Programmazione Orientata agli Oggetti in Linguaggio Java

Programmazione su XML: Strato di Persistenza

versione 2.0

G. Mecca – Università della Basilicata – mecca@unibas.it

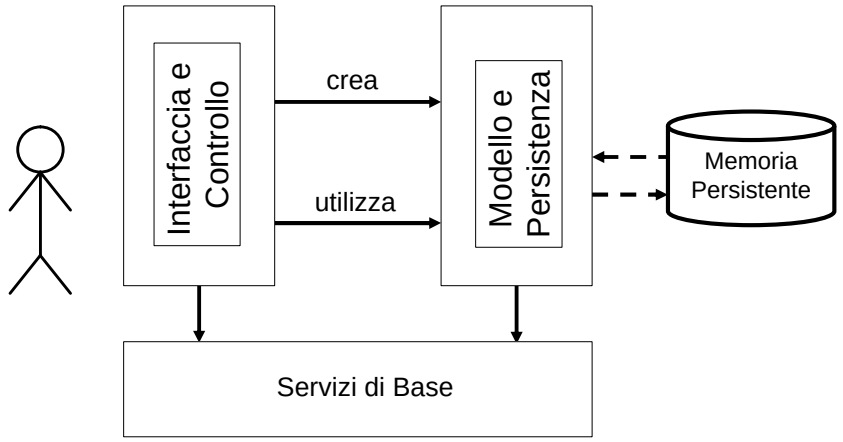
Sommario

- Lo Strato di Persistenza
- DAO
- Eccezioni
  - ⇨Trasformare le Eccezioni
  - ⇨Definire Nuove Eccezioni

Lo Strato di Persistenza

- Finora
  - ⇨tutte le applicazioni erano basate sull'architettura di base introdotta a lezione
  - ⇨due "macrostrati" applicativi
  - ⇨interfaccia e controllo assieme
  - ⇨modello e persistenza assieme

Lo Strato di Persistenza



Lo Strato di Persistenza

- In prospettiva
  - ⇨gli strati dovranno diventare quattro distinti
- In questo modulo
  - ⇨come separare lo strato di persistenza dagli altri, in particolare dal modello
  - ⇨introducendo un ulteriore strato di componenti specializzati esclusivamente in operazioni di persistenza

DAO

ATTENZIONE al concetto di DAO

- Data Access Objects (“DAO”)
  - ⇨componenti dello strato di persistenza
  - ⇨tipicamente sono componenti unici con visibilità globale (componenti statici)
  - ⇨contenuti in un package separato
- Due operazioni fondamentali
  - ⇨leggere dallo strato di persistenza
  - ⇨scrivere sullo strato di persistenza

DAO

- Obiettivi
  - migliorare la manutenibilità del codice
- Aumentare la coesione
  - concentrare tutto il codice relativo alla persistenza in componenti specializzati
- Ridurre l'accoppiamento
  - proteggere, per quanto possibile, lo strato di interfaccia e controllo e quello del modello dalle modifiche apportate alla persistenza

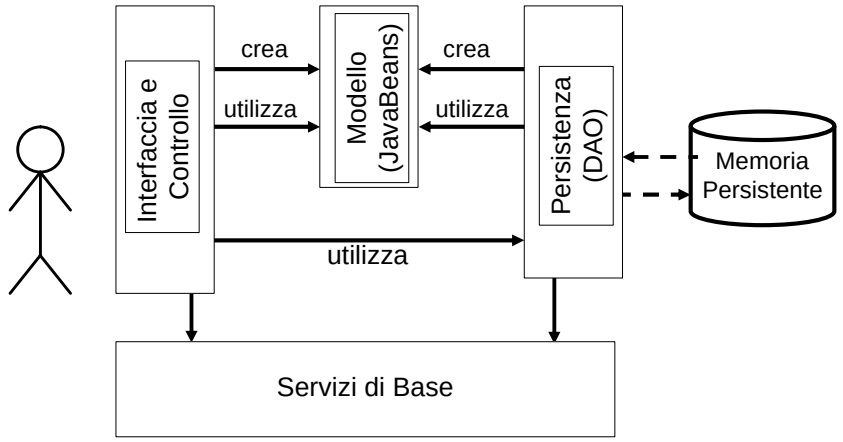
DAO

- L'approccio che utilizzeremo
  - i componenti del modello saranno completamente indipendenti da quelli di persistenza
  - JavaBeans che conterranno dati (get/set) e metodi di logica applicativa
  - l'unico strato ad utilizzare (dipendere da) la persistenza è lo strato di interfaccia e controllo

