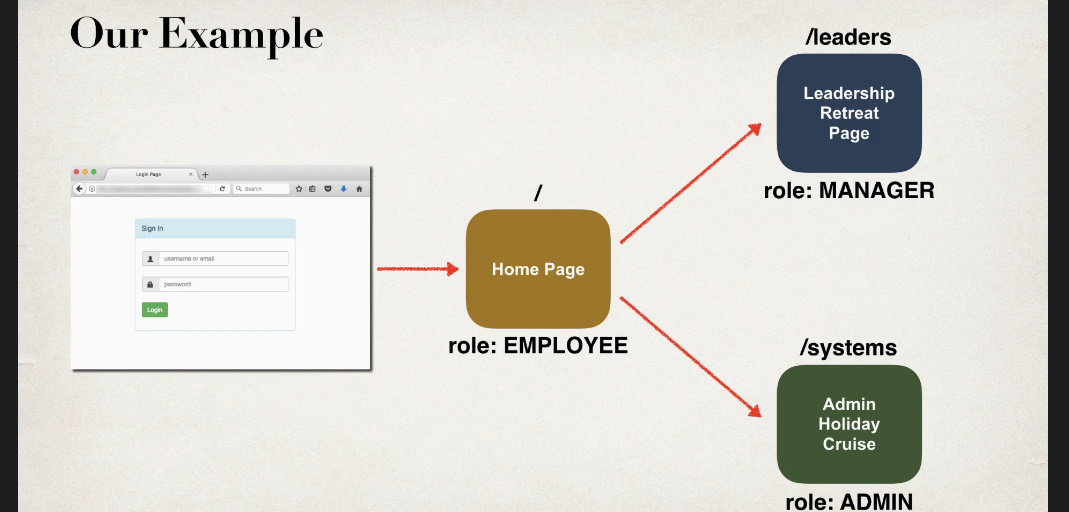
**Restrict Acces**

* 

Crem pagini la care doar anumite roluri au acces.

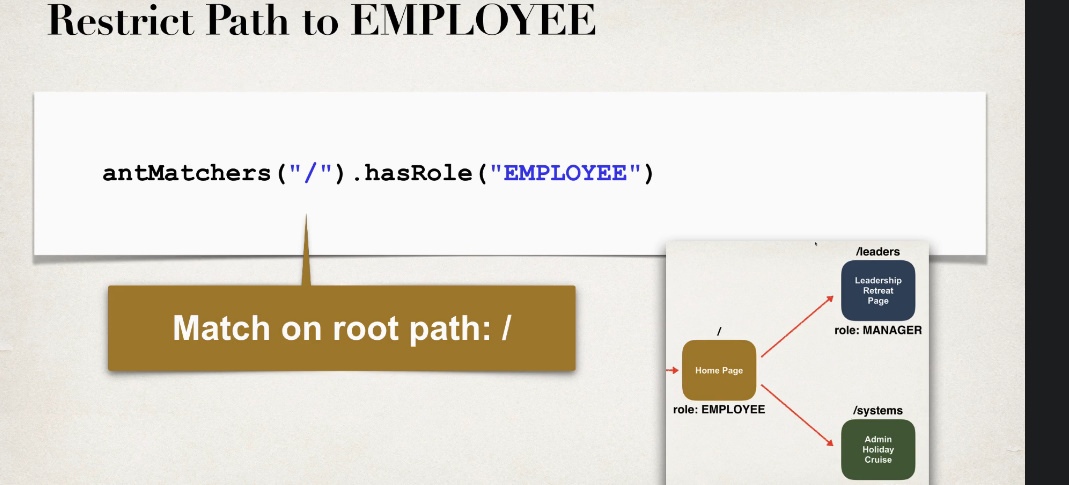
**Pasi**

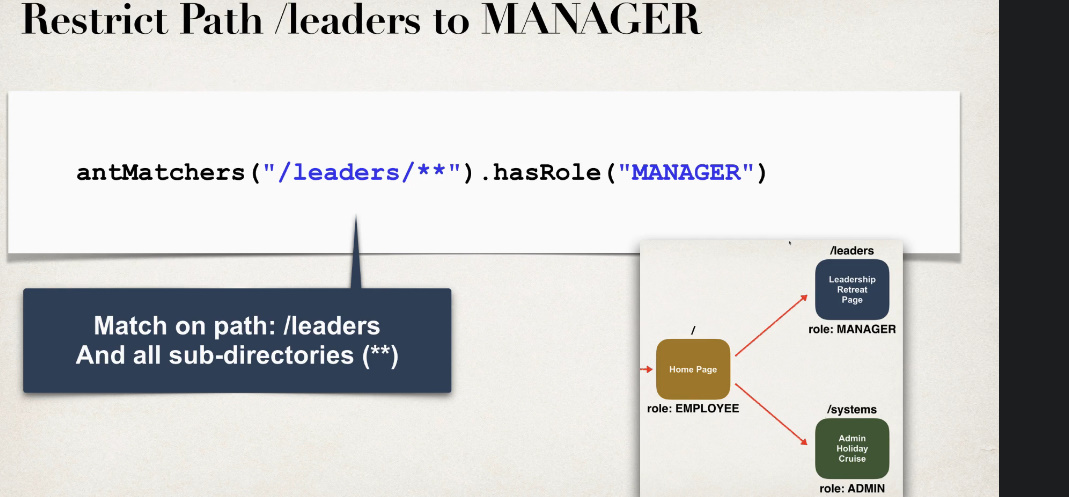
1. Cream requesturi pentru pagini in Controller la MVC
2. Restrictionam requesturile pentru roluri:



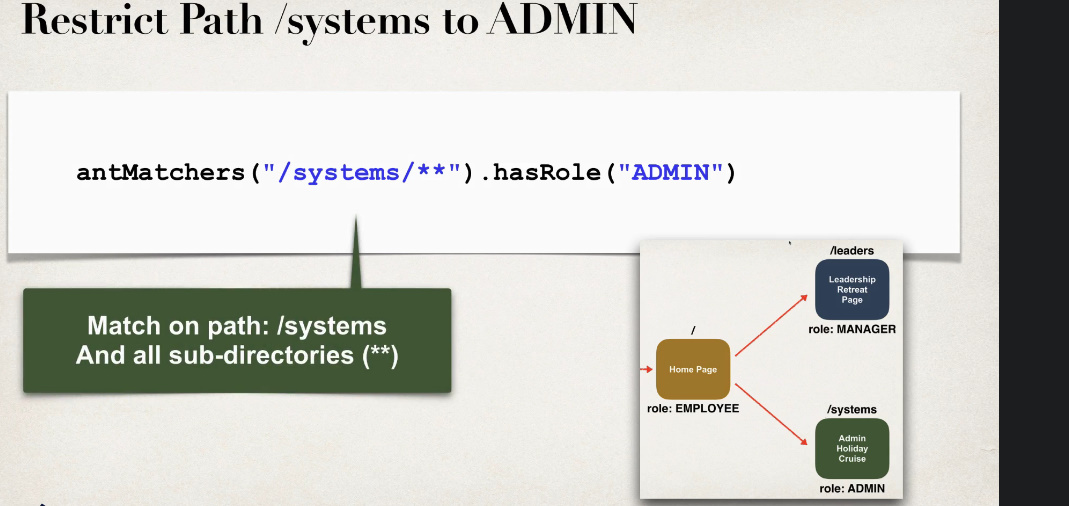
.antMatchers(“/request1”,”/request2”,..,).hasRole(“rol1”)– nu putem pune oricate roluri vrem, ci doar unul, dar putem pune mai multe requests

De ex:

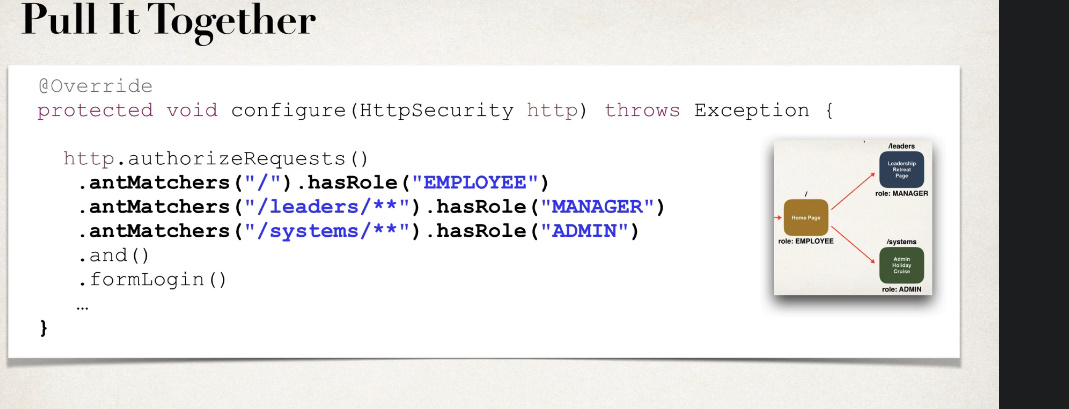




Aici spunem ca orice request care incepe cu **/leaders** va fi disponibil doar pentru MANAGER



Deci, va arata asa:



.anyRequest().authenticated()

asta trebuie sters, asa cum nu mai vrem ca orice request sa fie disponibil pentru cei autentificati, dar sa alegem noi care requesturi si pentru cine.

**Atentie!**

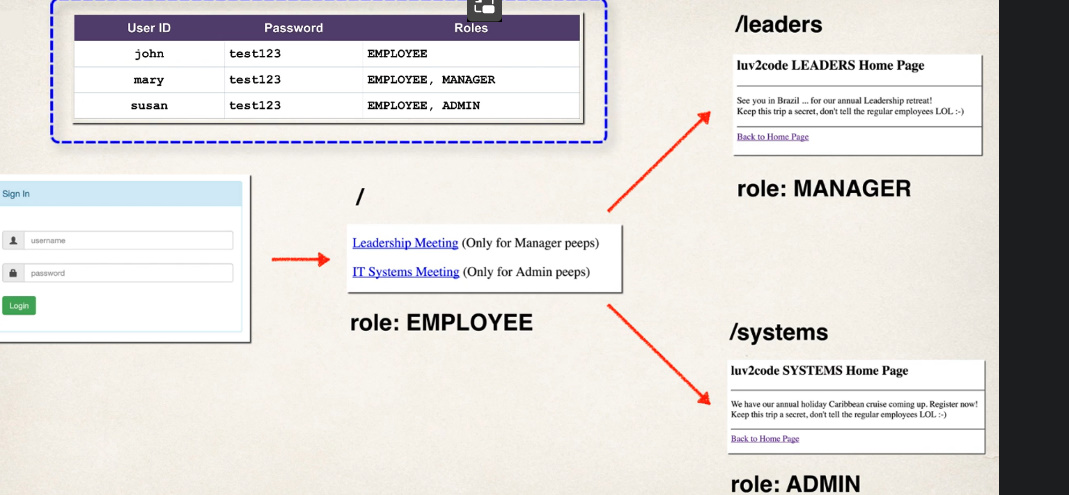
**Putem lasa si anyRequest().authenticated() daca vrem ca sa nu poata face niciun request cat nu e logat, dar eaa trebuie mereu sa fie la urma!!!**

**Asta e din cauza ca daca ea e prima, mereu se va executa prima si va returna true, si la celelalte nici nu se va ajunge, dar Spring nici nu ne va lasa sa rulam aplicatie, va da o eroare:**

[Can't configure antMatchers after anyRequest](https://stackoverflow.com/questions/62654582/nested-exception-is-java-lang-illegalstateexception-cant-configure-antmatchers)

hasRole() returneaza obiectul http, de aceea nu e necesar and().

1. Adaugam linkurile catre pagini in pagina index



Avem grija cand dam roluri la useri, caci daca e Admin, trebuie deci sa fie si Manager, si EMPLOYEE

auth.inMemoryAuthentication().withUser(userBuilder.username("kerbecs").password("test123").roles("EMPLOYEE","MANAGER","ADMIN"))  
 .withUser(userBuilder.username("Alin").password("test123").roles("EMPLOYEE","MANAGER"))  
 .withUser(userBuilder.username("Alex").password("test123").roles("EMPLOYEE"));  
}