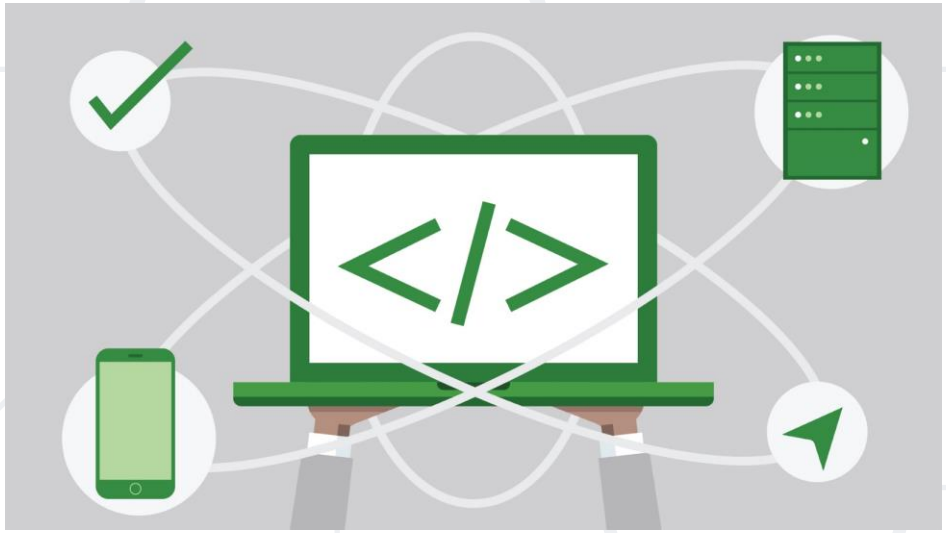
# State Management

## Cookies, Sessions and Headers




**SoftUni Team**

**Technical Trainers**

Software University

SoftUni

Software University

https://softuni.bg

# Have a Question?

Software University

sli.do

#java-web

# Table of Content

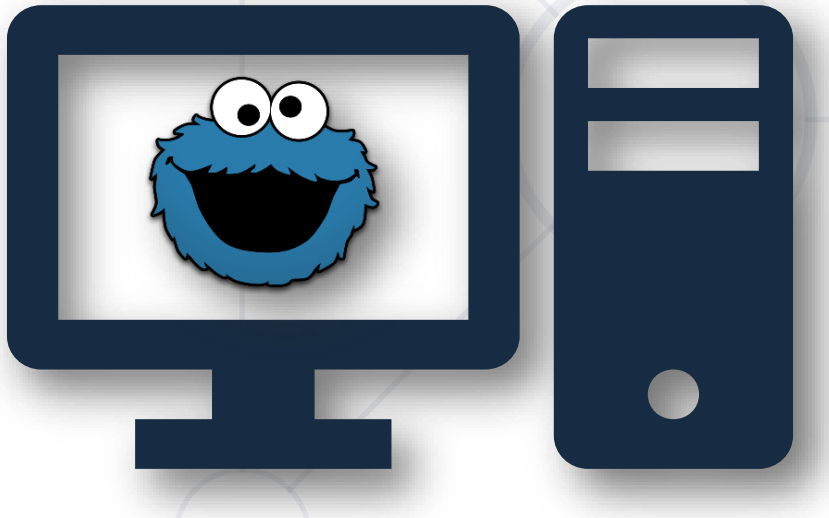
1. **HTTP** Cookies
2. **HTTP** Sessions
3. **Working** with **HTTP Sessions**, **Cookies** and **Headers**

# HTTP Cookies

## Usage and Control

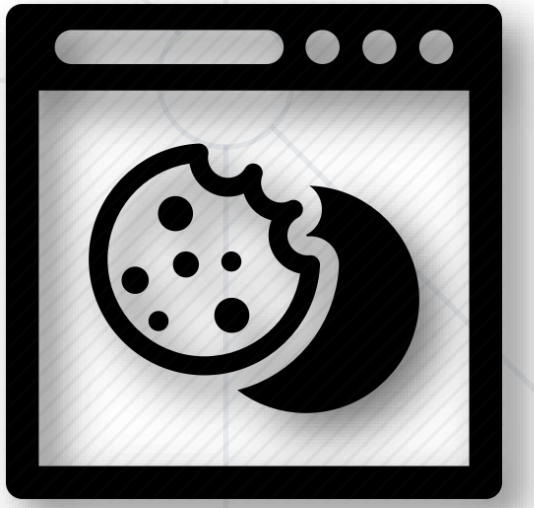
# What Are Cookies?

- A **small file** of **plain text** with no executable code
  - Sent by the server to the client's browser
  - Stored by the browser on the client's device (computer, tablet, etc.)
  - Hold small piece of data for a particular client and a website



# What Are Cookies Used for?

- **Session management**
  - Logins, shopping carts, game scores, or anything else the server should remember
- **Personalization**
  - User preferences, themes, and other custom settings
- **Tracking**
  - Recording and analyzing user behavior



# Session Management

- The HTTP object is **stateless**
  - It **doesn't store** information about the requests

# Stateless HTTP – the Problem

- The **server does not know** if two requests come from the same client
- **State management** problems
  - Navigation through pages requires **authentication each time**
  - Information about the pages is lost **between the requests**
  - **Harder personalization** of page functionality

# Stateless HTTP – the Cookie Solution

- A reliable **mechanism** for websites to **remember stateful information**
  - to know whether the user is **logged in or not**
  - to know **which account** the user is logged in with
  - to record the user's **browsing activity**
  - to remember **user preferences**, such as language preference or theme settings, selected options in a **form** (**excluding** sensitive information like **passwords** or **usernames**) or user-specific settings

# How Are Cookies Used?

- The response holds the cookies to be saved within the **Set-Cookie** header

```
HTTP/1.1 200 OK
Set-Cookie: lang=en
```

- The request holds the specific web site cookie within the **Cookie** header

```
GET /index HTTP/1.1
Cookie: lang=en
```

# Server-Client Cookies Exchange

# Cookie Structure

- The cookie consists of **Name**, **Value** and **Attributes** (optional)
- The attributes are **key-value pairs** with additional information
- Attributes are not included in the requests
- Attributes are used by the client to control the cookies

Name = Value

Attributes

```
Set-Cookie: SSID=Ap4P…GTEq; Domain=foo.com; Path=/;
Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
```

# Scope

- Defined by the attributes **Domain** and **Path**
- **Domain** – defines the website that the cookie belongs to
- **Path** – Indicates a **URL** path that must exist in the requested resource before sending the **Cookie** header

```
Set-Cookie: SSID=Ap4P…GTEq; Domain=foo.com; Path=/;
Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
```

# Lifetime

- Defined by the attributes **`Expires`** and **`Max-Age`**
- **`Expires`** – defines the date that the browser should delete the cookie
- By default the cookies are deleted after the end of the session
- **`Max-Age`** – interval of seconds before the cookie is deleted

```
Set-Cookie: SSID=Ap4P…GTEq; Domain=foo.com; Path=/;
Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
```