DAO

- Per ridurre l'accoppiamento
  - ⇨lo strato di persistenza deve essere scritto in modo tale da "nascondere" al controllo la tecnologia della persistenza
  - ⇨il controllo conosce esclusivamente i componenti di persistenza e chiede loro di eseguire operazioni senza preoccuparsi di come vengono effettuate
  - ⇨es: file con formato libero vs. XML

DAO



DAO

- Per ora
  - ⇨ci concentriamo su DAO che lavorano con la persistenza su file
  - ⇨ma sono utilizzati anche con altre tecnologie di persistenza (es: DBMS)
- Scelta dei DAO
  - ⇨tipicamente un DAO statico per ciascuna classe del modello per cui è richiesta la persistenza

DAO

- Metodi del DAO
  - ⇨costruttore privato
  - ⇨metodo statico di caricamento
    Oggetto carica(String nomeFile)
  - ⇨metodo statico di salvataggio
    salva (*Oggetto oggetto*, String nomeFile)
  - ⇨a volte, metodo statico di aggiornamento
    aggiorna (*Oggetto oggetto*, String nomeFile)

DAO

- Esempio: DAORisposte.java
  - componente specializzato nel leggere dal disco e scrivere risposte
  - public static Risposte carica(String nomeFile)
  - public static void salva(Risposte risposte, String nomeFile)
- Nota: differenza tra i due metodi
  - il metodo di caricamento è una funzione (deve creare e restituire un oggetto contenente i valori letti dal file)
  - il metodo di salvataggio è una procedura (lavora su un oggetto esistente)

DAO

- Vantaggi di questo approccio
  - il codice della persistenza è più facilmente localizzabile
  - si evitano classi molto lunghe a causa della presenza contemporanea di logica applicativa e logica di persistenza
  - i DAO incapsulano la strategia di persistenza: è facilmente possibile cambiare tecnologia per la persistenza

DAO

- Un esempio
  - ⇨potrei facilmente passare da file XML a file con formato libero
  - ⇨basta cambiare il codice dei DAO
  - ⇨il resto dei componenti non è sostanzialmente intaccato
- Attenzione, però
  - ⇨cambiano le eccezioni

Eccezioni

- Eccezioni nello strato di persistenza
  - ⇨possono essere di vari tipi
  - ⇨possono essere catturate dal DAO, per esempio a scopo di logging, ma non sempre devono essere gestite dal DAO
- Strategia di gestione dell'eccezione
  - ⇨due possibili casi

Eccezioni

- Caso n. 1
  - ⇨l'eccezione provoca un problema marginale nell'ambito della logica applicativa
  - ⇨es: il record di IndovinaIlNumero
  - ⇨in questo caso, il componente di persistenza può decidere di catturare e gestire l'eccezione, ponendo rimedio
  - ⇨l'eccezione viene nascosta al controllo
  - ⇨si tratta di un caso poco frequente

Eccezioni

- Caso n. 2: caso più frequente
  - ⇨l'eccezione provoca un problema sostanziale nell'ambito della logica applicativa
  - ⇨es: non è possibile acquisire il questionario
  - ⇨in questo caso è il controllo che deve gestirla, perchè si tratta di un problema troppo grave di funzionamento dell'appl.
  - ⇨Il il DAO deve necessariamente rilanciare l'eccezione al controllo

Eccezioni

- In questi casi
  - ⇨sarebbe opportuno trasformare l'eccezione prima di rilanciarla al controllo
  - ⇨in modo che il controllo non debba preoccuparsi di dettagli relativi alla specifica eccezione
  - ⇨ma sappia semplicemente che c'è stato un problema sulla persistenza

