07

Registrando os acessos da API

Vamos abordar agora uma funcionalidade ainda não mencionada, as ações, ou **Action**s. Elas servem como um tipo de interceptador que roda antes da lógica do controller nas requisições, com um detalhe: caso a rota seja autenticada esta é verificada antes e caso falhe, a **Action**, assim como a lógica do controller, não chega a ser executada. Vamos usar isso para fazer um registro dos acessos dos usuários à nossa API, para podermos gerar relatórios e usar os dados para analisar o tráfego do nosso servidor. Neste projeto vamos somente fazer uma contagem, mas poderíamos utilizar para identificar casos de abuso ou adequar tarefas para diversos níveis de utilização.

Para criar uma **Action**, basta estender a classe **play.mvc.Action.Simple**, sobrescrevendo o método `call()`.

```
package acoes;
public class AcaoDeRegistroDeAcesso extends Action.Simple {
    @Override
    public CompletionStage<Result> call(Context context) {
        Logger.debug("Ação de registro!");
        return delegate.call(context);
    }
}
```

Com a **Action** criada, anote os métodos do controller (ou o próprio controller) com `@With(AcaoDeRegistroDeAcesso.class)`. No caso, anotamos o próprio **ApiController**.

```
@Authenticated(AcessoDaApiAutenticado.class)
@With(AcaoDeRegistroDeAcesso.class)
public class ApiController extends Controller { ... }
```

Fazendo requisições autenticadas corretamente, podemos conferir que a frase **Ação de registro!** é impressa no log do servidor mas quando a autenticação falha o texto não é impresso.

Agora vamos criar o modelo dos registros, que consiste do **usuario** autenticado, da **uri** acessada e da **data** de acesso, tudo inicializado no construtor. Repare que a vinculação do usuário é `@ManyToOne`, pois teremos diversos registros para cada único usuário.

```
package models;
@Entity
public class RegistroDeAcesso extends Model {
    @Id
    @GeneratedValue
    private Long id;
    @ManyToOne
    private Usuario usuario;
    private String uri;
    private Date data;
    public RegistroDeAcesso (Usuario usuario, String uri) {
        this.usuario = usuario;
        this.uri = uri;
        this.data = new Date();
    }
```

```
    // getters
}
```

No modelo de usuário também queremos uma referência a esses acessos, com a anotação `@OneToMany` mapeada para o modelo de registros.

```
public class Usuario extends Model {
    // ...
    @OneToMany(mappedBy = "usuario")
    private List<RegistroDeAcesso> acessos;
    // getter e setter
}
```

E enfim criar a nova evolução da tabela de registros.

```
# --- !Ups
create table registro_de_acesso (
  id                            bigint auto_increment not null,
  usuario_id                    bigint,
  uri                           varchar(255),
  data                          datetime(6),
  constraint pk_registro_de_acesso primary key (id)
);
alter table registro_de_acesso add constraint fk_registro_de_acesso_usuario_id foreign key (usua
create index ix_registro_de_acesso_usuario_id on registro_de_acesso (usuario_id);
# --- !Downs
alter table registro_de_acesso drop foreign key fk_registro_de_acesso_usuario_id;
drop index ix_registro_de_acesso_usuario_id on registro_de_acesso;
drop table if exists registro_de_acesso;
```

Agora com a modelagem pronta podemos armazenar nosso **RegistroDeAcesso** no banco sempre que a **AcaoDeRegistroDeAcesso** for executada. Para isso precisamos recuperar o usuário que fez a requisição, mas como a autenticação já foi feita podemos pular a parte de conferir se ele existe. Precisamos também da **uri** requisitada, recuperada a partir do objeto de requisição.

```
@Inject
private UsuarioDAO usuarioDAO;
@Override
public CompletionStage<Result> call(Context context) {
    String codigo = context.request().getHeader("API-Token");
    String uri = context.request().uri();
    Usuario usuario = usuarioDAO.comToken(codigo).get();
    RegistroDeAcesso acesso = new RegistroDeAcesso(usuario, uri);
    acesso.save();
    return delegate.call(context);
}
```

E enfim mostrar a informação dos acessos no painel do usuário, adicionando ao final do arquivo dentro do bloco `@main {}` .

```
<p class="panel-body">
    Você já fez <strong>@usuario.getAcessos().size()</strong> acessos usando seu código.
</p>
```