**ENTERPRISE BEAN**

*In questo capitolo sono illustrate le scelte effettuate per la realizzazione di una versione dell’applicazione basata su Enterprise Bean.*

**Introduzione**

Nell’attività di sviluppo di un’applicazione, oltre all’implementazione della logica di business, è importante prestare attenzione ad aspetti come la gestione delle transazioni, concorrenza, scalabilità, affidabilità, sicurezza, portabilità e riusabilità. Per potersi concentrare sullo sviluppo della logica di business senza tralasciare le qualità appena elencate, è possibile usare gli **Enterprise Bean**. Per tali motivi, nell’ambito del lavoro di tesi, è stata realizzata una versione dell’applicazione che fa uso degli Enterprise Bean.

**Enteprise Bean**

Gli Enterprise Bean sono componenti lato server, scritti in Java, che incapsulano la logica di business di una applicazione ed implementano lo standard Enterprise Java Bean (**EJB**). Per logica di business si intende il codice realizzato per soddisfare gli scopi per i quali è stata creata l’applicazione. Gli EJB devono essere rilasciati all’interno di un EJB container che fornisce a tali componenti un certo numero di servizi come, per esempio, gestione della sicurezza, delle transazioni e supporto per i web-services.

Esistono diverse tipologie di Enterprise Bean:

* **Entity Bean**: il loro scopo è fornire le funzionalità di persistenza dei dati, inglobando oggetti lato server che memorizzano dati.
* **Session Bean**: incapsulano la logica applicativa e sono un’interfaccia tra i client e i servizi offerti dai componenti disponibili sul server.
* **Message Driven Bean**: permettono la ricezione e l’elaborazione di messaggi asincroni.

In particolare un Session Bean può essere di due tipi:

* **Stateful Session Bean**, in cui ciascuna istanza rappresenta lo stato di una sessione con un singolo client per tutta la durata della sessione.
* **Stateless Session Bean**, dove le istanze non mantengono informazioni sulle sessioni con i loro client.

**Benefici nell’uso degli Enterprise Bean**

Per vari motivi, gli Enterprise Bean semplificano lo sviluppo di grandi applicazioni distribuite. Il contenitore EJB, come già accennato, deve fornire dei servizi a livello di sistema per gli Enterprise Bean e lo sviluppatore si può quindi concentrare sulla risoluzione dei problemi di business.

In secondo luogo, poiché la logica di business è contenuta negli Enterprise Bean, gli sviluppatori dei client si possono focalizzare sugli aspetti di presentazione. Il client, libero dal dover implementare la logica di business oppure i metodi per accedere al database, sarà semplice e potrà andare in esecuzione anche su dispositivi di modesta potenza di elaborazione.

In terzo luogo, siccome gli enterprise bean sono componenti portabili, un’applicazione può essere assemblata a partire da bean esistenti. A patto di usare le API standard, queste applicazioni possono essere eseguite su qualsiasi server compatibile Java EE.

**Quando usare gli Enterprise Beans**

Si dovrebbe considerare l'utilizzo di Enterprise Bean se l'applicazione richiede uno dei seguenti requisiti.

* **L’applicazione deve essere scalabile**. Per soddisfare un numero crescente di utenti, è possibile distribuire i componenti dell’applicazione su più macchine. Non solo gli Enterprise Bean possono essere in esecuzione su macchine diverse, ma anche la loro posizione rimarrà trasparente ai clients.
* **Le operazioni devono garantire l’integrità dei dati**. Gli Enterprise Bean supportano le transazioni: meccanismi che gestiscono l'accesso simultaneo di oggetti condivisi.
* **L'applicazione può avere una varietà di clients**. Con solo poche righe di codice, client remoti possono facilmente individuare gli Enterprise Bean. Questi client possono essere semplici, numerosi e di varia natura.

**Car2go-ejb**

Sono state create due versioni dell’applicazione basate su Enterprise Bean.

Una prima versione è stata progettata creando:

* Un Session Bean di tipo stateful con nome Application EJB
* Un Session Bean di tipo stateless per ogni tipo di servizio

1. MacchinaServiceEJB, implementa l’interfaccia MacchinaService
2. UtenteServiceEJB, implementa l’interfaccia UtenteService
3. RuoloServiceEJB, implementa l’interfaccia RuoloService
4. PrenotazioneServiceEJB, implementa l’interfaccia PrenotazioneService
5. GeoEJB, implementa l’interfaccia GeoService

* Un Session Bean di tipo stateless per ogni tipo di dato per l’accesso al database

