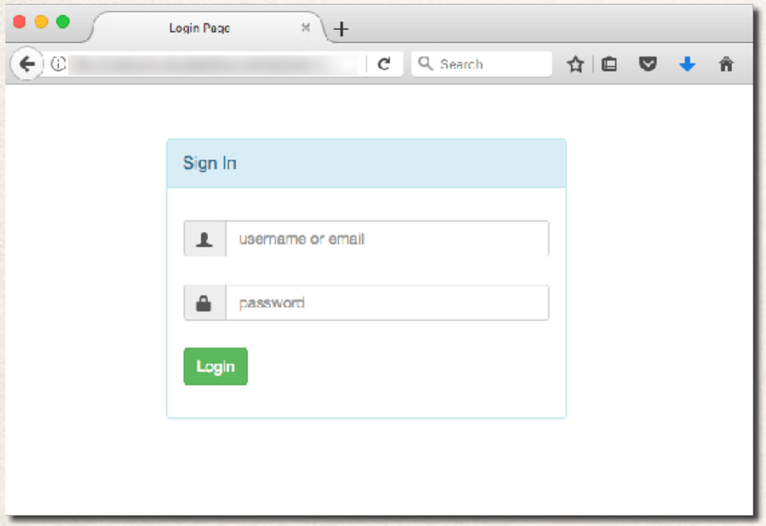
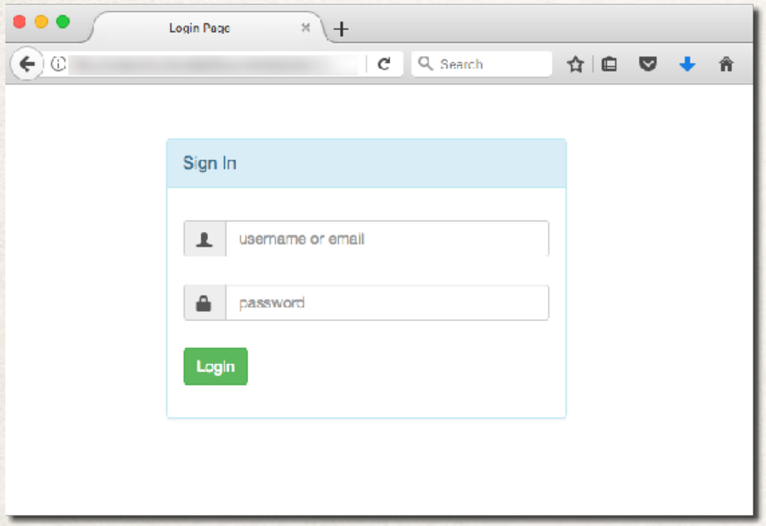
# Restrict Access Based on Roles