Trasformare le Eccezioni

ATTENZIONE
alla trasformazione delle eccezioni

- In concreto
  - ⇨questo è fattibile dichiarando una nuova classe di eccezione: DAOException
  - ⇨un semplice "involucro" per incapsulare eccezioni diverse verificatesi durante le operazioni di caricamento e salvataggio
  - ⇨le eccezioni vengono catturate e utilizzate per creare una DAOException che viene rilanciata al controllo

```
public static Risposte caricaRisposte(String nomeFile) throws DAOException {
    org.jdom.Document document = costruisciDOM(nomeFile);
    Risposte risposte = new Risposte();
    analizzaRadice(document, risposte);
    java.util.List listaNodiRisposta = estraiListaNodiRisposta(document);
    estraiRisposte(listaNodiRisposta, risposte);
    return risposte;
}

private static org.jdom.Document costruisciDOM(String nomeFile)
                                                    throws DAOException {
    org.jdom.input.SAXBuilder builder = new org.jdom.input.SAXBuilder();
    builder.setValidation(true);
    org.jdom.Document document = null;
    try {
        document = builder.build(nomeFile);
        return document;
    } catch (org.jdom.JDOMException jde) {
        throw new DAOException(jde);
    } catch (java.io.IOException ioe) {
        throw new DAOException(ioe);
    }
}
```

i metodi di caricamento e salvataggio lanciano DAOException

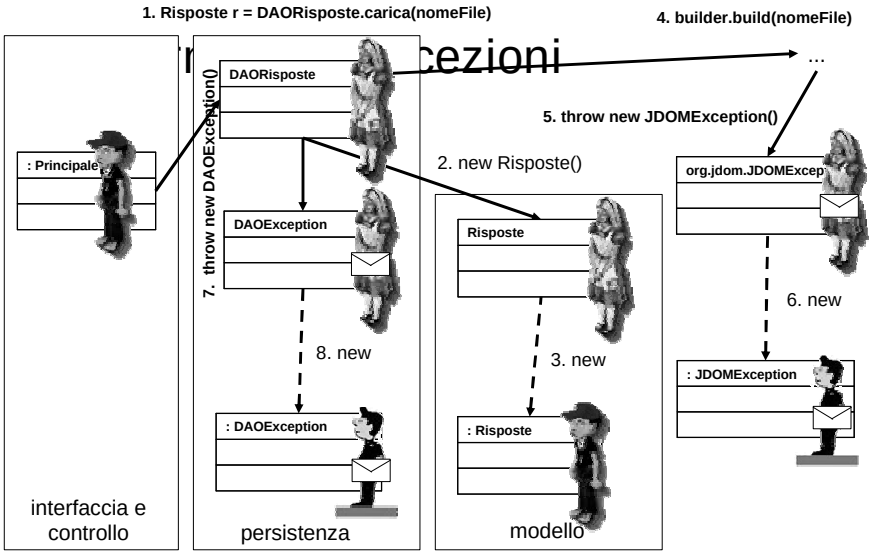
le varie eccezioni sollevate durante il processo di caricamento e salvataggio vengono catturate e rilanciate dopo essere state trasformate in DAOException

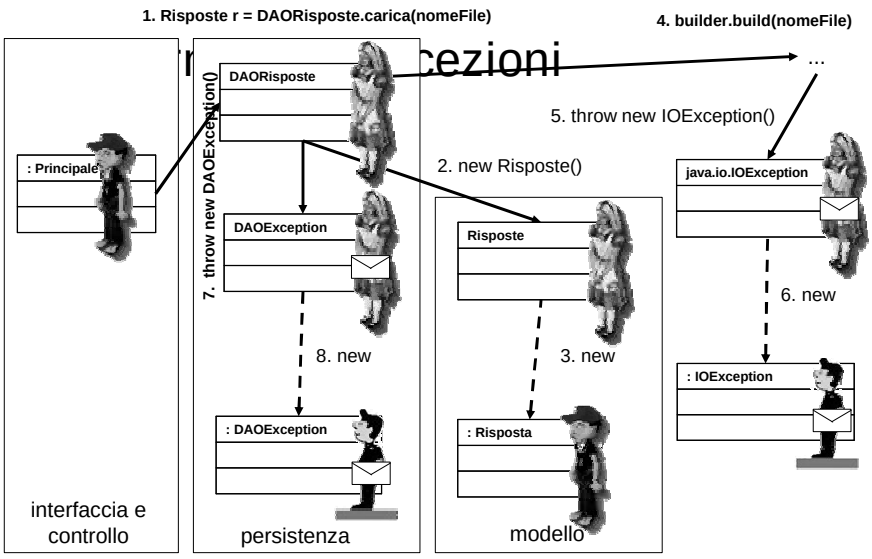

```
package it.unibas.questionari.controllo;

public class Principale {

    private Risposte schermoCaricaRisposte() {
        System.out.println("-----------------------------------");
        System.out.println("       Caricamento Risposte");
        System.out.println("-----------------------------------");
        System.out.print("Inserisci il nome del file --> ");
        String nomeFile = it.unibas.utilita.Console.leggiStringa();
        Risposte risposte = null;
        try {
            risposte = DAORisposte.caricaRisposte(nomeFile);
            System.out.println(" -- Caricamento effettuato -- ");
        } catch (DAOException pe) {
            System.out.println("ERRORE: " + pe);
        }
        return risposte;
    }
```

