Commenti sulla soluzione del compito

Un punto chiave è il fatto che i dati delle aste devono essere

- DINAMICI (ovvero alcuni attributi – prezzo e miglior offerente – cambiano nel tempo)
- CONDIVISI tra gli utenti (I dato devono essere lo stesso per tutti gli utenti)
- VISIBILI dalla varie unità logiche (Le varie servlet e JSP devono poter accedere ai dati)

Questo fa si che due utenti da macchine diverse o browser diversi possano competere sullo stesso dato.
Pertanto, ne discende che i dati DEVONO essere mantenuti a livello di WebApp (e dunque nel ServletContext – soluzioni basati su dati static di una servlet funzionano, ma è da preferire la prima strada).

I dati devono essere caricati dalla prima Servlet che si accorge che non sono presenti nel contesto (in genere la Home, ma se un utente ha bookmarkato un'altra pagina, è possibile che la usi come entry point: quindi anche altre servlet sono candidate a fare il primo caricamento). Questo va fatto nella init: non è corretto controllarlo ad ogni esecuzione della doGet/doPost. Peggio ancora metterlo nelle JSP. La soluzione più elegante è farlo tramite il ServletContextListener (qualcuno lo ha fatto) ma dato che a lezione abbiamo appena accennato all'ascolto degli eventi di contesto, non chiedo sia fatto così, e non lo faccio nella soluzione pubblicata che lavora con la init().

E' concettualmente sbagliato (gravemente sbagliato) tenere i dati in session: questa serve solo per i dati individuali, non può funzionare con i dati condivisi!
Quindi, ogni soluzione basata su dati in session è considerata insufficiente. L'unico dato da tenere in session è lo stato dell'utente (autenticato o no, ad esempio tenendo in session lo username).

Era richiesta una soluzione MVC: quindi soluzioni basate solo su JSP o solo su servlet non vanno bene. Dato che in una soluzione di questo tipo non ha mai senso chiamare direttamente una JSP, è bene nasconderle agli utenti mettendole in WEB-INF, così che siano richiamabili solo dalle servlet.

Mettere nelle JSP del codice statico HTML tramite delle out.println() è considerato sbagliato. Le out.println() vanno usate solo per stampare valori di variabili (dati dinamici), e comunque ad essa va preferita la sintassi <%= %>.

Il testo diceva che non viene gestita la persistenza, quindi che senso aveva aggiungere un DB? Se uno lo voleva fare "in più" ok, ma solo a patto che tutto il resto fosse fatto.

Piccole penalizzazioni di punteggio sono state date a chi non ha usato nomi di package canonici e a chi non ha seguito le regole per il nome del progetto date nelle istruzioni generali.

Un modo per far si che la web app si avvii con la pagina di catalogo è usare la <welcome-file-list> del web.xml

Il countdown dei secondi era facilmente realizzabile in JavaScript.

Il controllo ogni 10 sec. dello stato dell'applicazione per vedere se vi erano nuove offerte poteva essere fatto con Ajax, o più semplicemente con un aggiornamento della pagina con JavaScript: dato che la pagina era molto leggera, non vi è molta differenza tra le due strategie (anche dal punto di vista visivo). In presenza di una pagina molto ricca e pesante la scelta di Ajax sarebbe stata preferibile.