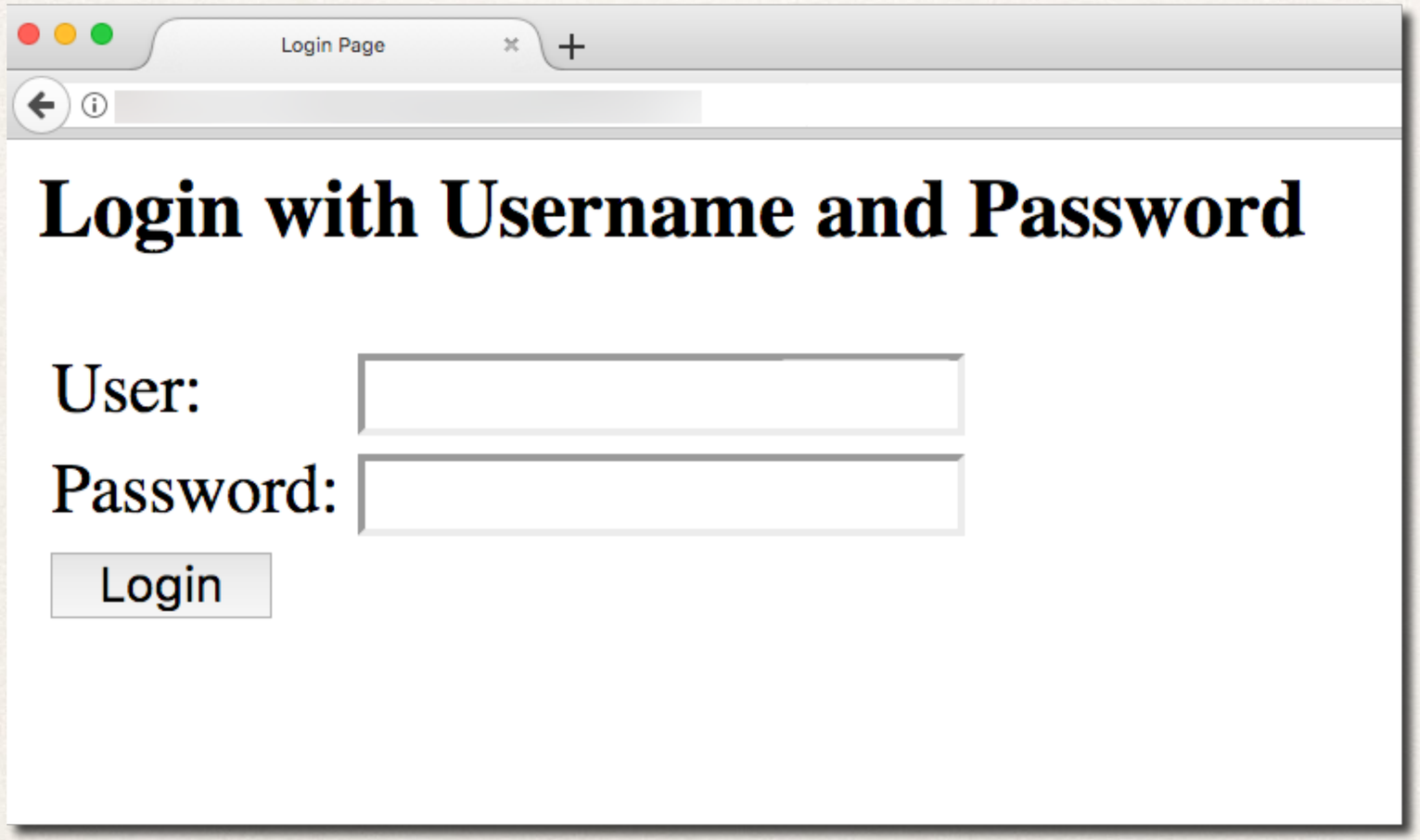
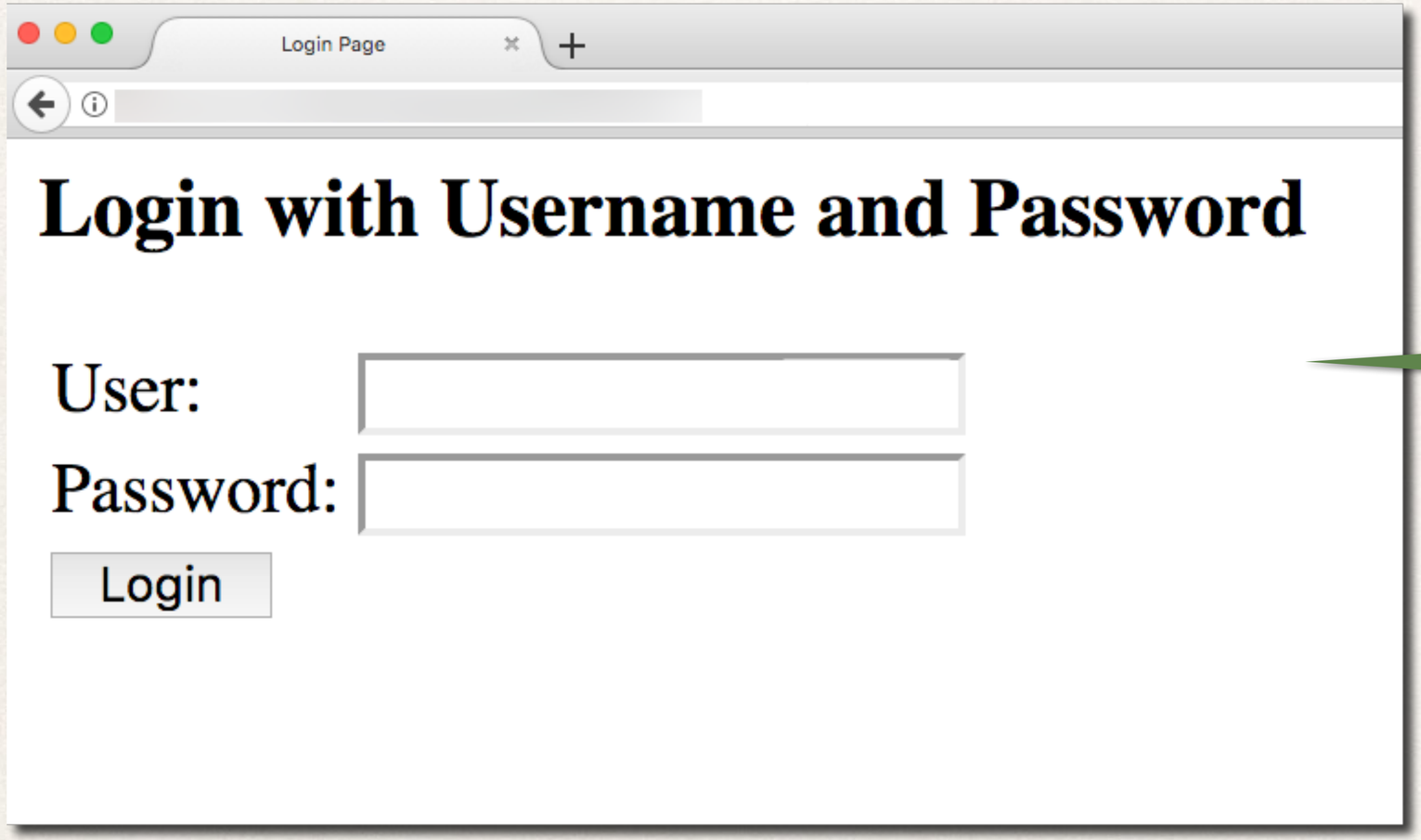
Spring Security - Custom Login Form

