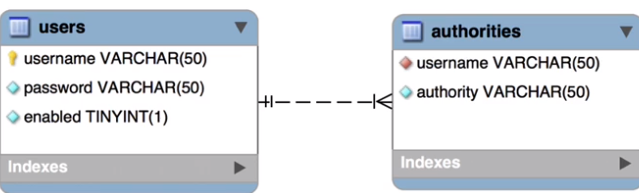
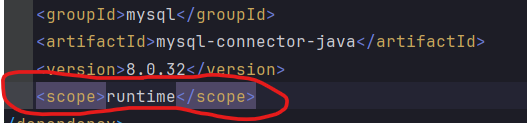
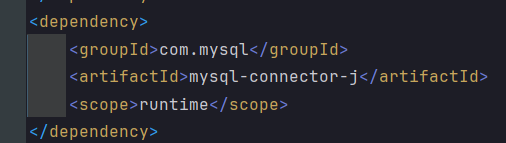
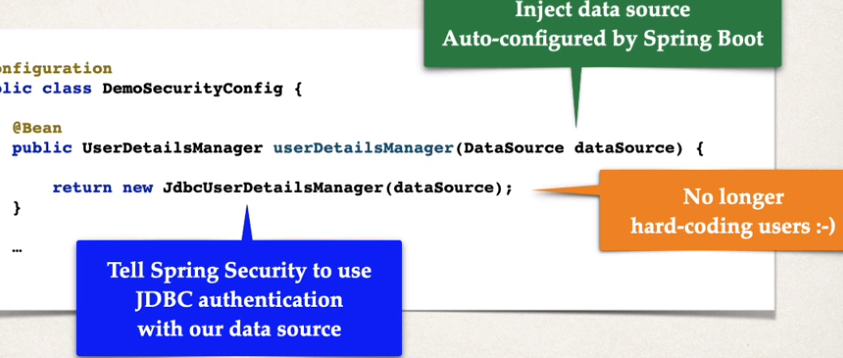
impareremo a usare Spring Security con gli account utente memorizzati nel database. Finora, i nostri account utente sono stati codificati nel codice sorgente Java, e lo abbiamo fatto solo per mantenere le cose semplici, ma ora vogliamo aggiungere l'accesso al database. Si tratta di una funzione avanzata di Spring Security, che utilizzeremo in questa serie di video.  
  
 Spring Security ha uno schema di database predefinito. È quindi necessario fornire due tabelle, una chiamata utenti e un'altra chiamata autorità. E bisogna usare questi nomi esatti delle tabelle. Inoltre, le tabelle devono avere queste colonne specifiche: nome utente, password e abilitato per la tabella degli utenti e anche nome utente e autorità per la tabella delle autorità.   
Quindi, ancora una volta, quando vedete le autorità, pensate ai ruoli.

  
  
Vedi file database-user-auth.sql  
  
Nella tabella users come possiamo notare abbiamo la colonna enabled, possiamo gestire la logica d’accesso ad un sito o altro, quindi per accesso abilitato metteremo 1, per accesso disabilitato 0  
  
Poi avremo la tabella authorities

la quale avrà due campi come possiamo vedere,   
**e una chiave unica**  → UNIQUE KEY `nome\_chiave` (`colonna1`, `colonna2); LA QUALE è COMPOSTA DAI DUE CAMPI, e poi una chiave esterna username della tabella authorities che punta alla chiave primaria della tabella users

create le tabelle, procediamo a collegare il progetto spring col database tramite jdbc,  
 e imposterò anche l'ambito per il **runtime**, perché abbiamo bisogno di questa dipendenza solo in fase di esecuzione.

 sbagliato quest’importazione  
  
 **OK**

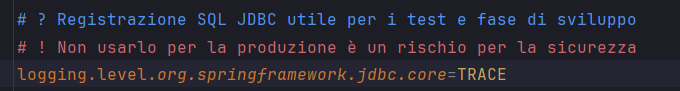
e successivamente andiamo a lavorare sulla classe di configurazione, dove una volta avevamo il codice sorgente degli utenti e degi ruoli  


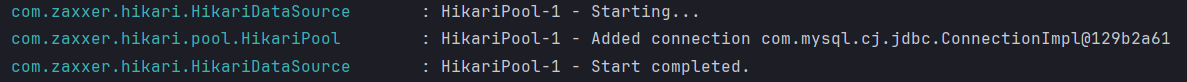
questo indica a Spring Security di usare l'autenticazione JDBC con la nostra fonte di dati DEFINITA nella file application.properties. E poi Spring Security sa che sta usando uno schema di tabelle predefinito.  
Quindi Spring Security cercherà in una tabella chiamata utenti e in un'altra tabella chiamata ruoli.

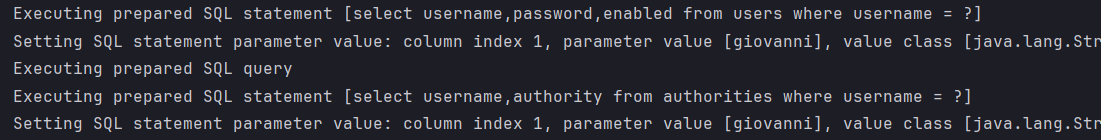
Conosce i nomi esatti delle colonne che utilizzerà. Ed è praticamente già pronto per l'uso. Tornando indietro, l'aspetto positivo dell'uso di questo UserDetailsManager JDBC è che non stiamo più codificando gli utenti.

**Diagramma eer in mysql**database → reverse engineer   
  
E poi un'altra cosa che vorrei fare per aiutarci, come esercizio accademico, è che vorrei essere in grado di registrare le istruzioni SQL JDBC. La cosa fondamentale è che questa configurazione o registrazione

sia impostata solo per lo sviluppo e i test. Non usarlo in produzione, perché registrerà effettivamente i nomi utente di springsecurity. Questo potrebbe essere un rischio per la sicurezza.



Questo è utile quando si esegue un'applicazione per verificare che l'applicazione stia davvero leggendo le informazioni dal database. E questo è un bene per scopi accademici.  
  
Ed ecco cosa possiamo dotare nella console quando viene effettuato un tentativo di login   
  
e quindi ciò ci dice che è stata avviata correttamente  
  
**Hikari** è un pool di connessioni JDBC ad alte prestazioni e leggero che viene utilizzato in Spring Boot per gestire le connessioni al database. Hikari è il pool di connessioni predefinito in Spring Boot

ma vorrei ancora verificare l'intero aspetto della sicurezza, per quanto riguarda l'autenticazione e l'autorizzazione. **Se mi sposto un po' più in basso, vedo che sta eseguendo un'istruzione SQL.  
**