# Security

- Security flags do not have associated values
- **Security** - tells the browser to use cookies only via **secure/encrypted** connections
- **HttpOnly** - defines that the cookie cannot be accessed via client-side scripting languages

```
Set-Cookie: SSID=Ap4P…GTEq; Domain=foo.com; Path=/;
Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
```

# What is in the Cookie?

- The cookie file contains a table with **key-value** pairs

| | |
|---|---|
| Name: | ELOQUA |
| Content: | GUID=50B3A712CDAA4A208FE95CE1F2BA7063 |
| Domain: | .oracle.com |
| Path: | / |
| Send for: | Any kind of connection |
| Accessible to script: | Yes |
| Created: | Monday, August 15, 2016 at 11:38:50 PM |
| Expires: | Wednesday, August 15, 2018 at 11:38:51 PM |

Remove

# Examine Your Cookies

- Most cookies are stored in a **RDBMS**, usually **SQLite**
- Download **SQLite browser** from **here**
- Location of Mozilla cookies

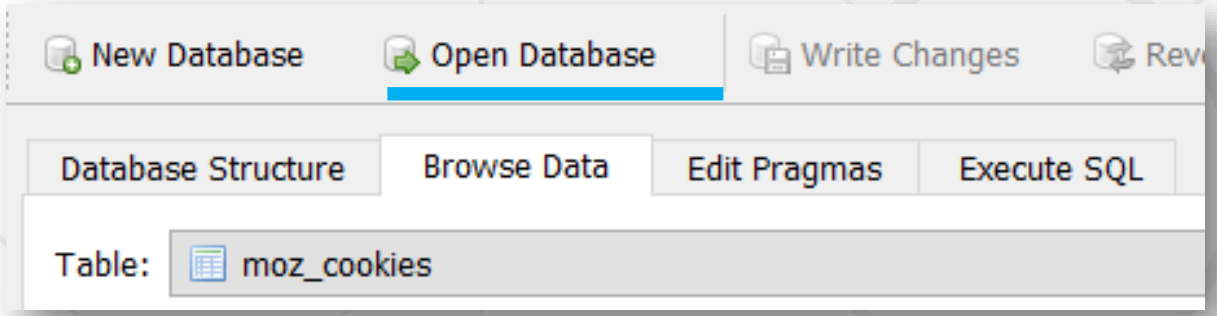
```
C:\Users\{username}\AppData\Roaming\Mozilla\Firefox\Profiles\{name}.default\cookies.sqlite
```

- Location of Chrome cookies

```
C:\Users\{username}\AppData\Local\Google\Chrome\User Data\Default\Cookies
```

# Examine Your Cookies

- Open the file with the **SQLite browser**




- Browse the cookies table

Table: moz_cookies

| | baseDomain | originAttributes | name | value | host | path | expiry | lastAccessed | creationTime |
|---|---|---|---|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | softuni | Filter | Filter | Filter | Filter |
| 1 | softuni.bg | | _ga | GA1.2.14749... | .softuni.bg | / | 1548331112 | 1485259173536000 | 1472458246652000 |
| 2 | softuni.bg | | cb-enabled | enabled | platform.soft... | / | 1512124532 | 1485213524987000 | 1480588532898000 |
| 3 | softuni.bg | | cookies-notifi... | ok | judge.softuni.... | / | 1787818276 | 1485259172447000 | 1472458276862000 |
| 4 | softuni.bg | | cb-enabled | accepted | softuni.bg | / | 1503994248 | 1485214353890000 | 1472458248921000 |