Spring Security - Default Login Form

Spring Security - Default Login Form



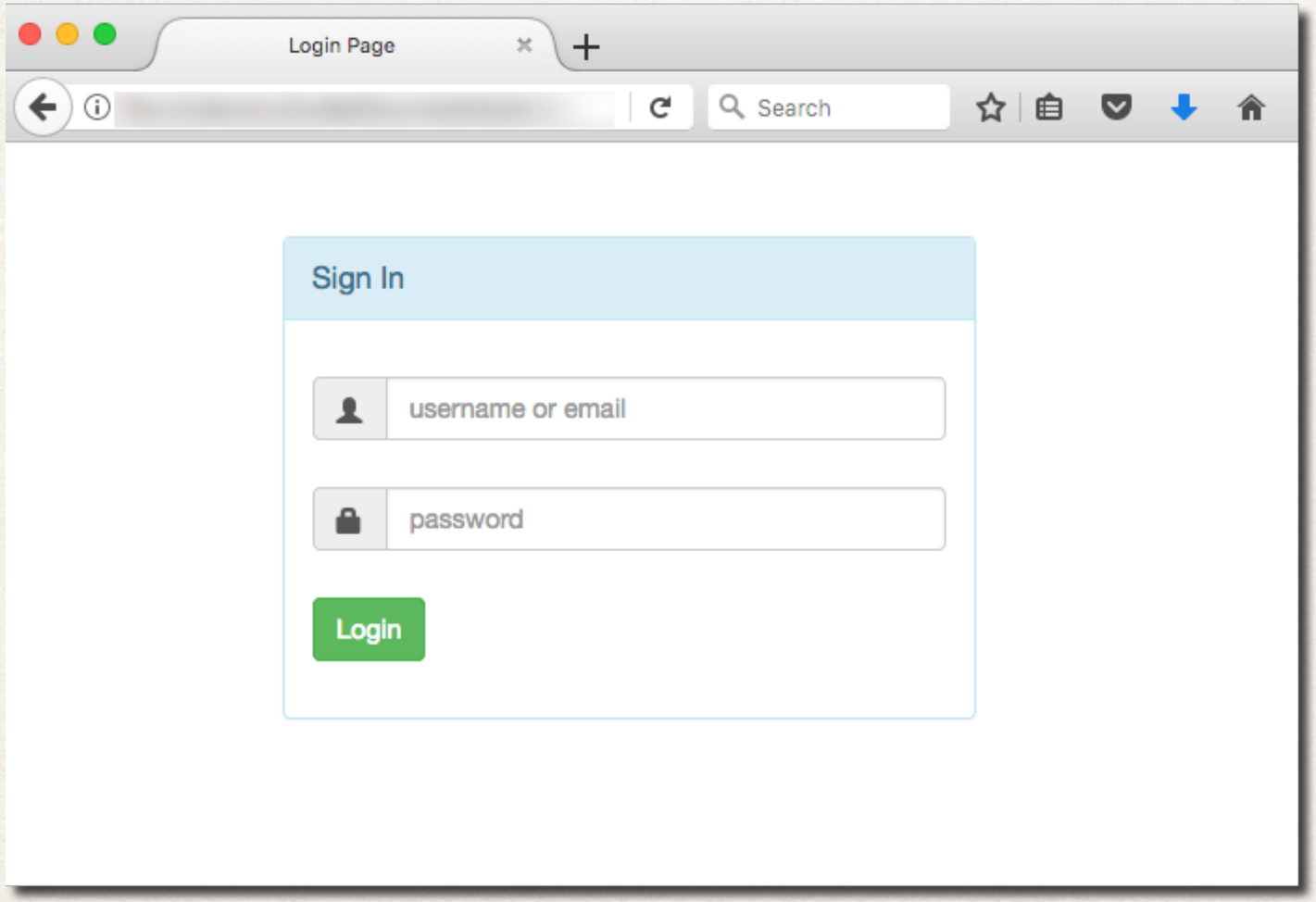
Spring Security - Default Login Form



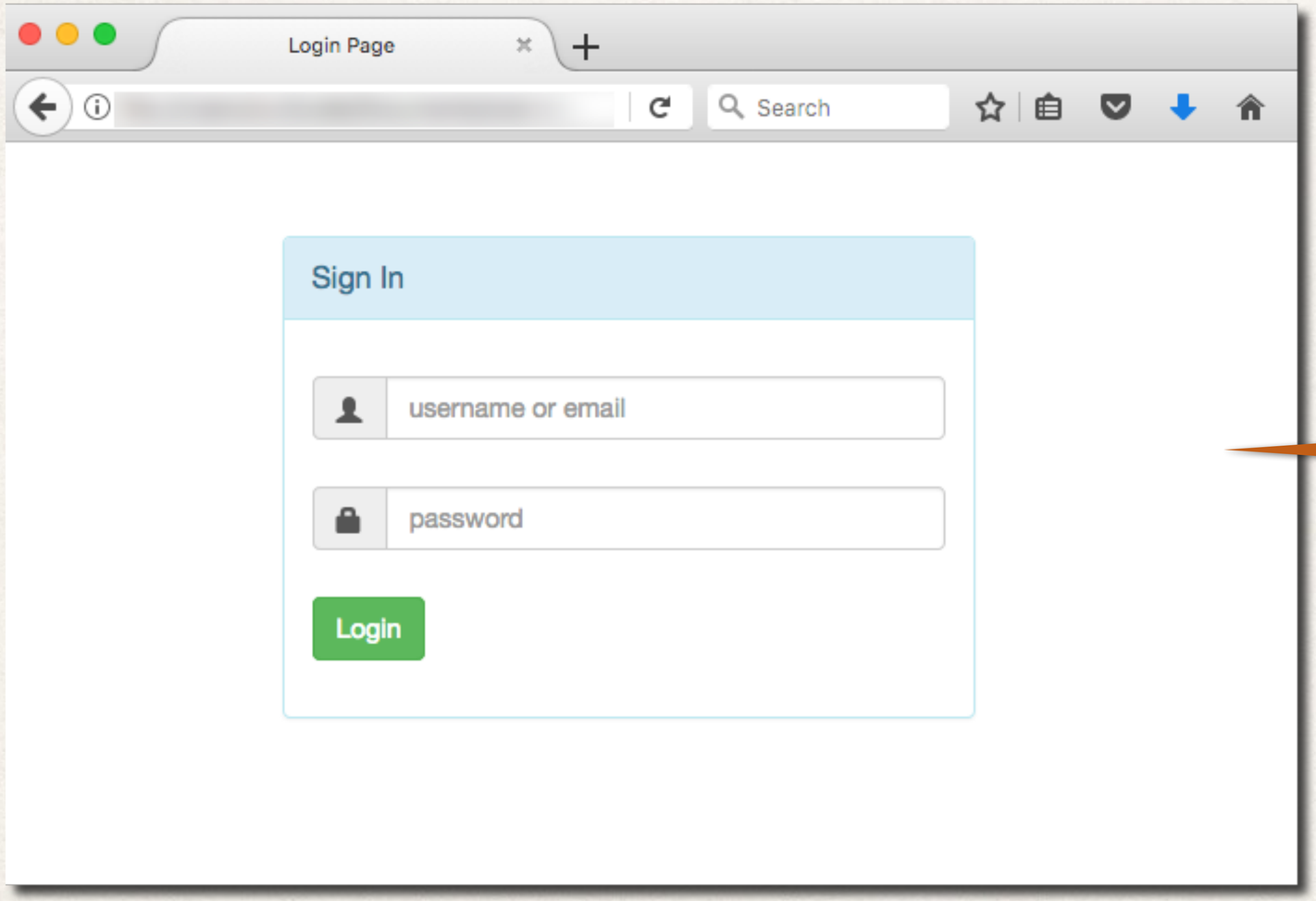
Good for quick start

Your Own Custom Login Form

Your Own Custom Login Form



Your Own Custom Login Form



Your own custom look-and-feel HTML + CSS

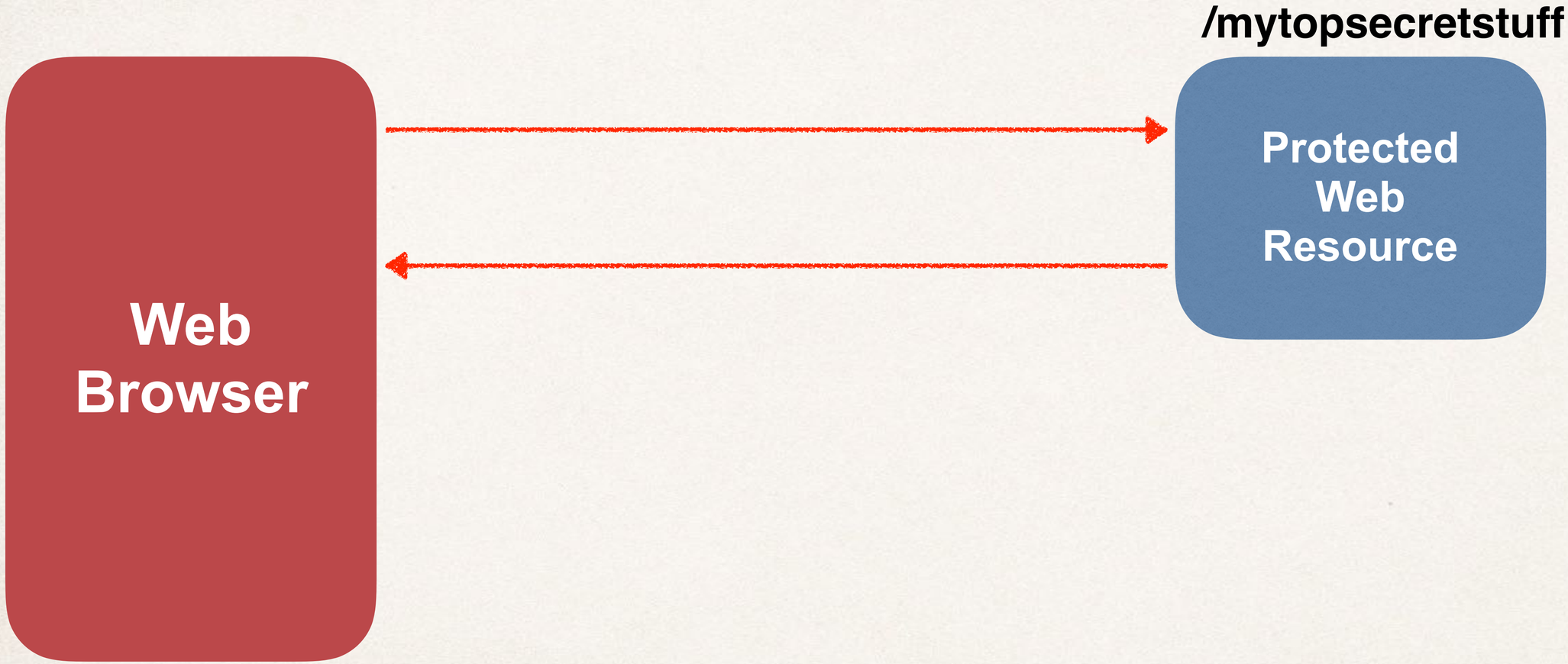
Spring Security Overview

Spring Security Overview

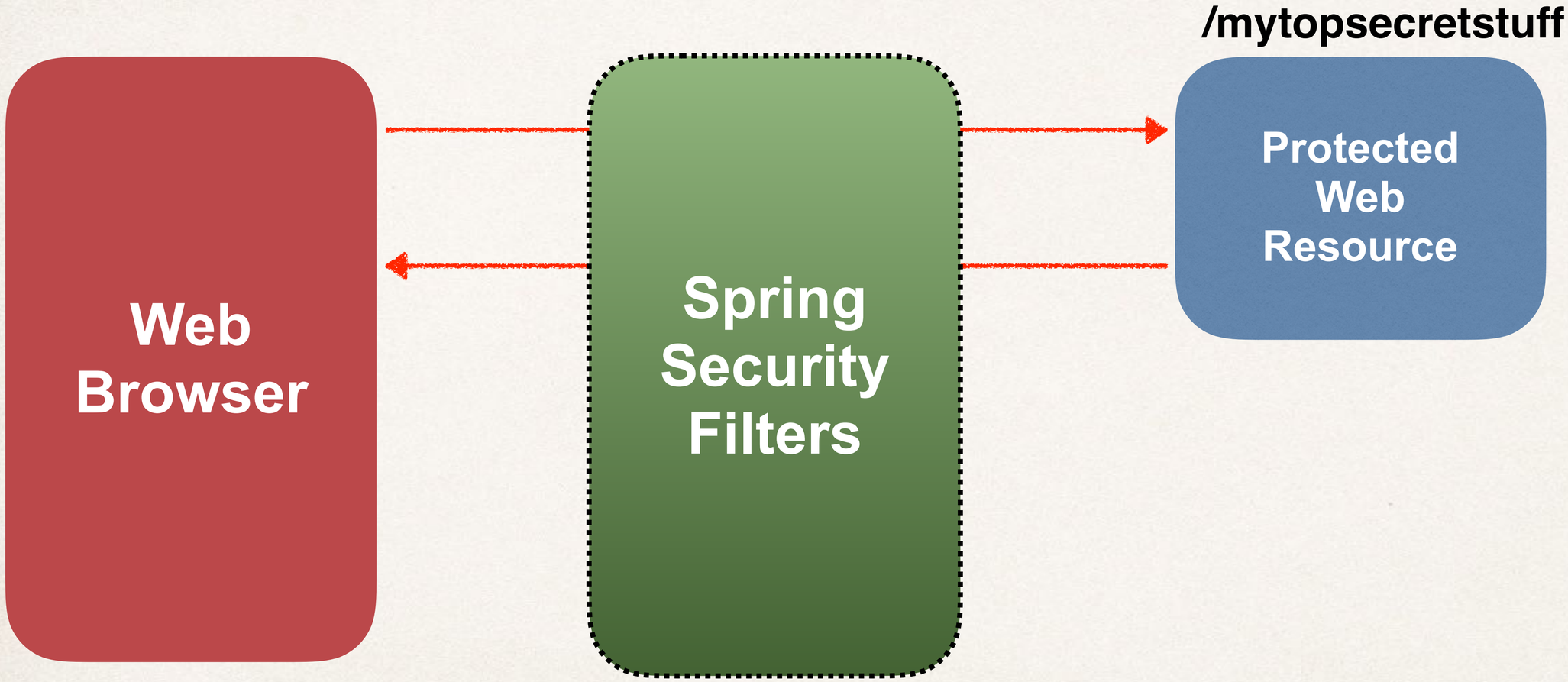
Spring Security Overview

Spring Security Overview

