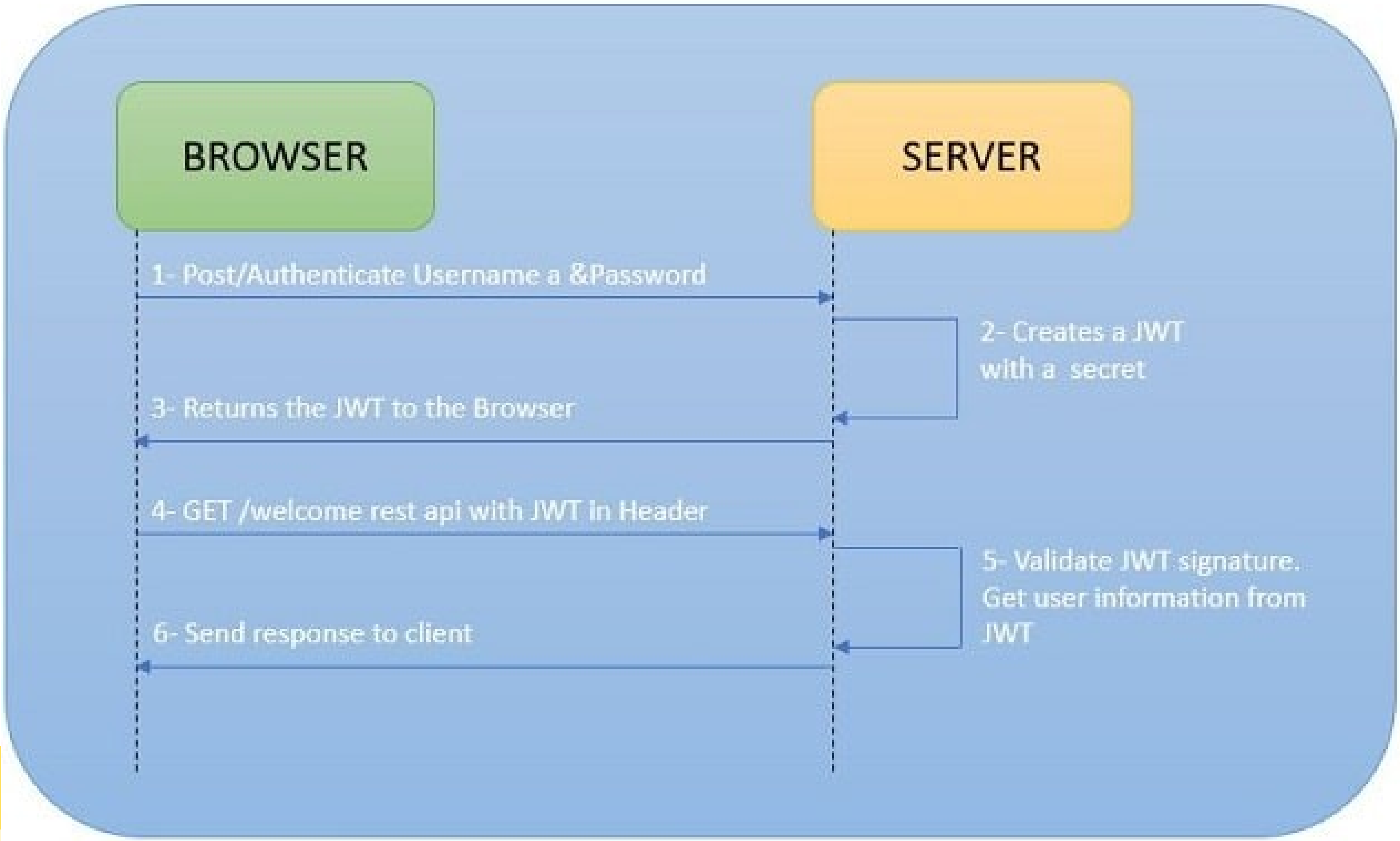
JSON Web Toolkit

JWT


BROWSER
SERVER
1- Post/Authenticate Username a &Password
2- Creates a JWT with a secret
3- Returns the JWT to the Browser
4- GET /welcome rest api with JWT in Header
5- Validate JWT signature. Get user information from JWT
6- Send response to client

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>spring-security-test</artifactId>
<scope>test</scope>
</dependency>
```

JWT Maven dependency

```
<dependency>
<groupId>io.jsonwebtoken</groupId>
<artifactId>jjwt</artifactId>
<version>0.9.1</version>
</dependency>
```

- Generate JWT : Use /authenticate POST endpoint by using username and password to generate a JSON Web Token (JWT).
- Validate JWT : User can use /greeting GET endpoint by using valid JSON Web Token (JWT).

```java
public static final long JWT_TOKEN_VALIDITY = 5 * 60 * 60;

@Value("$mysecrete.key")
private String sampleSecurityKey;
```

JWT Token Utility

```java
public String getUsernameFromToken(String token) {
return getClaimFromToken(token, Claims::getSubject);
}

public Date getIssuedAtDateFromToken(String token) {
return getClaimFromToken(token, Claims::getIssuedAt);
}

public Date getExpirationDateFromToken(String token) {
return getClaimFromToken(token, Claims::getExpiration);
}

public <T> T getClaimFromToken(String token, Function<Claims, T> claimsResolver) {
final Claims claims = getAllClaimsFromToken(token);
return claimsResolver.apply(claims);
}

private Claims getAllClaimsFromToken(String token) {
return Jwts.parser().setSigningKey(sampleSecurityKey).parseClaimsJws(token).getBody();
}
```

JWT Token Utility

```java
private Boolean isTokenExpired(String token) {
final Date expiration = getExpirationDateFromToken(token);
return expiration.before(new Date());
}

private Boolean ignoreTokenExpiration(String token) {
// here you specify tokens, for that the expiration is ignored
return false;
}

public String generateToken(UserDetails userDetails) {
Map<String, Object> claims = new HashMap<>();
return doGenerateToken(claims, userDetails.getUsername());
}

private String doGenerateToken(Map<String, Object> claims, String subject) {

return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
.setExpiration(new Date(System.currentTimeMillis() + JWT_TOKEN_VALIDITY * 1000))
.signWith(SignatureAlgorithm.HS512, sampleSecurityKey).compact();
}

public Boolean canTokenBeRefreshed(String token) {
return (!isTokenExpired(token) || ignoreTokenExpiration(token));
}

public Boolean validateToken(String token, UserDetails userDetails) {
final String username = getUsernameFromToken(token);
return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
}
```

Load Username and Password

- It searches the **username**, **password** and **GrantedAuthorities** for given user.
- This interface provide only one method called **loadUserByUsername**. **Authentication Manager** calls this method for getting the user details from the database when authenticating the user details provided by the user.
- Now we will using hard coded username password. Use BCrypt password, can use any online tool to BCrypt the password.

Load Username and Password

```
@Service
public class JwtUserDetailsService implements UserDetailsService{

@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
// TODO Auto-generated method stub
if ("tbarua1".equals(username)) {
return new User("tbarua1", "$2a$12$rT5KGSp7cpAQ3qku8MzKg.Fpay.Pe.yGbXCXKD.ujxSb80C5yak3W",new ArrayList<>());
} else {
throw new UsernameNotFoundException("User not found with username: " + username);
}
}

}
```

JWT Request Filter

- **JwtRequestFilter** class is executed for any incoming requests and validate JWT from the request and sets it in the context to indicate that logged in user is authenticated.