Name

Host

Expiration date

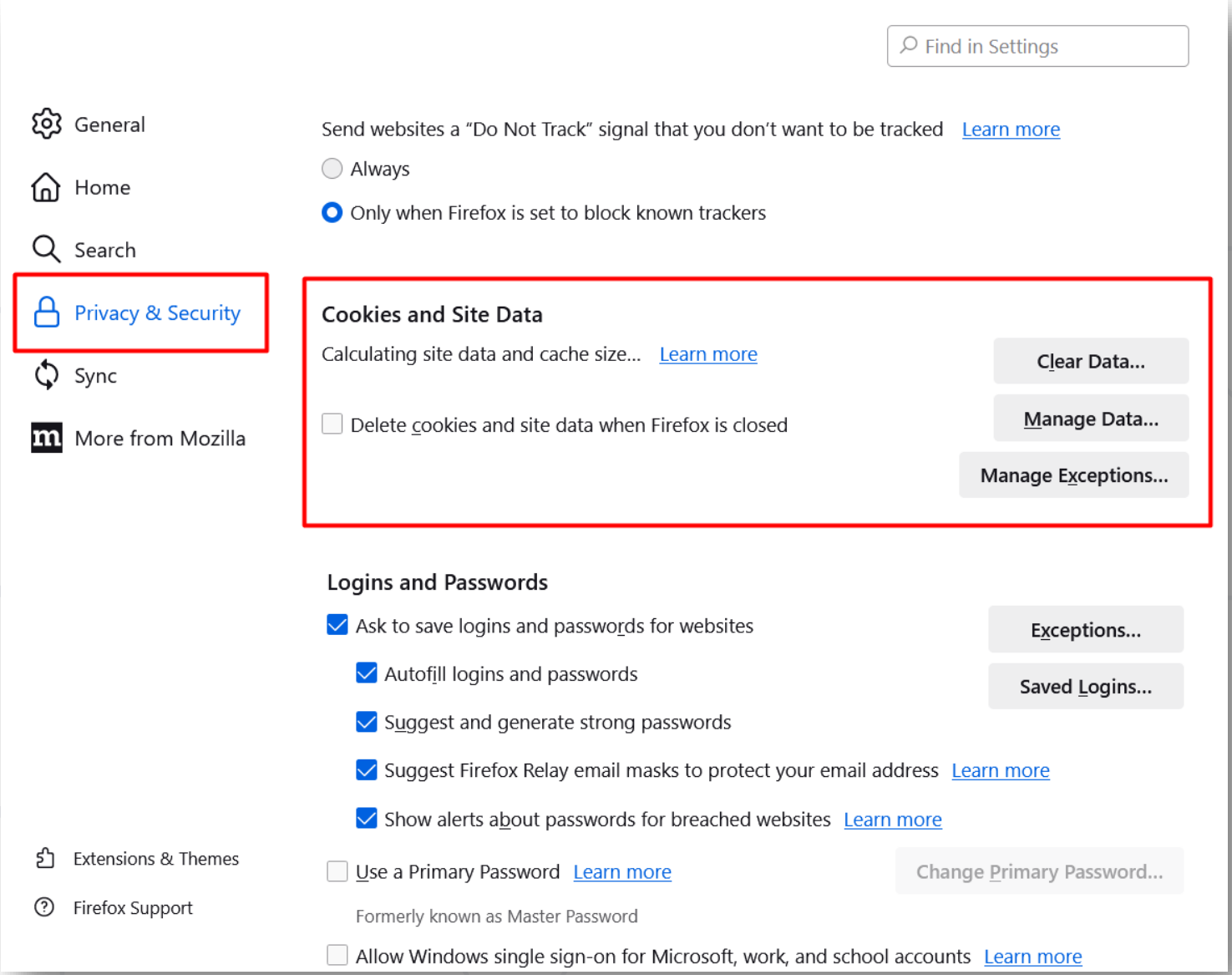
Created on

Value

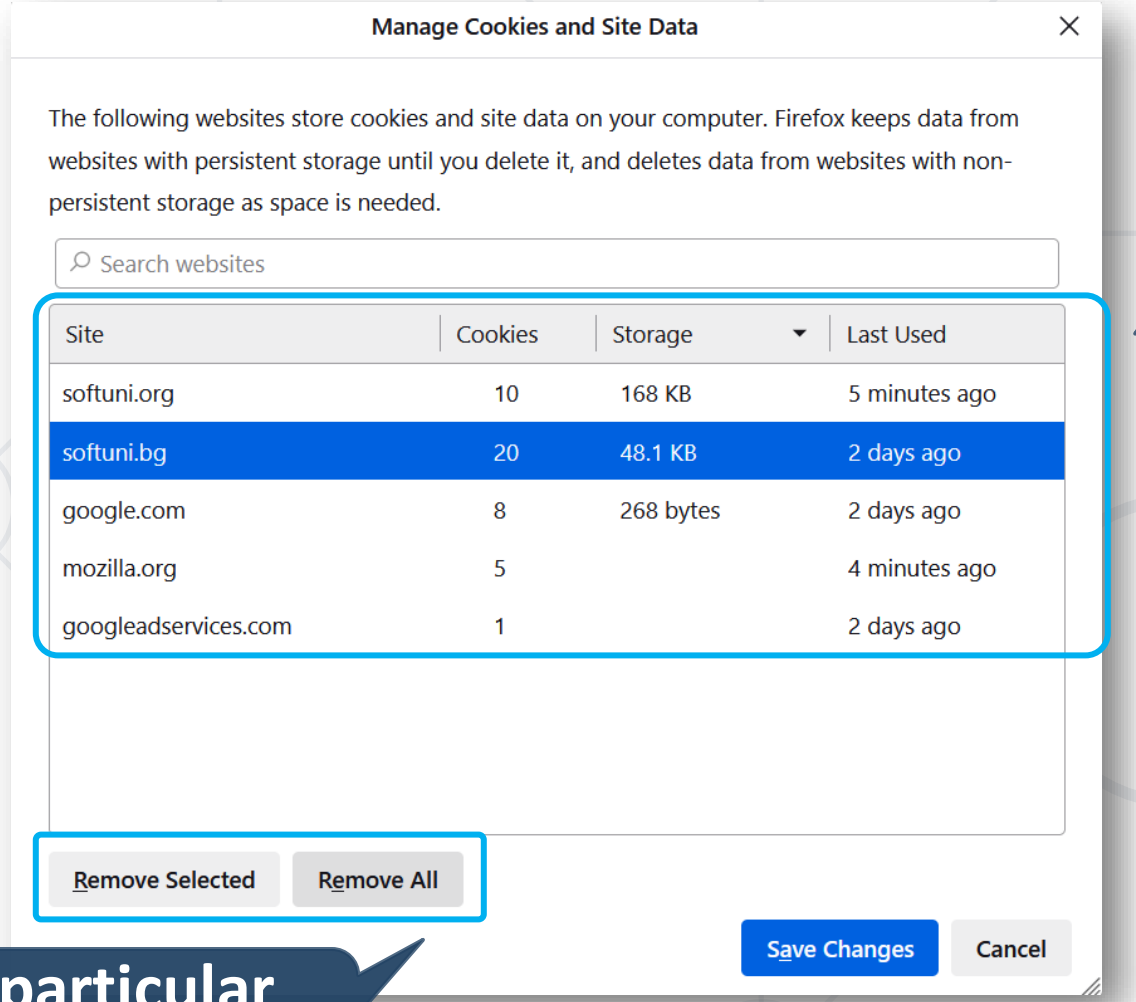
Paths

Last accessed on

# Control Your Cookies – Firefox Browser



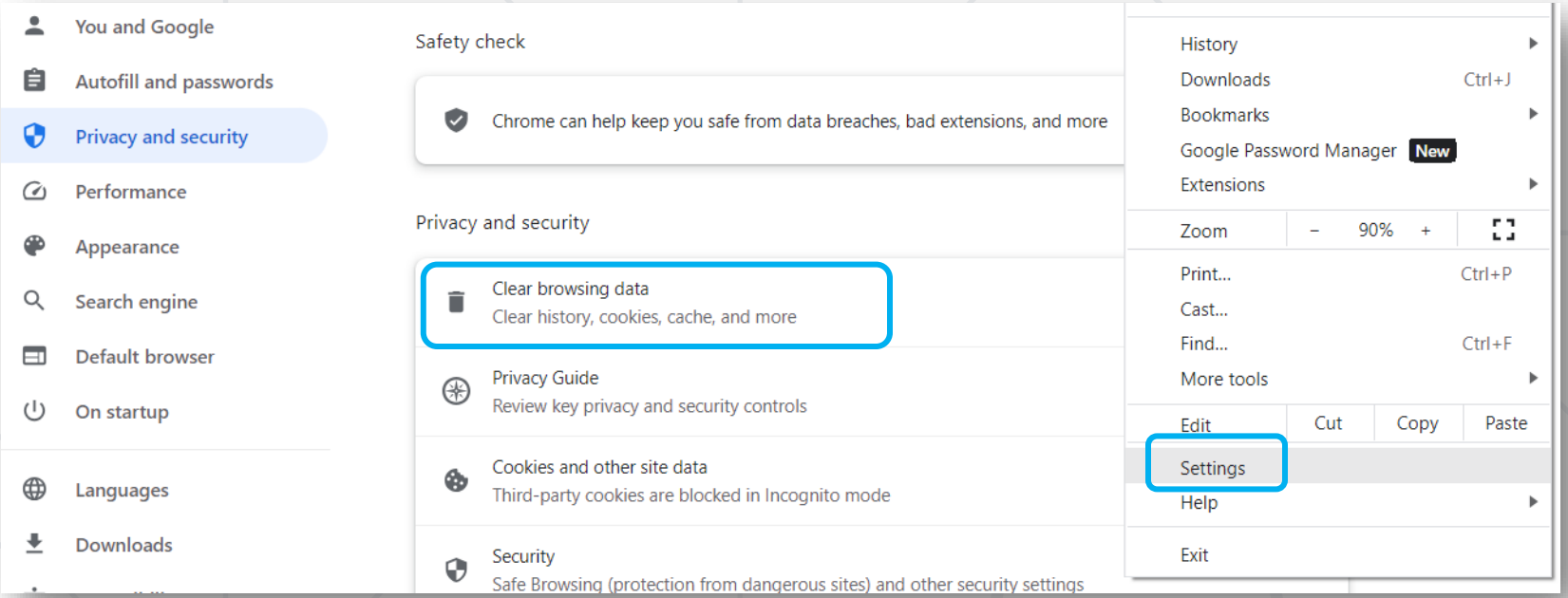
# Control Your Cookies – Firefox Browser