# Our Example

# Our Example



# Our Example

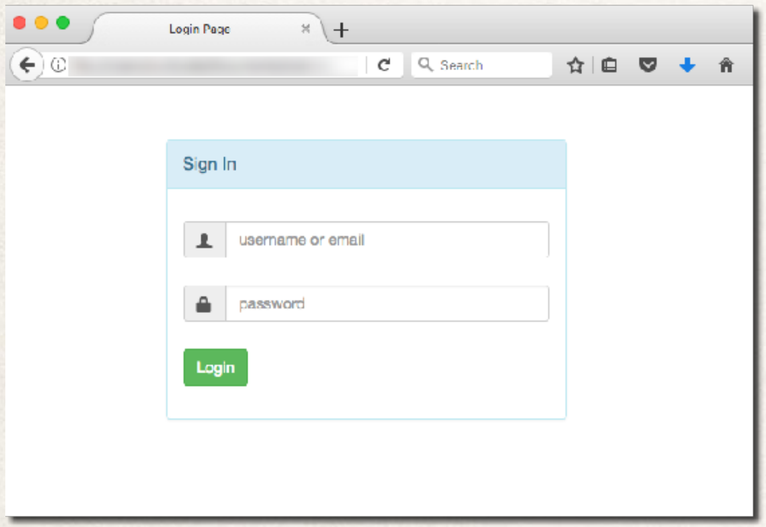



/

Home Page

role: EMPLOYEE

# Our Example




/

Home Page

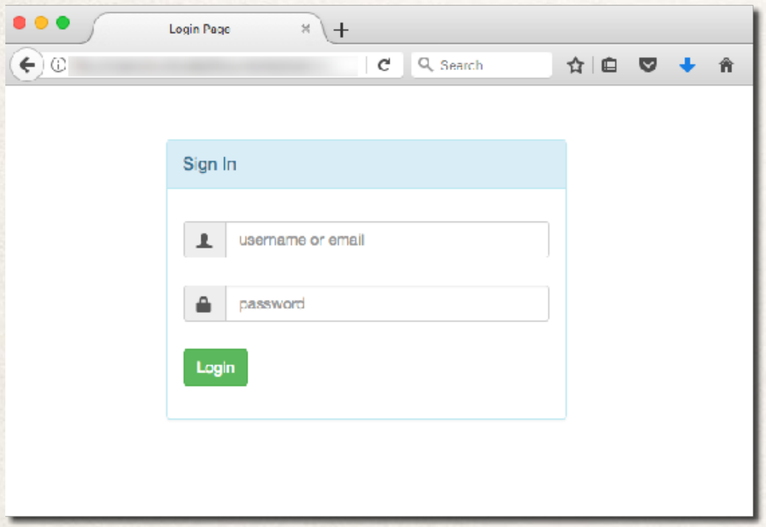
role: EMPLOYEE

/leaders

Leadership Retreat Page

role: MANAGER

# Our Example




/

Home Page

role: EMPLOYEE

/leaders

Leadership Retreat Page

role: MANAGER

/systems

Admin Holiday Cruise

role: ADMIN

# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Create supporting controller code and view pages

