07

Usando o Retrofit

Transcrição

Colocamos no formulário de ***Login*** o botão ***reCAPTCHA***. O que precisamos fazer é pegar o clique no método `login()` na classe `UsuarioController` , e enviar para o Google efetuar a validação.

De acordo com a documentação do Google, o clique do *reCAPTCHA* é passado com o parâmetro `g-recaptcha-response` . Passaremos como parâmetro no método `login()` o objeto `HttpServletRequest request` , por meio dele chamaremos `request.getParameter("g-recaptcha-response")` que nos retornará o clique do *reCAPTCHA*.

```
@RequestMapping(value="/login", method="RequestMethod.POST")
public String login(@ModelAttribute("usuario") Usuario usuario,
        RedirectAttributes redirect, Model model, HttpSession session, HttpServletRequest reque

    String recaptcha = request.getParameter("g-recaptcha-response");

    Usuario usuarioRetornado = dao.procuraUsuario(usuario);
    model.addAttribute("usuario", usuarioRetornado);
    if (usuarioRetornado == null) {
        redirect.addFlashAttribute("mensagem", "Usuário não encontrado");
        return "redirect:/usuario";
    }

    session.setAttribute("usuario", usuarioRetornado);
    return "usuarioLogado";

}
```

Para enviar o valor do clique para o Google, usaremos uma biblioteca chamada `Retrofit` para auxiliar na comunicação de requisições HTTP. Para usarmos a biblioteca, é necessário adicionarmos algumas dependências no projeto. Colocaremos no `pom.xml` as seguintes dependências:

```
<dependency>
    <groupId>com.squareup.retrofit2</groupId>
    <artifactId>retrofit</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>com.squareup.retrofit2</groupId>
    <artifactId>converter-gson</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>logging-interceptor</artifactId>
    <version>3.9.0</version>
</dependency>
```

Com as dependências baixadas, criaremos uma classe chamada `RetrofitInicializador` em um novo pacote `br.com.alura.owasp.retrofit`. Criaremos um construtor e dentro dele criaremos o objeto do *Retrofit* passando a URL base que ele irá se comunicar.

```
package br.com.alura.owasp.retrofit;

import retrofit2.retrofit;

public class RetrofitInicializador {

    public RetrofitInicializador() {
        new Retrofit.Builder().baseUrl("https://www.google.com/recaptcha/api");
    }
}
```

Utilizando de boas práticas de programação, criaremos uma constante para a URL. Com a String do link selecionada, clicaremos com o botão direito do mouse e selecionaremos a opção "Refactor > Extract Constant", chamaremos a constante de `BASE_URL`.

```
package br.com.alura.owasp.retrofit;

import retrofit2.retrofit;

public class RetrofitInicializador {

    private static final String BASE_URL = "https://www.google.com/recaptcha/api";

    public RetrofitInicializador() {
        new Retrofit.Builder().baseUrl(BASE_URL);
    }
}
```

Mas isso ainda não é o suficiente, o Google nos enviará um JSON e por isso precisamos chamar o método de conversão `addConverterFactory()`, passando como argumento a classe que fará essa função. Por fim, chamaremos o método `build()` para construir o objeto.

```
package br.com.alura.owasp.retrofit;

import retrofit2.retrofit;

public class RetrofitInicializador {

    private static final String BASE_URL = "https://www.google.com/recaptcha/api";

    public RetrofitInicializador() {
        new Retrofit.Builder().baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create()).build();
    }
}
```

Associaremos a instância do objeto em um atributo da classe chamado `retrofit`:

```
package br.com.alura.owasp.retrofit;

import retrofit2.retrofit;

public class RetrofitInicializador {

    private static final String BASE_URL = "https://www.google.com/recaptcha/api";
    private Retrofit retrofit;

    public RetrofitInicializador() {
        retrofit = new Retrofit.Builder().baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create()).build();
    }
}
```

Pronto, o objeto do *Retrofit* foi criado. Voltando para a documentação do Google, veremos que para o Google fazer a validação do *reCAPTCHA* precisamos passar para ele os parâmetros `secret` que é fornecido pelo Google e o `response` que o clique. Esses parâmetros devem ser passados ao Google por meio de uma requisição **POST** para o endereço:

URL: https://www.google.com/recaptcha/api/siteverify (https://www.google.com/recaptcha/api/siteverify)

Para isso criaremos uma **interface** chamada `GoogleService` , dentro da interface faremos a assinatura do método. Para fazer a comunicação com o Google, o Retrofit utiliza uma classe chamada `Call<T>` onde passamos o tipo de retorno. O Google retornará um tipo de JSON, mas como não sabemos exatamente a estrutura, colocaremos `Call<String>` .

Chamaremos o método de `enviaToken()` . Como os parâmetros `sercet` e `response` precisam ser passados pela URL, colocaremos neles a anotação `@Query` .

```
package br.com.alura.owasp.retrofit;

import retrofit2.Call;
import retrofit2.http.Query;

public interface GoogleService {

    Call<String> enviaToken(@Query("secret") String secret, @Query("response") String response);
}
```

Lembrando que essa requisição deverá ser feita pelo verbo HTTP **POST** para o ***siteverify*** - lembrando que já temos a URL base -, por isso colocaremos a anotação `@POST("siteverify")` .

```
package br.com.alura.owasp.retrofit;

import retrofit2.Call;
import retrofit2.http.Query;

public interface GoogleService {

    @POST("siteverify")
    Call<String> enviaToken(@Query("secret") String secret, @Query("response") String response);
}
```

Agora falta enviarmos de fato a informação.