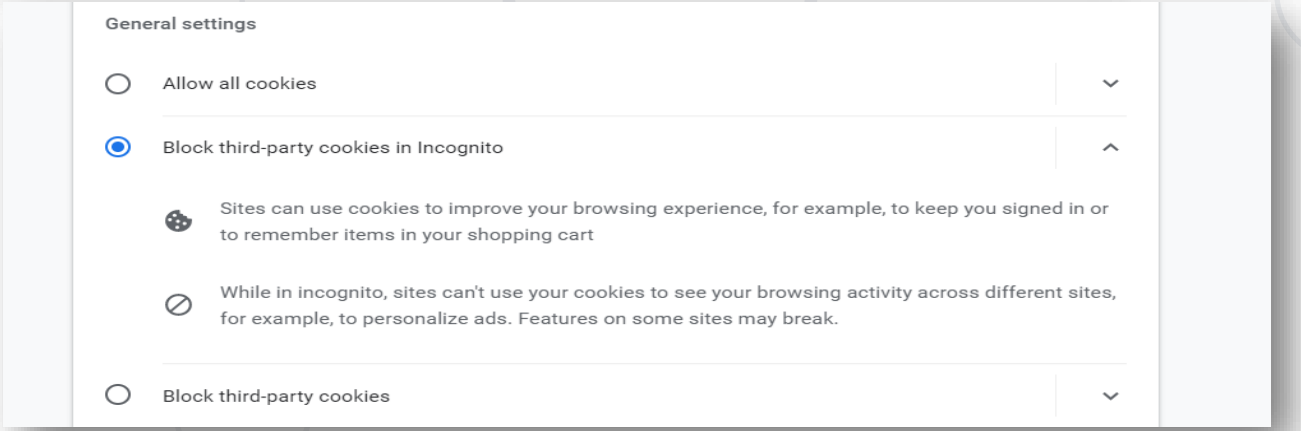



**Browse cookies from a selected website**

**Delete a particular cookie or all cookies**

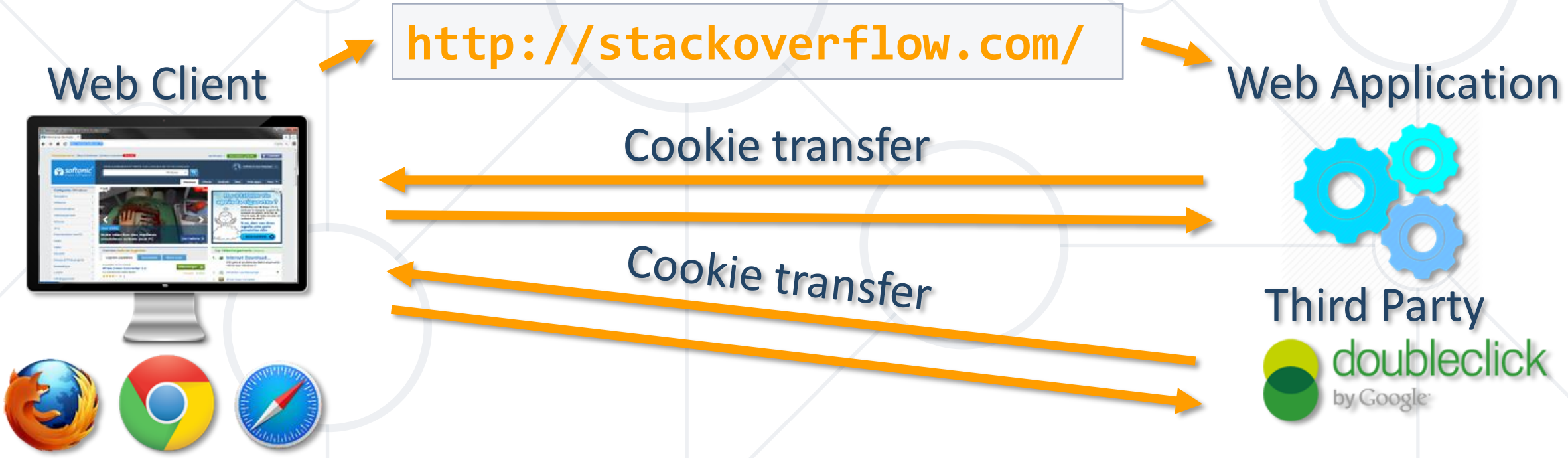
# Control Your Cookies – Chrome Browser



# Third Party Cookies

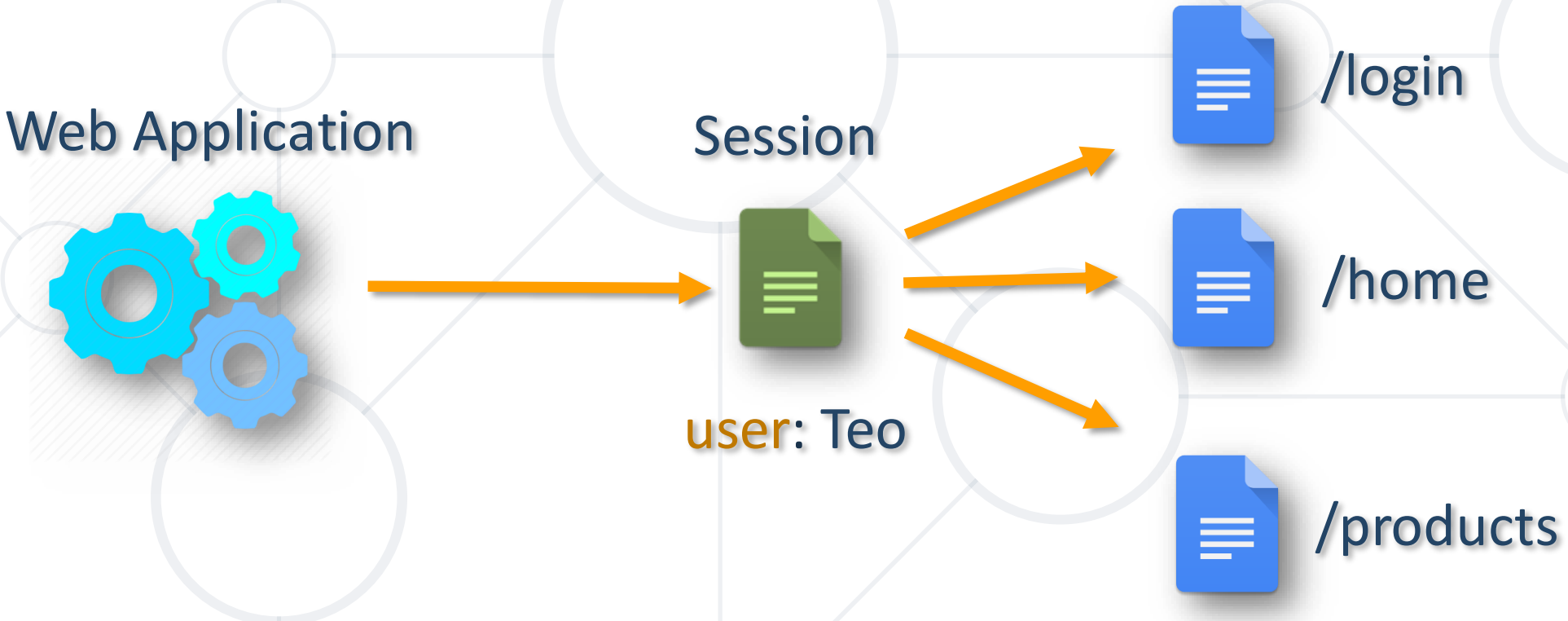
- Cookies stored by an **external party** (different **domain**)
- Mainly used for advertising and tracking across the web