1. MacchinaDAOEJB, implementa l’interfaccia MacchinaDAO
2. UtenteDAOEJB, implementa l’interfaccia UtenteDAO
3. RuoloDAOEJB, implementa l’interfaccia RuoloDAO
4. PrenotazioneDAOEJB, implementa l’interfaccia PrenotazioneDAO

Una seconda versione è stata progettata usando direttamente:

* Un Session Bean di tipo stateless per ogni tipo di servizio

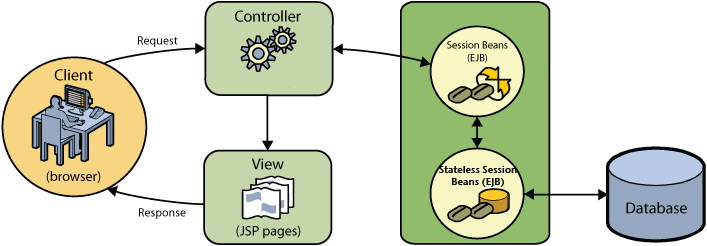
1. MacchinaServiceEJB, implementa l’interfaccia MacchinaService
2. UtenteServiceEJB, implementa l’interfaccia UtenteService
3. RuoloServiceEJB, implementa l’interfaccia RuoloService
4. PrenotazioneServiceEJB, implementa l’interfaccia PrenotazioneService
5. GeoEJB, implementa l’interfaccia GeoService

* Un Session Bean di tipo stateless per ogni tipo di dato per l’accesso al database

1. MacchinaDAOEJB, implementa l’interfaccia MacchinaDAO
2. UtenteDAOEJB, implementa l’interfaccia UtenteDAO
3. RuoloDAOEJB, implementa l’interfaccia RuoloDAO
4. PrenotazioneDAOEJB, implementa l’interfaccia PrenotazioneDAO

In ambedue le versioni dell’applicazione, ciascun Enterprise Bean implementa un’interfaccia remota (che costituisce l’interfaccia fornita) e può utilizzare, mediante l’iniezione delle dipendenze (grazie all’annotazione @EJB), altri Bean (ovvero l’interfaccia richiesta).

L’immagine seguente mostra come sono state strutturate, in linea generale, entrambe le versioni delle due applicazioni.



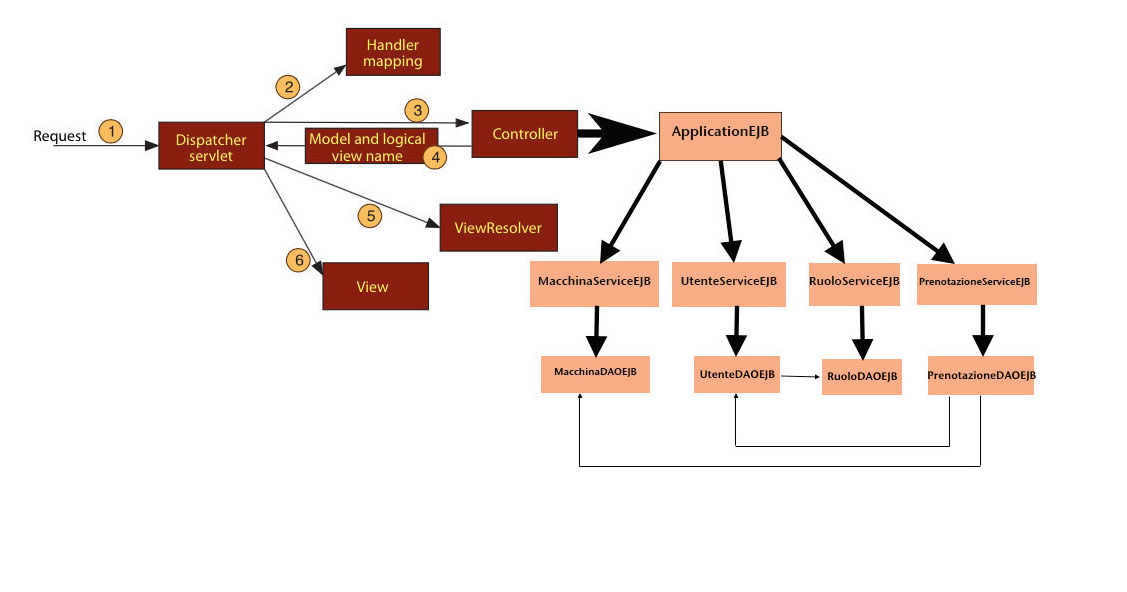
Dai Controller realizzati mediante l’approccio suggerito da Spring MVC, vengono richiamati gli EJB che contengono le funzionalità di business. Il compito di interfacciarsi con il database per recuperare i dati e gestire la persistenza è affidato ad Enterprise Bean di tipo Stateless.

Data una visione d’insieme della struttura delle due applicazioni, è necessario analizzare nel dettaglio come sono state realizzate ambedue le versioni.

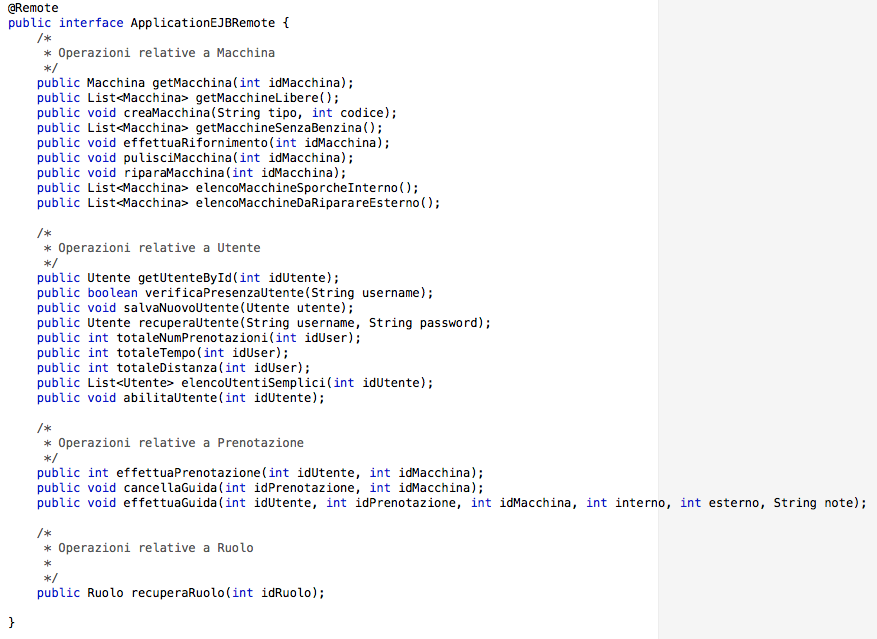
**Prima versione (uso di Stateful Session Bean)**

La prima versione dell’applicazione è caratterizzata dall’uso di tre strati di Enterprise Bean. Al primo livello c’è un Enterprise Bean di tipo Stateful nominato ApplicationEJB. Questo bean ha il compito di ricevere le richieste provenienti dai Controller, gestire lo stato della sessione e utilizzare i bean di tipo Stateless (secondo livello) per poter completare le operazioni di sistema che gli sono state delegate dall’applicazione web. Infine al terzo livello troviamo gli EJB di tipo Stateless per gestire la persistenza dei dati.

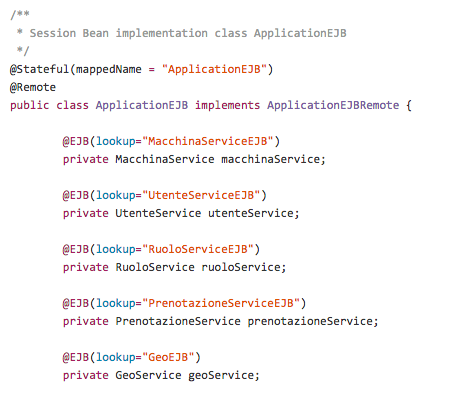
La figura sottostante riassume le relazioni esistenti tra il Controller, l’ApplicationEJB e tutti gli altri Enterprise Bean di tipo Stateless.



L’ApplicationEJB implementa l’interfaccia remota ApplicationEJBRemote.

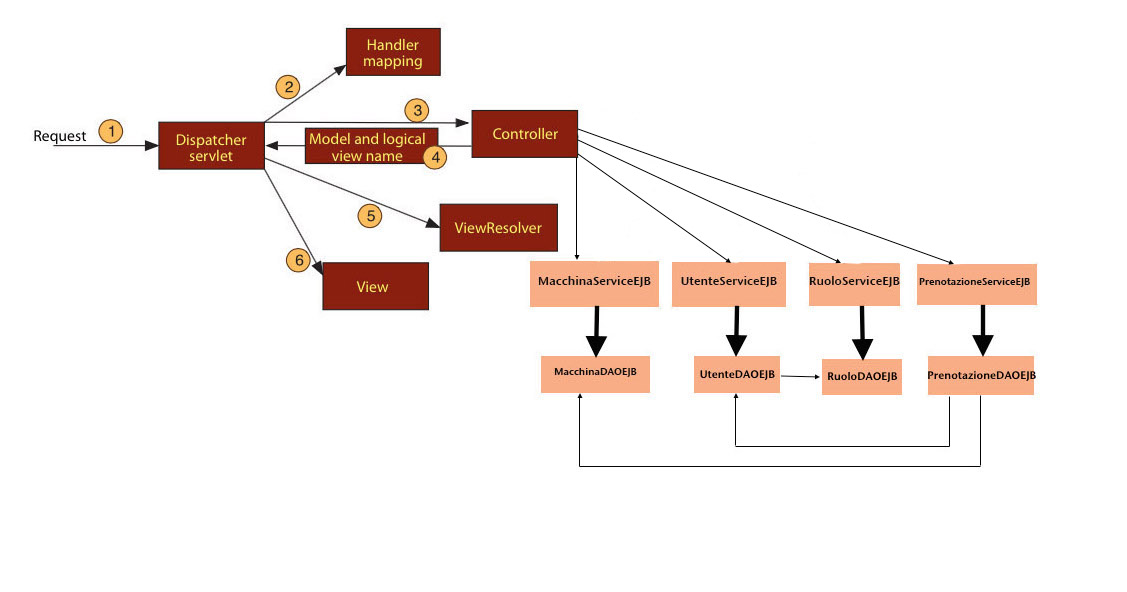


Per utilizzare i bean MacchinaServiceEJB, UtenteServiceEJB, RuoloServiceEJB e PrenotazioneServiceEJB viene sfruttata l’iniezione delle dipendenze.



**Seconda versione (senza uso di Stateful Session Bean)**

Nella seconda versione dell’applicazione non è stato fatto uso del Bean Stateful di nome ApplicationEJB. I Controller dell’applicazione web utilizzano direttamente i Bean Stateless che implementano la logica di business come MacchinaServiceEJB, UtenteServiceEJB, RuoloServiceEJB e PrenotazioneServiceEJB.

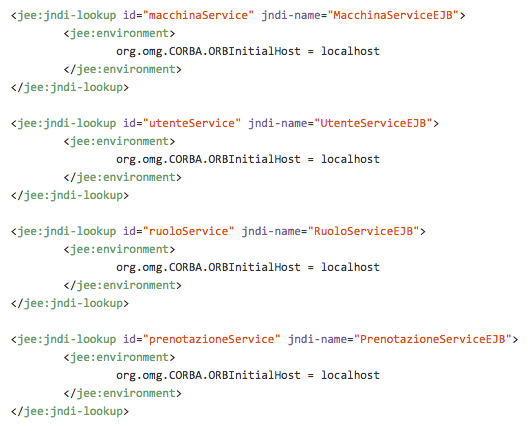


Sarà l’applicazione web a mantenere le informazioni di sessione di un utente.

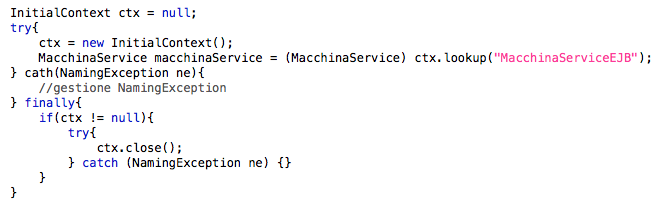
**Integrazione Spring MVC con EJB**

Nei capitoli precedenti è stato ampliamente spiegato come una delle caratteristiche più importanti di Spring Framework sia l’integrabilità. Spring è facilmente integrabile nei diversi livelli di una applicazione, dal frontend al backend, dimostrandosi utilissimo anche nello strato applicativo nell’implementazione della logica di business. L’intenzione di questo paragrafo è quella di evidenziare come sia semplice integrare Spring all’interno di una applicazione distribuita che fa uso di EJB3 e come facilmente i servizi da essi esposti possano essere iniettati nel contesto applicativo ed utilizzati all’interno di un Controller.

Il primo passo da compiere è quello di settare all’interno del file di configurazione di Spring le linee di codice necessarie per recuperare i Session Bean. Per tale scopo è possibile usare il tag <jee:jndi-lookup> con l’attributo jndi-name. Tale tag indica di effettuare una ricerca JNDI andando a rintracciare un bean il cui nome è specificato nell’attributo jndi-name. Nell’immagine seguente è mostrato il frammento del file di configurazione che permette di collegare gli EJB all’interno dell’applicazione web gestita da Spring.



Senza l’utilizzo di Spring, all’interno di un client per ottenere il riferimento ad un EJB, è necessario scrivere delle righe di codice simili alle seguenti.



Dall’analisi del listato precedente si nota come con Spring il codice per l’integrazione sia più semplice.

Per effettuare l’iniezione delle dipendenze di un bean all’interno del controller è sufficiente usare l’annotazione @Autowired.