# Development Process

Step-By-Step

1. Create supporting controller code and view pages

2. Update user roles

# Development Process

Step-By-Step

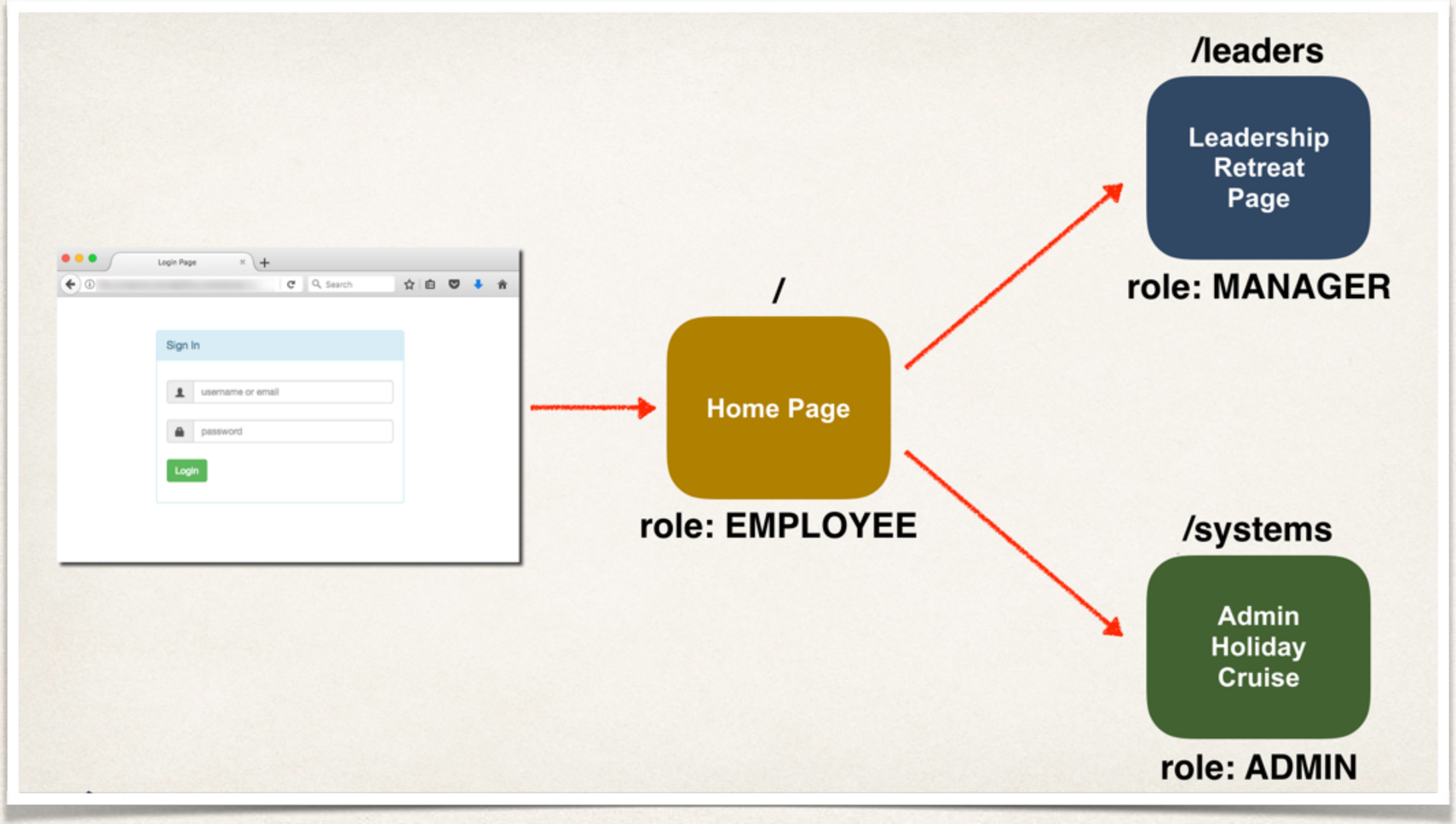
1. Create supporting controller code and view pages