# HTTP Sessions

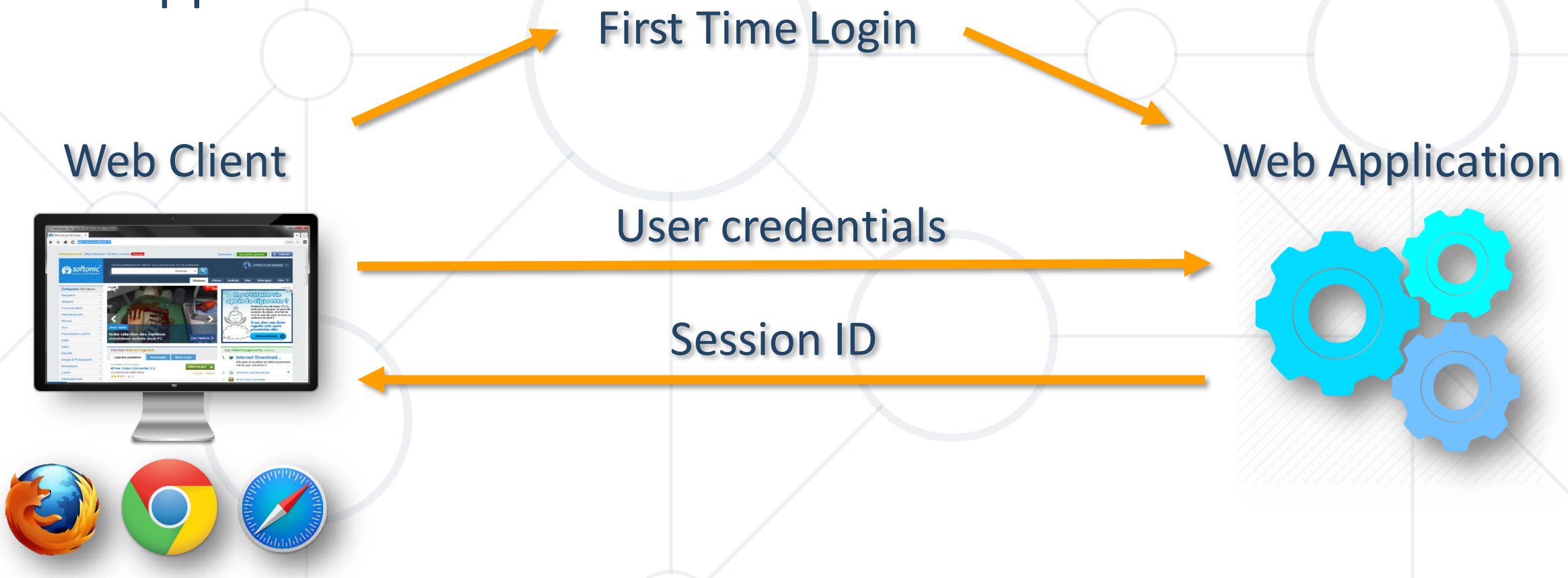
# What Are Sessions?

