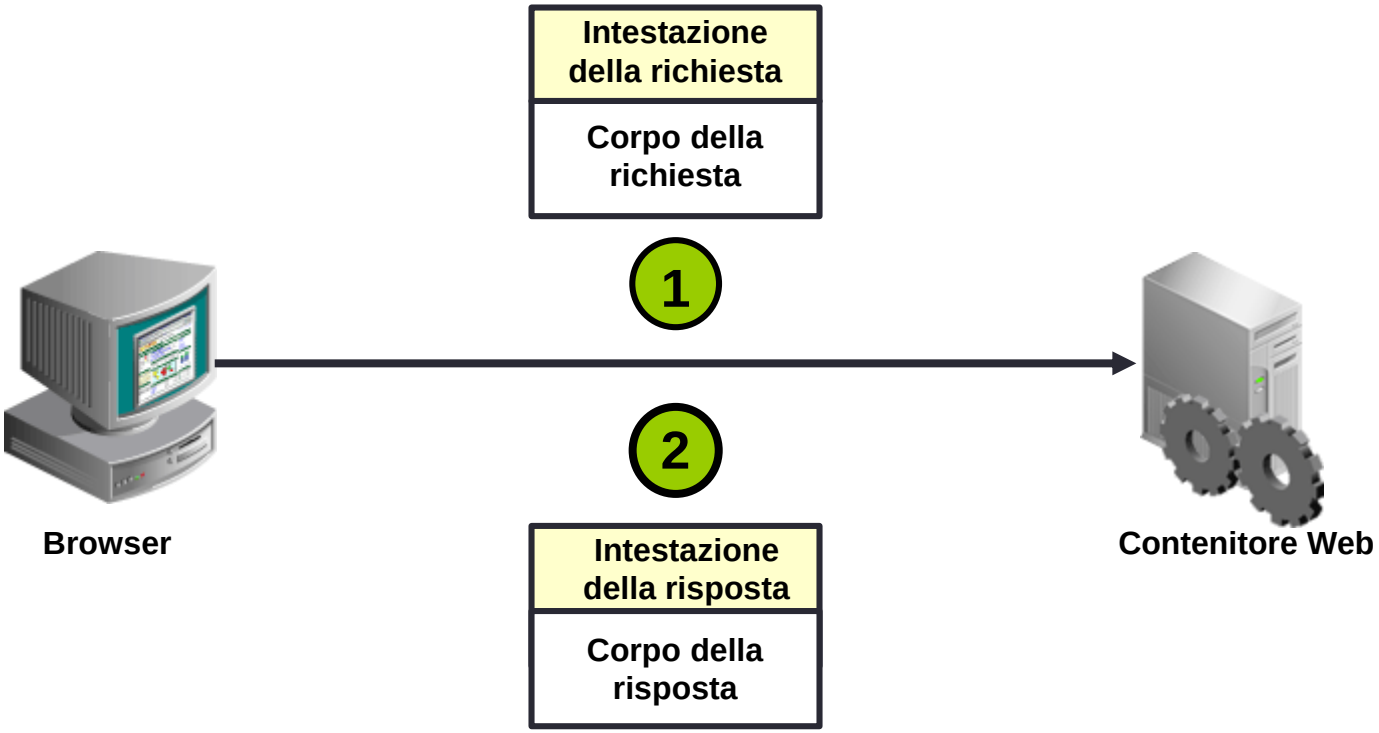
# MODELLO BASATO SU COMPONENTI WEB

# Modello di richiesta-risposta HTTP



# Richieste GET e POST

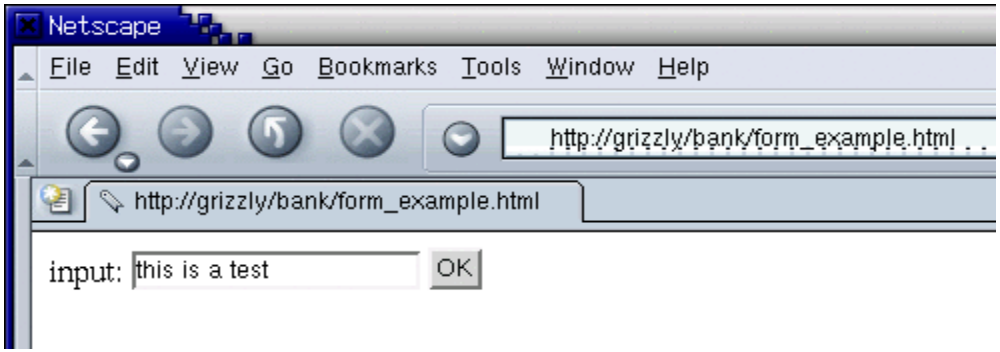
| | Richiesta GET | Richiesta POST |
|---|---|---|
| Tipo di utilizzo | Predefinito | Invio form |
| Metodo di invio dei dati di form | • Inviata con l'URI<br>• Con limite di dimensione | • Inviata nel corpo della richiesta<br>• Senza limiti di dimensione |
| Visualizzazione dei dati di form | Il browser viene visualizzato nell'area URI. | Il browser non viene generalmente visualizzato con l'URI. |

# Dati di form

- Snippet HTML:

```
<FORM ACTION='form_test' METHOD='POST'>
