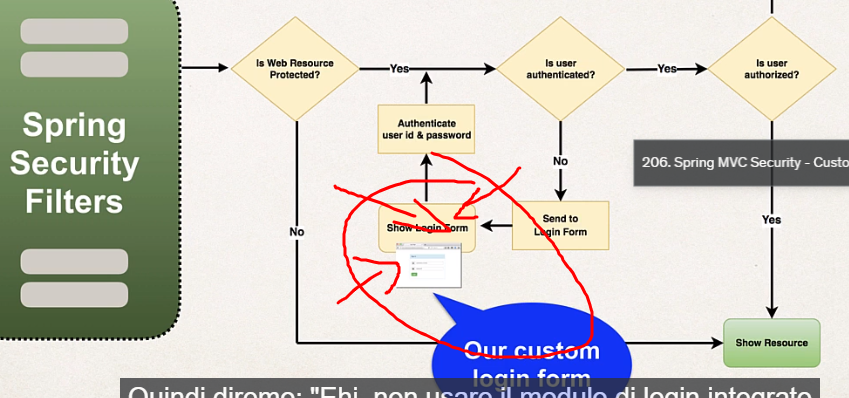
  
  
Realizzata la nostra home page,   
- **procediamo** ad inserire una classe di configurazione @Configuration

dentro la quale aggiungeremo le passwor i ruoli degli utenti.  
  
Le password di Spring Security sono memorizzate in un formato specifico. Tra le parentesi graffe, si indica l'id effettivo e poi la password codificata.  
  
Questa classe sarà cosi composta: Quindi creeremo un metodo di gestione dei dettagli dell’utente che restituisce un InMemory…..  


Quindi abbiamo realizzato correttamente l’accesso con questi dati al nostro sito.  
  
Ora il nostro obiettivo è dare come pagina di login, il nostro template, per cui quello che dobbiamo fare è andare ad intercettare quando viene mostrato il login di accesso di spring. Spring Security, avete il vostro browser web e state cercando di accedere a una risorsa web protetta. E ricordate che abbiamo dei piccoli filtri di Spring Security che in realtà effettuano una sorta di pre-processamento e -processamento.

In pratica utilizzano la configurazione delle loro app e riutilizzano ID, password e rotoli e gestiscono l'intero processo di sicurezza con Spring. E quello che faremo con i filtri di Spring Security è impostare una configurazione. Quindi, quando si accede e occorre inviare l'utente al modulo di login, configureremo Spring in modo da utilizzare il nostro modulo di login personalizzato. Quindi diremo: "Ehi, non usare il modulo di login integrato o il modulo di login predefinito, usa il nostro modulo di login personalizzato". Bene, ecco il processo di sviluppo di base. Quindi la prima cosa da fare è modificare la configurazione

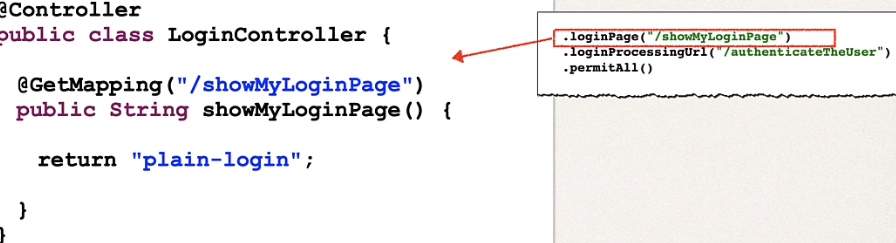
di Spring Security per fare riferimento al nostro form di login, sviluppare un controllore per mostrare il form di login personalizzato e infine creare il form di login personalizzato.

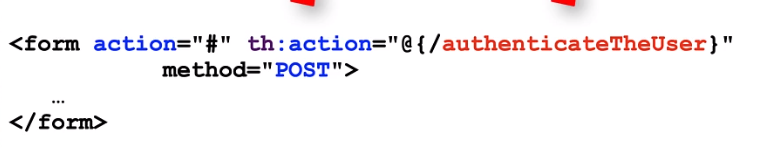
  
Lo facciamo andando a creare bean specifco nella nostra classe  
  
  
il nostro bean filtro di sicurezza. Passerà in una sicurezza HP che possiamo utilizzare. All'interno di questo metodo, ho il punto HTTP che autorizza la richiesta HTTP. Quindi diremo dot any request dot authenticated significa che ogni richiesta che arriva all'applicazione deve essere autenticata. L'utente deve essere connesso. E per il nostro modulo di login, personalizzeremo il processo di login. Per quanto riguarda la pagina di login vera e propria, mostreremo il nostro modulo personalizzato per questa determinata richiesta di mappatura di **/showMyLoginPage**. Quindi il modulo di accesso e invierà i dati o a questo URL per l'elaborazione. Quindi faremo in modo che sia **/authenticateTheUser**.

