**Servlet e il loro contesto in Java EE**

Le **Servlet** sono componenti lato server scritte in Java, progettate per gestire richieste da parte dei client (ad esempio un browser) e generare risposte dinamiche. Esse fanno parte del mondo **Java EE (Jakarta EE)** e vengono usate soprattutto per creare applicazioni web scalabili. Sono alla base del funzionamento di framework più complessi, come Spring o Struts.

**Struttura e funzionamento di una Servlet**

Una servlet è una classe Java che estende la classe HttpServlet. Il ciclo di vita di una servlet è gestito dal container (ad esempio Tomcat, Jetty), che si occupa di:

1. **Caricare** la servlet quando viene richiesta la prima volta (o al deployment).
2. **Inizializzarla** invocando il metodo init().
3. **Gestire richieste e risposte** tramite i metodi doGet() o doPost().
4. **Distruggerla**, invocando il metodo destroy() quando non è più necessaria.

**Metodi principali**

Le servlet sovrascrivono i metodi chiave della classe HttpServlet:

1. **doGet(HttpServletRequest req, HttpServletResponse resp)**: gestisce richieste HTTP di tipo GET (es. quando un utente visita un URL).
2. **doPost(HttpServletRequest req, HttpServletResponse resp)**: gestisce richieste HTTP di tipo POST (es. quando inviamo dati via form).
3. **init()**: viene eseguito una sola volta, quando la servlet viene caricata.
4. **destroy()**: usato per liberare risorse quando la servlet viene rimossa dal container.

Esempio di base:

|  |
| --- |
| **import** javax.servlet.\*;  **import** javax.servlet.http.\*;  **import** java.io.IOException;  ​  **public** **class** MiaServlet **extends** HttpServlet {     @Override  **public** void doGet(HttpServletRequest req, HttpServletResponse resp) **throws** IOException {         resp.setContentType("text/html");         resp.getWriter().println("<h1>Ciao dal server!</h1>");    }  }  ​ |

**Configurazione delle servlet con il file web.xml**

Per fare in modo che il container riconosca una servlet, è necessario configurarla nel file web.xml, noto anche come *deployment descriptor*. Esempio:

|  |
| --- |
| <web-app>     <servlet>         <servlet-name>MiaServlet</servlet-name>         <servlet-**class**>MiaServlet</servlet-**class**>     </servlet>     <servlet-mapping>         <servlet-name>MiaServlet</servlet-name>         <url-pattern>/ciao</url-pattern>     </servlet-mapping>  </web-app>  ​ |

Questo permette di accedere alla servlet tramite l'URL: <http://localhost:8080/nome-applicazione/ciao>.

**Differenza tra GET e POST**

* **GET**: i parametri sono inclusi nell’URL e visibili (es. ?param=valore). È adatto per richieste di lettura o quando non ci sono dati sensibili.
* **POST**: i parametri sono inclusi nel corpo della richiesta. È più sicuro e viene usato per inviare dati (es. form).

Esempio di utilizzo dei parametri in GET/POST:

|  |
| --- |
| @Override  **protected** void doGet(HttpServletRequest req, HttpServletResponse resp) **throws** IOException {     String nome = req.getParameter("nome");     resp.getWriter().println("Ciao " + nome);  }  ​ |

**Reindirizzamenti: Forward e Redirect**

Quando una servlet vuole inviare l'utente verso un'altra pagina o servlet:

* **Forward**: trasferisce la richiesta lato server. Non cambia l'URL visibile al client. Usa

|  |
| --- |
| RequestDispatcher dispatcher = req.getRequestDispatcher("pagina.jsp");  dispatcher.forward(req, resp);  ​ |

* **Redirect**: invia una nuova richiesta lato client, cambiando l’URL visibile. Usa:

|  |
| --- |
| resp.sendRedirect("http://www.esempio.com");  ​ |

**Sessioni e Cookie**

Le servlet supportano la gestione di **sessioni** per mantenere informazioni tra richieste successive.