2. Update user roles

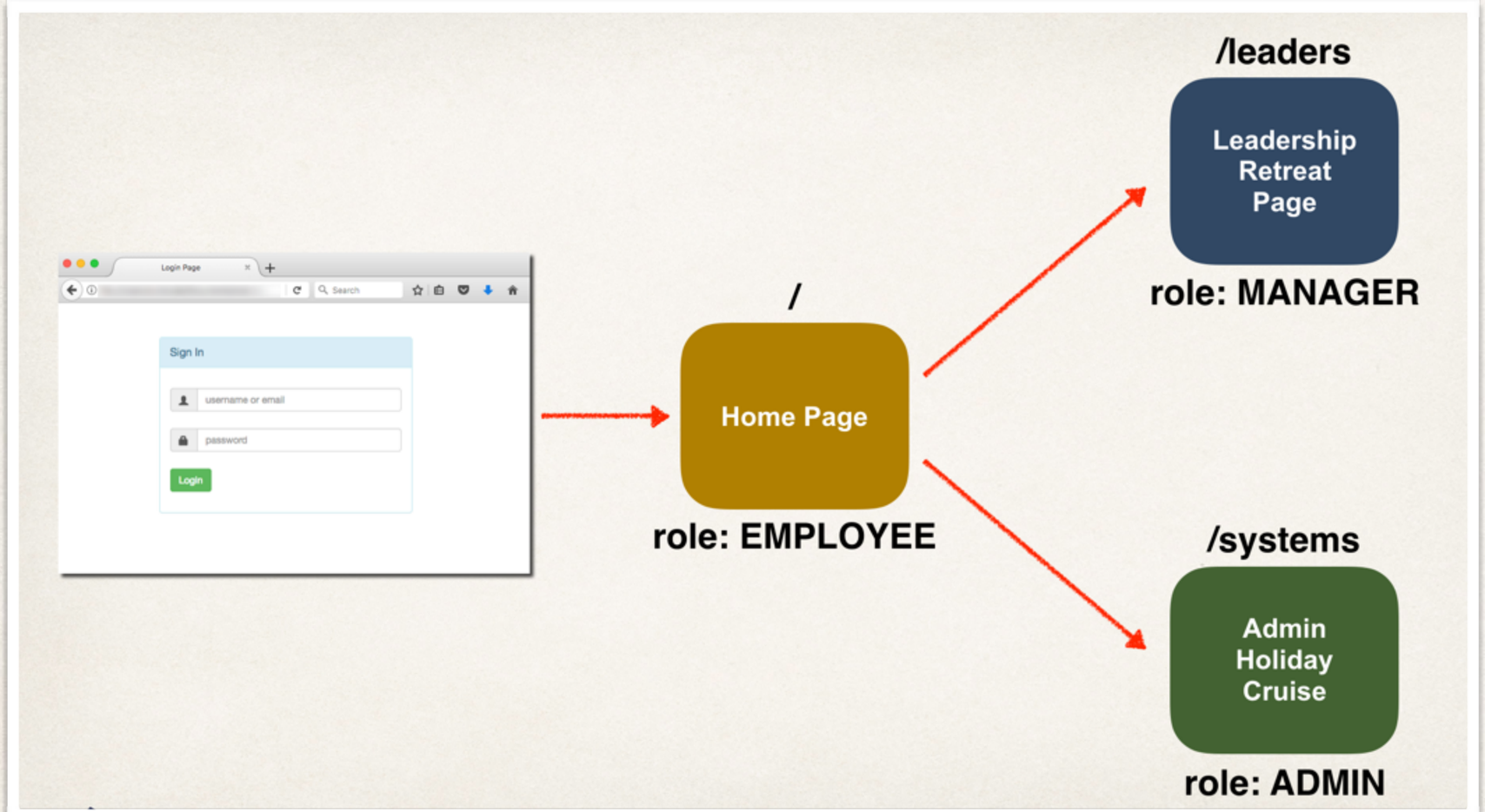
3. Restrict Access based on Roles

# Step 1: Create Supporting Controller code and View Pages

# Step 1: Create Supporting Controller code and View Pages



# Step 1: Create Supporting Controller code and View Pages




Cover these steps in the video

# Step 2: Update Our User Roles

# Step 2: Update Our User Roles

| User ID | Password | Roles |
| --- | --- | --- |

# Step 2: Update Our User Roles

| User ID | Password | Roles |
|---|---|---|
| `john` | `test123` | `EMPLOYEE` |

# Step 2: Update Our User Roles

| User ID | Password | Roles |
|---|---|---|
| john | test123 | EMPLOYEE |
| mary | test123 | EMPLOYEE, MANAGER |

# Step 2: Update Our User Roles

| User ID | Password | Roles |
|---|---|---|
| john | test123 | EMPLOYEE |
| mary | test123 | EMPLOYEE, MANAGER |
| susan | test123 | EMPLOYEE, ADMIN |

# Step 2: Update Our User Roles

| User ID | Password | Roles |
|---|---|---|
| john | test123 | EMPLOYEE |
| mary | test123 | EMPLOYEE, MANAGER |
| susan | test123 | EMPLOYEE, ADMIN |

We can give ANY names for user roles

# Step 2: Update Our User Roles

**File: DemoSecurityConfig.java**

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {

  // add our users for in memory authentication





}
```

# Step 2: Update Our User Roles

| User ID | Password | Roles |
|---|---|---|
| john | test123 | EMPLOYEE |
| mary | test123 | EMPLOYEE, MANAGER |
| susan | test123 | EMPLOYEE, ADMIN |

**File: DemoSecurityConfig.java**

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {

  // add our users for in memory authentication





}
```