il controllo vede un'unica eccezione rilanciata dalla persistenza: DAOException

Trasformare le Eccezioni

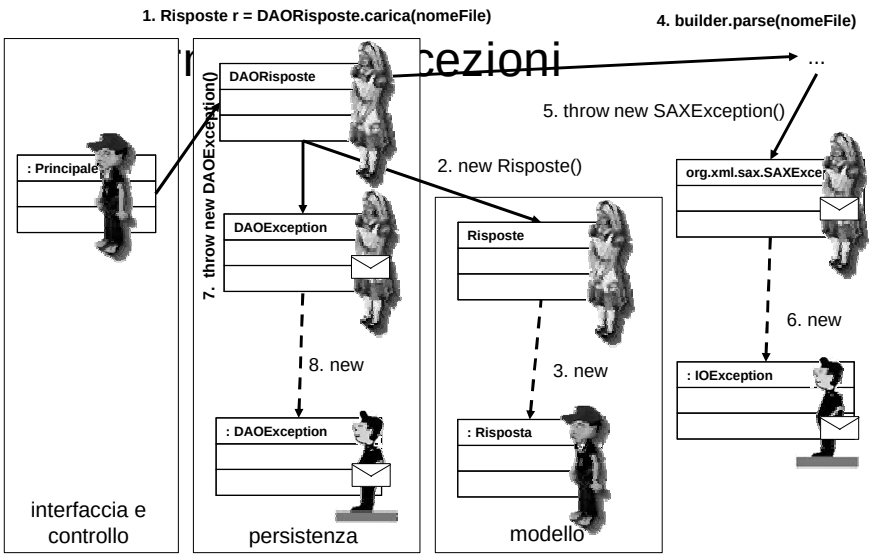
- In questo modo
  - ⇨il controllo è di fatto disaccoppiato completamente dalla persistenza
  - ⇨vede un'unica eccezione, DAOException
  - ⇨e non le diverse eccezioni che possono verificarsi
  - ⇨es: IOException oppure ParserException oppure JDOMException





Trasformare le Eccezioni

- Vantaggio di questo approccio
  - ⇨posso facilmente cambiare la tecnologia usata per la persistenza senza avere alcuna influenza sul controllo
- Esempio
  - ⇨passo da JDOM a JAXP
  - ⇨cambia il codice dei DAO
  - ⇨cambiano le eccezioni; es: SAXException
  - ⇨il codice del controllo non cambia



Definire Nuove Eccezioni

- La definizione di DAOException
  - ⇨si tratta di un nuovo tipo di eccezione non previsto dalle API di Java
  - ⇨scelgo che sia un'eccezione controllata
  - ⇨estende java.lang.Exception
  - ⇨eredita la proprietà message
  - ⇨è necessario definire esclusivamente i costruttori

```
package it.unibas.questionari.persistenza;

public class DAOException extends Exception {

    public DAOException() {
        super();
    }

    public DAOException(String s) {
        super(s);
    }

    public DAOException(Exception e) {
        super(e);
    }

}
```


Riassumendo

- Lo Strato di Persistenza
- DAO
- Eccezioni
  - ⇨Trasformare le Eccezioni
  - ⇨Definire Nuove Eccezioni

Termini della Licenza