Pattern Role-Based Access Control

RBAC [Schumacher]

- Intento: descrive come gli utenti possono acquisire i diritti in base alle funzioni del loro lavoro o dei compiti loro assegnati
- Contesto: ogni ambiente in cui necessitiamo il controllo degli accessi alle risorse computazionali e dove vi è un gran numero di utenti, tipi di informazioni, e grande varietà di risorse
- Problema
  - Per convenienza di amministrazione dei diritti di accesso bisogna aver un modo per separare i diritti dal resto, altrimenti assegnare i diritti ai singoli utenti richiederebbe l'immagazzinamento di tante regole di autorizzazione, e sarebbe difficile per gli amministratori tener traccia di queste regole

Role-Based Access Control

- Problema: per assegnare i diritti alle persone in base alle funzioni o ai compiti bisogna bilanciare le seguenti forze
  - Le persone possono essere classificate in base alle loro funzioni o compiti
  - Compiti comuni possono richiedere un insieme di diritti simili
  - Aiutare l'organizzazione a definire diritti di accesso in base alla regola di **dare il minimo di informazioni** ai diretti interessati (need-to-know policy)
- Soluzione
  - Molte organizzazioni hanno varietà di ruoli che richiedono diverse abilità e responsabilità. Per ragioni di sicurezza, gli utenti dovrebbero avere i diritti in base alle loro funzioni e sui compiti a loro assegnati. Questo corrisponde al principio need-to-know
  - Le funzioni nell'ambiente di lavoro possono essere interpretate come ruoli delle persone durante lo svolgimento del lavoro

Role-Based Access Control

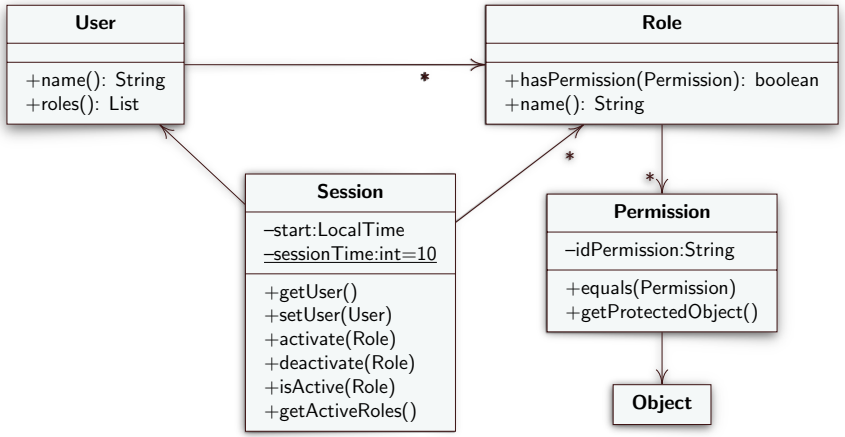
- Soluzione
  - **User** rappresenta un utente registrato precedentemente
  - **Role** descrive un ruolo predefinito
  - **Permission** definisce il tipo di accesso permesso ad un certo ruolo su un oggetto protetto
  - **ProtectionObject** è l'oggetto (o risorsa) da proteggere
  - **Session** mappa ciascun utente ad un insieme di ruoli attivi che gli sono stati assegnati
  - Ovvero, all'interno di una sessione, a un utente viene assegnato un ruolo (o più ruoli), e ciascun ruolo ha permessi predefiniti in base alle funzioni che si sa che quel ruolo deve svolgere

Esempio Di Applicazione

- Permettere a diverse organizzazioni di partecipare a varie missioni
  - Durante una missione un utente può mandare allarmi (Alert) per richiedere servizi di trattamento per persone ferite
  - Ogni utente appartiene ad almeno un'organizzazione e può essere assegnato a una o più missioni. Ogni missione ha un nome, una data di inizio e di fine, e un luogo
  - L'appartenenza di un utente a un'organizzazione o a una missione non gli dà automaticamente accesso a risorse. L'accesso è permesso in base al ruolo che ha l'utente
- **Ruoli**: SecurityOfficer, Participant, Trainee
- **Utenti**: Alice è un Participant; Bob è un SecurityOfficer
- Oggetti: Refugee, Device, Alert, Disk
- Permessi per Participant: readRefugee e updateRefugee su Refugee
- Permessi per SecurityOfficer: sendAlert su Alert

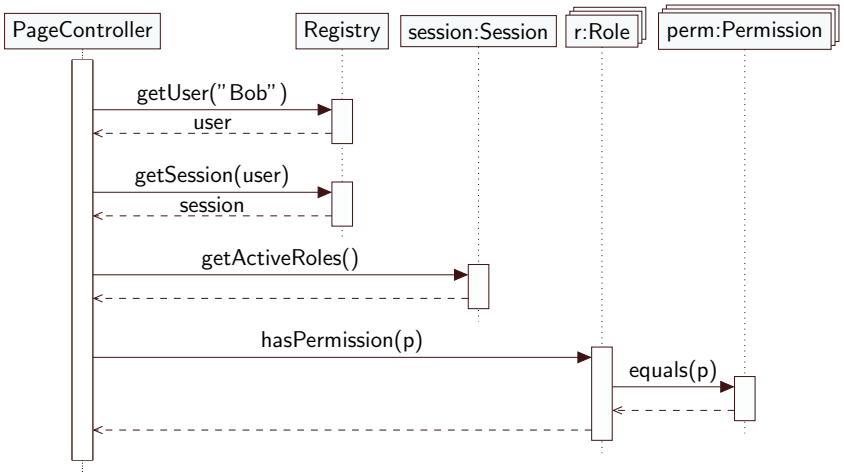
Role-Based Access Control

- Soluzione: diagramma UML delle classi



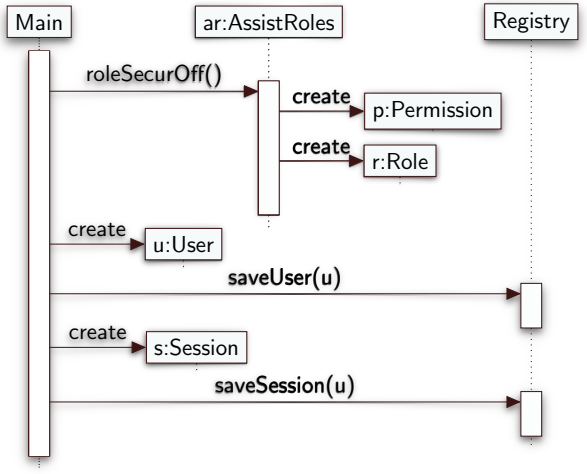
Role-Based Access Control

- Soluzione: prima di eseguire l'operazione desiderata, si controlla che l'utente (Bob), per un suo ruolo attivo (securityOfficer), abbia il permesso appropriato (sendAlert)



Role-Based Access Control

- Soluzione: creazione dell'utente, assegnamento di un ruolo e creazione sessione



Design Pattern RBAC (init user and save session)

Role-Based Access Control

Conseguenze

- L'applicazione di questo pattern fornisce i seguenti benefici
  - Permette agli amministratori di ridurre la complessità della sicurezza, poiché vi sono molti più utenti che ruoli
  - Politiche dell'organizzazione che riguardano le funzioni del lavoro possono essere direttamente mappate sulla definizione dei ruoli e sull'assegnazione degli utenti ai ruoli
  - E' semplice soddisfare gli utenti che arrivano, che lasciano l'organizzazione o che sono riassegnati. Tutte queste attività richiedono solo la manipolazione dell'associazione fra utenti e ruoli