<INPUT NAME='input1' SIZE='20'/>
<INPUT TYPE='SUBMIT' VALUE='OK'/>
</FORM>
```

- Form del browser:




- Richiesta del browser:

```
POST /bank/form_test HTTP/1.1
... request headers...
input1=this+is+a+test
```

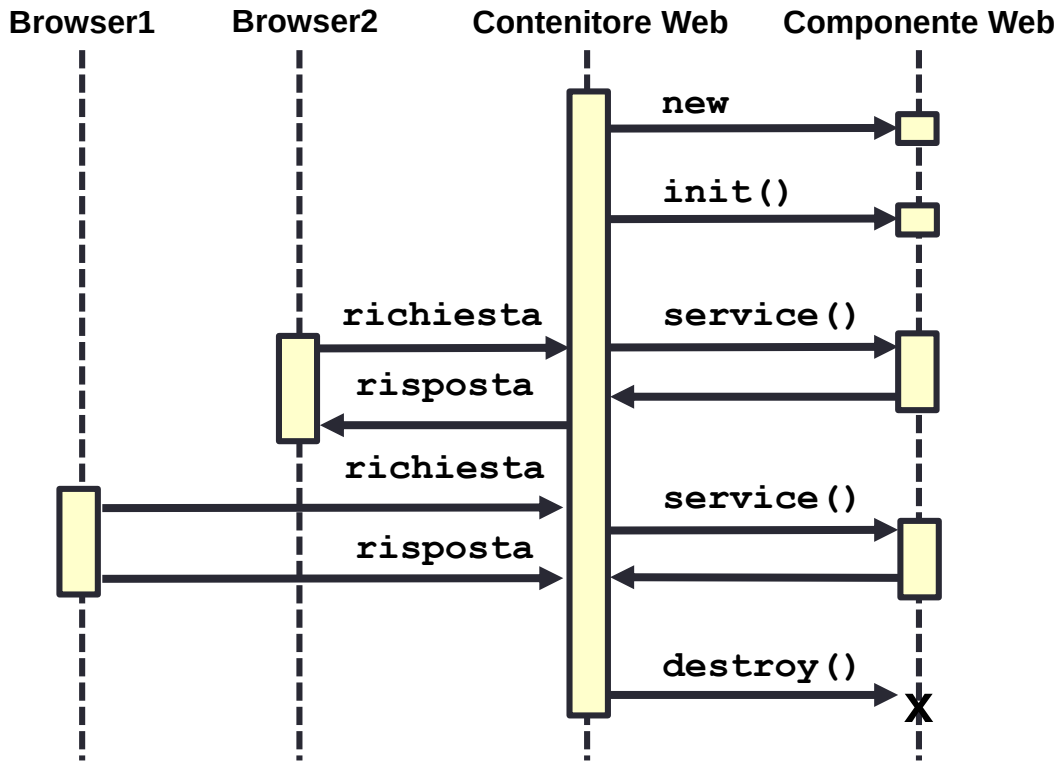
# Intestazione del tipo di contenuto e della risposta

- La risposta del server include un'intestazione `Content-Type` contenente valori di tipo MIME che includono, tra gli altri, i valori seguenti:
  - `text/html`
  - `text/xml`
  - `image/jpeg`
- Esempi di ulteriori intestazioni della risposta includono:
  - `Content-Encoding`
  - `Content-Length`
  - `Cache-Control`

# Confronto tra servlet e componenti JSP

| | Servlet | Componenti JSP |
|---|---|---|
| Descrizione | Componenti Web creati nel linguaggio di programmazione Java | Contenuto di presentazione con elementi di programmazione incorporati |
| Caratteristiche | Estendono le classi di base generiche nell'API, generalmente l'interfaccia `HttpServlet` | • Possono essere migliorate con tag personalizzati<br>• Vengono convertiti in servlet dal contenitore Web |
| Creati o gestiti da | Sviluppatori | Autori di contenuto |

# Ciclo di vita di un componente Web



# Il metodo `service`

- Il contenitore Web chiama il metodo `service` una volta per ogni richiesta in entrata. Il metodo `service` quindi generalmente completa le operazioni seguenti:
  - Convalida tutti i dati dei form
  - Aggiorna il modello di dati dell'applicazione
  - Raccoglie i dati dal modello da visualizzare
  - Visualizza i dati in formato HTML *oppure passa la richiesta e i dati a un altro componente che dovrà visualizzarli*

# Esempio di servlet

```
1    package com.example;
2
3    import java.io.*;
4    import java.util.Date;
5    import javax.servlet.annotation.WebServlet;
6    import javax.servlet.http.*;
7
8    @WebServlet("/hello")
9    public class HelloServlet extends HttpServlet {
10
11        @Override
12        protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws IOException {
13            response.setContentType("text/html");
14            PrintWriter out = response.getWriter();
15            out.println("<html><head/><body>");
16            out.println("<h1>Hello, World!</h1>");
17            out.println("The date is:" + new Date());
18            out.println("</body></html>");
19            out.close();
20        }
21   }
```

# Esempio di componente JSP

- Il codice seguente genera un output simile a quello generato dall'esempio di servlet precedente:

```
1 <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
2 <html>
3       <head/>
4       <body>
5               <h1>Hello, World!</h1>
6               <jsp:useBean id="date" class="java.util.Date" />
7       The date is: <fmt:formatDate value="${date}" type="both" />
8       </body>
9 </html>
```

# Collaborazione tra servlet e componenti JSP

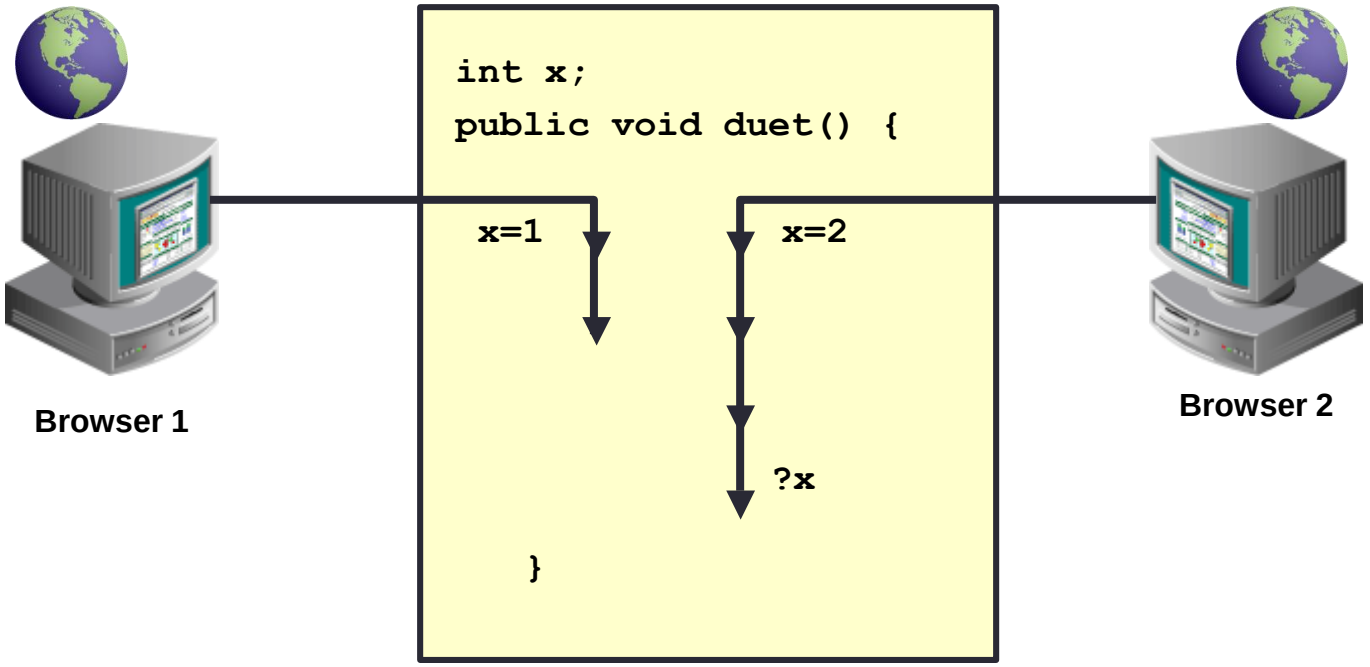
- Le applicazioni Web di ultima generazione utilizzano insieme servlet e componenti JSP. Le funzionalità sono simili ma vengono espresse diversamente.