- A way to store information about a user to be used across **multiple pages**



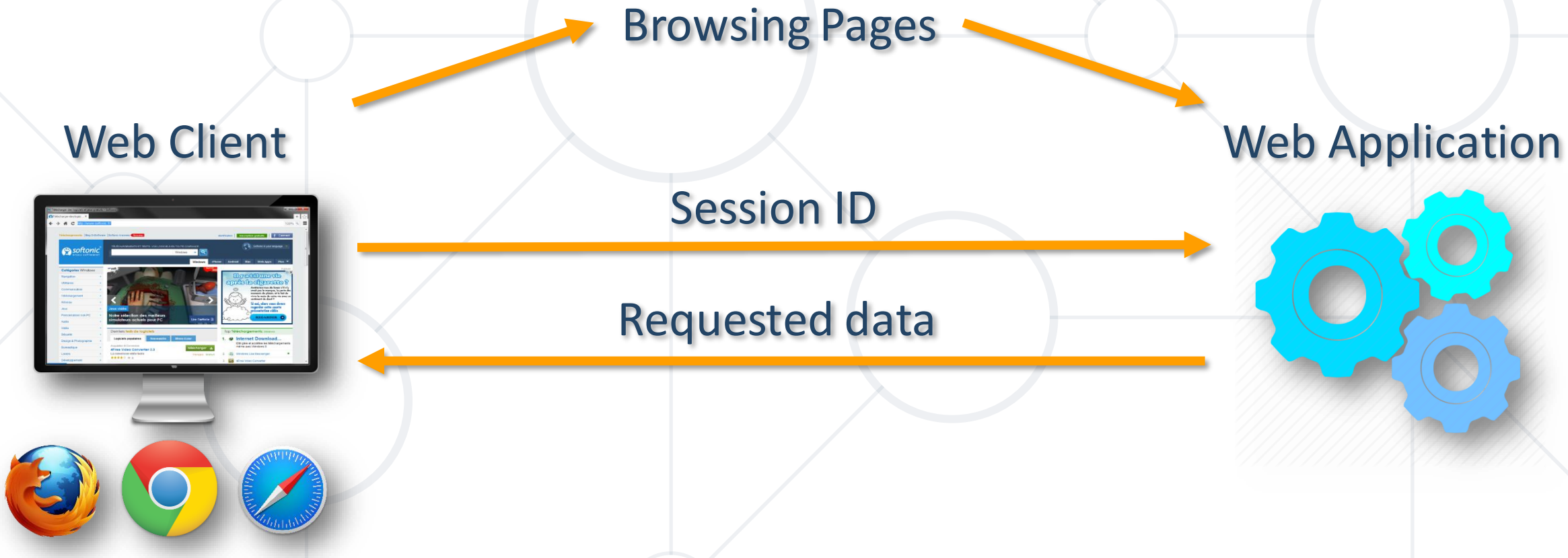
# Session Management

- The exchange mechanism be used between the user and the web application



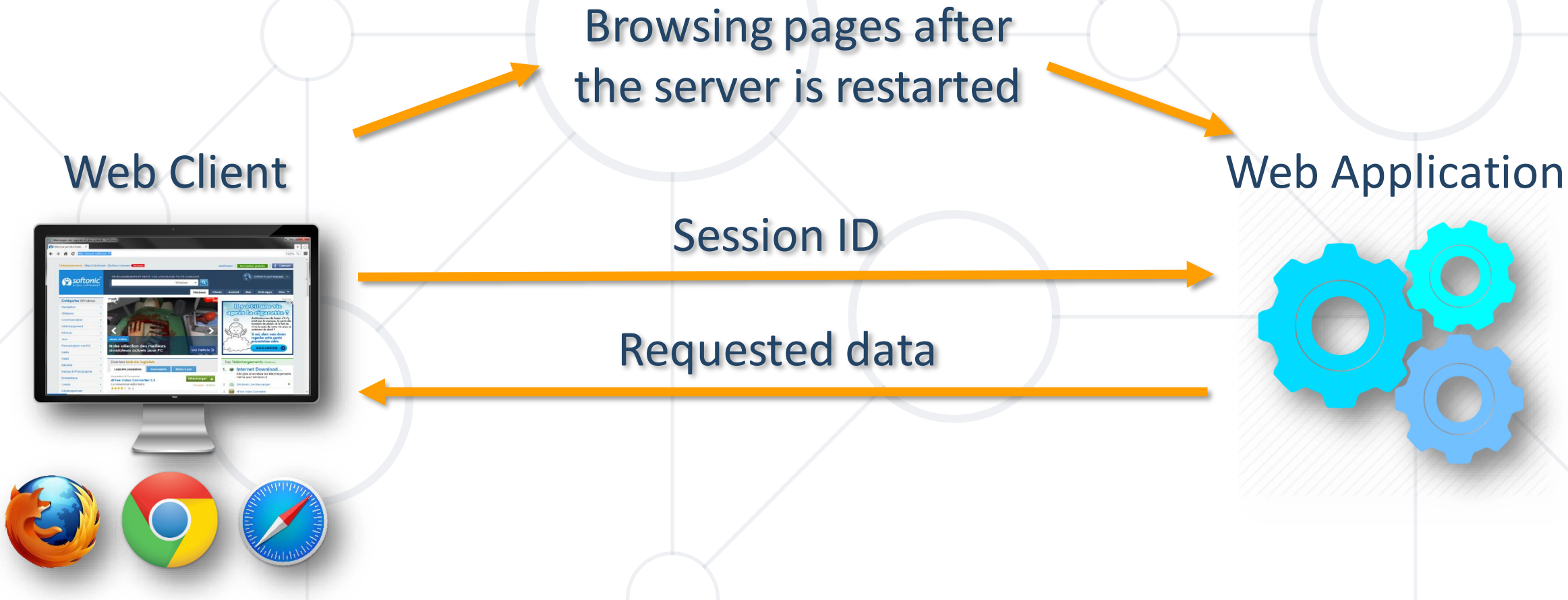
# Session Management

- The exchange mechanism be used between the user and the web application

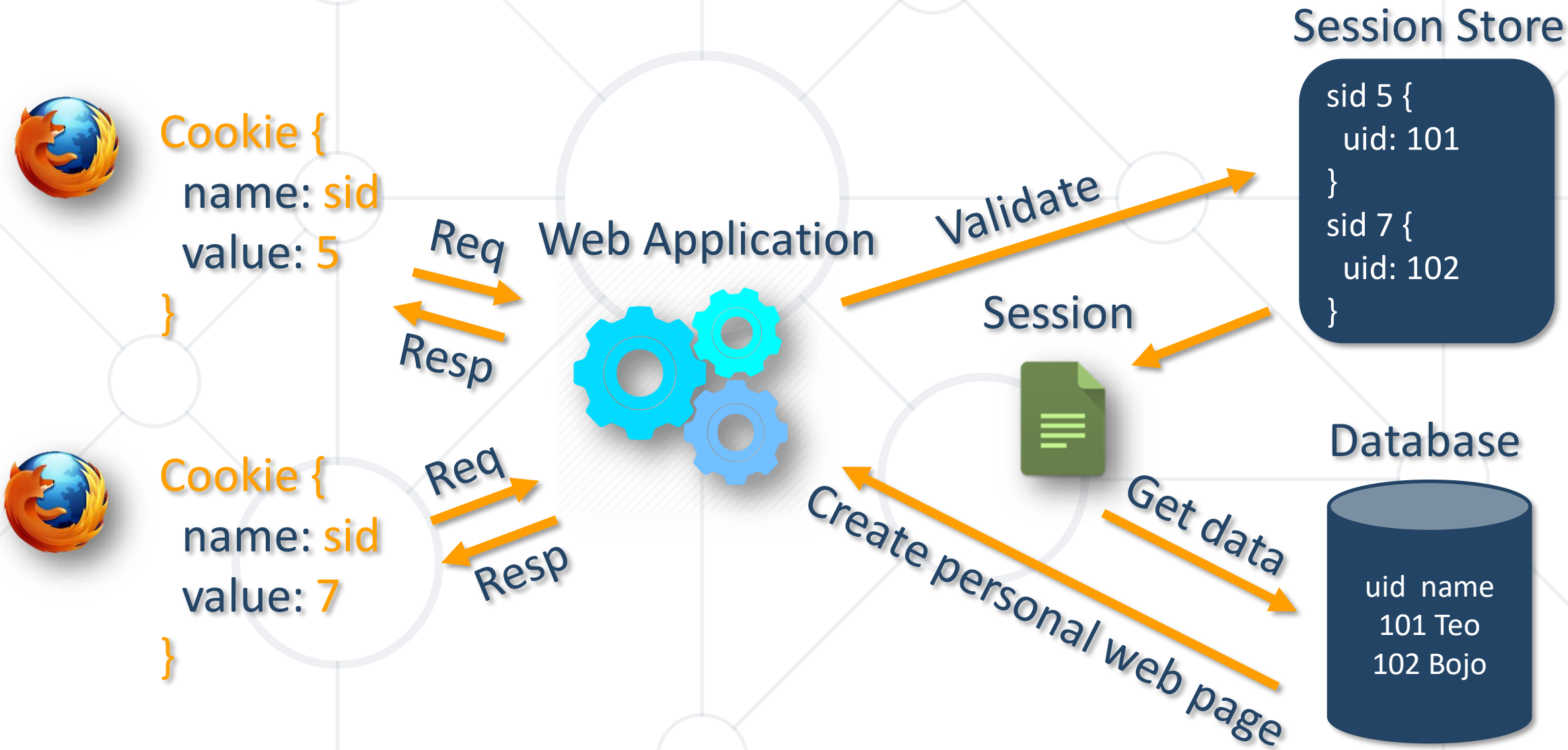


# Session Management

- The exchange mechanism be used between the user and the web application



# Relation with Cookies



# Session Structure

Unique Session ID

```
"hje85d3" : {
            user_id: 789
            username: FirstUser
          },
"af354dd" : {
            user_id: 456
            username: SecondUser
          },
"fg78e5s" : {
            user_id: 654
            username: ThirdUser
          }
```