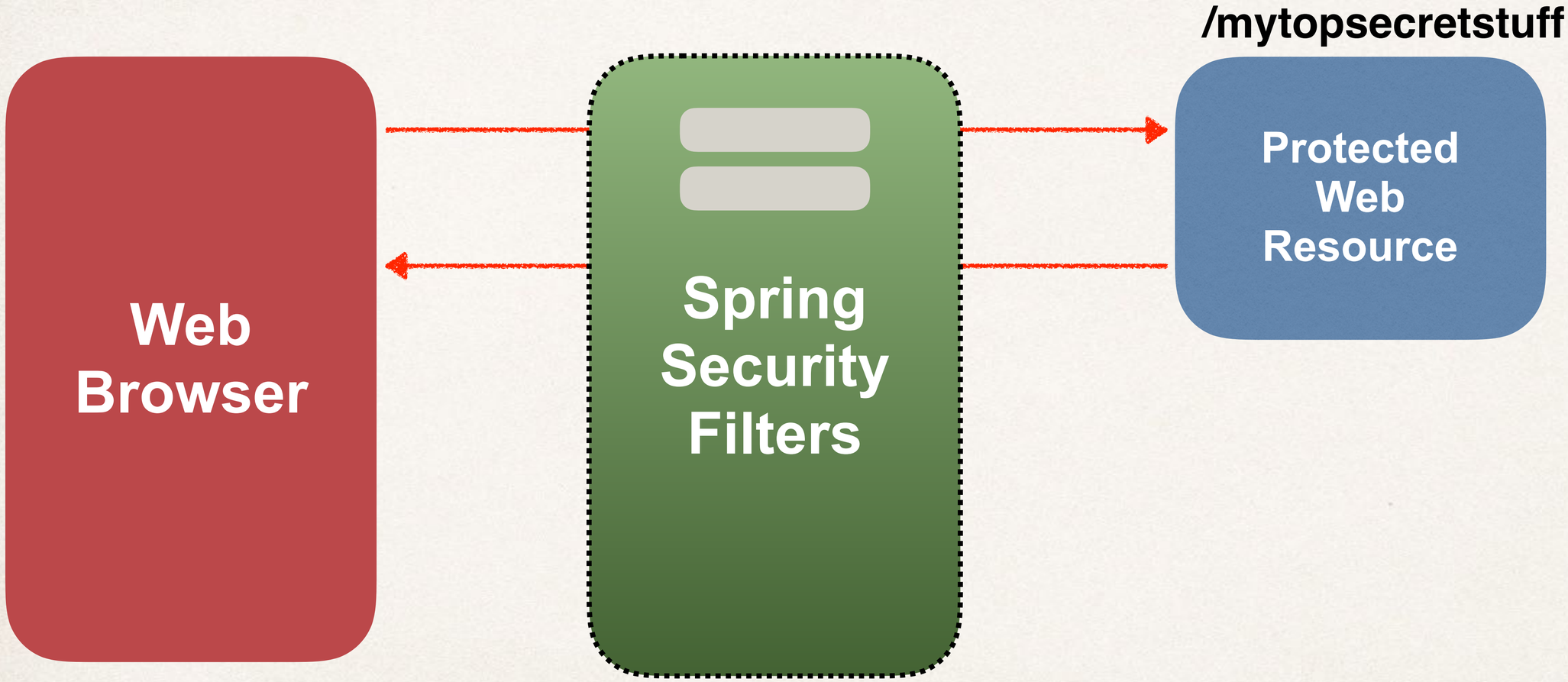
Spring Security Overview



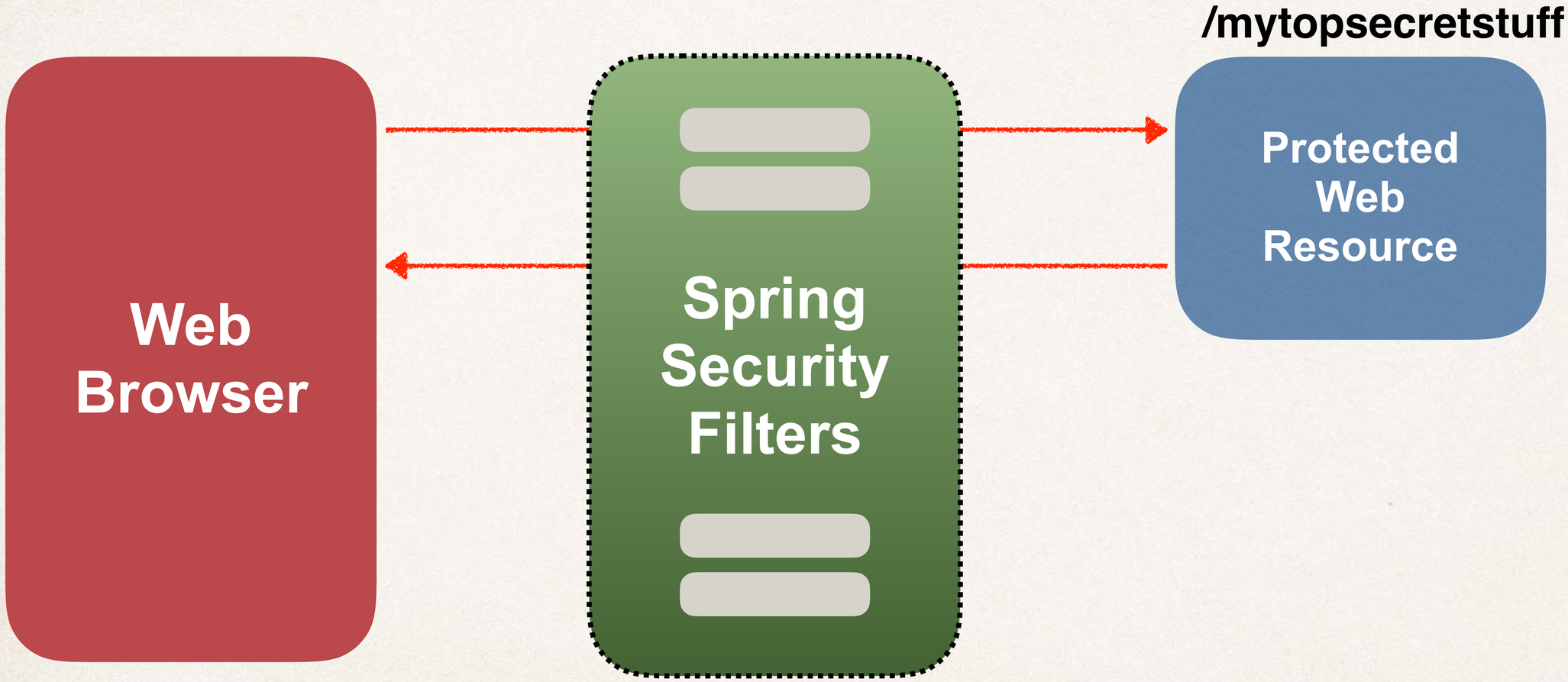
Spring Security Overview



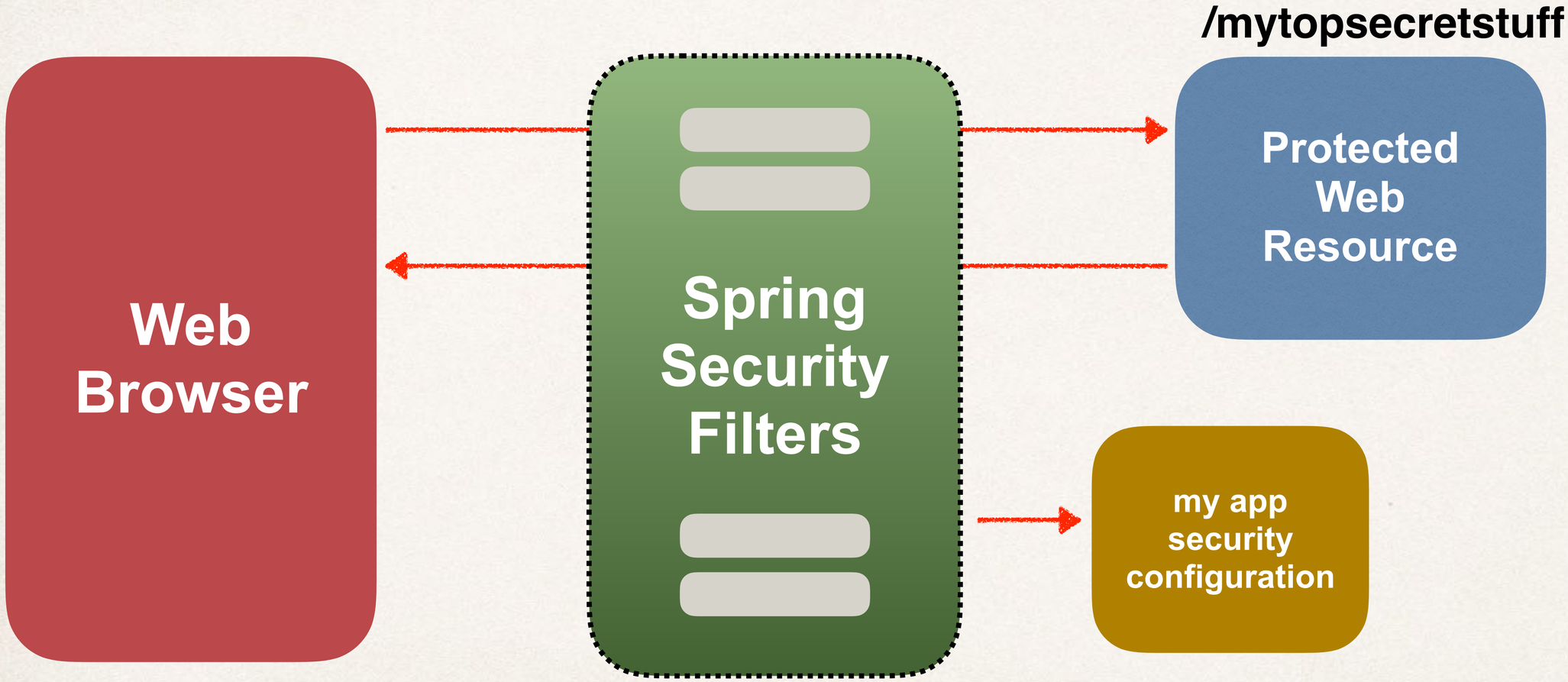
Spring Security Overview



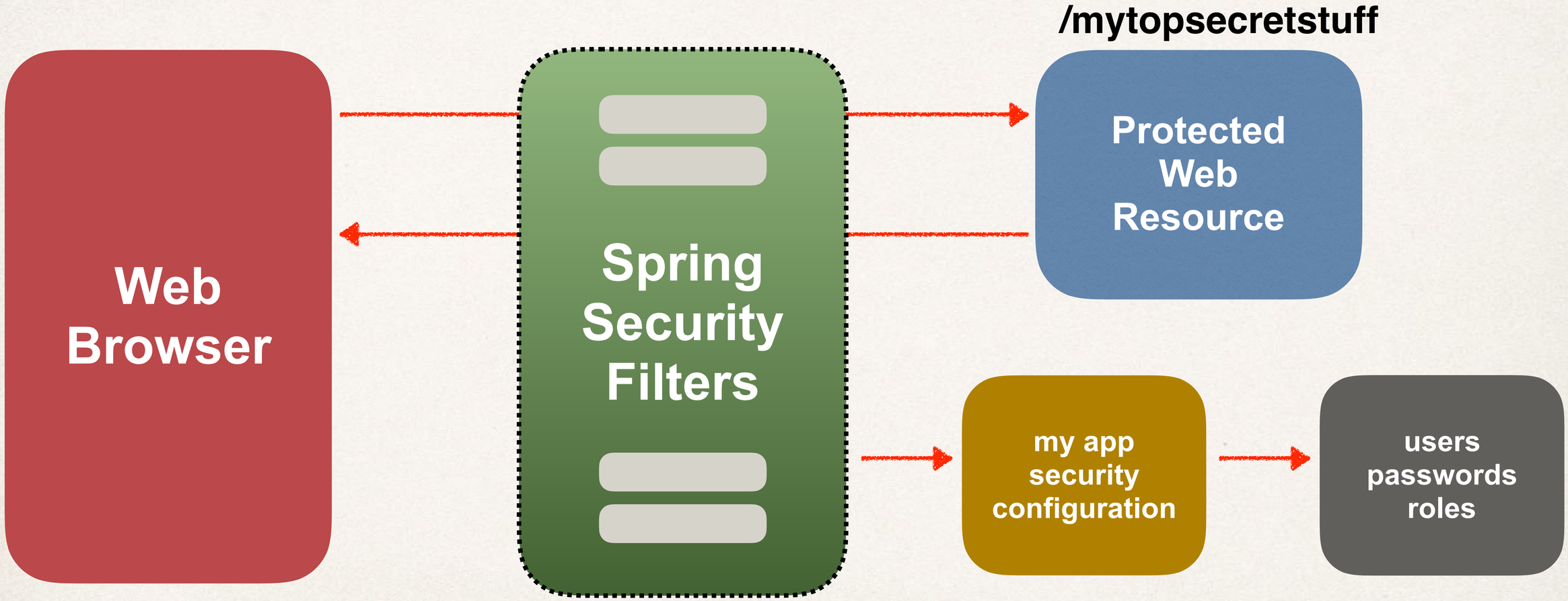
Spring Security Overview



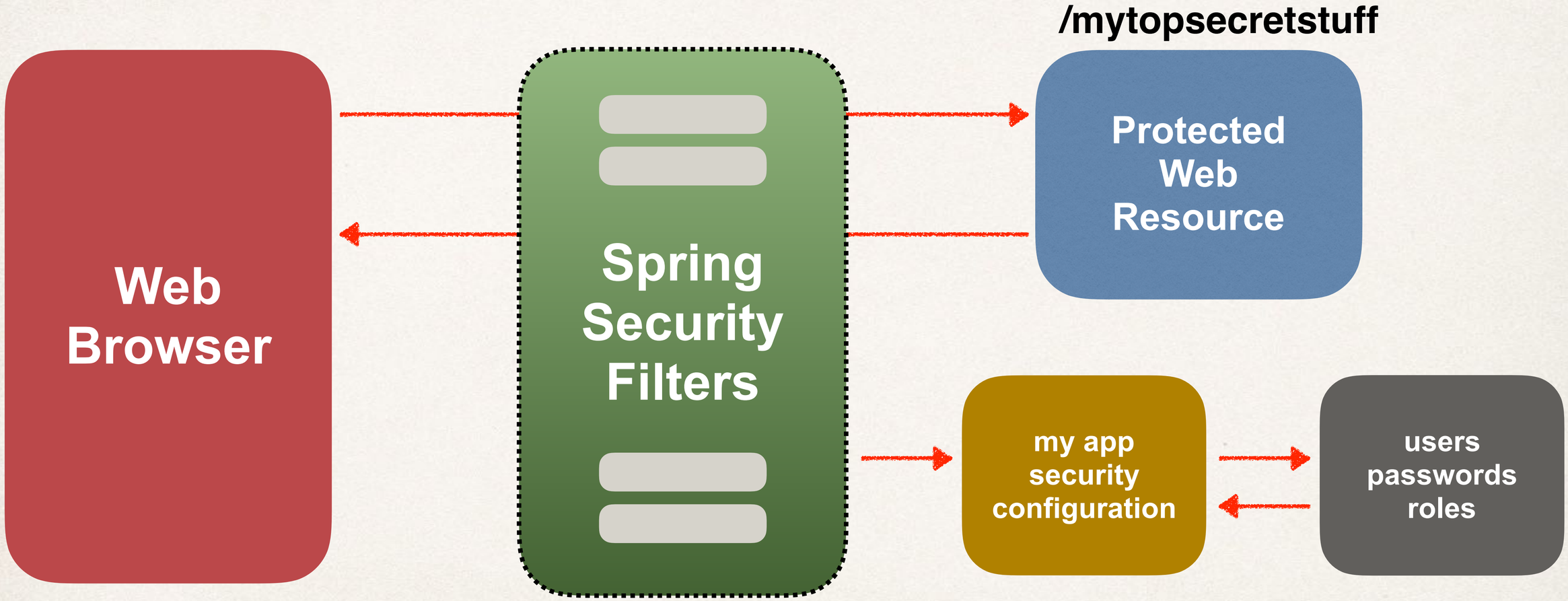
Spring Security Overview



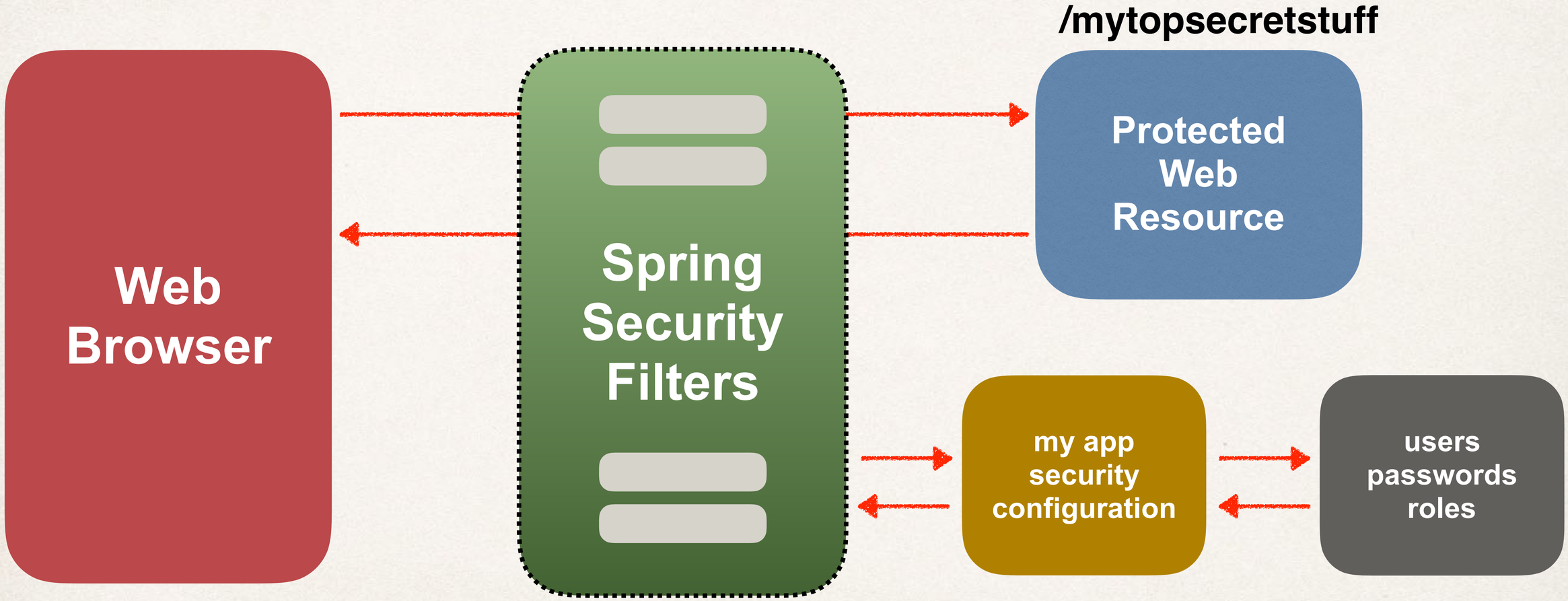
Spring Security Overview



Spring Security Overview

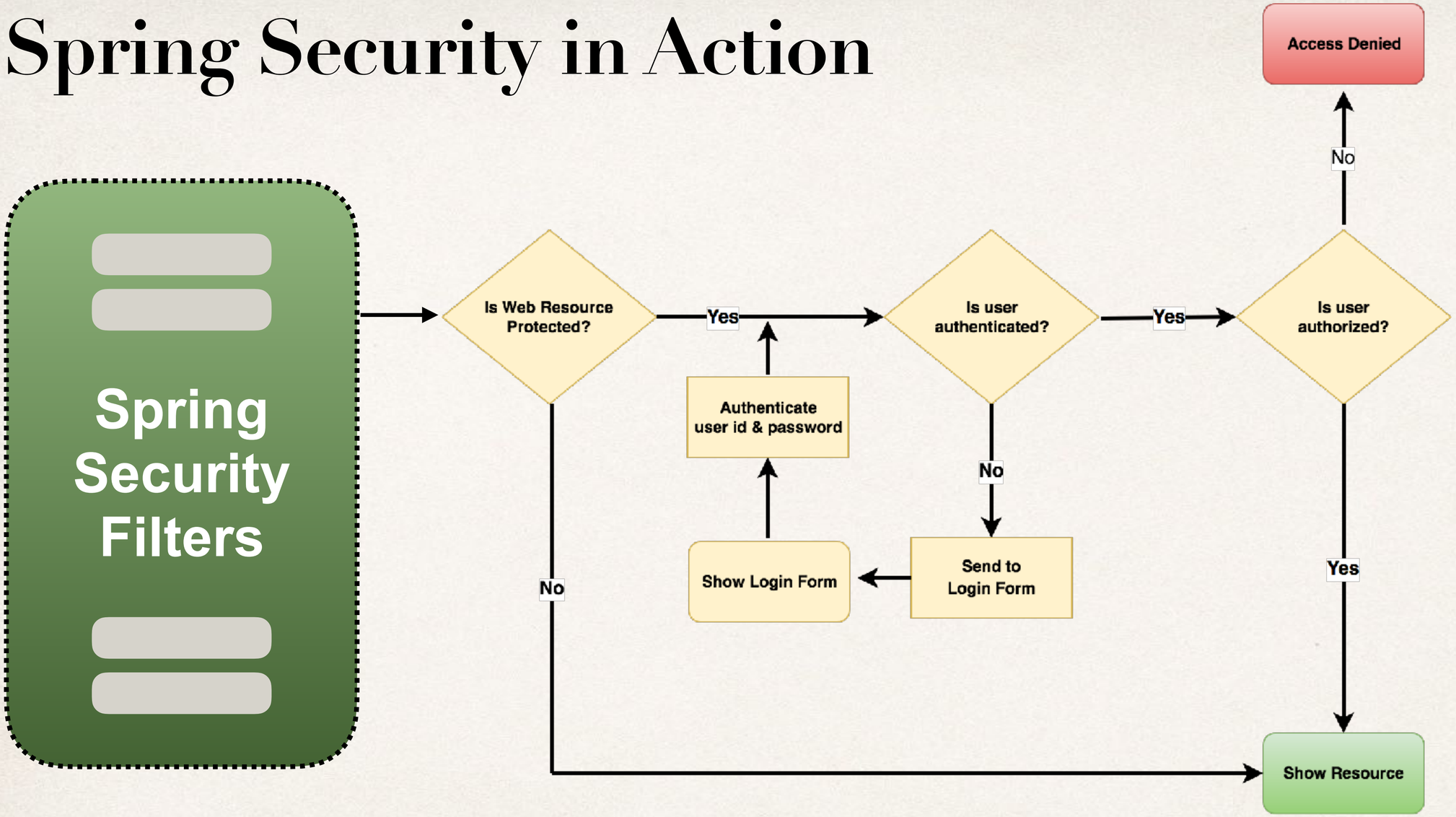


Spring Security Overview

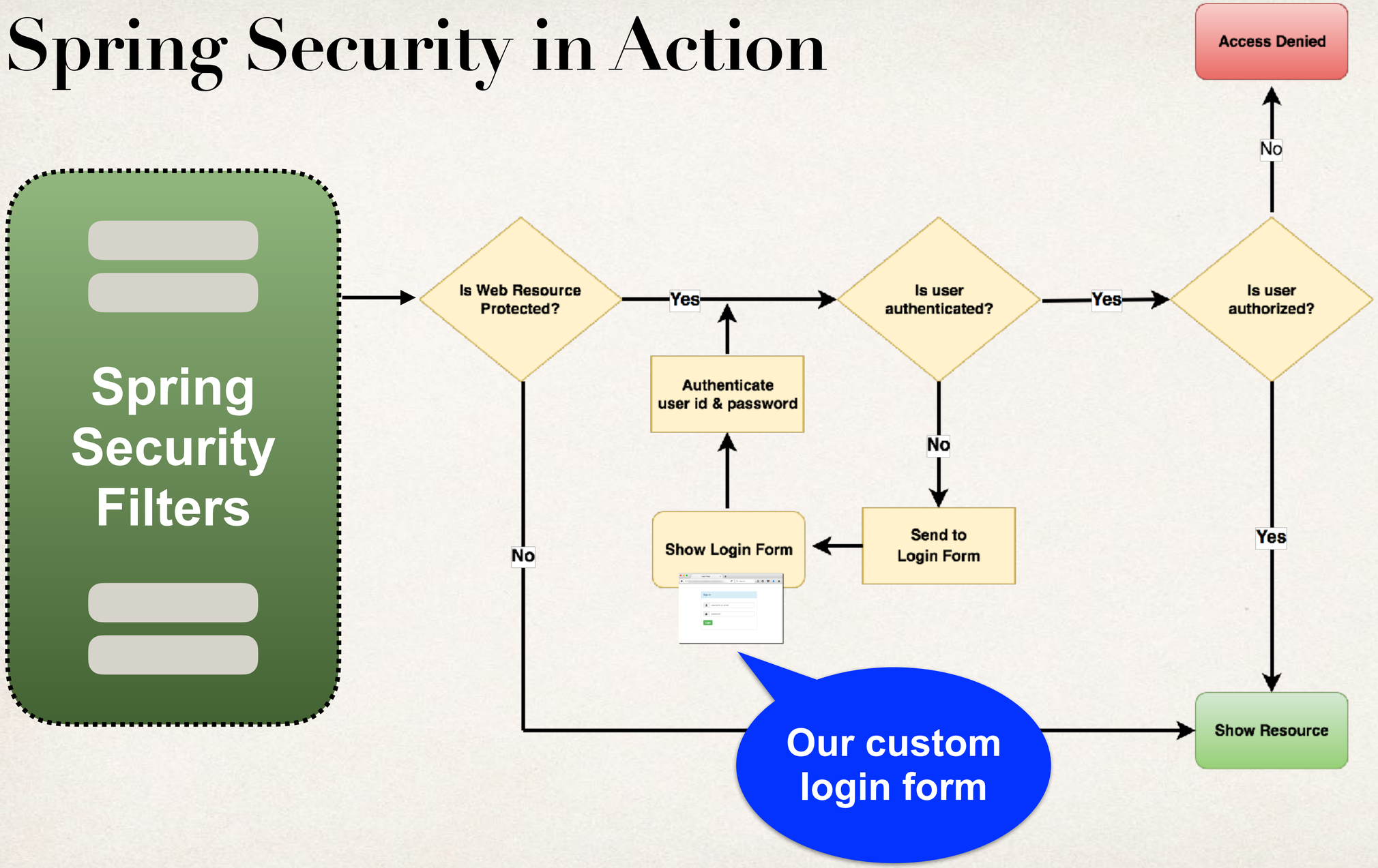