# Step 2: Update Our User Roles

| User ID | Password | Roles |
|---|---|---|
| john | test123 | EMPLOYEE |
| mary | test123 | EMPLOYEE, MANAGER |
| susan | test123 | EMPLOYEE, ADMIN |

**File: DemoSecurityConfig.java**

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {

  // add our users for in memory authentication

  UserBuilder users = User.withDefaultPasswordEncoder();

  auth.inMemoryAuthentication()
    .withUser(users.username("john").password("test123").roles("EMPLOYEE"))
    .withUser(users.username("mary").password("test123").roles("EMPLOYEE", "MANAGER"))
    .withUser(users.username("susan").password("test123").roles("EMPLOYEE", "ADMIN"));

}
```

# Step 2: Update Our User Roles

| User ID | Password | Roles |
|---|---|---|
| john | test123 | EMPLOYEE |
| mary | test123 | EMPLOYEE, MANAGER |
| susan | test123 | EMPLOYEE, ADMIN |

**File: DemoSecurityConfig.java**

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {

  // add our users for in memory authentication

  UserBuilder users = User.withDefaultPasswordEncoder();

  auth.inMemoryAuthentication()
    .withUser(users.username("john").password("test123").roles("EMPLOYEE"))
    .withUser(users.username("mary").password("test123").roles("EMPLOYEE", "MANAGER"))
    .withUser(users.username("susan").password("test123").roles("EMPLOYEE", "ADMIN"));

}
```

# Step 3: Restricting Access to Roles

# Step 3: Restricting Access to Roles

- Update your Spring Security Java configuration file (.java)

# Step 3: Restricting Access to Roles

- Update your Spring Security Java configuration file (.java)
- General Syntax

# Step 3: Restricting Access to Roles

- Update your Spring Security Java configuration file (.java)
- General Syntax

```
antMatchers(<< add path to match on >>).hasRole(<< authorized role >>)
```

# Step 3: Restricting Access to Roles

- Update your Spring Security Java configuration file (.java)
- General Syntax

```
antMatchers(<< add path to match on >>).hasRole(<< authorized role >>)
```

**Restrict access to a given path "/systems/**"**

# Step 3: Restricting Access to Roles

- Update your Spring Security Java configuration file (.java)
- General Syntax

Single role

```
antMatchers(<< add path to match on >>).hasRole(<< authorized role >>)
```

Restrict access to a given path "/systems/**"

# Step 3: Restricting Access to Roles

- Update your Spring Security Java configuration file (.java)
- General Syntax

Single role

```
antMatchers(<< add path to match on >>).hasRole(<< authorized role >>)
```

Restrict access to a given path "/systems/**"

"ADMIN"

# Step 3: Restricting Access to Roles

# Step 3: Restricting Access to Roles

```
antMatchers(<< add path to match on >>).hasAnyRole(<< list of authorized roles >>)
```

# Step 3: Restricting Access to Roles

Any role in the list, comma-delimited list

```
antMatchers(<< add path to match on >>).hasAnyRole(<< list of authorized roles >>)
```

# Step 3: Restricting Access to Roles

Any role in the list, comma-delimited list

```
antMatchers(<< add path to match on >>).hasAnyRole(<< list of authorized roles >>)
```

"ADMIN", "DEVELOPER", "VIP", "PLATINUM"

