**Ciclo di Vita di un Oggetto in un'Applicazione Web**

Nella programmazione di applicazioni web, è cruciale definire il ciclo di vita degli oggetti.  
  
Questo significa decidere quando un oggetto verrà creato, quanto a lungo "vivrà" in memoria e quando verrà rimosso. In fase di progettazione dell'architettura, dobbiamo stabilire come e quando ciascun oggetto dovrà essere gestito, soprattutto perché in un contesto web, gli oggetti non sempre hanno una durata breve.

Quando si risponde a una richiesta del browser, spesso si creano oggetti per generare una pagina web o per elaborare i dati dell'utente. Tuttavia, alcuni di questi oggetti potrebbero rimanere allocati in memoria anche dopo che la richiesta è stata servita, e questo può avere implicazioni sulle prestazioni dell’applicazione. Una gestione attenta del ciclo di vita degli oggetti evita sprechi di risorse e assicura che l’applicazione resti reattiva ed efficiente.

**HTTP e Come Funziona una Richiesta al Server**

Il protocollo HTTP (HyperText Transfer Protocol) è lo standard di comunicazione utilizzato per trasferire dati tra il client (browser) e il server. Quando digitiamo un URL nella barra degli indirizzi, il browser invia una richiesta HTTP al server che ospita la risorsa richiesta. La richiesta è strutturata in modo specifico: la prima riga contiene un verbo HTTP (ad esempio, GET o POST), l’URL della risorsa e il protocollo (es. HTTP/1.1).

L'URL (Uniform Resource Locator) è univoco e indirizza una specifica risorsa; non è possibile che un URL punti a due risorse diverse. Una stessa risorsa, tuttavia, può essere raggiunta attraverso più URL. Questo sistema assicura che ogni risorsa sia sempre identificabile e accessibile in modo chiaro e coerente.

**Tipologie di Risorse: JSP e Servlet**

Le applicazioni web Java prevedono due principali tipologie di risorse:

* **JSP (Java Server Pages)**: sono pagine HTML con codice Java integrato, utilizzate per generare contenuti dinamici come le informazioni personalizzate di un utente.
* **Servlet**: sono componenti Java che risiedono sul server e gestiscono le richieste HTTP in modo programmatico, offrendo un controllo completo su come la richiesta viene processata e risolta.

**HTTP e le Servlet: Gestione dei Verbi**

HTTP ha una serie di verbi (GET, POST, PUT, DELETE, ecc.), ciascuno con uno scopo specifico. Le servlet di Java sono progettate per sfruttare questi verbi, rispondendo a ognuno di essi attraverso metodi specifici: ad esempio, doGet() gestisce richieste GET, doPost() gestisce richieste POST, e così via. Questo approccio permette di gestire diversi tipi di richieste HTTP nello stesso componente Java, semplificando la struttura e consentendo di utilizzare un’unica servlet per più funzionalità.

Ogni metodo rappresenta un "punto d'ingresso" nel componente, rispondendo a una specifica richiesta del client. In pratica, questo consente di realizzare un unico componente Java che può rispondere a diverse esigenze e interagire con lo stesso URL a seconda delle operazioni richieste dall'utente.

**Application Server e Configurazione della Porta**

Un application server è un software che gestisce e serve applicazioni. Mentre i web server (come Apache) si occupano della gestione delle pagine web statiche, un application server (come Tomcat o WildFly) è in grado di elaborare richieste che richiedono una logica applicativa complessa.

Quando configuriamo un application server, è importante specificare la **porta applicativa** su cui dovrà ricevere le richieste. Questa porta è come un “identificatore” che dice al server quale programma deve gestire la richiesta in arrivo. Ad esempio, quando una richiesta arriva da un client, la porta determina quale application server dovrà occuparsene.

**Differenza tra JRE e JDK nel Runtime**

Per far girare il server e garantire piena compatibilità con il codice Java, è preferibile configurare il runtime usando il **JDK** (Java Development Kit) invece della **JRE** (Java Runtime Environment). Il JDK include tutto ciò che offre la JRE ma aggiunge strumenti per sviluppatori, rendendolo più adatto per lo sviluppo e l’esecuzione di applicazioni Java complesse.

**La Cartella WEB-INF e la Configurazione dell’Applicazione**

Nelle applicazioni web Java, la cartella WEB-INF è uno spazio riservato e protetto in cui è possibile inserire file di configurazione e risorse non accessibili direttamente tramite URL. Qui viene spesso inserito un file XML di configurazione, web.xml, che definisce impostazioni specifiche dell’applicazione come le servlet, i filtri e altre risorse. La cartella WEB-INF è essenziale per garantire la sicurezza e organizzazione dell’applicazione.

**Tipologie di File: JAR, WAR, EAR**

Nell’ecosistema Java, esistono diverse tipologie di file di archivio utilizzate per distribuire applicazioni e componenti:

* **JAR (Java ARchive)**: contiene librerie e componenti Java.
* **WAR (Web Application Archive)**: contiene applicazioni web pronte per essere distribuite su un application server.
* **EAR (Enterprise Application Archive)**: contiene applicazioni aziendali complesse e permette di gestire applicazioni con moduli multipli.

**Differenza tra HTML e JSP**

Una pagina **HTML** è statica, significa che i contenuti sono definiti una volta e non cambiano fino a quando non viene modificata manualmente. Una **JSP (Java Server Page)**, invece, è un mix tra HTML e Java, che permette di generare contenuti dinamici. Ad esempio, in una JSP è possibile definire variabili, creare oggetti e gestire i dati dell'utente in modo interattivo.

**Inserire Codice Java all'interno di JSP**

In una pagina JSP, è possibile inserire codice Java usando una sintassi specifica. Ad esempio:

<%="Ciao da JSP" %>

​ Questa linea stampa “Ciao da JSP” sulla pagina, in modo simile a un System.out.println. L’integrazione di Java all’interno di una JSP permette di rendere la pagina dinamica e interattiva, rispondendo in tempo reale ai dati dell’utente.