Spring Security in Action

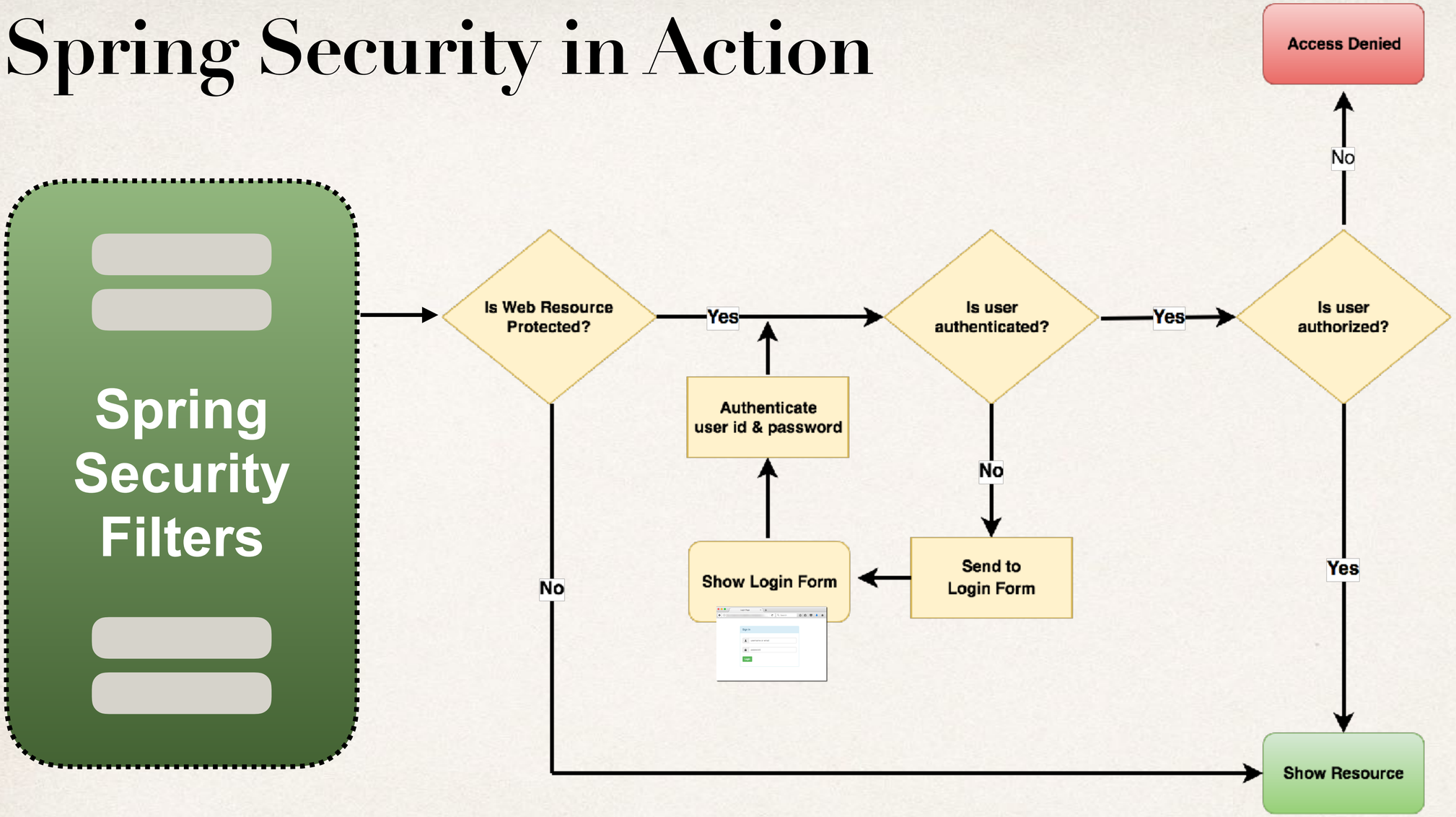
Spring Security in Action



Spring Security in Action



Spring Security in Action



Development Process

Step-By-Step

Development Process

Step-By-Step

1. Modify Spring Security Configuration to reference custom login form

Development Process

Step-By-Step

1. Modify Spring Security Configuration to reference custom login form

2. Develop a Controller to show the custom login form

Development Process

Step-By-Step

1. Modify Spring Security Configuration to reference custom login form

2. Develop a Controller to show the custom login form

3. Create custom login form

Development Process

1. Modify Spring Security Configuration to reference custom login form

2. Develop a Controller to show the custom login form

3. Create custom login form

   - HTML (CSS optional)

Development Process

Step-By-Step

1. Modify Spring Security Configuration to reference custom login form

2. Develop a Controller to show the custom login form

3. Create custom login form

- HTML (CSS optional)
- Spring MVC form tag <form:form>

Step 1: Modify Spring Security Configuration

Step 1: Modify Spring Security Configuration

File: DemoSecurityConfig.java

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

   // override method to configure users for in-memory authentication …

}
```

Step 1: Modify Spring Security Configuration

File: DemoSecurityConfig.java

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

   // override method to configure users for in-memory authentication …

}
```

Method	Description

Step 1: Modify Spring Security Configuration

File: DemoSecurityConfig.java

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

   // override method to configure users for in-memory authentication …

}
```

Method	Description
`configure(AuthenticationManagerBuilder)`	Configure users (in memory, database, ldap, etc)

Step 1: Modify Spring Security Configuration

File: DemoSecurityConfig.java

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

   // override method to configure users for in-memory authentication …

}
```

