Segurança com Token em WebService REST

Eduardo Rodrigues

Tópicos Especiais 6M1/2017

Passo Fundo 15 de Dezembro de 2017

1. Introdução

O presente trabalho consiste em uma aplicação desenvolvida na linguagem Java, utilizando a tecnologia JSF e como estilização o framework Primefaces. A aplicação disponibiliza um serviço web, que se baseia no método “REST”. O Web Service REST é uma das formas de criar um serviço web, que utilizada muito o protocolo HTTP para realizar essa integração entre as aplicações . (SAKURAI[1], 2012).

A aplicação realiza um cadastro de uma classe pré determinada no modelo, possibilitando manutenções de GET, POST, PUT e DELETE. Através de uma base de dados são guardados todos os registros feitos pelo usuário através da aplicação, tanto com a utilização de um cliente quando diretamente persistido pelo lado do servidor. As operações de CRUD não possuem acesso total a qualquer usuário, por esse motivo para inserir registros e alterá-los, foi implantado o padrão de segurança via Token. Para protagonizar a segurança do sistema, será implementado com o auxílio de Web Token, ou seja o serviço do JAVA WT. O JWT (JSON Web Token) é um sistema de transferência de dados que pode ser enviado via POST ou em um cabeçalho HTTP (header) de maneira “segura”, essa informação é assinada digitalmente por um algoritmo HMAC, ou um par de chaves pública/privada usando RSA. Podemos ver na imagem abaixo um cenário onde será requisitado um token através do Verbo HTTP POST, que irá devolver um token validado para que nas próximas requisições que utilizem os Verbos HTTP possam utilizar. Todos os métodos de manipulação de dados que o sistema web oferece é programado em uma classe principal que através de “Paths”, controla a entrada e saída de informações do WebService, assim tornasse possível proteger determinado “Path” com a segurança por token, apenas inserindo uma anotação de instância da própria classe que contém um usuário e senha válidos.

2. Objetivo Geral

O objetivo do projeto é realizar um estudo de Web Service, implementado e testando suas funcionalidades e possibilidades. Deverá ser estudado a tecnologia JSF, para a criação de um cliente, o framework Primefaces que auxiliará na

[1] "Web Service REST - Chamando um web service ... - Universidade Java." 6 dez. 2012, http://www.universidadejava.com.br/materiais/webservice-rest-jsf/. Acessado em 13 set. 2017.

estilização do sistema, a conexão da aplicação com uma base de dados, a implementação de um sistema de login e tratamento de permissões de usuários.

3. Objetivos específicos

- Concluir o desenvolvimento da aplicação, sem bugs, erros ou falhas
- Construir um sistema que consiste em listar filmes, como seus atributos de título, sinopse etc…
- Os filmes listados devem estar cadastrados em uma base de dados
- Possibilidade do cliente fazer alterações nos filmes ou exclusões, até o cadastramento de novos
- Realizar login ao entrar no sistema
- Controlar as permissões de usuários
- Realizar a chamada de um WebService REST com JSF

4. Referencial teórico/prático para desenvolvimento

Artigos e Tutoriais, bem como vídeo aulas pela internet.

4.1. Tecnologia JSF

JSF é uma tecnologia que nos permite criar aplicações Java para Web utilizando componentes visuais pré-prontos, de forma que o desenvolvedor não se preocupe com Javascript e HTML. Basta adicionar os componentes (calendários, tabelas, formulários) e eles serão renderizados e exibidos em formato html. Além disso o estado dos componentes é sempre guardado automaticamente. Isso nos permite, por exemplo, criar formulários de várias páginas e navegar nos vários passos dele com o estado das telas sendo mantidos.

4.2. Aplicação com Banco de dados

Trata da criação de variáveis na classe, já pré definidas para facilitar futuras alterações, e são preenchidas conforme a configuração da sua aplicação e configuração de banco de dados como por exemplo, atributos de usuário e senha; é necessário que seja declarado também o driver que nesse caso é padrão por ser do PostgreSQL e depende do banco que irá usar na sua aplicação.

4.3. Web Service

O serviço web (web service) é uma forma de integração bem utilizada atualmente para realizar a integração entre aplicações. O Web Service REST é uma das formas de criar um serviço web, que utilizada muito o protocolo HTTP para realizar essa integração entre as aplicações.

