Esercitazione di Sistemi Distribuiti e Pervasivi

Suggerimenti di Sviluppo e Progettazione I

Gabriele Civitarese
gabriele.civitarese@unimi.it

EveryWare Lab
Università degli Studi di Milano
-
Docente: Claudio Bettini

Slide tratte da versioni precedenti di Letizia Bertolaja e Dario Freni

Laboratorio

- 4 lezioni dedicate all'assistenza allo svolgimento del progetto
  - Voi potete sfruttare le ore di lezione per sviluppare il progetto
  - Potete nel frattempo chiarire dubbi o chiedere consigli
  - Nel caso in cui abbiate concluso il progetto in anticipo, potete sfruttare l'ultima lezione per discuterlo
- Le lezioni partiranno come sempre alle 8:45 (puntuali!), con la possibilità di rimanere fino alle 11:30
- Prima parte di "lezione frontale"
- È possibile lavorare sul proprio portatile
- L'importante è che lavoriate individualmente
  - Questo non vuol dire che non possiate parlare tra di voi...
  - Due progetti copiati SONO EVIDENTI

Statistiche

Giusto un po' di dati degli anni scorsi...

- Circa il 67% degli studenti frequentanti ha passato il progetto entro l'appello di febbraio
- Di questi, circa il 63% ha verbalizzato entro luglio
- Il 76% dei voti assegnati ai progetti è stato $>= 27$
- Solo il 10% degli studenti ha preso un voto < 23
- Spero che queste statistiche vi abbiano rincuorato sulla fattibilità dell'esame :)

Disclaimer

- Il contenuto di questi lucidi è da considerarsi puramente indicativo
  - Verranno forniti dei suggerimenti di sviluppo e di progettazione
  - Ogni studente può effettuare scelte differenti
- Le direttive su come svolgere il progetto sono riportate nel testo del progetto presente sul sito del corso.

Identificazione delle applicazioni

Il sistema richiede due applicazioni:

- *Peer:* il client di gioco
- *Server:* server REST che gestisce le partite del gioco e i relativi utenti

Un consiglio..

Flusso di sviluppo consigliato

- Primo step (lezione di oggi): sviluppo del server
  - Progettazione server (risorse e metodi)
  - Sviluppo servizio REST
  - Analisi e risoluzione di problemi di sincronizzazione
  - Testing con tool dedicati
- Secondo step (prossima lezione): sviluppo del client
  - Progettazione token ring (inserimento, passaggio token, rimozione)
  - Sviluppo di token ring (senza logica di gioco)
  - Progettazione protocollo per comunicazione di mosse
  - Sviluppo funzionaltià "*spostamento*"
  - Sviluppo funzionalità "*bomba*"
- È assolutamente necessario considerare attentamente tutti i problemi di sincronizzazione *interna* e sincronizzazione *distribuita*

Server

- Il *Server* è un'unica applicazione che ha il compito di:
  - Creare nuove partite
  - Visualizzare le partite esistenti
  - Permettere ad utenti di aggiungersi a nuove partite
  - Cancellare partite
- Questo servizio deve esere erogato tramite architettura REST
- Sono quindi necessari meccanismi di sincronizzazione per gestire l'accesso alle risorse condivise

Risorse e operazioni

Quali risorse?

- Il primo passo è sicuramente quello di individuare:
  - Le risorse da modellare
  - Le operazioni CRUD effettuabili sulle risorse e il mapping con HTTP
- Una *partita* può sicuramente essere considerata una risorsa
- Un *utente* invece?
  - Conviene se volete gestire operazioni per gli utenti separatamente (esempio registrazione con username)

Esempio mapping CRUD - HTTP

- GET $\rightarrow$ ottenere elenco partite o dettaglio di una partita
- POST $\rightarrow$ creare una nuova partita
- PUT $\rightarrow$ aggiungere utente a partita
- DELETE $\rightarrow$ eliminare partita o eliminare utente da partita

Gestione partite (I)

Creazione/cancellazione partite

- Quando un giocatore richiede di creare una nuova partita, il *Server*:
  - Crea una nuova partita con il nome specificato e la aggiunge all'elenco delle partite in corso
  - Se il nome della partita è già stato usato, viene restituito un messaggio di errore
  - La lista di giocatori attivi viene inizializzata
  - Ritorna un messaggio di avvenuta creazione della partita al giocatore
- La rimozione di una partita avviene eliminando la partita dall'elenco delle partite in corso

Gestione partite (II)

Inserimento/uscita da partite

- Quando un giocatore richiede di entrare in una partita:
  - Il *Server* controlla che non ci siano altri giocatori con lo stesso id nella partita
  - Se il controllo va a buon fine, restituisce un messaggio positivo con la lista dei giocatori attualmente presenti
  - Altrimenti restituisce un messaggio di errore
- L'uscita dalla partita viene gestita semplicemente rimuovendo quel giocatore dalla lista
  - Se il server si accorge che la partita non ha più giocatori, la elimina automaticamente

Reminder

Ricordatevi che...

- Ogni classe che gestisce una risorsa (annotata con *@Path*) viene istanziata ogni volta che viene effettuata una singola richiesta HTTP
  - È quindi necessario gestire adeguatamente la memoria condivisa
- Viene automaticamente gestito il multi-threading: chiamate concorrenti eseguono in concorrenza il codice di diverse istanze della classe che gestisce la risorsa
  - Bisogna fare attenzione ai problemi di concorrenza

Possibili problemi di sincronizzazione

- All'interno dello sviluppo del server sono presenti svariati problemi di sincronizzazione:
  - Cosa succede se due partite vengono create contemporaneamente con lo stesso nome?
  - Cosa succede se due giocatori con lo stesso nome vogliono entrare nella stessa partita?
  - Cosa succede se due giocatori efettuano contemporaneamente l'ingresso in una partita? (ricordatevi che devono ricevere l'elenco dei giocatori attivi)
  - ...

Come testare il Server?

- Il primo passo è di testare ogni singolo metodo con tool comodi
  - Ad esempio *Advanced REST Client*
- Per testare invece i problemi di concorrenza (e anche per iniziare a metterci le mani) vi conviene scrivere codice Java
- Il consiglio è quello di utilizzare le API Client di Jersey
  - Il vantaggio è anche quello di sfruttare nuovamente marshalling e unmarshalling offerto da JAXB

Nella prossima puntata...

Spoiler

- Nella prossima lezione vedremo invece lo sviluppo del client
  - La gestione della rete ad anello
  - Il passaggio del token
  - La comunicazione tra processi
  - ...

Buon lavoro!