Key-Value pairs with user data


# Working with HTTP Sessions, Cookies and Headers

# State Management

- **State management** refers to the ability to maintain and manage the state of an application across multiple requests and sessions
- It is **crucial** for web applications to remember **user data**, session information and application state as users navigate through different pages and interact with the application

# State Management

- **Form Data Binding**
  - Spring MVC supports data binding between HTML form fields and Java objects using **@ModelAttribute** and **@RequestParam** annotations
  - This allows form data to be automatically **bound** to Java objects in controller methods
  - Thymeleaf templates can bind form fields to model attributes using Thymeleaf **expressions**, allowing for two-way data binding between HTML forms and server-side Java objects

# State Management

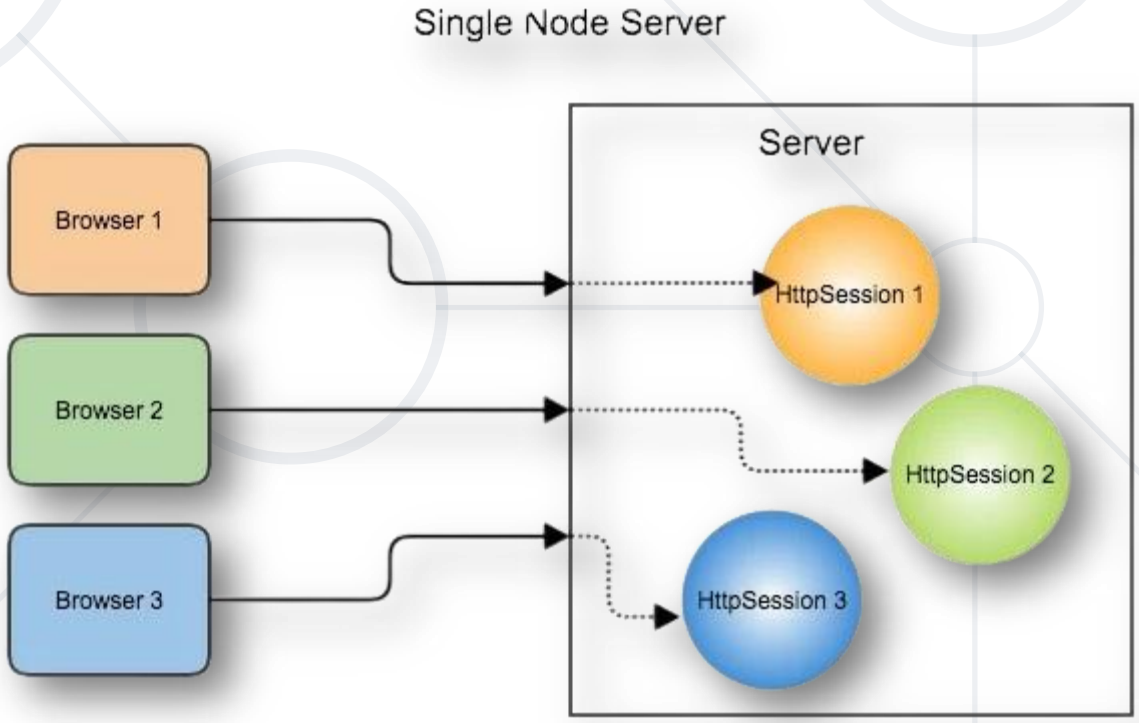
- **Flash Attributes**
  - Spring provides **flash attributes**, which are used to store data temporarily and pass it to the next request
  - Flash attributes are typically used for **redirect** scenarios, where data needs to be **preserved** across redirects

# State Management

- **Client-side State Management**
  - In addition to server-side state management, client-side state management techniques such as cookies and local storage can also be used in conjunction with Spring and Thymeleaf to store user **preferences**, session **tokens**, and other **client-specific data**
  - Thymeleaf templates can use **JavaScript** to interact with client-side storage mechanisms and manipulate client-side state as needed

# Working with HTTP Sessions

- **Spring** provides support for managing HTTP sessions through the HttpSession **object**. You can access the session object in Spring MVC controllers by injecting it as a method parameter or using the **@SessionAttributes** annotation
- To **create** or **access** session attributes, you can use methods provided by the HttpSession interface, such as **setAttribute**, **getAttribute** and **removeAttribute**

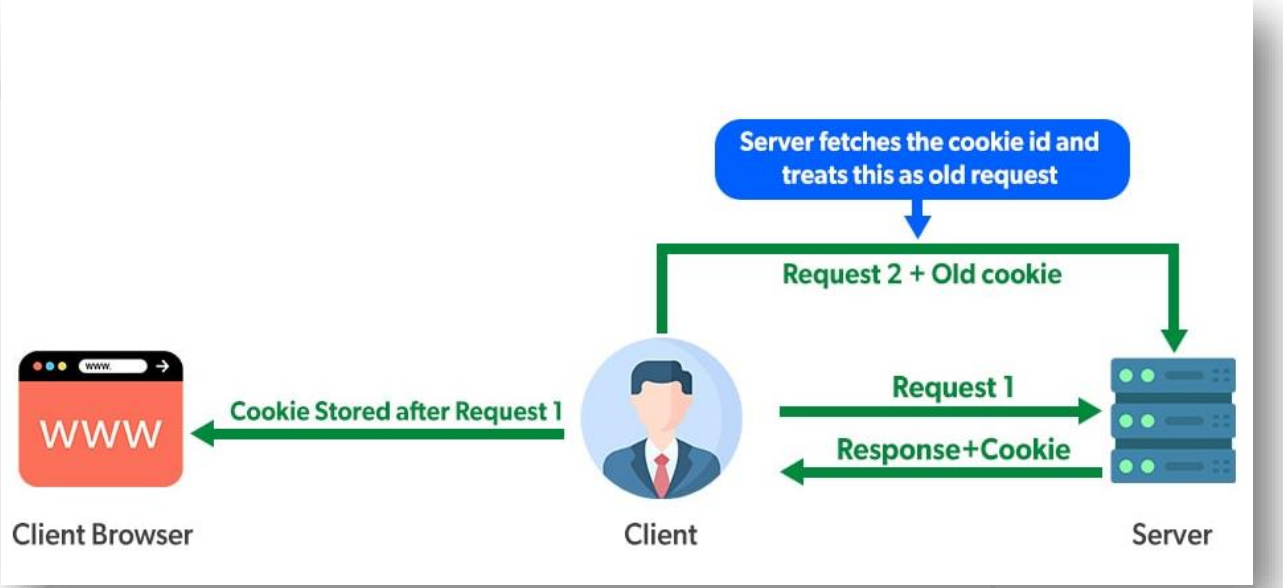


# HTTP Session in Java Servlets

```
 public class MyServlet extends HttpServlet {
        protected void doGet(HttpServletRequest request,
HttpServletResponse response) {

            HttpSession session = request.getSession();
            // Storing data in the session
            session.setAttribute("username", "exampleUser");
            // Retrieving data from the session
            String username = (String)
            session.getAttribute("username");
        }
    }
```