Method	Description
`configure(AuthenticationManagerBuilder)`	Configure users (in memory, database, ldap, etc)
`configure(HttpSecurity)`	Configure security of web paths in application, login, logout etc

Step 1: Modify Spring Security Configuration

File: DemoSecurityConfig.java

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

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

      // add our users for in memory authentication
      …
   }









}
```

Step 1: Modify Spring Security Configuration

File: DemoSecurityConfig.java

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

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

      // add our users for in memory authentication
      …
   }

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

      http.authorizeRequests()
            .anyRequest().authenticated()
          .and()
          .formLogin()
            .loginPage("/showMyLoginPage")
            .loginProcessingUrl("/authenticateTheUser")
            .permitAll();

   }
}
```

Step 1: Modify Spring Security Configuration

File: DemoSecurityConfig.java

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

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

      // add our users for in memory authentication
      …
   }

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

      http.authorizeRequests()
            .anyRequest().authenticated()
          .and()
          .formLogin()
            .loginPage("/showMyLoginPage")
            .loginProcessingUrl("/authenticateTheUser")
            .permitAll();

   }
}
```

Configure security of web paths in application, login, logout etc

Step 1: Modify Spring Security Configuration

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

  http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

Step 1: Modify Spring Security Configuration

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

  http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

Restrict access based on the HttpServletRequest

Step 1: Modify Spring Security Configuration

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

  http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

Any request to the app must be authenticated (ie logged in)

Step 1: Modify Spring Security Configuration

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

  http.authorizeRequests()
        .anyRequest().authenticat
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

We are customizing the form login process

Step 1: Modify Spring Security Configuration

```
@Override
protected void configure(HttpSecurity ht

  http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

Show our custom form at the request mapping

"/showMyLoginPage"

Step 1: Modify Spring Security Configuration

```
@Override
protected void configure(HttpSecurity http

  http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

Login form should POST data to this URL for processing

(check user id and password)

Step 1: Modify Spring Security Configuration

```
@Override
protected void configure(HttpSecur

  http.authorizeRequests()
        .anyRequest().authenticated
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

You can give ANY values
for this configuration.

Just stay consistent in your app

Step 1: Modify Spring Security Configuration

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

  http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

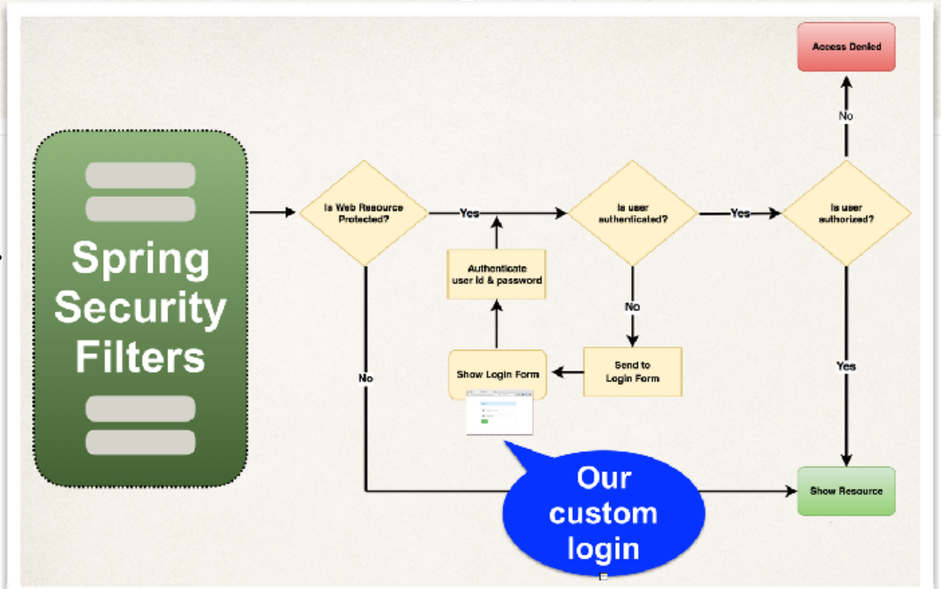
Allow everyone to see login page.
No need to be logged in.

Step 1: Modify Spring Security Configuration

```
@Override
protected void configure(HttpSecurity

  http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```



Step 1: Modify Spring Security Configuration

```
@Override
protected void configure(HttpSecuri

  http.authorizeRequests()
        .anyRequest().authenticated(
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

TO DO
We need to create a controller for this request mapping

"/showMyLoginPage"

Step 1: Modify Spring Security Configuration

```
@Override
protected void configure(HttpSecuri

  http.authorizeRequests()
        .anyRequest().authenticated(
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();

}
```

TO DO
We need to create a controller for this request mapping

"/showMyLoginPage"

No Controller Request Mapping required for this.
We get this for free :-)

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {









}
```

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```

```
http.authorizeRequests()
      .anyRequest().authenticated()
    .and()
    .formLogin()
      .loginPage("/showMyLoginPage")
      .loginProcessingUrl("/authenticateTheUser")
      .permitAll();
```

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```

View name

/WEB-INF/view/plain-login.jsp

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```

/WEB-INF/view/plain-login.jsp

Step 2: Develop a Controller to show the custom login form

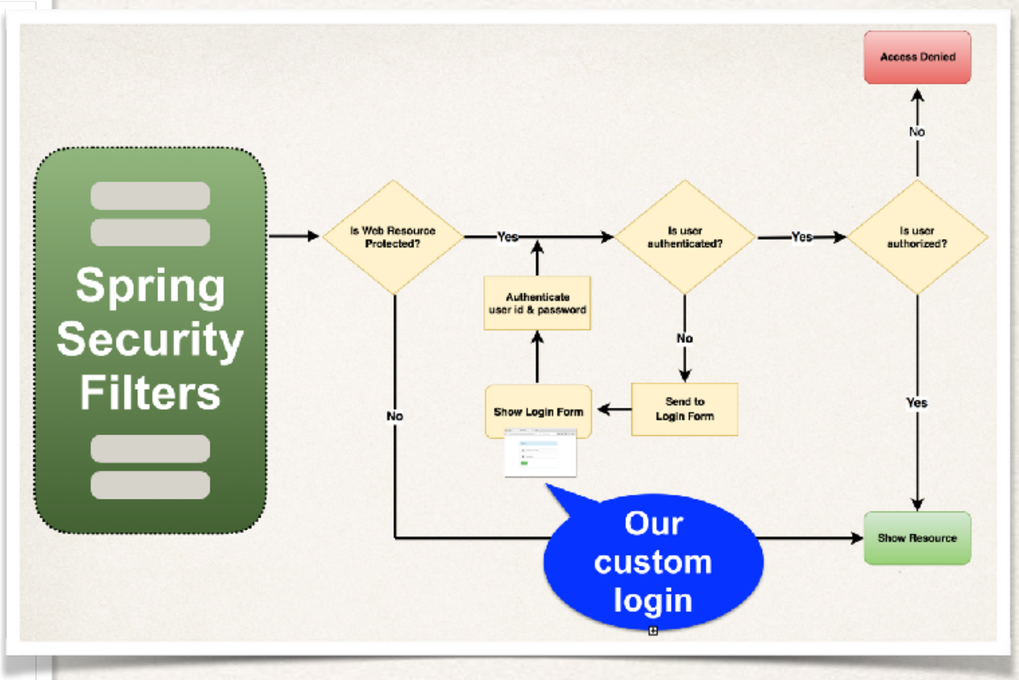
File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```



/WEB-INF/view/plain-login.jsp

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```

/WEB-INF/view/plain-login.jsp

Step 2: Develop a Controller to show the custom login form

File: LoginController.java

```
@Controller
public class LoginController {

  @GetMapping("/showMyLoginPage")
  public String showMyLoginPage() {

    return "plain-login";

  }
}
```

/WEB-INF/view/plain-login.jsp

TO DO
We need to create this file