O Web Service REST pode disponibilizar através do protocolo HTTP métodos para manipulação de informações, por exemplo: ao fazer uma requisição HTTP através do método GET para a URL http://localhost:8080/CinemaREST/servico/filmes é possível obter um recurso, que nesse caso é uma lista de filmes. Se chamar essa URL através de um navegador podemos verificar o retorno desse Web Service REST.

4.4. Json

O JSON além de ser um formato leve para troca de dados é também muito simples de ler. É interessante compará-lo com o formato XML, que analisado junto ao Json pode ficar de lado, por sua gama de opcionalidades ser mais restrita comparadas às do Json.

5. Método REST

Serviços	Entrada	Saída
addFilme	Todos os atributos	OK,ERRO,EXISTE ...
updateFilme	id do filme	ok,erro
deleteFilme	id do filme	ok,erro

Acesso : http://localhost:8080/MeuCinema/webresources/filmes

{} = Variável

! = Método

ROTA	**Método HTTP**	**Parâmetros**	**Retorno**
\filmes	GET	nenhum	Todos os filmes
\filmes	POST	filme	insere um filme
\filmes	PUT	lista de filme	atualiza a lista de filmes
\filmes	DELETE	nenhum	exclui filme
\filmes{id}	GET	nenhum	retorna o filme que possui o id
\filmes{id}	PUT	o filme (baseado pelo id).	atualiza filme que possui o id

\filmes{id}	DELETE	nenhum	exclui filme que possui o id

MENSAGENS :

“1xx" : Continue
"2xx" : OK
"3xx" : Accepted
"4xx" : Erros
“5xx” : informações

URI	HTTP	Sucesso STATUS	Erro STATUS
\filmes	GET	200	404, 401, 403, 404, 405
\filmes	POST	200,100	‘’
\filmes	PUT	200	‘’
\filmes	DELETE	200	‘’
\filmes{id}	GET	200	‘’
\filmes{id}	PUT	200	‘’
\filmes{id}	DELETE	200	‘’

Json :

```
{
    “filme” :  “As aventuras de PI”,
    “sinopse” : ” Pi e sua família decidem ir para o Canadá”,
    “gênero” :  “Ficção”,
    “duração” : “120 minutos” ,
    “trailer”  :  “código de video do youtube”;
}
```

6- Conexão com Base de dados

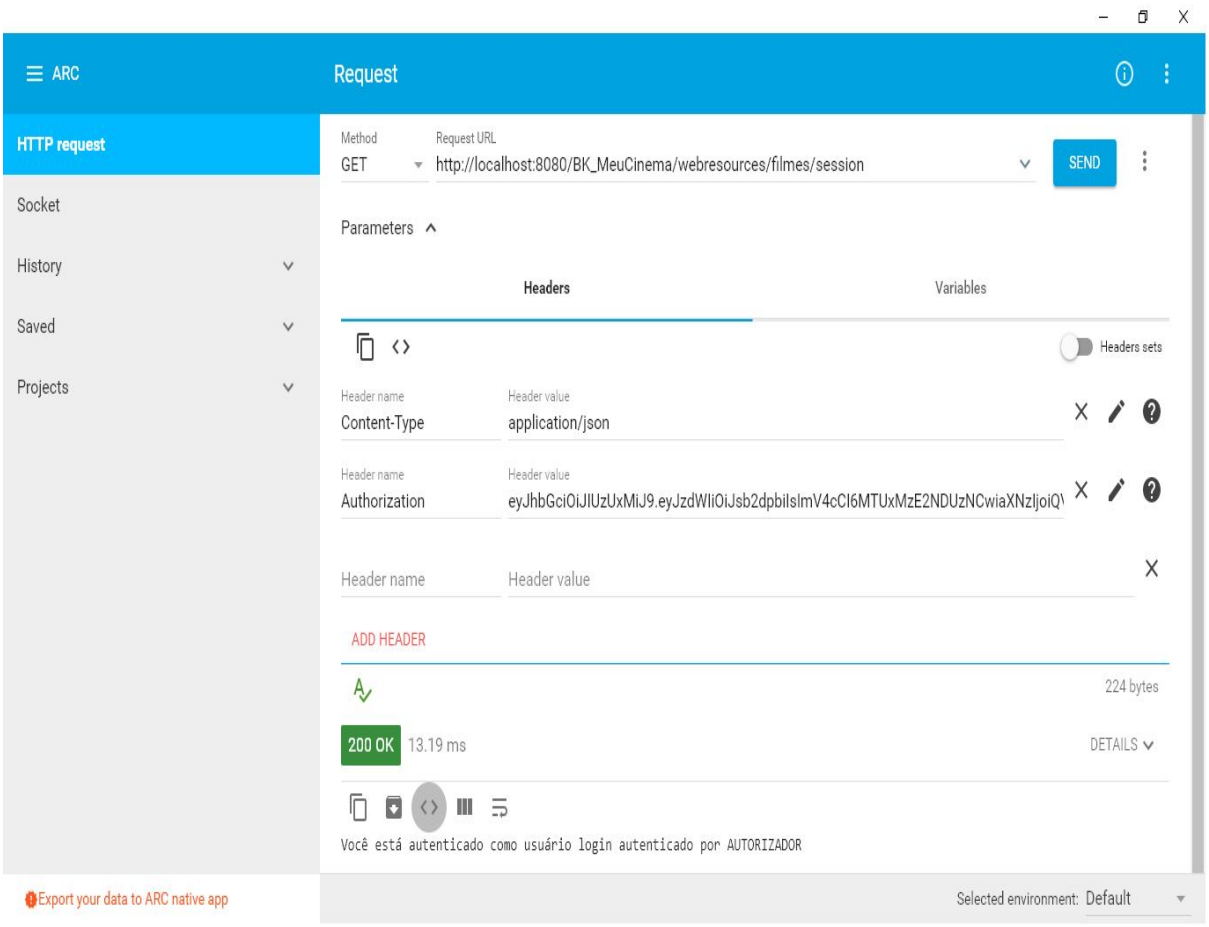
A base de dados que recebe os dados do servidor é o MySQL e o SGBD utilizado como gerenciador de dados foi o PhpMyAdmin. O mysql é um servidor que funciona por TCP, mas não por HTTP, ele é um servidor de banco de dados totalmente separado que pode ser instalado até em um computador diferente de uma mesma rede ou de uma rede externa que seja acessível e fica em uma porta diferente do HTTP. O phpmyadmin não é um servidor ele é um "gerenciador" escrito em php e html ou seja ele é uma aplicação e não deve ser usado por outras aplicações acessíveis a outros usuários, é uma ferramenta que deve ser usada apenas pelo administrador do banco de dados e por vezes por quem mantém os scripts , não é parte do mysql, não é um banco dados, apenas é uma ferramenta para facilitar a gestão do banco de dados e existem alternativas pra ele.