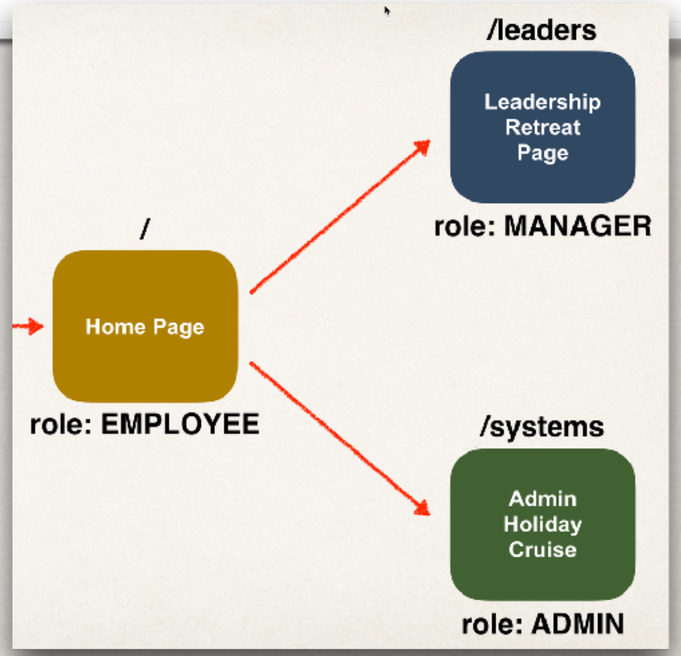
# Restrict Path to EMPLOYEE

# Restrict Path to EMPLOYEE

```
antMatchers("/").hasRole("EMPLOYEE")
```

# Restrict Path to EMPLOYEE

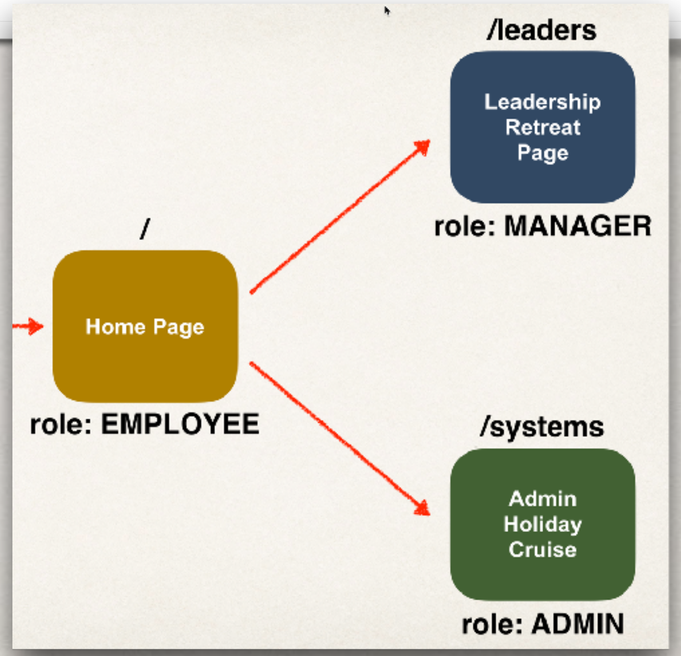
```
antMatchers("/").hasRole("EMPLOYEE")
```



# Restrict Path to EMPLOYEE

```
antMatchers("/").hasRole("EMPLOYEE")
```

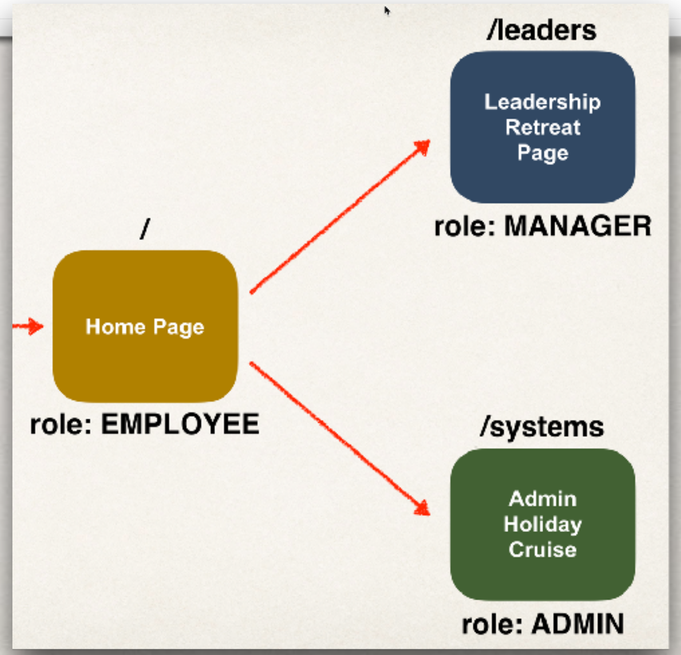
**Match on root path: /**



# Restrict Path /leaders to MANAGER

# Restrict Path /leaders to MANAGER

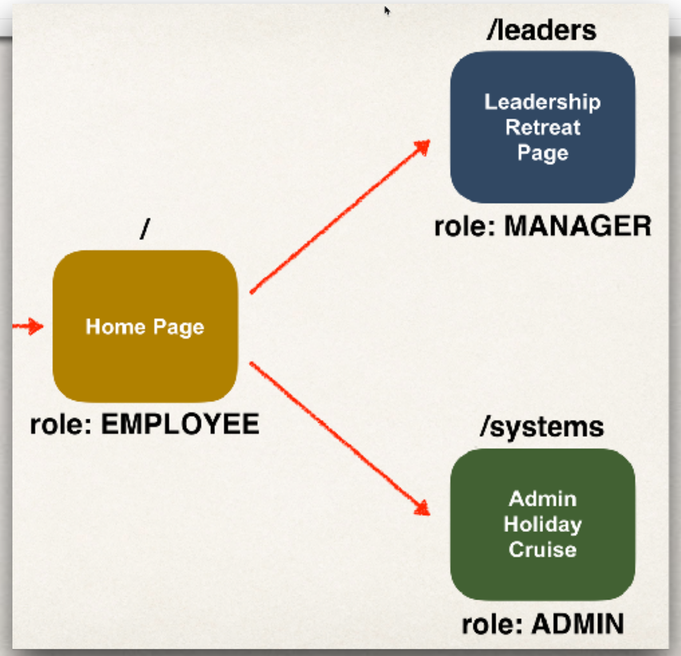
```
antMatchers("/leaders/**").hasRole("MANAGER")
```



# Restrict Path /leaders to MANAGER

```
antMatchers("/leaders/**").hasRole("MANAGER")
```

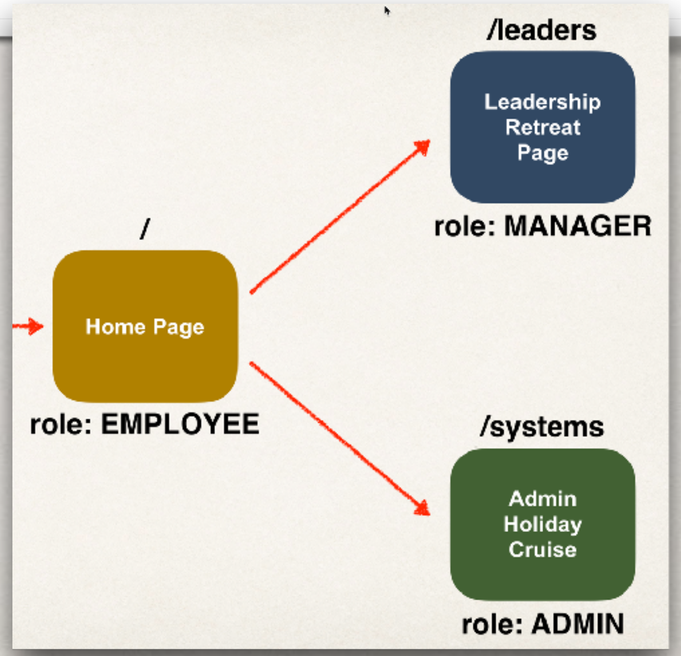
**Match on path: /leaders**
**And all sub-directories (**)**



