Data and Authentication

Remote Collections and User Sessions

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#js-advanced

Table of Contents

Remote Storage

Accessing Remote Data

Sending Data

- The client can **send data** to the server, usually via **POST request**

```
const data = {title:'First Post',content:'Hello, Server!'};

fetch('/articles', {
    method: 'post',
    headers: { 'Content-type': 'application/json' },
    body: JSON.stringify(data),
});
```

- This allows:
  - Specialized requests, such as **filtering** collections
  - **Permanent storage** and **sharing** of content

Request Options

- Provide an **options** object to **Fetch API** to send data:
  - `method` – can be **POST**, **PUT**, **PATCH** or **DELETE**
  - `body` – contains the **data** to be sent, usually as **JSON** string
  - `headers` – common headers include:
    - `Content-Type` specifies the **format** of the data (**manual**)
    - `Content-Length` specifies the **size** of the data (**automatic**)
    - `Cookie` can be used with **authentication** (**automatic**)
    - Custom **authorization** headers (**manual**)

Database Principles

Relational and Non-Relational Databases

Backend As a Service

- Solutions that provide **pre-built**, **cloud** hosted components for developing **application backends**
- Reduce the **time** and **complexity** required
- Allow developers to focus on **core features** instead of low-level tasks
- Types:
  - Cloud BaaS
  - Open-source BaaS

Relational Databases

- Represent and store **data** in tables and rows
- Use **S**tructured **Q**uerying **L**anguage (**SQL**)
- Allows you to **link information** from different tables through the use of **foreign keys** (or indexes)

Microsoft® SQL Server®

ORACLE DATABASE

MySQL™

Non-Relational Databases

- No-SQL databases
- More **flexibility** and **adaptability**
- Allow us to **store unstructured data** in a single document *(**not** a good idea)*
- **Additional processing** effort and **more storage** as the document sizes grow

Relational and Non-Relational Pros

- Relational
  - Work with **structured data**
  - They support **ACID** transactional consistency and support "**joins**"
  - Built-in **data integrity** and a large eco-system
  - Relationships in this system have **constraints**
  - Limitless **indexing**
- Non-Relational
  - They **scale** out **horizontally**
  - Work with **unstructured** and semi-structured **data**
  - Schema-free or Schema-on-read options
  - **High availability**
  - Many are **open source** and so "free"

Working with NoSQL Collections

Software University

- **Records** in a database have **unique identification keys**
  - New records are usually **assigned** an Id **automatically**
  - This allows a record to be **retrieved directly**
  - **Keys** can be used to create a **relationship** between records
- It's best to impose a **structure** on all records
  - Every entry has the **same properties**
- **De-normalize** data
  - E.g., article comments can be stored **inside** the article

Handling Forms

Grouping Related Request Values

HTML Form Standard

- The **<form>** element groups many **<input> fields**
  - Attribute **method** specifies which **HTTP method** to use
  - Attribute **action** specifies to which **URL** the requests is sent

```
<form method="POST" action="/articles">
  <input type="text" name="title" />
  <textarea name="content"></textarea>
  <input type="submit" value="Create Article" />
</form>
```

- On **submit**, the browser **sends all values** to the server
  - Every input is identified by its **name attribute**

Handling Submit Request

- Browser form submission causes the **page to reload**
  - Our application will be **closed** or **restarted**
- The **submit event** can be **intercepted**

```
const formElement = document.querySelector('form');
formElement.addEventListener('submit', event => {
  event.preventDefault();
  // collect values and send via fetch
});
```

- A **fetch request** can be made using the **input values**

Working with FormData

- The **FormData** object **automatically serializes** all input values
  - No need to select them manually

```
formElement.addEventListener('submit', event => {
  event.preventDefault();
  const data = new FormData(formElement);

  const email = data.get('email');      // Read single value
  const entries = [...data.entries()]; // Get array of values
});
```

Authentication

Working with user Credentials

Authentication and Authorization

Software University

- **Authentication**
  - The process of verifying the identity of a user or computer
  - Questions: "**Who are you?**", "**How do you prove it?**"
  - Credentials can be password, smart card, external token, etc.
- **Authorization**
  - The process of determining what a user is permitted to do on a computer or network
  - Questions: "**What are you allowed to do?**", "**Can you see this page?**"

Common Authentication Techniques

Software University

- **HTTP Basic Authentication** – credentials with **every request**
  - Username and password sent in a **request header**:

```
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

- **Cookie** – upon login, **server returns** authentication **cookie**
- **Token-based** – upon login, **server returns** signed **token**
  - Usually sent in a **request header** (name varies):

```
Auth-Token: d50d5f194848683ec68d2d0c4595128b146551249…
```

- Other methods: One Time passwords, Oauth, OpenID, etc.

Registration Request

```
<form method="POST" action="/users/register">
  <input type="text" name="email" />
  <input type="password" name="password" />
  <input type="password" name="repass" />
  <input type="submit" value="Register" />
</form>
```

