

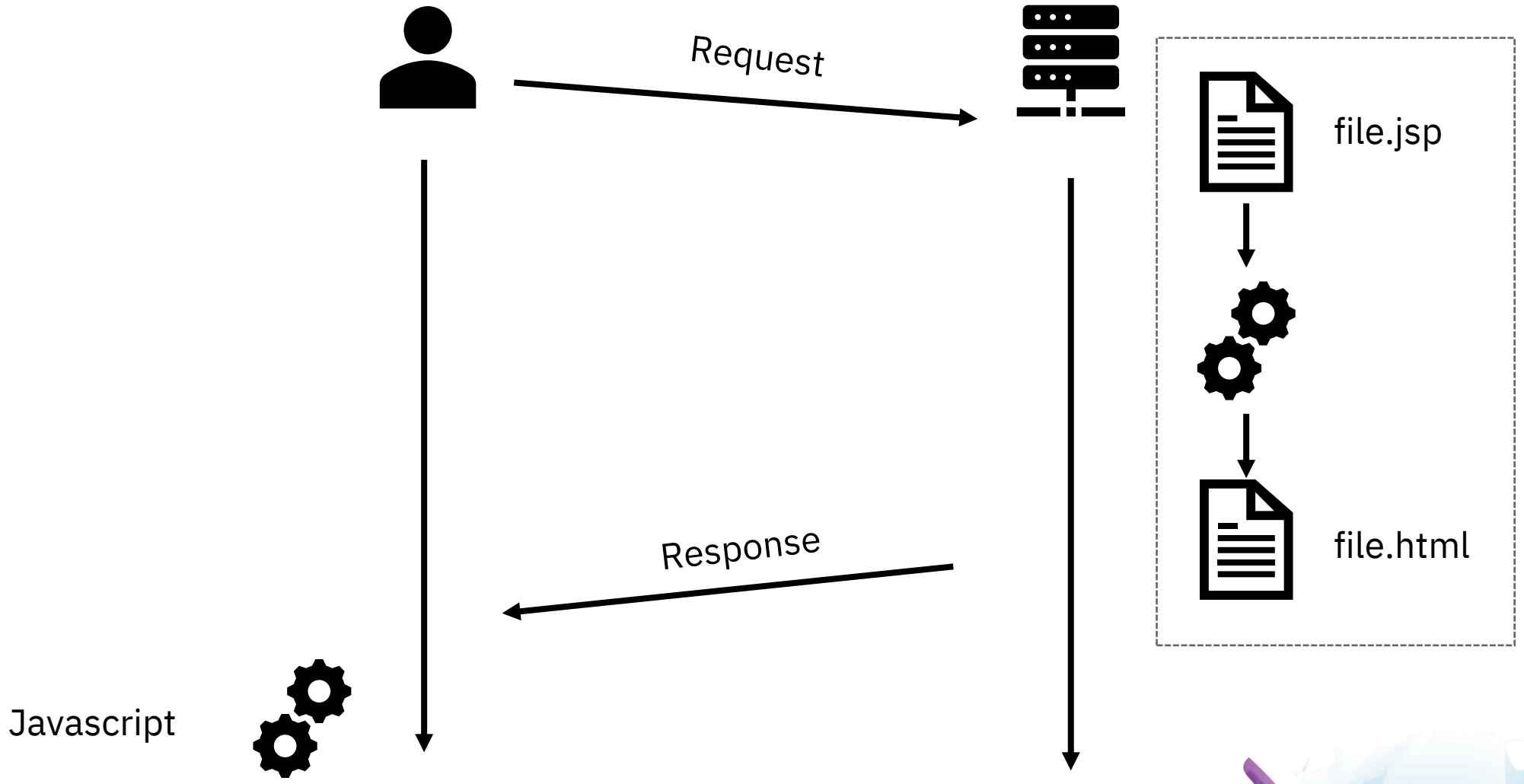
Risultato:

Data attuale: 09-Feb-2023 21:24:25

JSP esempio

```
<%! int day = 3; %>
 <html>
            <head>
                        <title>IF...ELSE Example</title>
            </head>
            <body>
                        <% if (day == 1 || day == 7) { %>
                        <p> Today is weekend</p>
                        <% } else { %>
                        <p> Today is not weekend</p> <% } %>
            </body>
</html>
```

JSP vs Javascript



JSTL: JSP Standard Tag Library

JSTL espone una collezione di funzionalità core a molti applicativi che utilizzano JSP.

Per utilizzare i tag, è necessario importare la libreria:

<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>

Tag Utili
<c:out>
<c:set >
<c:if>
<c:choose>, <c:when> <c:otherwise >
<c:forEach >

JSTL: Esempio

- <c:url> provvede a costruire l'indirizzo considerando il dominio del sito e la pagina di origine della JSP, esempio: *http:localhost:8080/search*

- Con <c:forEach> è possibile stampare una lista di oggetti.

