# Authentifizierung & Session Managment

aka. Login



## Registrierung

```
import javax.persistence.Entity;
@Entity
public class User {
   P.L.
   @GeneratedValue
   private int id;
   private String username;
   private String password;
   // getter und setter...
```

Bei der Registrierung wird schlicht und einfach ein User-Objekt erstellt und in der Datenbank gespeichert.



## Login: Erwartungen als Nutzer

- Ich kann mich nur mit Nutzername und korrektem Passwort einloggen
- Einmal eingeloggt, werde ich wiedererkannt und muss für absehbare Zeit mein Passwort nicht erneut eingeben
- Ich kann auf mehreren Geräten/Browsern gleichzeitig eingeloggt sein
- Mein Passwort wird so wenig wie möglich übertragen und gespeichert
- Ich kann mich wieder ausloggen
- Nach einer gewissen Zeit werde ich automatisch ausgeloggt



## Login: Erwartungen als Entwickler

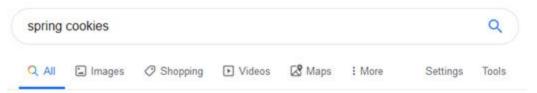
- Mehrere Nutzer können gleichzeitig eingeloggt sein
- Ich kann in allen Controller-Methoden einfach feststellen, welcher Nutzer den Request gemacht hat



## Wiederholung: Cookies

- Teil des HTTP-Protokolls
- Kleine Datenmengen werden bei der Response vom Servern an den Client geschickt
- Client speichert die Daten und schickt sie bei folgenden Requests mit





About 948 000 000 results (0,56 seconds)

#### Images for spring cookies



→ More images for spring cookies

Report images

#### Our Most Delicious Spring Cookies - Two Sisters

https://www.twosisterscrafting.com/our-most-delicious-spring-cookies/ \*

Feb 14, 2019 - We've compiled all of Our Most Delicious Spring Cookies in one place for you to enjoy. ... Our Carrot Cake Cookies with Cream Cheese Frosting feature a light, and fluffy cookie chocked full of carrots and cinnamon and topped with delicious cream cheese frosting. ... Make a batch of these ...

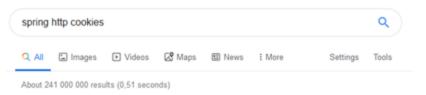
#### 43 Dainty Spring Cookies to Bake for the Ones You Love | Taste of ...

https://www.tasteofhome.com/collection/dainty-spring-cookies/ \*

These spring cookies are sweet, petite and so fun to eat! Make a batch today and share them with somebody special.

#### Spring Cookies Recipes | Land O'Lakes





#### A Guide To HTTP Cookies In Java | Baeldung

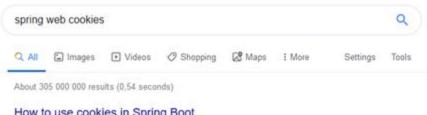
https://www.baeldung.com/cookies-java •

May 8, 2019 - Overview. In this article, we are going to explore low-level operations with Java network programming. We'll be taking a deeper look at Cookles ...

#### How to use cookies in Spring Boot

https://attacomsian.com/blog/cookies-spring-boot \*

Apr 24, 2019 - Spring framework provides @CookieValue annotation to get the value of any HTTP cookie without iterating over all the cookies fetched from the ...



#### How to use cookies in Spring Boot

https://attacomsian.com/blog/cookies-spring-boot .

Apr 24, 2019 - In this tutorial, we will learn how to read, set and remove HTTP cookies in a Spring Boot application. ... Spring framework provides @CookieValue annotation to get the value of any HTTP cookie without iterating over all the cookies fetched from the request. ... To set a cookie in Spring ...

#### Spring cookies tutorial - working with cookies in Spring application

zetcode.com/spring/cookies/ \*

May 4, 2019 - Cookies. Cookie is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with the next ...



About 74 700 000 results (0,55 seconds)

#### How to use cookies in Spring Boot

https://attacomsian.com/blog/cookies-spring-boot \*

Apr 24, 2019 - To set a cookie in Spring Boot, we can use HttpServletResponse class's method addCookie() . All you need to do is to create a new instance of ...

