11

Comunicando com o Google

Transcrição

Criamos o objeto **Retrofit** e configuramos com a URL base. Essa URL é o endereço para qual devemos enviar a informação de clique do ***reCAPTCHA***. Em seguida criamos uma interface de ***Service*** que irá enviar uma requisição do tipo **POST** para o endereço "siteverify", passando os parâmetros `secret` e o `response`. Com essas informações o Google irá informar se o clique foi válido ou inválido.

Criaremos uma classe que será o cliente na comunicação com o servidor do Google. Dentro do pacote `br.com.alura.owasp.retrofit`, criaremos a classe `GoogleWebClient`. Dentro da classe criaremos o método `verifica()`, que terá a função de verificar a validade do clique.

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

public class GoogleWebClient {

    public void verifica() {

    }
}
```

Na classe `UsuarioController` estamos pegando o valor do clique e armazenando na variável `recaptcha`. Passaremos a variável `recaptcha` como argumento para o método `new GoogleWebClient().verifica(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");

    new GoogleWebClient().verifica(recaptcha);

    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";

}
```

O Eclipse irá reclamar pelo fato do método `verifica()` não receber parâmetros. Com o cursor no método, usaremos o atalho "Ctrl + 1" e selecionaremos a opção ***Change Method 'verifica()': Add parameter 'String'***. Pronto, agora o método passou parâmetros.

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

public class GoogleWebClient {

    public void verifica(String recaptcha) {

    }
}
```

Dentro do método `verifica()` criaremos o objeto do **Retrofit**. Instanciaremos a classe `new RetrofitInicializador()`, mas precisamos de um método que pegue o *service* que faz a comunicação com o servidor do Google, então faremos a chamada ao método `getGoogleService()`.

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

public class GoogleWebClient {

    public void verifica(String recaptcha) {

        new RetrofitInicializador().getGoogleService();

    }
}
```

Como o método não existe, usaremos o atalho "Ctrl + 1" e selecionaremos a opção ***Create Method***. Na classe `RetrofitInicializador`, mudaremos o retorno do método `getGoogleService()` para `GoogleService`. Em seguida pediremos para o método retornar o objeto criado chamando o `retrofit.create(GoogleService.class)`.

```
public class RetrofitInicializador {

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

    public RetrofitInicializador() {
        retrofit = new Retrofit.Builder().baseUrl(BASE_URL).addConverterFactory(GsonConverterFa
    }

    public GoogleService getGoogleService() {
        return retrofit.create(GoogleService.class);
    }
}
```

De volta a classe `GoogleWebClient`, a partir do método `getGoogleService()` chamaremos o `enviaToken()` da interface passando os parâmetros `secret` e o `response`. O `secret` foi fornecido pelo Google na documentação, para manter as boas práticas criaremos um constante `SECRET`. Já o `response` é o valor da variável `recaptcha` passada como parâmetro para o método.

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

public class GoogleWebClient {
```

```
    private static final String SECRET = "6LddPTUUAAAAAOkAx05jP2N9rIPlhf3WQHMuHMjZ";

    public void verifica(String recaptcha) {

        new RetrofitInicializador().getGoogleService().enviaToken(SECRET, recaptcha);

    }
}
```

Com o retorno dessa chamada, armazenaremos em variável `Call<String> token`. Com o objeto `token`, poderemos executar a chamada e definir a resposta pelos métodos `token.execute().body()`.

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

public class GoogleWebClient {

    private static final String SECRET = "6LddPTUUAAAAAOkAx05jP2N9rIPlhf3WQHMuHMjZ";

    public void verifica(String recaptcha) {

        Call<String> token = new RetrofitInicializador().getGoogleService().enviaToken(SECRET,
        token.execute().body();

    }
}
```

O Eclipse irá reclamar, é necessário tratar as exceções que podem ser lançadas. Usaremos o atalho "Ctrl + 1" e selecionaremos a opção ***Add throws declaration***. Novamente o Eclipse irá reclamar no método `login()` da classe `UsuarioController`, é necessário tratar a chamada `new GoogleWebClient().verifica(recaptcha)`, também usaremos o *Add throws declaration*.

Com tudo configurado, testaremos a aplicação. Acessaremos o formulário de *Login*, no campo **E-mail** colocaremos `ana@gmail.com`, no campo **Senha** colocaremos `789`, clicaremos no botão do *reCAPTCHA* e em "LOG IN".

Repare que recebemos um ***Exception***, isso aconteceu porque colocamos o tipo de retorno da requisição como String na `Call<String>`.

Na próxima aula veremos como tratar o tipo de retorno.