1. **Sessione tramite HttpSession:**
   * Si utilizza per salvare dati associati all'utente, validi durante la sessione.  
     Esempio:

|  |
| --- |
| HttpSession session = req.getSession();  session.setAttribute("utente", "Mario Rossi");  String utente = (String) session.getAttribute("utente");  ​ |

1. **Cookie:**

* Si utilizzano per salvare dati lato client.  
  Esempio di creazione e lettura di un cookie:

|  |
| --- |
| Cookie cookie = **new** Cookie("preferenze", "tema-scuro");  resp.addCookie(cookie);  ​  // Leggere i cookie  Cookie[] cookies = req.getCookies();  **for** (Cookie c : cookies) {  **if** ("preferenze".equals(c.getName())) {         String valore = c.getValue();    }  }  ​ |

**Struttura di una Web Application**

Una web application segue una struttura predefinita quando viene distribuita come file **WAR** (*Web Application Archive*). La struttura è la seguente:

|  |
| --- |
| nome-applicazione/  │  ├── WEB-INF/  │   ├── web.xml  │   ├── classes/ (dove risiedono le servlet compilate)  │   └── lib/ (librerie esterne, es. file JAR)  ├── index.html (file pubblico)  └── pagina.jsp (altro file pubblico)  ​ |

**Funzionalità aggiuntive**

1. **welcome-file-list nel file web.xml:** Permette di definire la pagina predefinita.

|  |
| --- |
| <welcome-file-list>     <welcome-file>index.html</welcome-file>  </welcome-file-list>  ​ |

1. **Gestione avanzata tramite Listener e Filter:**

* **Filter**: intercettano e manipolano richieste o risposte.

|  |
| --- |
| **public** **class** MioFiltro **implements** Filter {  **public** void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)  **throws** IOException, ServletException {         // Modifica la richiesta o risposta         chain.doFilter(req, resp); // Continua il ciclo    }  }  ​ |

* **Listener**: ascoltano eventi legati al ciclo di vita dell'applicazione (es. creazione sessioni).

**Domande sulle Servlet**

1. **Cos'è una servlet e qual è il suo scopo principale in un'applicazione Java?**
   * Qual è il ruolo delle servlet rispetto al client e al server?
2. **Come si configura una servlet in un progetto?**
   * Qual è il ruolo del file web.xml?
   * Come configuri una servlet usando annotazioni?
3. **Qual è il ciclo di vita di una servlet?**
   * Quali sono i metodi principali che vengono chiamati (es. init(), service(), doGet(), doPost(), destroy())?
   * Cosa avviene in ogni fase del ciclo di vita?
4. **Qual è la differenza tra doGet() e doPost()?**
   * Quando conviene usare uno rispetto all’altro?
   * Come gestisci parametri passati tramite URL o body?
5. **Come si inviano dati dalla servlet al client?**
   * Qual è il ruolo dell’oggetto HttpServletResponse?
   * Come invii una risposta in formato HTML o JSON?
6. **Come si ricevono dati da una richiesta HTTP?**
   * Qual è il ruolo di HttpServletRequest?
   * Come recuperi i parametri di una richiesta?
7. **Cos’è il deployment di una servlet?**
   * Come viene gestito il deployment su un application server?
   * Cos’è un contesto servlet e come viene configurato?
8. **Come si gestiscono gli stati nelle servlet?**
   * Qual è la differenza tra oggetti HttpSession, cookies e URL rewriting?
   * Quando conviene usare uno rispetto all’altro?
9. **Qual è la differenza tra ServletContext e HttpSession?**
   * Quando utilizzi uno invece dell’altro?
   * Come condividi dati tra più servlet?
10. **Come si inoltra una richiesta da una servlet a un’altra risorsa?**
    * Qual è la differenza tra RequestDispatcher.forward() e HttpServletResponse.sendRedirect()?
    * Quando è opportuno usarli?
11. **Quali sono le limitazioni delle servlet?**
    * Perché spesso vengono integrate con JSP o framework come Spring?
12. **Quali sono le best practice per scrivere servlet?**
    * Come gestire eccezioni in modo efficace?
    * Come assicurarti che una servlet sia thread-safe?
13. **Come implementi un semplice esempio CRUD con servlet?**
    * Quali metodi del ciclo di vita usi per ogni operazione (create, read, update, delete)?
14. **Cos’è il metodo service()?**
    * Perché viene sovrascritto raramente rispetto a doGet() o doPost()?
15. **Cos’è un filtro (filter) e come si integra con una servlet?**
    * Quali sono gli usi pratici dei filtri?
    * Come si configura un filtro nel file web.xml o con annotazioni?
16. **Come puoi servire file statici come immagini o CSS con una servlet?**
    * Qual è il ruolo di un RequestDispatcher in questo caso?
17. **Come si testano le servlet?**
    * Quali strumenti o framework puoi usare per testare le richieste HTTP e le risposte generate dalle servlet?