# Restrict Path /systems to ADMIN

# Restrict Path /systems to ADMIN

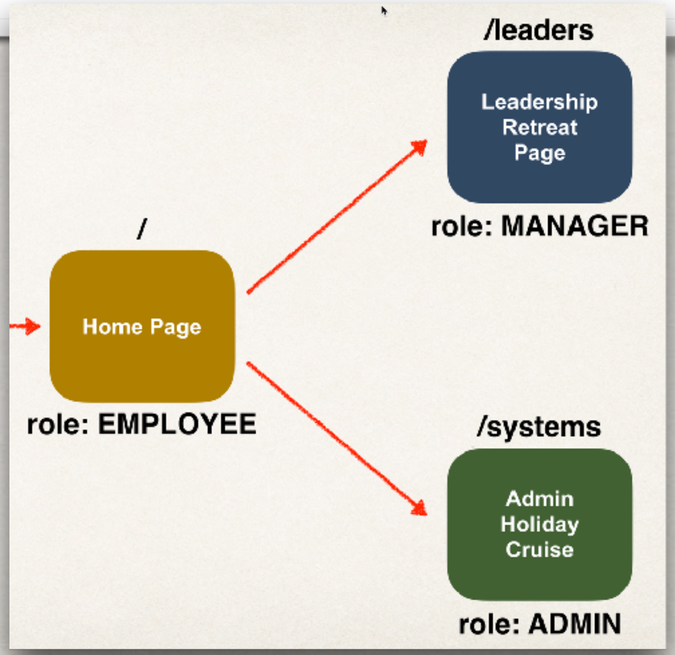
```
antMatchers("/systems/**").hasRole("ADMIN")
```



# Restrict Path /systems to ADMIN

```
antMatchers("/systems/**").hasRole("ADMIN")
```

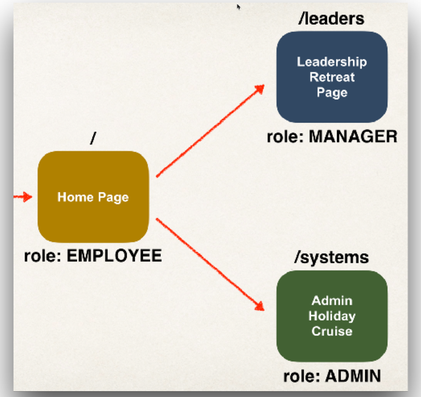
**Match on path: /systems**
**And all sub-directories (**)**



# Pull It Together

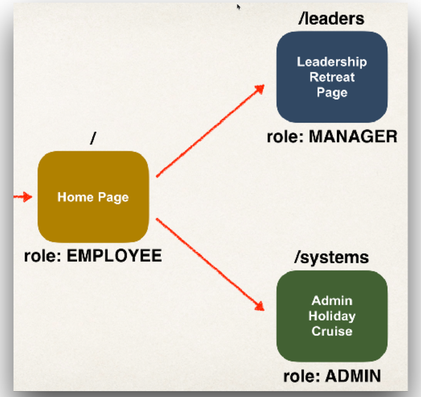
# Pull It Together

```
@Override
protected void configure(HttpSecurity http) throws Exception {

  http.authorizeRequests()
    .antMatchers("/").hasRole("EMPLOYEE")
    .antMatchers("/leaders/**").hasRole("MANAGER")
    .antMatchers("/systems/**").hasRole("ADMIN")
    .and()
    .formLogin()
   …
}
```



# Pull It Together

```
@Override
protected void configure(HttpSecurity http) throws Exception {

  http.authorizeRequests()
    .antMatchers("/").hasRole("EMPLOYEE")
    .antMatchers("/leaders/**").hasRole("MANAGER")
    .antMatchers("/systems/**").hasRole("ADMIN")
    .and()
    .formLogin()
   ...
}
```




```
auth.inMemoryAuthentication()
  .withUser(users.username("john").password("test123").roles("EMPLOYEE"))
  .withUser(users.username("mary").password("test123").roles("EMPLOYEE", "MANAGER"))
  .withUser(users.username("susan").password("test123").roles("EMPLOYEE", "ADMIN"));
```