# Esercitazione del week end Francesco Randillo

Modello Entità-Relazioni

n

n

Visit

+id

+date

+name

+result

1

1

n

n

extends

extends

Reparto

+Id

+Nome

Patient

+admissionDate

+exitDate

+totDays

Doctor

+registrationNumber

Person

+id

+name

+surname

+cf

+dateOfBirth

+placeOfBirth

+gender

Item-Type

Gli item type che ho creato seguono il precedente modello Entità-Relazioni, la relazione n-n tra Pazient e Doctor con la relazione Visit intermedia è stata risolta tramite una relazione 1-n tra Patient e Visit, e 1-1 tra Visit e Doctor aggiungendo Doctor agli attributi di Visit).

Ho creato due Enum-type: Gender (M o F) e Result(POSITIVE o NEGATIVE).

Le relazioni sono le seguenti:

* *Department2DoctorRelation;* Department 1, Doctor n
* *Patient2VisitRelation;* Patient 1, Visit n
* *Departed2VisitRelation;* Department 1, Visit n
* la relazione 1-1 tra Visit e Doctor, risolta aggiungendo l’attributo Doctor in Visit

l’Item Patient ha inoltre un attributo dinamico totDays, relativi ai giorni totali in ospedale, calcolati dalla differenza dei giorni tra la admissionDate ed exitDate, questo attributo è stato risolto tramite una classe DaysHandler (package: my.hospitaluiipcore.handler) che calcola la differenza dei giorni.

Dopo gli item-type, ho popolato il Db tramite gli impex (resource/impex/hospitaluiipcore.impex)

DAO/SERVICE

Per la risoluzione dell’esercizio ho creato un’interfaccia VisitDao, con due metodi:

List<VisitModel> findVisitByPatient(PatientModel patient);

List<VisitModel> findVisitofTomorrow(Date date);

il primo cerca tutte le visite di un paziente, utile per l’interceptor/event che deve stampare su file la cartella clinica del paziente,il secondo metodo cerca tutte le visite per una determinata data, utile per la risoluzione del cronjob che deve visualizzare tramite Log le visite del giorno successivo, la data viene calcolata dal service che chiama il metodo del Dao.

I metodi sono stati implementati dalla classe DefaultVisitDao, che eseguono delle Flexible Search per ottenere i dati necessari.

Ho creato anche un’interfaccia VisitService con i seguenti metodi:

List<VisitModel> getVisitForPatient(PatientModel patient);

List<VisitModel> getVisitofTomorrow();

i metodi sono stati implementati dalla classe DefaultVisitService, che implementano la logica dell’applicazione.

Interceptor/Event

L’interceptor che ho utilizzato è un Prepare-Interceptor, implementato nella classe VisitPrepareInterceptor, questo interceptor attende una modifica di VisitModel, quando avviene lancia il metodo onPrepare: controlla se il valore result del model modificato corrisponde a “POSITIVE”, se è vero allora lancia l’evento VisitEvent, passandoci come parametro il patient associato al model modificato. La classe VisitEvent estende AbstractEvent e fornisce i metodi get, set e Costruttore del patient passato. Inoltre ho creato la classe VisitEventListener che estende AbstractEventListener<VisitEvent>, ed implementa la logica dell’ event:

* tramite visitService ottiene una lista di Visit del Patien passato all’event
* dichiaro un oggetto PrintWriter per la scrittura sul file csv
* tramite un ciclo for each mi scorro il contenuto della lista ed inserisco i valori nel file

la Path del file csv è stata scritta all’interno del file project.properties, e la richiamo tramite :

**final** String path = Config.getParameter("visitecsv").toString();

il file risiede nella cartella visitcsv, all’interno dell’estensione hospitaluiipcore.

Prima di stampare su file, visualizzo un Log di livello Info, che mi notifica la scrittura su file.

Sono stati creati i relativi Bean per l’interceptor e per il mapping, e quello relativo all’event.

CronJob

Per la realizzazione del CronJob ho creato una classe VisitJob che estende AbstractJobPerformable<CronJobModel>.

Nel metodo perform ho implementato la logica del CronJob:

* tramite l’oggetto visitService ottengo una lista di visite corrispondenti al giorno successivo
* con un for each mi scorro il contenuto della lista
* visualizzo il contenuto tramite un’oggetto LOG con visibilità Info

per il conjob ho creato l’impex per schedularlo ogni notte alle 23:30 (resources/impex/cronjob.impex), ed il relativo bean.

FACADE/FRONTEND

In aggiunta alla traccia ho realizzato anche un’estensione hospitaluiipfacade, dove ho creato i DATA, le varie classi populator per ogni item, e un’interfaccia ed una classe per il facade di Visit, ed infine i relativi bean per ogni classe creata.

Ed un’estensione hospitaluiipfrontend, dove ho creato un controller ed una jsp, che mi visualizzano le visite del giorno successivo