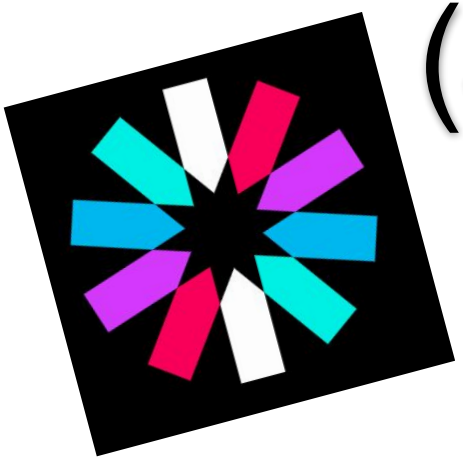
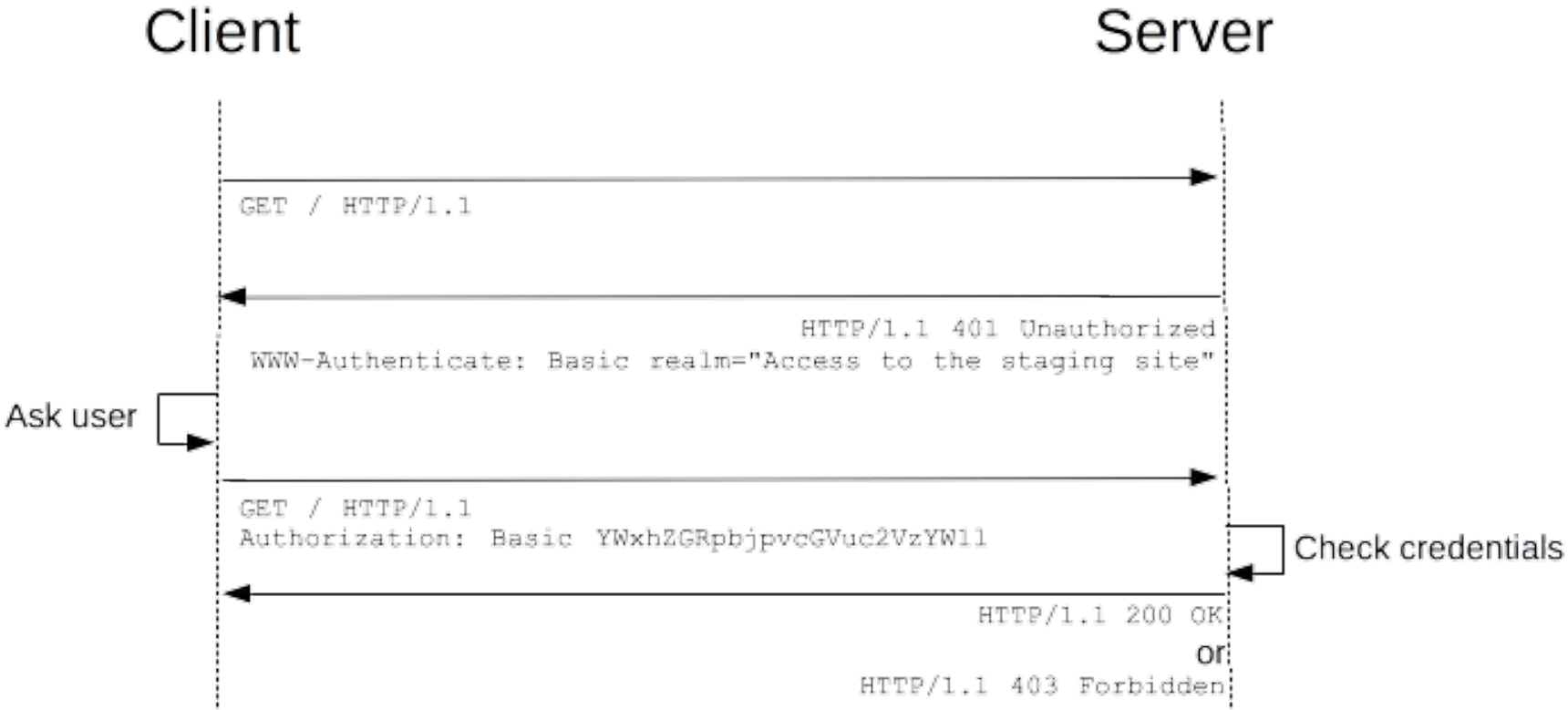
Basic Authentication
Soteria
(Json)WebToken