| | Servlet | Componente JSP |
|---|---|---|
| Tipo di operazione | • Elaborare i dati di form<br>• Eseguire calcoli<br>• Raccogliere dati per il rendering | Generare presentazioni, in particolare HTML |
| Ruolo | Gestire richieste, eseguire calcoli, trasferire il controllo ai componenti JSP | Visualizzare una risposta alla richiesta iniziale |

# Comportamento di runtime dei servlet e dei componenti JSP

- I componenti JSP vengono convertiti in servlet, pertanto componenti JSP e servlet condividono i comportamenti di runtime:
  - Gestione del ciclo di vita e del contenitore
  - Servizi contenitore e API
  - Accesso alla sessione client
- Entrambi possono essere immessi contemporaneamente in più thread e devono essere implementati di conseguenza.

# Modello thread basato su componenti Web



# Implicazioni per lo sviluppatore

- Poiché il contenitore Web immette il metodo `service` in più thread concorrenti per consentire il supporto di più richieste simultanee, lo sviluppatore deve garantire che i componenti
Web siano basati su thread sicuri mediante l'applicazione dei principi seguenti:
  - Utilizzare le variabili di istanza con cautela.
  - Utilizzare le variabili di classe con *molta cautela.*
  - Consentire l'accesso alle risorse esterne con cautela
  - Utilizzare i costrutti di sincronizzazione per indicare le sezioni critiche:

```
synchronized (this) {
// This section is only entered by one thread at a time
}
```

  - Utilizzare il package `java.util.concurrent` anziché i costrutti di sincronizzazione per creare codice basato su thread sicuri.

# Radice del contesto Web e mapping di alias

- I servlet e i componenti JSP vengono inseriti in package in un'applicazione Web.
  - È incluso contenuto statico, come HTML o immagini.
  - Il formato dell'URI di un'applicazione Web è il seguente:
    `http://server:port/context_root/resource`
  - La radice di contesto viene mappata a un'applicazione Web.
  - Un alias viene mappato da un URL a un componente JSP o a un servlet.
    `http://www.mybank.com/bank/main`
- La risorsa `main` potrebbe essere specificata:
  - Nel descrittore di distribuzione mediante i pattern di un URL
  - Nella classe di servlet utilizzando un'annotazione per specificare i pattern di un URL

# Gestione delle sessioni

- HTTP è un protocollo senza stato. Lo stato basato su conversazione può essere memorizzato nel browser o nel server:

| | Browser | Server |
|---|---|---|
| Descrizione | Semplice e non consuma risorse del server | Deve trasportare un ID di sessione tra il browser e il server |
| Disponibilità di memoria | Limitata | Meno limitata |

- Il modello Java EE offre un semplice meccanismo per la memorizzazione nel server dello stato basato su conversazione.

# Problemi con lo sviluppo del livello Web

- Le complessità dello sviluppo dell'interfaccia utente basata su Web includono:
  - Gestire la sequenza richiesta-risposta senza stato di HTTP
  - Evitare che vengano creati numeri elevati di servlet e componenti JSP che gestiscono tipi diversi di richieste
  - Utilizzare componenti JSP e servlet in modo efficace

**Grazie**