#### A Guide To HTTP Cookies In Java | Baeldung

https://www.baeldung.com/cookies-java \*

May 8, 2019 - A quick and practical quide to HTTP Cookies in Java. ... new Learn Spring course. focused on the fundamentals of Spring 5 and Spring Boot 2:

#### Videos





## Spring: Cookie lesen

Parameter für Controller-Methode:

```
@GetMapping("/")
public String home(@CookieValue(value="food", defaultValue = "pizza") String food, Model model) {
    model.addAttribute("food", food);
    return "home";
}
```



## Spring: Cookie schreiben

```
@GetMapping("/")
public String home(HttpServletResponse response, Model model)
{
   food = "Pizza";
   Cookie newCookie = new Cookie("food", food);
   response.addCookie(newCookie);

   return "home";
}
```

Obacht: Cookie-Werte dürfen nur bestimmte Zeichen enthalten. Verboten sind u.a. Leerzeichen und Komma!



## Vorschlag: Eingeloggter Nutzer wird in Service/Datenbank gespeichert

Nur ein eingeloggter Nutzer 😑



## Vorschlag: Nutzername/-ld im Cookie

Mit den Entwickler-Tools kann ich mich ohne Passwort einloggen ®



## Vorschlag: Nutzername und Passwort in Cookies

Passwort wird im Klartext im Browser gespeichert



## Lösungsansatz: Sessions

Für jeden erfolgreichen Login wird eine zufällige Session-ID erzeugt und mit Verbindung zum User in der Datenbank gespeichert.

Die Session-ID wird im Cookie gespeichert.

Sessions können Ablaufdatum haben und einzeln gelöscht (ausgeloggt) werden.



## Session Entity

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import java.time.Instant;
import java.util.UUID;
@Entity
public class Session {
   6 Td
  private String id = UUID.randomUUID().toString();
   @ManyToOne
   private User user;
  private Instant expiresAt;
   // Constructor, getter, setter...
```

Ein Objekt pro Nutzer und Gerät/Browser, erstellt bei erfolgreichem Login.

Wenn im Request ein SessionId-Cookie mitgeschickt wird, so suchen wir nach dem passenden, nicht-abgelaufenen Eintrag in der Datenbank.

Der zugehörige Nutzer gilt für den Request als eingeloggt.



## Der Login

```
@PostMapping("/login")
public String login(HttpServletResponse response) {
   Optional < User > optional User = userRepository.findByUsernameAndPassword ("...", "...");
   if (optionalUser.isPresent()) {
       Session session = new Session(optionalUser.get(), Instant.now().plusSeconds(7*24*60*60));
       sessionRepository.save(session);
       Cookie cookie = new Cookie("sessionId", session.getId());
       response.addCookie(cookie);
       // Login erfolgreich
       return "redirect:/";
   // Login nicht erfolgreich
   return "login";
```



## Session Cookie behandeln

```
@GetMapping("/")
public String home(@CookieValue(value="sessionId", defaultValue = "") String sessionId) {
   if (!sessionId.isEmpty()) {
       Optional < Session > optional Session = sessionRepository.findByIdAndExpiresAtAfter(
           sessionId, Instant.now());
       if (optionalSession.isPresent()) {
           Session session = optionalSession.get();
           // neues Ablaufdatum für die Session
           session.setExpiresAt(Instant.now().plusSeconds(7*24*60*60));
           User user = session.getUser();
           // User ist eingeloggt....
   return "home";
```

## Session Cookie behandeln

... in jeder einzelnen Controller-Methode?

Der Code könnte in einen Service ausgelagert werden, allerdings müsste dann immer noch jede Controller-Methode den CookieValue weitergeben.

Ziel: Der eingeloggte Nutzer (oder null) steht überall als ModelAttribute zur Verfügung

Lösung dafür: eine ControllerAdvice Klasse



## SessionControllerAdvice

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.CookieValue;
import org.springframework.web.bind.annotation.ModelAttribute;
import java.time.LocalDateTime;
import java.util.Optional;
@ControllerAdvice
public class SessionControllerAdvice {
  private SessionRepository sessionRepository;
   @Autowired
  public SessionControllerAdvice(SessionRepository sessionRepository) {
       this.sessionRepository = sessionRepository;
   @ModelAttribute("sessionUser")
  public User sessionUser(@CookieValue(value = "sessionId", defaultValue = "") String sessionId) {
        // TODO: Logik aus dem Controller
```

## Logout

- Session aus der Datenbank löschen
- Cookie mit leerem String überschreiben