Basic Authentication



Basic authentication (Header)

```
Authorization: Basic username:password
```

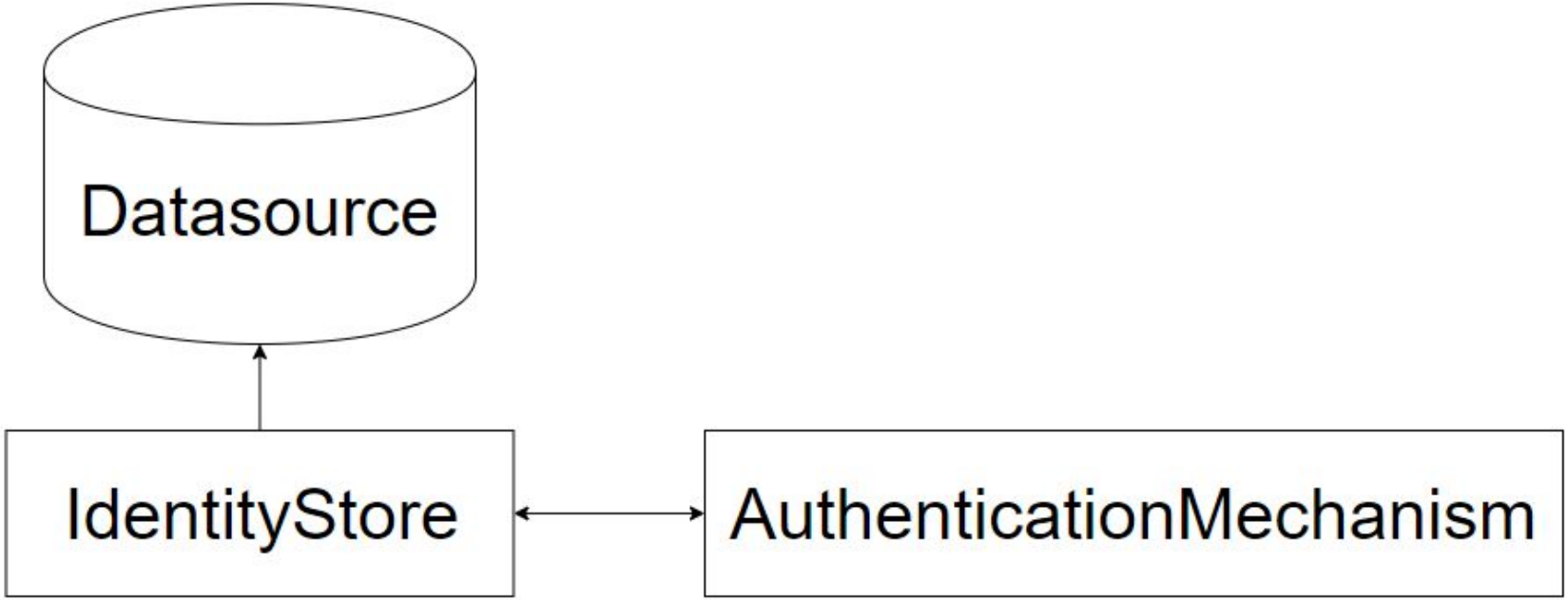
Base64-codiert

```
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

Soteria

```
<dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-api</artifactId>
    <version>8.0</version>
</dependency>
```

Soteria



Identity Store

```
public interface IdentityStore {
    CredentialValidationResult validate(Credential credential);
    Set<String> getCallerGroups(CredentialValidationResult result);
    int priority();
    Set<ValidationType> validationTypes();
    enum ValidationType { VALIDATE, PROVIDE_GROUPS }
}
```

Rückgabewert CredentialValidationResult

```
public class CredentialValidationResult {
    public Status getStatus() {...}
    public CallerPrincipal getCallerPrincipal() {...}
    public Set<String> getCallerGroups() {...}
    public enum Status { NOT_VALIDATED, INVALID, VALID }
}
```

IdentityStore basic implementations

@LdapIdentityStoreDefinition

@DatabaseIdentityStoreDefinition

@EmbeddedIdentityStoreDefinition

AuthenticationMechanism

```
public interface HttpAuthenticationMechanism {
    AuthenticationStatus validateRequest(
            HttpServletRequest request,
            HttpServletResponse response,
            HttpMessageContext httpMessageContext) throws AuthenticationException;
    AuthenticationStatus secureResponse(...) throws AuthenticationException;
    void cleanSubject(...) throws AuthenticationException;
}
```

AuthenticationMechanism “Response”

