MODALITÀ DI LABORATORIO ONLINE

Per le esercitazioni di oggi, troverete il meeting nel canale "Lezioni" del Teams del corso. Il Docente presenterà gli esercizi da svolgere: per ogni esercizio avremo una breve introduzione seguita dallo svolgimento da parte degli studenti. Potete usare la chat per richiedere aiuto da parte del Docente. Usare il microfono solo per comunicare qualche urgenza.

Esercizio 1: Calcolatrice EJB

Scrivere un client Java che invoca degli EJB sul server che implementano un servizio di calcolatrice per tipi `float`. La calcolatrice offre tre metodi, tutti prendono in input due operandi di tipo float e restituiscono un float :

- `add(float,float)`
- `sub(float,float)`
- `multiply(float,float)`

Il server offre inoltre un servizio di counting `count()` che restituisce il numero di operazioni effettuate da tutti i clienti dall'avvio del server. Il client deve offrire da console un interfaccia che permetta di effettuare tutte le operazioni da remoto.

Esercizio 2: Calcolatrice EJB basata su stack

Scrivere un client Java che invoca degli Enterprise EJB sul server che implementano un servizio di calcolatrice basata su stack per tipi float. La calcolatrice offre cinque metodi. Tre di questi metodi rappresentano operazioni aritmetiche che non prendono input e operano direttamente sui dati nello stack: `add()`, `sub()`, `multiply()`. In aggiunta, il server offre il metodo `void push(float)` e `float pop()` per la gestione dello stack. Nota che l'unico metodo che restituisce valori è la `pop()`. Le operazione aritmetiche eseguono implicitamente due `pop()` per prelevare gli operandi e una `push()` per salvare il valore di ritorno.

Esempio di esecuzione del servizio:

- `push(3), push(4), add(), pop()` -> restituisce 7
- `push(3), push(4), multiply(), pop()` -> restituisce 12
- `push(3), push(4), push(1), add(), pop()` -> restituisce 5

- `push(3), push(4), push(1), add(), add(), pop()` -> restituisce 8

Note:

- Si consiglia di testare il servizio prima localmente
- Lo stack è visibile solo all'interno di un'iterazione del client

Avanzato: gestire i meccanismi di attivazione e passivazione.

Esercizio 3: Calcolatrice EJB basata su stack persistente

Estendere il servizio di calcolatrice basata su stack implementata nell'esercizio precedente aggiungendo un livello di persistenza dello stack. In particolare, ad ogni operazione, lo stack deve essere salvato su database. All'interfaccia del servizio verrà aggiunto il metodo `loadStack(String stackName)` e `saveStack(String stackName)` che consentono all'utente di caricare lo stack da database . Se il client non invoca la `loadStack()`, la sessione si avvia con uno stack vuoto. Lo stack puo essere salvato nel formato piu appropriato, anche come stringa.

Al fine di completare l'esercizo, occorre

1. Implementare il metodo `loadStack(String username)`
2. Implementare il metodo `saveStack(String username)`
3. Implementare un metodo `stackList()` che restituisce tutti i nomi degli stack salvati su database.
4. Implementare un interceptor che, alla fine dell'interazione con un client, stampa una lista dei stackName usati da quell'utente in quella sessione.

Per la parte di persistenza da server EJB, fare riferimento alle slides su EJB