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

I prossimi passi

- 3 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
- Le lezioni partiranno ufficialmente alle 08:45 (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

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 lo sviluppo di tre applicazioni diverse:

- *Casa*
- *Server amministratore*
- *Amministratore*

Un consiglio..

Flusso di sviluppo consigliato

- Primo step (lezione di oggi): sviluppo del server
  - Progettazione server (risorse e metodi)
  - Sviluppo servizi REST
  - Analisi e risoluzione di problemi di sincronizzazione
  - Testing con tool dedicati
  - Sviluppo client *Amministratore*
- Secondo step (prossima lezione): sviluppo della rete di *case*
  - Progettazione architettura e protocolli per la rete peer-to-peer
  - Implementazione rete dinamica peer-to-peer (inserimenti ed uscite dinamici di case)
  - Implementazione di raccolta/comunicazione dati smart meter
  - Implementazione algoritmo di mutua esclusione distribuita
- È assolutamente necessario considerare attentamente tutti i problemi di sincronizzazione *interna* e sincronizzazione *distribuita*

Server amministratore

- Il *server amministratore* è un'unica applicazione che ha il compito di:
  - Gestire l'inserimento e rimozione di *case*
  - Ricevere le statistiche locali e globali relative al consumo energetico condominiale
  - Permettere agli amministratori di analizzare le statistiche
- Questi servizi devono esere erogati 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
- Il condominio può essere considerato una risorsa?(probabilmente si)
- Come modellare le statistiche?

Esempio mapping CRUD - HTTP

- GET $\rightarrow$ ottenere le informazioni relative alle case presenti nel condominio
- POST $\rightarrow$ inserire una nuova casa nel condominio
- PUT $\rightarrow$ modificare le informazioni relative ad una casa (ci serve?)
- DELETE $\rightarrow$ rimuove una casa dal condominio

Gestione *case*

Inserimento/uscita di una casa

- Quando una *casa* fa richiesta di entrare, il *server amministratore*:
  - Cerca di inserirla nel condominio
  - Se esiste già una *casa* con lo stesso identificatore viene restituito un messaggio di errore un messaggio di errore
  - Se invece è tutto regolare viene aggiunta all'elenco di case del condominio e viene restituita la lista di *case* presenti nella città
- L'uscita di una *casa* avviene semplicemente rimuovendola dall'elenco
- Che tipo di sincronizzazione è necessaria?

Va reso atomico il controllo che una casa esista e il suo inserimento.

Gestione statistiche

- Lato *casa*:
  - Il condominio invia periodicamente le statistiche globali insieme alle statistiche locali più recenti di ogni *casa*.
  - Queste misurazioni vanno salvate internamente in una struttura dati per permettere successivamente l'analisi
- Lato amministratore: il server REST deve avere delle interfacce usate dall'amministratore
  - Sono necessarie delle interfacce per analizzare i dati come da specifiche
- Cercare di realizzare la sincronizzazione più a grana fine possibile:
  - Ha senso bloccare ogni operazione lato server (ad esempio aggiunta/rimozione di case) mentre vengono calcolate le statistiche? No, si possono controllare le statistiche mentre si fanno aggiunte/rimozioni
  - Ha senso bloccare l'intera struttura dati con tutte le statistiche anche se vogliamo analizzare le statistiche di una specifica *casa*?

Jersey: Reminder

Il multithreading in Jersey esiste ma è nascosto. Ogni volta che viene fatta una richiesta HTTP viene creata una nuova classe. Synchornized non funziona, va usato un singleton.

Ricordatevi che...

- Ogni classe che gestisce una risorsa (annotata con *@Path*) viene istanziata (circa) 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:
  - L'elenco di case viene modificato e letto in maniera concorrente
  - Le statistiche vengono aggiunte e lette in maniera concorrente
- Gestite la sincronizzazione in maniera snella
- Utilizzare *synchronized* alla rinfusa non è assolutamente una best practice

Come testare il server amministratore?

- Il primo passo è quello di testare ogni singolo metodo REST con tool comodi
  - Ad esempio *Advanced REST Client*
- Testate i problemi di concorrenza con le *sleep()*
- Per automatizzare test più complicati, 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 della rete di *case*
  - Progettazione dell'architettura
  - La comunicazione tra processi
  - Mutua esclusione distribuita
  - ...

Buon lavoro!