06

Fazendo autenticação com o GoogleRecaptcha

Transcrição

Criamos a classe `Resposta` com um atributo `success`, o nome do atributo é o mesmo da propriedade recebida no JSON enviado pelo Google. Na classe `GoogleWebClient`, estamos chamando o método `isSuccess()` para verificar se o valor é `true` ou `false`. Precisamos retornar esse valor para o método `login()` do `UsuarioController`, por isso usaremos o `return` e mudaremos o tipo de retorno do método para `boolean`.

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

import java.io.IOException;

import retrofit2.Call;

public class GoogleWebClient {

    private static final String SECRET = "6LddPTUUAAAAAOkAx05jP2N9rIPlhf3WQHMuHMjZ";

    public boolean verifica(String recaptcha) {

        Call<Resposta> token = new RetrofitInicializador().getGoogleService().enviaToken(SECRET
        return token.execute().body().isSuccess();
    }
}
```

Como estamos trabalhando com o **Spring**, podemos pedir para ele fazer o gerenciamento do estado dos objetos da classe `GoogleWebClient`, dessa forma na classe `UsuarioController` podemos pedir a injeção desses objetos. Para isso anotaremos a classe `GoogleWebClient` com a *annotation* `@Component`.

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

import java.io.IOException;

import org.springframework.stereotype.Component;

import retrofit2.Call;

@Component
public class GoogleWebClient {

    private static final String SECRET = "6LddPTUUAAAAAOkAx05jP2N9rIPlhf3WQHMuHMjZ";

    public boolean verifica(String recaptcha) {

        Call<Resposta> token = new RetrofitInicializador().getGoogleService().enviaToken(SECRET
        return token.execute().body().isSuccess();
    }
}
```

Com o atalho "Ctrl + Shift + R" acessaremos a classe `UsuarioController`. Pediremos para o Spring injetar um objeto da classe `GoogleWebClient`:

```
@Controller
@Transactional
public class UsuarioController {

    @Autowired
    private UsuarioDao dao;

    @Autowired
    private GoogleWebClient cliente;

    // ...
}
```

No método `login()`, não precisaremos instanciar o objeto na mão, basta trocarmos para o `cliente.verifica(recaptcha)`. Como o método `verifica()` retorna `true` ou `false`, armazenaremos o retorno em uma variável `boolean verificaRecaptcha`.

```
@RequestMapping(value="/login", method=RequestMethod.POST)
public String login(@ModelAttribute("usuario") Usuario usuario, RedirectAttributes redirect, Mo

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

    boolean verificaRecaptcha = cliente.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";
}
```

Para que o código não fique muito extenso, vamos extrair o trecho de código que verifica o e-mail e senha do usuário para um método próprio. Com o trecho selecionado, clicaremos com o botão direito do mouse e em "Refactor > Extract Method". Chamaremos o novo método de `procuraUsuario()`:

```
@RequestMapping(value="/login", method=RequestMethod.POST)
public String login(@ModelAttribute("usuario") Usuario usuario, RedirectAttributes redirect, Mo

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

    boolean verificaRecaptcha = cliente.verifica(recaptcha);

    return procuraUsuario(usuario, redirect, model, session);
}
```

```
private String procuraUsuario(Usuario usuario, RedirectAttributes redirect, Model model, HttpSe

    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";
}
```

No método `login()` se a variável `verificaRecaptcha` for `true`, então faremos a lógica de verificar e-mail e senha de usuário. Caso a variável `verificaRecaptcha` for `false`, nós enviaremos uma mensagem e retornaremos para o formulário.

```
@RequestMapping(value="/login", method=RequestMethod.POST)
public String login(@ModelAttribute("usuario") Usuario usuario, RedirectAttributes redirect, Mo

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

    boolean verificaRecaptcha = cliente.verifica(recaptcha);

    if(verificaRecaptcha){
        return procuraUsuario(usuario, redirect, model, session);
    }

    redirect.addFlashAttribute("mensagem", "Por favor, comprove que você é humano!");
    return "redirect:/usuario"
}
```

Reiniciaremos o Tomcat e tentaremos fazer o *login* com a conta da **Ana**, no campo **E-mail** colocaremos `ana@gmail.com`, no campo **Senha** colocaremos `789`, clicaremos no ***reCAPTCHA*** e em "LOG IN".

Conseguimos efetuar a autenticação e acessa o perfil da Ana! Para verificar se tudo está funcionando como imaginamos, tentaremos fazer o *login* mas **sem clicar no botão *reCAPTCHA***. Maravilha! Recebemos a mensagem pedindo a comprovação de que somos humanos.

Com essa nova camada de segurança, tentaremos novamente executar o teste de força bruta que o Alex utilizou para descobrir a senha da Ana.