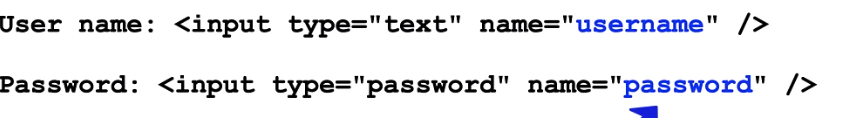
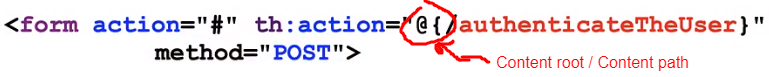
Qui Spring Security controllerà l'ID utente e la password. Ora, a queste due voci si può dare qualsiasi valore per questa configurazione (**loginPage e loginProcessingUrl**). È sufficiente essere coerenti nell'applicazione.

E qui in basso vogliamo permettere a tutti di vedere la pagina di login (**permitAll()**).

Non è necessario effettuare il login, giusto? Perché sai che prima devono vedere qualcosa, vero? Quindi lasciate che chiunque veda la pagina di accesso. E questo è praticamente tutto. Ecco come si configura Spring Security per fare riferimento al modulo di login personalizzato.

Quindi PASSIAMO AL CODICE ORA!  
  
andiamo a creare il nostro controller per fornire il **loginPage** tramite l’esposizione di un end-point, servizio che ritorna una pagina html  
  
  
  
Quindi poi successivamente invieremo i dati all’url di elaborazione del login e ricordate, questo si basa sulle informazioni di configurazione che abbiamo impostato nel file di configurazione di Spring .Quindi, l'invio dei dati all'URL di elaborazione del login, che abbiamo impostato, sarà gestito dai filtri di Spring Security.



dobbiamo inviare i dati. Questo fa parte dei requisiti qui richiesti. Ora, ricordiamo che il motivo per cui lo inviamo ad authenticateTheUser è che è basato sulla configurazione che abbiamo impostato nelle slide precedenti. Ora, Spring Security ha definito alcuni nomi predefiniti per i campi del modulo di login. Bene, quando si crea il modulo per il campo nome utente, lo si chiamerà nome utente. Per il campo della password, si chiamerà password. Queste sono le impostazioni predefinite.  
  
 Quindi, quando si crea il modulo, è sufficiente impostare un input di tipo text, name uguale a username. Per il tipo di immissione della password, il nome è uguale alla password. Quindi Spring Security leggerà i dati del modulo e li userà per autenticare l'utente. Quindi Spring farà tutte le operazioni di autenticazione e verifica in background, purché si forniscano i nomi dei campi del modulo che Spring si aspetta. Quindi andiamo avanti e mettiamo tutto insieme.   
  
  
  
Context root e context path sono praticamente la stessa cosa. Bene, sentirete i nomi passare

in diversi modi. Alcuni dicono percorso contestuale, altri dicono radice contestuale, ma in pratica sono la stessa cosa. D'accordo, cosa ci dà questo? In pratica, questo ci dà accesso al percorso del contesto in modo dinamico. Così possiamo includerlo al volo per qualsiasi link che possiamo avere nella nostra applicazione, ad esempio per l'invio di moduli, link HREF e così via. Possiamo aggiungerlo dinamicamente. Perché utilizzare il percorso contestuale? Come ho detto prima, ci consente di fare riferimento dinamicamente al percorso del contesto dell'applicazione, aiutandoci a mantenere i collegamenti relativi al percorso del contesto dell'applicazione. In questo modo, se si cambia il percorso del contesto dell'applicazione, i collegamenti funzioneranno comunque, il che è molto meglio che codificare il percorso del contesto, e questo approccio è una best practice del settore. Pertanto, vi consiglio vivamente di utilizzarlo nelle vostre applicazioni.