# Working with Cookies

- In **Spring** we use the **Cookie class** and **HttpServletResponse** object
- They are used to **create** cookies, to **add** them to the response and to **read** cookies from the request
- You can set cookies using the **addCookie** method of the HttpServletResponse object and retrieve cookies using the **getCookies** method of the **HttpServletRequest** object

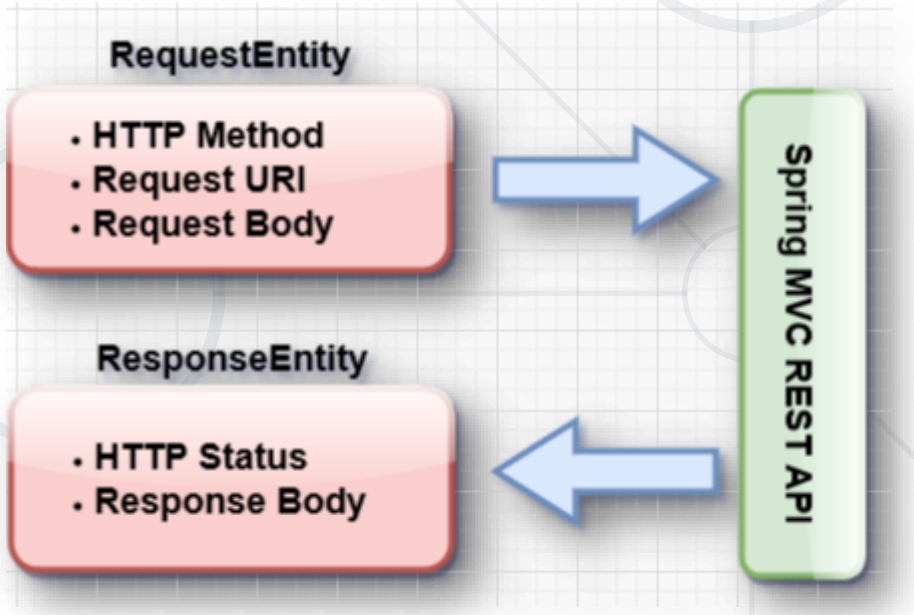


# HTTP Cookie in Java Servlets

```
public class MyServlet extends HttpServlet {
        protected void doGet(HttpServletRequest request,
     HttpServletResponse response) {
            // Here you can access the request and response objects
            // and perform operations such as getting parameters,
            setting attributes, etc.
            // Example of creating a cookie
            Cookie cookie = new Cookie("username", "exampleUser");
            cookie.setMaxAge(3600); // Cookie expires in 1 hour
            response.addCookie(cookie);

        }
    }
```

# Working with HTTP Headers

- Spring allows you to work with HTTP headers using the **HttpHeaders** class and **ResponseEntity** object. You can add custom headers to HTTP responses and retrieve headers from HTTP requests
- You can set headers using the **add** and **set** methods of the HttpHeaders class and retrieve headers using the **getFirst** and **get** methods



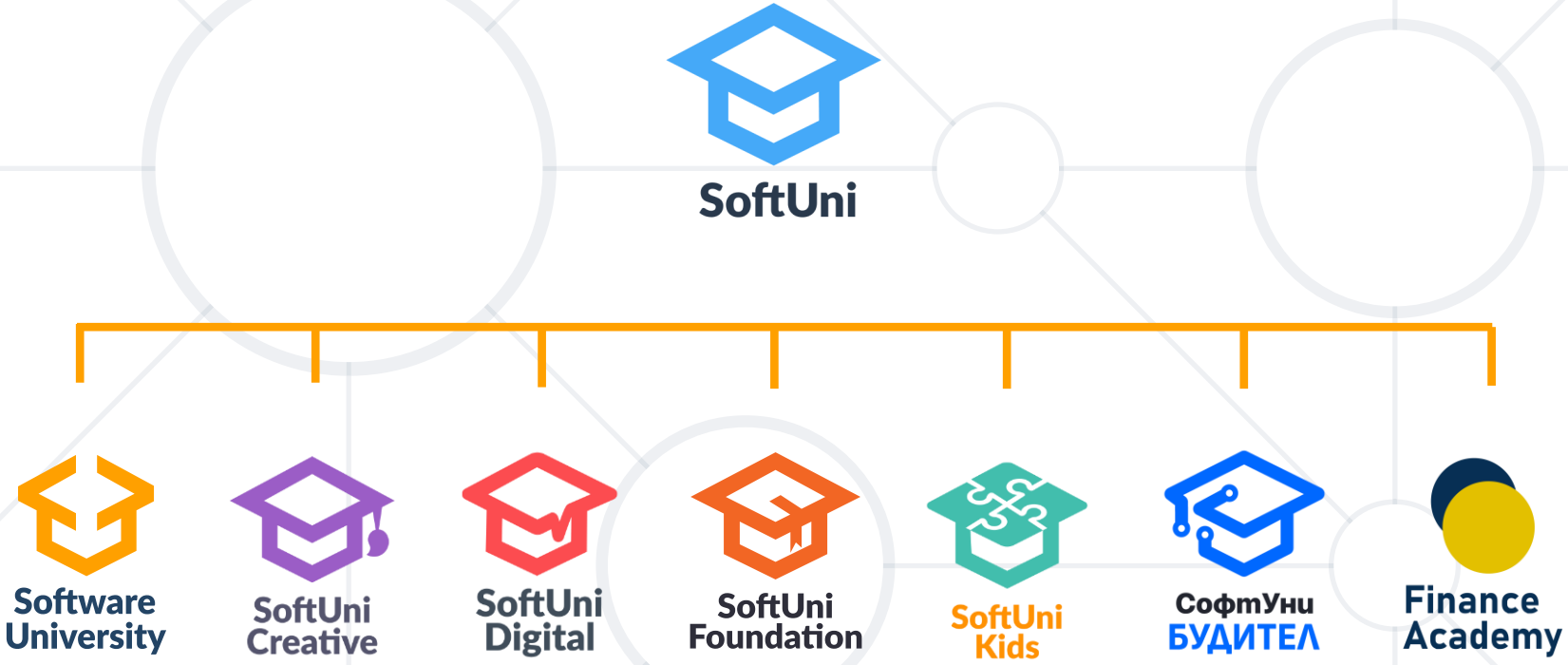
# HTTP Headers in Spring Boot

```java
@RestController
public class MyController {

    @GetMapping("/example")
    public ResponseEntity<String> getExample() {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Custom-Header", "Example");
        headers.add("Another-Header", "Value");

        return ResponseEntity.ok()
                .headers(headers)
                .body("Hello, World!");
    }
}
```

# Summary

- HTTP Cookies
- HTTP Sessions
- Working with HTTP Sessions, Cookies and Headers

# Questions?



# SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank
Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR
Serving the high achievers

SOFTWARE GROUP

PHAR VISION

# Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

# License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg