Appunti Parte due 21-01

Context

La classe `Context` rappresenta un pattern di **factory centralizzata** che utilizza membri statici per gestire le dipendenze dell'applicazione in modo globalmente accessibile. Questo approccio crea un punto di accesso unico da cui tutte le parti del sistema possono ottenere le istanze di cui hanno bisogno, come la connessione al database, senza doverle passare esplicitamente attraverso catene di chiamate.

Static initialization block

Il blocco static è un **blocco di inizializzazione statico**, una struttura che viene eseguita automaticamente dal classloader quando la classe viene caricata in memoria, prima di qualsiasi altro codice che usi quella classe. Questo blocco ha la garanzia di eseguire esattamente una volta durante l'intero ciclo di vita dell'applicazione, prima del metodo main e prima della creazione di qualunque istanza della classe. Il suo scopo principale è inizializzare le variabili statiche quando tale inizializzazione richiede logica complessa che non può essere espressa in una singola assegnazione inline.

Membri static e contesto globale

La lista `dependencies` dichiarata come `static List<Object>` rappresenta una variabile di classe condivisa da tutte le parti del programma. Le variabili statiche esistono a livello di classe anziché a livello di istanza, il che significa che ne esiste una sola copia in memoria indipendentemente dal numero di istanze della classe Context create. Questa caratteristica permette di usare Context come un registro centralizzato dove memorizzare oggetti che devono essere accessibili globalmente, come connessioni, configurazioni o servizi.

Autowiring

Autowiring è un meccanismo automatico fornito dal framework Spring per implementare la dependency injection senza dover configurare manualmente ogni dipendenza. Mentre la dependency injection è il pattern architetturale generale che prevede di fornire a un oggetto le sue dipendenze dall'esterno anziché fargliele creare internamente, l'autowiring è la specifica tecnica con cui Spring risolve automaticamente quali oggetti iniettare e dove.

Il concetto fondamentale è che, quando si annota una classe o un suo componente con `@Autowired`, si delega al container di Spring (l'ApplicationContext) il compito di trovare il bean appropriato nel suo registro e iniettarlo automaticamente. Spring utilizza la reflection per analizzare le dipendenze richieste e cerca nel suo contesto un bean compatibile per tipo, collegandolo automaticamente senza che lo sviluppatore debba scrivere codice di configurazione esplicito.

La differenza rispetto al pattern della factory statica vista in precedenza è che con autowiring non si chiama esplicitamente nessun metodo per ottenere le dipendenze: il framework le inietta automaticamente dove necessario semplicemente dichiarando l'annotazione. Nel codice Context mostrato prima, si creava manualmente la connessione e la si memorizzava in una variabile statica; con autowiring in Spring, si annotarebbe semplicemente il campo Connection con `@Autowired` e Spring gestirebbe automaticamente l'intera catena di creazione e iniezione. Questo approccio inverte il controllo (Inversion of Control) spostando la responsabilità della gestione delle dipendenze dall'applicazione al framework.