LU08.L03 - Bookshelf mit Authentifikation mit Bruno

Musterlösung zu LU08.A03 - Bookshelf mit Authentifikation mit Bruno.

Zwei Korrekturen gegenüber den Beispielen in der Aufgabe:

- Den Body liest man in Bruno mit `res.getBody()`, nicht mit `res.json()`.
- Das Token setzt man mit `bru.setEnvVar("token", …)` und liest es mit `bru.getEnvVar("token")`. `bru.getVar` liest Laufzeitvariablen, nicht Environment-Variablen.

Projektstruktur

```
bookshelf-bruno/
  bruno.json
  .gitignore              -> environments/dev.local.bru
  environments/
    dev.bru               baseUrl
    dev.local.bru         token, nicht einchecken
  1-auth/
    login-invalid.bru
    login-user.bru
    login-admin.bru
  2-books/
    list-books.bru
    read-book.bru
    save-book.bru
    delete-book.bru
```

Die Ordner beginnen mit einer Zahl, damit der Runner `auth` vor `books` ausführt. Innerhalb von `auth` steht `login-admin` zuletzt, damit bei einem Gesamtlauf das Admin-Token aktiv ist.

Environments

environments/dev.bru

```
vars {
  baseUrl: https://it.bzz.ch/book/ext
}
```

environments/dev.local.bru

```
vars {
  baseUrl: https://it.bzz.ch/book/ext
  token:
}
```

.gitignore

```
environments/dev.local.bru
```

1-auth/login-admin.bru

```
meta {
  name: login-admin
  type: http
  seq: 3
}

post {
  url: {{baseUrl}}/login
  body: json
  auth: none
}

body:json {
  {
    "username": "admin",
    "password": "admin"
  }
}

tests {
  test("Login admin -> 200 und Token", function () {
    expect(res.status).to.equal(200);
    const data = res.getBody();
    expect(data).to.have.property("token");
    bru.setEnvVar("token", data.token);
  });
}
```

`login-user.bru` ist gleich, mit `musterh`/`geheim`.

1-auth/login-invalid.bru

```
meta {
  name: login-invalid
```

```
  type: http
  seq: 1
}

post {
  url: {{baseUrl}}/login
  body: json
  auth: none
}

body:json {
  {
    "username": "wrong",
    "password": "wrong"
  }
}

tests {
  test("Login invalid -> 401", function () {
    expect(res.status).to.equal(401);
  });

  test("kein Token in der Antwort", function () {
    const data = res.getBody() || {};
    expect(data.token).to.be.undefined;
  });
}
```

2-books/read-book.bru

```
meta {
  name: read-book
  type: http
  seq: 2
}

get {
  url: {{baseUrl}}/read/c746a291-0ef9-4b2a-8268-392b12d636bd
  body: none
  auth: bearer
}

auth:bearer {
  token: {{token}}
}

tests {
  test("Read Book – rollenabhängig", function () {
    if (bru.getEnvVar("token")) {
```

```
      expect(res.status).to.equal(200);
      const book = res.getBody();
expect(book.book_uuid).to.equal("c746a291-0ef9-4b2a-8268-392b12d636bd");
      expect(book).to.have.property("title");
    } else {
      expect(res.status).to.equal(403);
    }
  });
}
```

Statt den Header von Hand zu setzen, verwendet die Lösung die Bruno-Authentifizierung `auth: bearer`. Bruno schreibt daraus `Authorization: Bearer <token>`.

2-books/list-books.bru (Tests)

```
test("List Books – rollenabhängig", function () {
  if (bru.getEnvVar("token")) {
    expect(res.status).to.equal(200);
    expect(res.getBody()).to.be.an("array").with.lengthOf(3);
  } else {
    expect(res.status).to.equal(403);
  }
});
```

2-books/save-book.bru und delete-book.bru

Nur `admin` darf schreiben. Am Token allein erkennt der Test die Rolle nicht. Darum speichern die Login-Requests zusätzlich die Rolle:

```
// in login-admin.bru, nach setEnvVar("token", ...)
bru.setEnvVar("role", "admin");
// in login-user.bru
bru.setEnvVar("role", "user");
```

```
test("Save Book – nur admin", function () {
  const role = bru.getEnvVar("role");
  if (bru.getEnvVar("token") && role === "admin") {
    expect(res.status).to.equal(201);
  } else {
    expect(res.status).to.equal(403);
  }
});
```

Drei Läufe

Lauf	Vorbereitung	list / read	save / delete
guest	`token` und `role` in `dev.local` leer	403	403
user	`login-user` senden	200	403
admin	`login-admin` senden	200	201 / 200

Auf der Kommandozeile:

```
bru run 2-books --env dev.local          # guest, Token leer
bru run 1-auth/login-user.bru --env dev.local
bru run 2-books --env dev.local          # user
bru run 1-auth/login-admin.bru --env dev.local
bru run 2-books --env dev.local          # admin
```

Die Tests enthalten eine Fallunterscheidung nach Rolle, aber keine Ablaufsteuerung (`setNextRequest`). Die Reihenfolge bestimmen die Ordner und die Läufe. Damit bleibt jeder Request für sich ausführbar.

M450-LU08, M450-C3I, M450-B1I

Kevin Maurizi

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu08/loesungen/automationbruno

Last update: **2026/11/17 17:53**