```
async function onRegister(ev) {
  const response = await fetch('/users/register', {
    method: 'post',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(ev.formData);
}
```

Login Request

```
<form method="POST" action="/users/register">
  <input type="text" name="email" />
  <input type="password" name="password" />
  <input type="submit" value="Register" />
</form>
```

```
async function onLogin(ev) {
  const response = await fetch('/users/login', {
    method: 'post',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(ev.formData);
  };
  // handle authentication token
}
```

Handling Authentication Token

- Upon **successful login**, the server returns authentication **token**
  - This token must be **attached** to every **subsequent request**
- **Save** it using **`sessionStorage`**:

```
const authToken = response.authToken;
sessionStorage.setItem('authToken', authToken);
```

- **Send** it in a **request header**:

```
fetch('/articles', {
    method: 'get',
    headers: { 'X-Authorization': authToken }
};
```

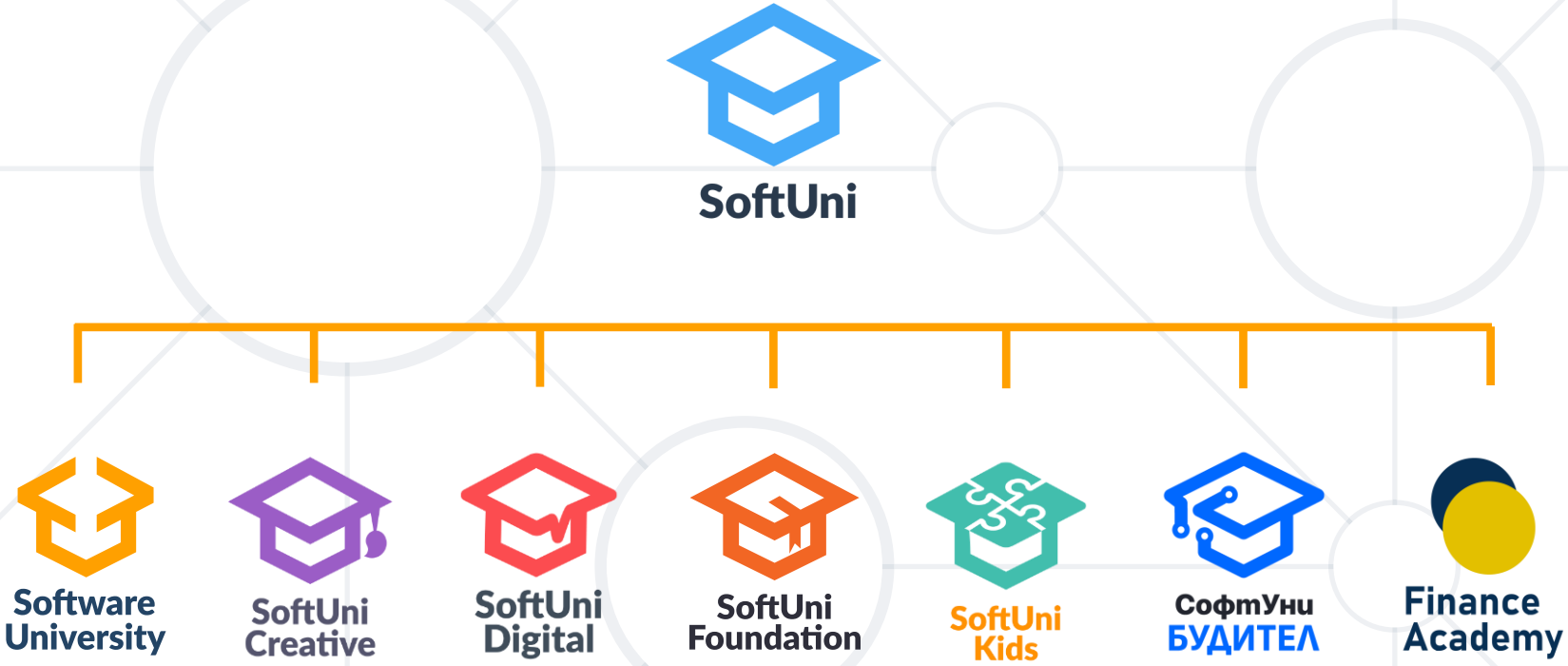
Data Ownership and Authorization

- Most APIs will **record** the data's **author**
  - Stored as **`ownerId`**, **`creator`** or similarly named **property**
  - Can be used to e.g., identify an article's or comment's author
- Depending on the service's **access rules**, only the author (and possibly administrators) **can modify** their records
- Display **edit controls** for records owned by the **current user**
  - Note that visibility **does not** provide security – this is done **on the server**, using access rules

Summary

- Data can be sent to the server
- Databases store records with unique keys
- HTML forms group input values
  - Have `submit` and `formdata` events
- Users can be authenticated with the service
  - Tokens are a common method

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- 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://softuni.org
- © Software University – https://softuni.bg