A estrutura do código de conexão é relativamente simples, basta integrar os endereços de conexão com uma variável, segue o código:

```
public class conexao {
    private static final String URL = "jdbc:mysql://localhost/webservice";
    private static final String DRIVER = "com.mysql.jdbc.Driver";
    private static final String USER = "root";
    private static final String SENHA = "root";

    private static Connection con;

    public static Connection conectaBanco() throws ClassNotFoundException, SQLException{
        Class.forName(DRIVER);
        con = DriverManager.getConnection(URL,USER,SENHA);
        return con;
    }

    public static void deconctaBanco(){
        try {
            con.close();
        } catch (SQLException ex) {
            Logger.getLogger(conexao.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

7 - Cliente para o WebService

Após o término da programação do servidor, foi proposto a sua interação com uma aplicação que faz o papel de cliente. No presente trabalho foi integrado o plugin “Advanced REST Client”, que através de métodos como: GET, POST, PUT E DELETE, foi possível a integração com o serviço disponibilizado pelo servidor. Os parâmetros de conexão com o WebService para o cliente são: Declarar o método a ser usado, inserir a url referente ao “PATH” que foi designado para determinada manutenção no lado servidor e dependendo de qual é a manutenção desejada necessita a inserção de parâmetros. Isso ocorre em casos de persistência de Json, texto ou XML e caso necessite de algum tipo de manutenção de segurança, envolvendo token e autorizações. A interface inicial do cliente é simples, porém intuitiva, como mostra a imagem :



8 - Manutenção CRUD

Todas as manutenções do servidor foram escritas na classe “FilmesResources”, ambas são nomeadas por “@PATH” e possuem a rota de acesso que disponibiliza seus serviços ao cliente. Para disponibilizar os dados obtidos nas consultas ao servidor, foi implementado a conversão dos dados para Json, através da biblioteca Gson (criado pelo google).

A primeira manutenção é o método de listagem, que implica em percorrer uma lista de filmes e listá los na tela. Segue o código:

```
@GET
@Produces(MediaType.APPLICATION_JSON)
@Path("Listar/filmes")
public String getFilmes() throws ClassNotFoundException, SQLException{
    Gson g = new Gson();
    FilmeDAO dao = new FilmeDAO();
    List<Filme> filmes = dao.listaFilmes();

    return g.toJson(filmes);
}
```

A manutenção de inserção, que está descrita como “inserir”, dispõe dos atributos da classe que vai ser persistida, e recebe um Json com o preenchimento dos parâmetros que equivalem a cada atributo.Segue o código:

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
@Path("/inserir")
public boolean inserirFilme(String content, @HeaderParam("Authorization")String token){
    String user = testaToken(token);
    System.out.println("Usuario " + user);

    Gson g = new Gson();
    Filme f = (Filme) g.fromJson(content, Filme.class);

    FilmeDAO dao = new FilmeDAO();
    return dao.inserirFilme(f);
}
```

A manutenção para deletar um registro, faz a busca do objeto que é existente dentro da lista que percorre a classe, e através do seu identificador, chama o método de exclusão. Segue o código:

```
@GET
@Produces(MediaType.APPLICATION_JSON)
@Path("/deletar/{id}")
public String deletarFilme(@PathParam("id")int id){
    FilmeDAO dao = new FilmeDAO();
    if(dao.deletarFilme(id)){
        return "true";
    }else{
        return "false";
    }
}
```

Por fim, o método alterar, que lista os dados que já estão cadastrados no banco, e através do ID selecionado, é substituído o parâmetro antigo pelo Json que será gerado novamente após a sua chamada. Segue o código:

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
@Path("/alterar")
public boolean alterarFilme(String content, @HeaderParam("Authorization")String token){
    String user = testaToken(token);
    System.out.println("Usuario " + user);


    Gson g = new Gson();
    Filme f = (Filme) g.fromJson(content, Filme.class);