- Con <c:choose>, <c:when>, <c:if> è possibile definire delle istruzioni condizionali.

```
<a href="<c:url value="/search"/>">Accedi alla pagina di ricerca per Product Line</a>
<br><br>
<c:choose>
    <c:when test="${not empty products}">
        <c:forEach var="product" items="${products}">
            Codice: <c:out value="${product.productcode}" />
            Nome: <c:out value="${product.productname}" />
            Prezzo: <c:out value="${product.buyprice}" />&euro;
            <br>
            <br>
        </c:forEach>
    </c:when>
    <c:otherwise>
        <p>Nessun prodotto presente all'interno del catalogo.</p>
    </c:otherwise>
</c:choose>
```

JSF

Java Server Faces (JSF) è un framework

- java basato sul design pattern architetturale Model-View-Controller
- Interfaccia utente per la creazione di applicazioni web

VANTAGGI PRINCIPALI

JSF mette a disposizione dei **componenti predefiniti**, che lo sviluppatore deve semplicemente richiamare nelle proprie pagine Web. Ciò rende possibile sviluppare l'interfaccia Web di una applicazione secondo la logica RAD, Rapid Application Development, tipica degli ambienti di sviluppo .NET.

JSF è basato sugli eventi. **Gli eventi** sono costituiti dalle azioni che l'utente effettua sui componenti di interfaccia. A questi eventi vengono registrati dei listener per la gestione lato server degli stessi.

I componenti predefiniti sono considerati "intelligenti", cioè sono in grado di **validare automaticamente i dati** inseriti dall'utente e di memorizzare il proprio stato.

Molti ambienti di sviluppo, tra i quali Eclipse e NetBeans, presentano **tool per la gestione visuale**, sia della parte relativa alla composizione delle pagine in JSF, sia per quanto riguarda la gestione dei suoi file di configurazione.

JSF

Il cuore del framework è il file *faces-config.xml*, nel quale è possibile definire le regole di navigazione all'interno delle pagine dell'applicazione.

`Nel file web.xml, della nostra applicazione, occorre configurare il framework JSF.`

E' necessario definire, mediante il parametro javax.faces.CONFIG_FILES, il path del file faces-config.xml e occorre dichiarare la FacesServlet e il suo relativo URL-pattern.

La versione standard di JSF prevede due taglibrary:

core: definisce i tag di base di JSF, indipendenti dalla tecnologia di rendering della pagina;

HTML: fornisce componenti riusabili specifici per renderizzare in HTML i componenti server di JSF.

WebService

I Web service sono applicazioni client e server che comunicano tramite il protocollo HTTP (HyperText Transfer Protocol) del World Wide Web (WWW)

A livello concettuale, un servizio è un componente software fornito attraverso un endpoint accessibile dalla rete. Il consumatore del servizio e il fornitore utilizzano i messaggi per scambiare informazioni

VANTAGGI

- permettono l'interoperabilità tra diverse applicazioni software e su diverse piattaforme hardware/software
- Utilizzano un formato dei dati di tipo testuale, quindi più comprensibile e più facile da utilizzare per gli sviluppatori
- Normalmente, essendo basati sul protocollo HTTP, non richiedono modifiche alle regole di sicurezza utilizzate come filtro dai firewall
- Sono semplici da utilizzare e possono essere combinati l'uno con l'altro (indipendentemente da chi li fornisce e da dove vengono resi disponibili) per formare servizi "integrati" e complessi
- Permettono di riutilizzare applicazioni già sviluppate.

WebService

A livello tecnico, i servizi Web possono essere implementati in vari modi.

I due tipi di servizi Web più discussi possono essere distinti in :

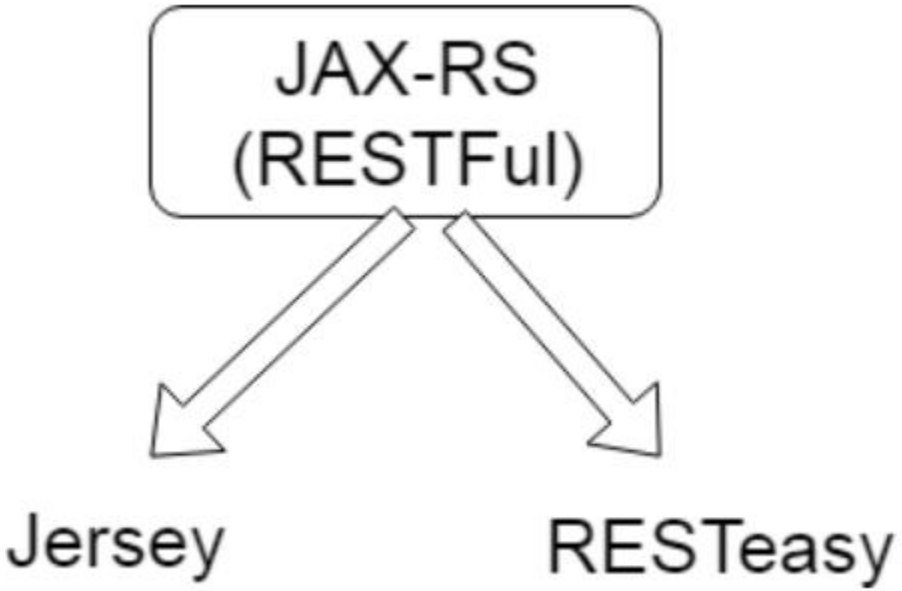
- " Big" web service basato sul protocollo SOAP (Simple Object Access Protocol è un protocollo di trasporto per l'invio e la ricezione di richieste e risposte in formato XML)
- "RESTful" web service (isipirato ai principi architetturali del web, si concentra sulla descrizione delle risorse, sul modo di individuarle nel web)