```
public interface HttpMessageContext {
    boolean isProtected();
    AuthenticationStatus responseUnauthorized();
    AuthenticationStatus responseNotFound();
    AuthenticationStatus notifyContainerAboutLogin(String callername, Set<String> groups);
    AuthenticationStatus doNothing();
}
```

AuthenticationMechanism basic implementation

@BasicAuthenticationMechanismDefinition

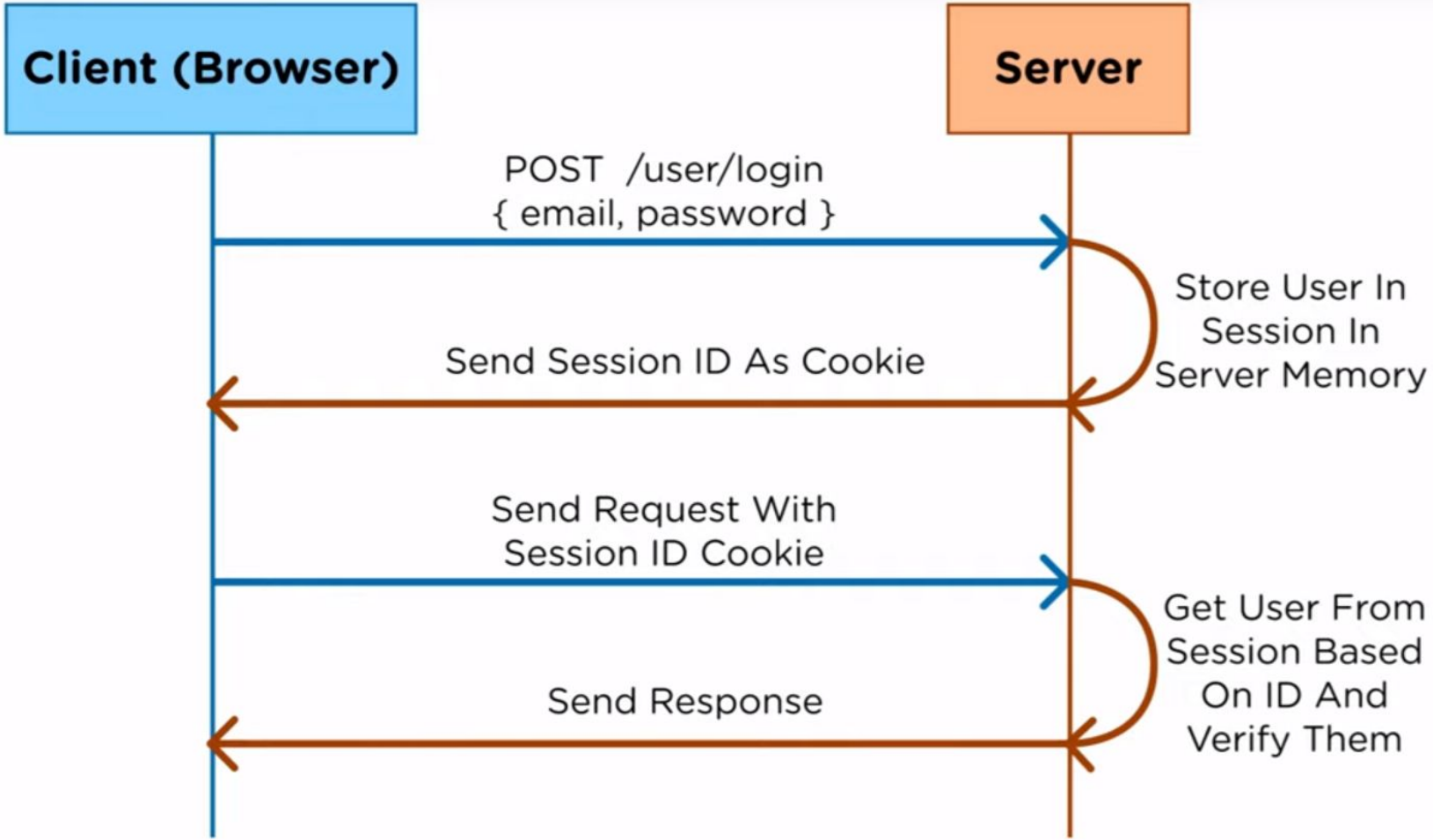
@FormAuthenticationMechanismDefinition

(loginToContinue = @LoginToContinue())

@CustomFormAuthenticationMechanismDefinition

(loginToContinue=@LoginToContinue())

JSON Web Token


Client (Browser)
Server
POST /user/login
{ email, password }
Store User In Session In Server Memory
Send Session ID As Cookie
Send Request With Session ID Cookie
Get User From Session Based On ID And Verify Them
Send Response

Json Web Token Aufbau

HEADER.PAYLOAD.SIGNATURE

https://jwt.io/

bei erhalt dann im Http-Header wie folgt angeben:

```
Authorization: Bearer <token>
```

predefined claims (Key:Value)

iss = Issuer

exp = expiration time

sub = subject

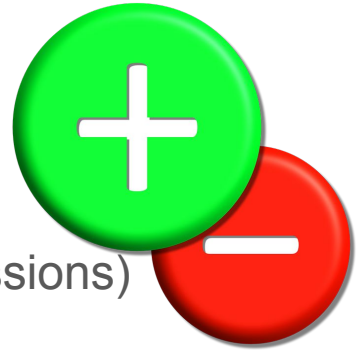
aud = audience

nbf = Not Before

iat = Issued At

...

Json Web Token



+ braucht nur das secret um zu validieren(keine gespeicherten Sessions)
+ mehrer Server -> nicht mehrmals anmelden
+ Änderungen machen das Token ungültig

- http://cryto.net/~joepie91/blog/2016/06/13/stop-using-jwt-for-sessions/
- groß
- können nicht gelöscht werden
- kann alte Daten enthalten
- ...