# Basic Authentication

## Tài liệu tham khảo

1. Website [gpcoder](https://gpcoder.com/5724-rest-web-service-basic-authentication-trong-jersey-2-x/#Gioi_thieu_Basic_Authentication_trong_Jersey_REST_API) (lý thuyết và mã nguồn ví dụ) – Tiếng Việt
2. Website [simplapi](https://simplapi.wordpress.com/2015/09/19/jersey-jax-rs-securitycontext-in-action/) (lý thuyết và mã nguồn ví dụ) – Tiếng Anh

## Tổng quan

1. Nói chung, với **basic authentication** chúng ta sẽ gửi **username** và **password** được mã hóa **Base64** theo sau mỗi request đến server, xem nhanh mục [[4]](#_Luồng_thực_hiện) để có cái nhìn tổng quát.
2. Server sẽ có bộ lọc để lấy thông tin được mã hóa từ header (hoặc bất kỳ cách nào khác như cookie, body, hidden, …) sau đó tiến hành decode, phân tích… để lấy ra được **username** và **password**. Tiến hành so khớp thông tin với thông tin được lưu trữ tại server (database, ram, rom, …). Nếu quá trình so khớp là thành công, đối tượng sẽ được gói vào 1 lớp thực thi giao diện **SecurityContext** và được quản lý bởi **ContainerRequestContext**.
3. Lớp thực thi giao diện SecurityContext được dùng trong quá trình **authorization** (phân quyền). Ta dùng Annotation **@RolesAllowed(<<Role>>)** để xác định quyền truy cập của User. Hai annotation khác cũng dùng trong authorization là **@PermitAll** và **@DenyAll**.
4. Ngoài ra ta cũng có thể lấy được đối tượng SecurityContext với annotation **@Context**.
5. Lưu ý, đăng ký lớp RolesAllowedDynamicFeature để bật tính năng **authorization**.

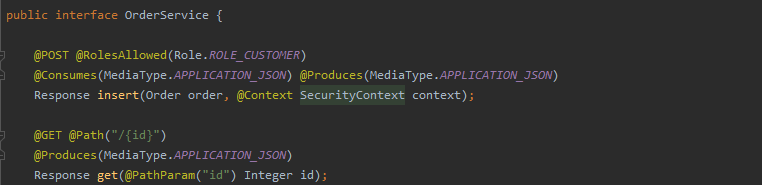
## Giải thích step-by-step

[2.1] – Chúng ta sẽ dùng lớp **HttpAuthenticationFeature** để tiến hành gửi kèm username và password theo mỗi request, lớp này hỗ trợ 4 cách khác nhau để thao tác khi gửi thông tin bao gồm BASIC, BASIC NON-PREEMPTIVE, DIGEST và UNIVERSAL thông tin chi tiết ở trên Google hoặc tham khảo mục [1.1].

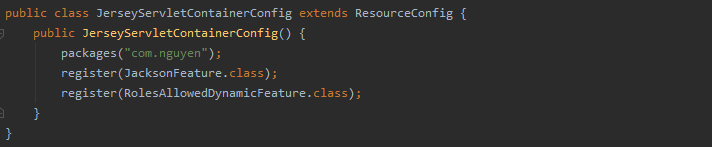


[2.2] – Server sẽ lấy thông tin xác thực từ **header Authorization** và thực hiện các thao tác logic để đến điểm đến cuối cùng là **một user được gói trong lớp thực thi SecurityContext** hoặc **1 ngoại lệ** được ném ra (bắt hay không tùy vào nhu cầu). Mã nguồn filter này tham khảo [tại đây](https://github.com/ytbhoanghai/Jersey-JAXRS/blob/master/REST/Jersey%20-%202.x/BasicAuthentication/src/main/java/com/nguyen/filter/AuthFilter.java). Chú ý annotation **@Priority(Priorities.AUTHENTICATION)** -> filter thực hiện trước authentication.

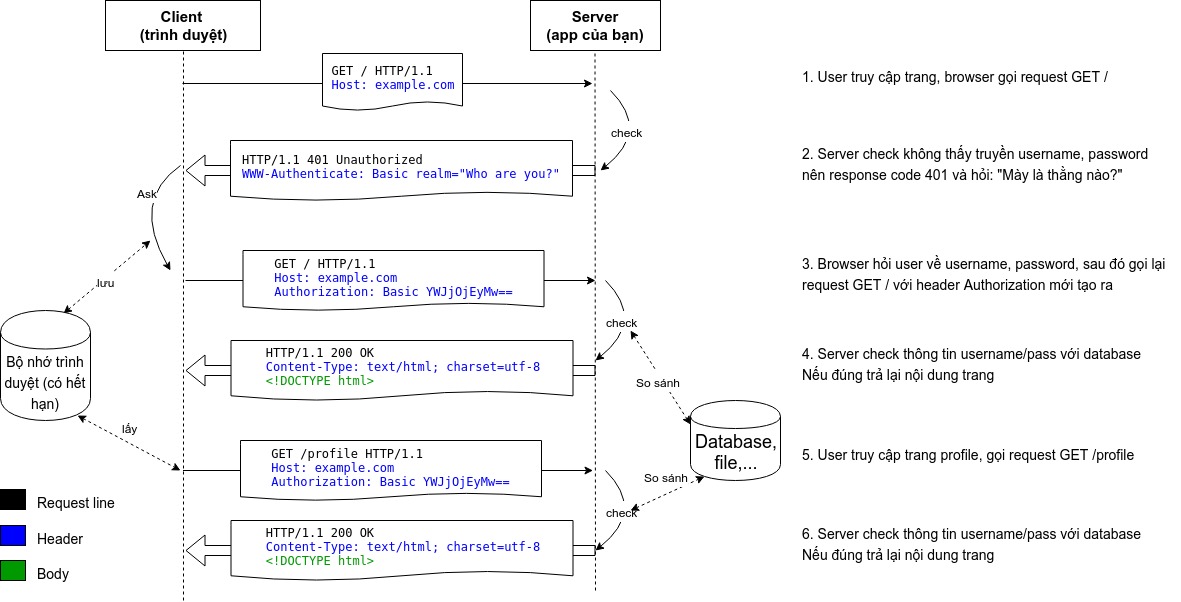
[2.3 and 2.4] – Liên hệ với **Spring Security** thì nó giống với annotation **@Secured(<<role>>)** hoặc **@PreAuthorize(“hasRole(‘<<Role>>’)”)** và **@PostAuthorize(<<EL>>)**, tham khảo [tại đây](http://websystique.com/spring-security/spring-security-4-method-security-using-preauthorize-postauthorize-secured-el/). Code thực hiện [tại đây](https://github.com/ytbhoanghai/SpringSecurity/tree/master/%5Bexample%5D_spring_session).



[2.5] – Đăng ký tại lớp cấu hình hoặc tại file web.xml



## Luồng thực hiện



## Code ví dụ

[BasicAuthentication](https://github.com/ytbhoanghai/Jersey-JAXRS/tree/master/REST/Jersey%20-%202.x/BasicAuthentication) giành cho server và [BasicAuthenticationClient](https://github.com/ytbhoanghai/Jersey-JAXRS/tree/master/REST/Jersey%20-%202.x/BasicAuthenticationClient) giành cho client.