JAX-RS

Set di API Java che fornisce support nella creazione di API REST .

JAX-RS 2.0 e JAX-RS 2.1 forniscono un’API JAVA per servizi Web RESTFull tramite protocollo HTTP

Le implementazioni più note sono : Jersey e RESTEasy



Identificazione delle risorse

Per risorsa si intende un qualsiasi elemento oggetto di elaborazione. Per fare qualche esempio : libro, utente

Ciascuna risorsa viene identificata univocamente (concetto di URI).

Esempi di URI:

- http://www.myapp.com/utente/123
- http://www.myapp.com/ordini/2011

Risorse autodescrittive

I servizi Rest non pongono nessun limite alla rappresentazione delle risorse

Il tipo di rappresentazione inviato dal WebService al client è indicato nella risposta HTTP tramite un tipo MIME.

Un client a sua volta ha la possibilità di richiedere una risorsa in uno specifico formato sfruttando l'attributo *Accept*

Comunicazione senza stato

Il principio della **comunicazione stateless** è ben noto a chi lavora con il Web. Questa è infatti una delle caratteristiche principali del protocollo HTTP, cioè ciascuna richiesta non ha alcuna relazione con le richieste precedenti e successive. Lo stesso principio si applica ad un Web Service **RESTful**, cioè le interazioni tra client e server devono essere senza stato.

È importante sottolineare che sebbene REST preveda la **comunicazione stateless**, non vuol dire che un'applicazione non deve avere stato. La responsabilità della gestione dello stato dell'applicazione non deve essere conferita al server, ma rientra nei compiti del client.

REST Service (Negozio on line)

Iniziamo a individuare le risorse : non tutte le risorse devono essere rese accessibili ma solo quelle che sono necessarie, nel nostro caso :

- Selezionare articoli
- Effettuare ordini
- Elenco fatture

Proseguiamo con la definizione degli URI che identificano le risorse

Ad esempio, nel nostro caso un ordine con identificatore *123* può essere individuato da un URI della forma *http://www.mionegozio.com/ordini/123*

http://www.mionegozio.com/ordini/?id=123.

WebService

Un documento WSDL definisce un Web service. All'interno del documento esistono quattro elementi principali:

- **<type>** definizione del tipo di dati utilizzati
- **<message>** definizione di uno dei messaggi impiegati dal web service per comunicare con l'applicazione client
- **<portType>** definisce una "porta" e le operazioni che possono essere eseguite dal web service. Definisce inoltre i messaggi coinvolti nelle operazioni elencate
- **<binding>** definisce il formato del messaggio ed i dettagli di protocollo per ogni porta

```
<message name="get_userRequest">
        <part name="id" type="xs:integer" />
</message>
 <message name="get_userResponse">
        <part name="utente" type="tns:utente" />
</message>
```

```
<portType name="gestioneUtentiType">
<operation name="getUserById">
<input message="tns:get_userRequest" />
<output message="tns:get_userResponse" />
</operation>
<operation name="saveUser">
<input message="tns:save_userRequest" />
</operation>
</portType>
```

WebService

Il punto di partenza per lo sviluppo di un servizio Web JAX-WS è una classe Java annotata con l'annotazione javax.jws.WebService. L'annotazione @WebService

Un'interfaccia dell'endpoint del servizio è un'interfaccia o una classe Java che dichiara i metodi che un client può invocare sul servizio.

I passaggi di base per la creazione di un servizio Web e un client sono i seguenti:

- Codificare la classe di implementazione.
- Compilare la classe di implementazione.
- Comprimere i file in un file WAR.
- Distribuire il file WAR.
- Codificare la classe client.
- Utilizzare un'attività Ant wsimport per generare e compilare gli artefatti del servizio Web necessari per connettersi al servizio.
- Compilare la classe client.
- Esegui il client.

RESTFull Webservice

JAX-RS fornisce la funzionalità per i servizi Web RESTful (Representational State Transfer).

Non richiedono messaggi XML o definizioni API-servizio WSDL.

I principi che rendono il web adatto alla realizzazione di webservice rest sono :

- Identificazione delle risorse:
- Utilizzo esplicito dei metodi HTTP
- Risorse autodescrittive
- Comunicazione senza stato

Experience.
Create.
Inspire.

Gianluca Avizzano

IBM Client Innovation Center
Italy