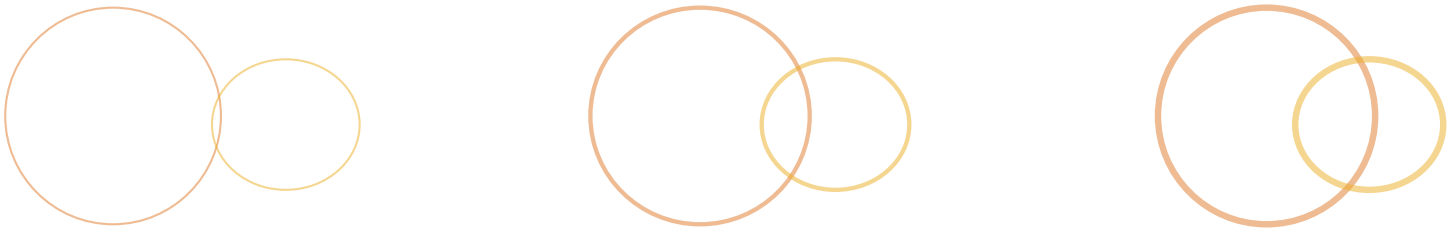
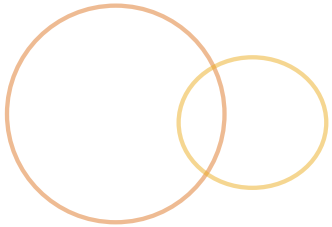
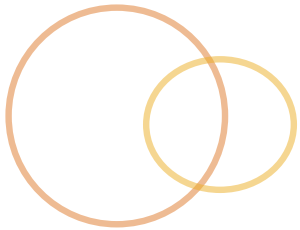


Spring REST Security

Objectives

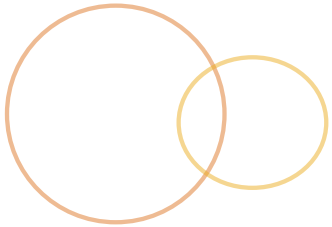


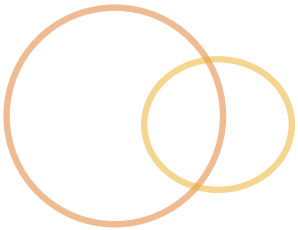


When we are done, you should be able to:

- Explain how security works with a Spring web application

Security





- Who gets access to what, and how do we ensure that
  - Authentication – Are you who you say you are?
  - Authorization – Are you allowed here?
- Many different providers for each of these
- Many different technologies that can be mixed and matched

Spring Security

- We need spring security JAR
  - Security JAR: `spring-security-web` and `spring-security-config`
  - Technology's JAR
    - Spring has several packages of security classes, many of which are specific to particular technologies
- We configure
  - Authentication connection information
    - This is just what is needed for Spring to find it
    - Says little to nothing about how the data is actually sent
  - Authorization information
    - Java is role based authorization
    - Most of configuration is which roles a resource is restricted to

Web Security Configuration

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends
    WebSecurityConfigurerAdapter{


  @Override
  public void configure
   (AuthenticationManagerBuilder amb)throws Exception{
     ...
  }
  @Override
  public void configure(HttpSecurity http)throws Exception{
     ...
  }
}
```

HttpSecurity

```
  @Override
  public void configure(HttpSecurity http)throws Exception{
    http.authorizeRequests().
    antMatchers("/", "/home").permitAll().
    anyRequest().authenticated().and().
    formLogin().loginPage("/login").permitAll().and().
    logout().permitAll();
  }
}
```

- Each step is considered in order
  - The above says that we are using authorization
  - Access to / or /home is allowed to anyone
  - All other requests must be authenticated using the login page defined and they have the ability to logout

Configuration Explanation

- `@EnableWebSecurity`
  - Turns on web security
- `WebSecurityConfigurerAdapter`
  - Contains methods to override if needed for security configuration
    - Authentication managers
    - Authentication configuration
    - Trust resolvers
    - User details
    - Etc.

Configure HttpSecurity

- Method to configure authorization for web pages
  - Includes login pages, logout mechanism and pages for HttpStatus errors

```
  @Override
  public void configure(HttpSecurity http)throws Exception{
    http.authorizeRequests().
    antMatchers("/", "/home").permitAll().
    antMatchers("/admin/**").hasRole("ADMIN").
    anyRequest().authenticated().and().
    formLogin().loginPage("/login").permitAll().and().
    logout().permitAll();
  }
}
```

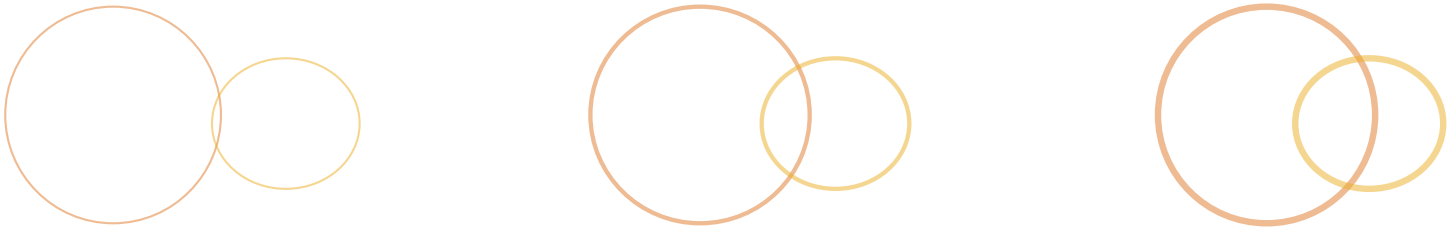
Configuring Authentication Managers

- Method to override if you need to modify or add information to gain access to your authentication manager

```
@Override
public void configure(AuthenticationManagerBuilder auth)
   throws Exception{
     auth.inMemoryAuthentication().
     withUser("person").password("pass").roles("USER").and().
     withUser("administrator").password("aPass").roles("USER",
   "ADMIN");
}
```

Miscellaneous Security Notes

- `@EnableGlobalMethodSecurity`
  - Generally added to a configuration file
    - Allows configuration using expression-based annotations
    - Turns on AOP based security



Lab 7 – Web Security