    FilmeDAO dao = new FilmeDAO();
    return dao.alterarFilme(f);
}
```

9 - Segurança

Em uma aplicação REST a formas tradicionais de fazer a autenticação e segurança podem não ser melhor caminho, por isso existem autenticações baseadas em token.
Resumindo esses são os passos de uma autenticação com token tem:

1. O cliente envia as credenciais para o servidor.

2. O servidor autentica as credenciais e gera um token.

3. O servidor envia o token para o cliente.

4. O cliente salva esse token e envia ele em um header em cada requisição

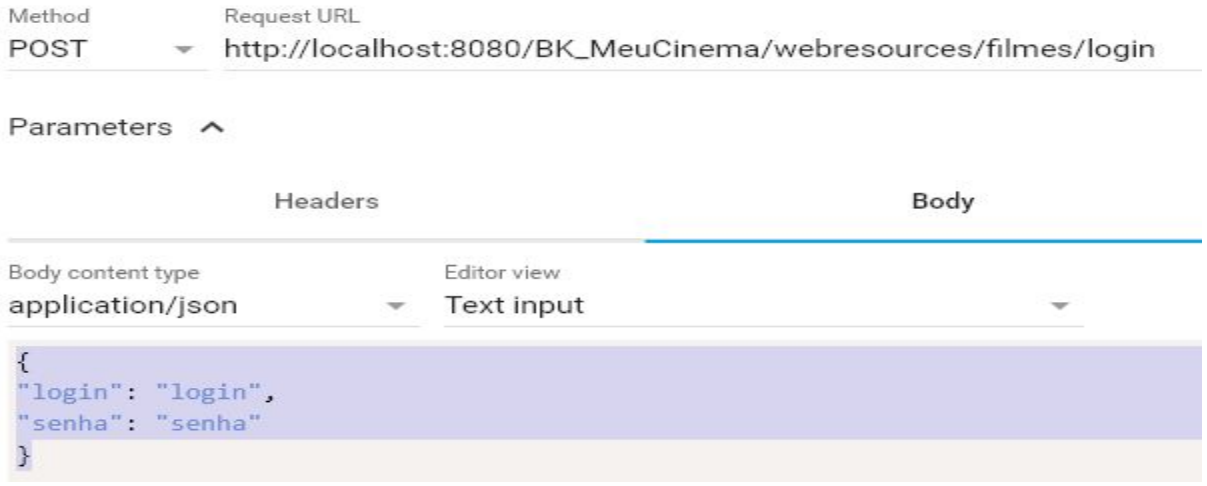
6. O servidor, em cada requisição extrai o token e verifica se o token é válido ou não

- Se o token é válido, o servidor aceita a requisição

- Se o token é inválido, o servidor rejeita a requisição

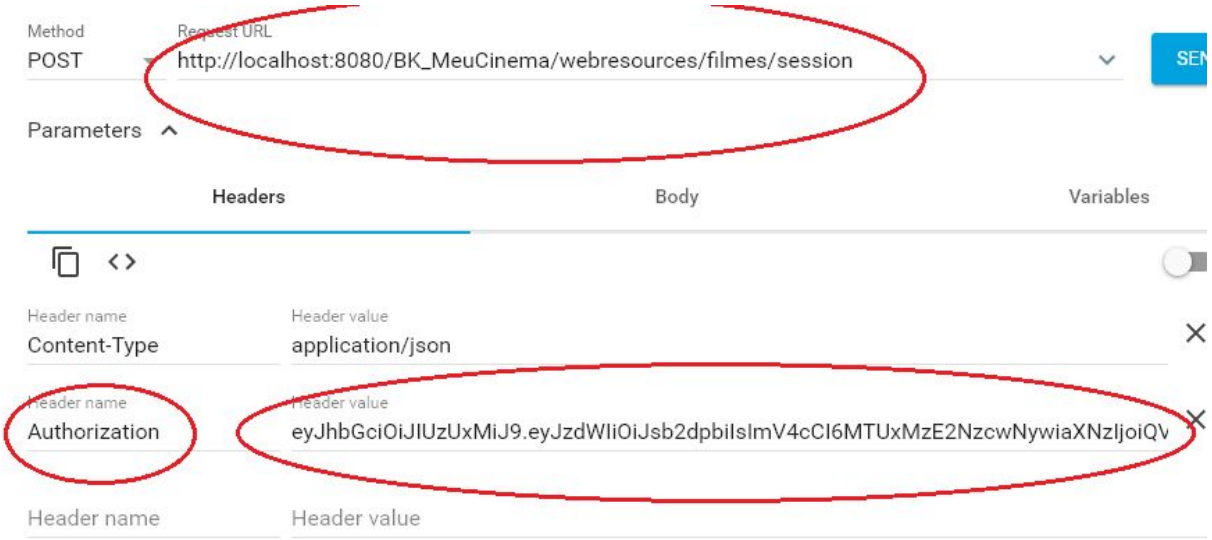
7. O servidor pode ter um endpoint que renova o token

No caso da aplicação desenvolvida no trabalho, é necessário acessar o Patch : “http://localhost:8080/BK_MeuCinema/webresources/filmes/login” , logo após inserir na aba “Body” as suas respectivas credenciais em formato Json, como mostra a imagem :

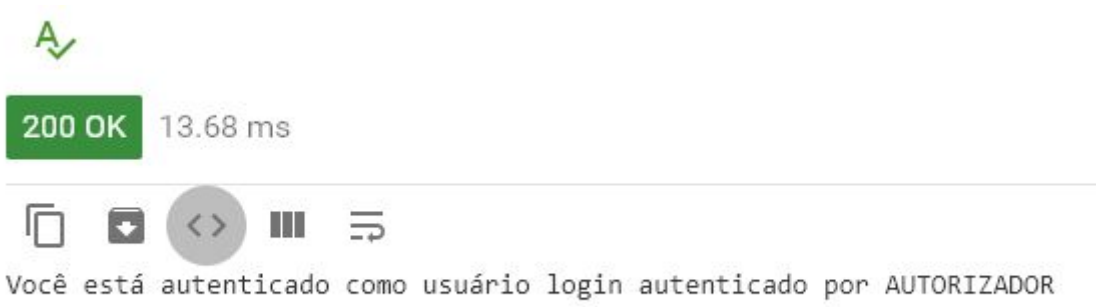


Logo após o envio, na guia de depuração o cliente irá responder se o token foi gerado com sucesso ou não:

Então basta acessar a rota Session : http://localhost:8080/BK_MeuCinema/webresources/filmes/session, e inserir na aba Headers o token e o parâmetro de autenticação :



Caso tudo ocorra bem, ou seja, se o token for válido (incluindo seu tempo de renovação) e o header name estar correto, no campo de depuração vai constar a seguinte mensagem :



10. Referencias

http://www.devmedia.com.br/o-que-e-json/23166

http://www.universidadejava.com.br/materiais/webservice-rest-jsf/

http://narinhachaves.blogspot.com.br/2013/02/classe-java-de-conexao-com-banco-de.html

https://www.caelum.com.br/apostila-java-testes-jsf-web-services-design-patterns/introducao-ao-jsf-e-primefaces/#7-1-desenvolvimento-desktop-ou-web