

Segurança em REST

Prof. Dr. Alexandre L'Erario



Considerações iniciais

- Aplicativos Restful utilizam HTTP, portanto:
 - Mesmos problemas do HTTP
 - Segurança
 - Mesmas vantagem do HTTP
 - Portabilidade
 - Certificação digital/criptografia
 - Universalmente aceito

Elementos essenciais da segurança

- **Autenticação:** envolve a verificação da identidade do usuário que está tentando acessar o aplicativo ou serviço da web. Normalmente, isso é realizado obtendo as credenciais de login e validando-as em relação aos detalhes do usuário configurados no servidor.
- **Autorização:** Envolve verificar o que um usuário autenticado tem permissão para fazer no aplicativo ou serviço.

Autenticação básica HTTP

- A autenticação HTTP básica funciona enviando o nome de usuário e a senha codificados em Base64 e UTF-8 como um par no cabeçalho de autorização HTTP.
- O nome de usuário e a senha devem ser enviados para cada solicitação HTTP feita pelo cliente.

Implementando uma autenticação básica

- Como interceptar uma request, antes de um objeto?
- Interceptor e Filter resolvem o problema
 - ContainerRequestFilter deve ser implementado

Criar uma classe filtro

```
@Provider

public class JaxRsFilter implements ContainerRequestFilter {

    @Context

    private ResourceInfo resourceInfo;

    private static final String AUTHORIZATION_PROPERTY = "Authorization";

    private static final String AUTHENTICATION_SCHEME = "Basic";

    public void filter(ContainerRequestContext requestContext) {

        //recupera os cabecalhos da requisicao

        final MultivaluedMap<String, String> headers = requestContext.getHeaders();

        //recupera o cabecalho de autenticacao

        final List<String> authorization = headers.get(AUTHORIZATION_PROPERTY);

        //Bloqueia o acesso caso nao haja informacao de autorizacao

        if (authorization == null || authorization.isEmpty()) {

            requestContext.abortWith(Response.status(Response.Status.FORBIDDEN)

                .entity("Nao possui informacao de autenticacao! (adicione o header Authorization = user1:pass2)").build());

            return;

        }

    }

}
```

```
//Recupera o usuario e senha codificados
    final String encodedUserPassword =
authorization.get(0).replaceFirst(AUTHENTICATION_SCHEME + " ", "");
    //Decodifica-os
    String usernameAndPassword;
    boolean autenticar = false;
    try {
        usernameAndPassword = new
String(Base64.getDecoder().decode(Base64.getEncoder().encode(encodedUserP
assword.getBytes("UTF-8"))));
        //Separa os tokens
        final StringTokenizer tokenizer = new
StringTokenizer(usernameAndPassword, ":");
        final String username = tokenizer.nextToken();
        final String password = tokenizer.nextToken();

    }
```



```
if (username.equals("user1") && password.equals("pass2")) {
    autenticar = true;
}

if (!autenticar) {
    requestContext.abortWith(Response.status(Response.Status.
FORBIDDEN).entity("usuario e senha invalidos").build());
}
```

Cliente Rest – Autenticação Básica

- Login e Senha devem ser encaminhados no header do HTTP
- Deve ser incluído após o request

```
loginSenha= "user1"+":"+"pass2";
```

```
public void add(int codigo, String nome) throws ClientErrorException {  
    Response response = webTarget.path("cidade")  
        .path("/") + codigo + "/" + nome)  
        .request()  Após request  
        .header(HttpHeaders.AUTHORIZATION, loginSenha)  
         método .put(Entity.entity("", MediaType.APPLICATION_JSON));  
}
```

Considerações finais

- Implementação simples
- Pode suportar diversos estilos em um mesmo software