DBWT Dossier

Denis Behrends

1.2 Wie würde der Shop umgesetzt werden müssen (statisches HTML), wenn der Kunde 10 Detailseiten für die Produkte fordern würde?

- Jede Detailseite für ein Produkt müsste neu angelegt werden
- Es entsteht viel redundanter HTML Code
- Die Verlinkungen von der Produkte.html müsste manuell für jedes Produkt bearbeitet werden
- Bei kleiner Designänderung auf den Detailseiten müsste die Veränderung auf allen zehn Seiten vorgenommen werden.

1.6 Dropdownelemente im HTML Formularen anbieten und mehrfach verschachteln?

- Ein Dropdown kann mit dem <section> Tag eingeleitet werden.
- Die einzelnen Auswahlmöglichkeiten können mit <option> Tags erstellt werden
- Für eine mehrfache Verschachtelung der Auswahlelemente gibt es das <optgroup> Tag

https://wiki.selfhtml.org/wiki/HTML/Formulare/Auswahllisten

1.6 Wie kann man <option> Elemente in einem Dropdown nicht auswählbar machen?

- Indem man <option disabled></option> schreibt.

1.6 Welche Attribute sind bei <option> noch nützlich?

- Das „selected“ Attribut kann dazu verwendet werden eine Vorauswahl zu treffen
- Andernfalls wird das oberste Dropdownelement genommen

2.1 Der `Studiengang` {ET, INF, ISE, MCD, WI}

Vorteil durch Datentyp Enum: einfacher gut verständlicher Datentyp.

Nachteil durch Datentyp Enum: nicht in allen Datenbank DDL Dialekten enthalten

2.2 Welche Relationen lassen sich ohne eigene Tabelle abbilden?

Rekursive Beziehungen die keine Netzwerke sind können ihre Rekursive Relation in die eigene Tabelle aufnehmen. (z.B. Kategorien -> Oberkategorien)

Diese Verschmelzung ist immer bei 1:n und 1:1 Relationen möglich.

Paket 2 binären Relationstypen (1:1, 1:N, N:M) abgebildet

1:1 Durch FOREIGN KEY CONSTRAINTS ohne zwischen Tabelle

1:n Durch FOREIGN KEY CONSTRAINTS auf n Seite ohne zwischen Tabelle

n:m Durch FOREIGN KEY CONSTRAINTS mit zwischen Tabelle

Unterschied Tabellen und Spalten CONSTRAINTS

Tabellenbedingungen (TABLE CONSTRAINTS) sind CONSTRAINTS, die sich auf mehrere Spalten der Tabelle beziehen können und nicht nur auf eine einzelne Spalte. Notwendig sind Tabellenbedingungen z.B. für die Definition von Primärschlüsseln oder Fremdschlüsseln, die sich auf mehr als eine Spalte beziehen.

Spaltenbedingungen (COLUMN CONSTRAINTS) sind CONSTRAINTS, die sich auf eine einzelne Spalte, nicht auf die gesamte Tabelle, beziehen. Nur der NOT NULL-Constraint kann ausschließlich als Spaltenbedingung definiert werden und nicht als Tabellenbedingung, alle anderen Spaltenbedingungen lassen als sich auch als Tabellenbedingung formulieren.

Welche CONSTRAINTS erfüllen welchen zweck?

Constraint	Description
`PRIMARY KEY`	Sets the column for referencing rows. Values must be unique and not null.
`FOREIGN KEY`	Sets the column to reference the primary key on another table.
`UNIQUE`	Requires values in column or columns only occur once in the table.
`CHECK`	Checks whether the data meets the given condition.

https://mariadb.com/kb/en/library/constraint/

Aufzählungen ENUM in anderen DBMS Systemen?

ENUM kann in anderen DBMS mit CHECK([Spaltenname]IN([[[a], b], c])) nachgebildet werden.

Semikolon am Ende einer Anweisung?

Das Semikolon bewirkt die Beenedigung des vorangegangenen Statements.

4.1 Filter Methode

Das Filter Formular wird per `method="get"` gesendet, damit die Parameter im Link angezeigt werden. Damit können die Filterparameter auch als Link versendet werden um zum Beispiel Freunden zu zeigen was es neues in der E-Mensa gibt.

4.2. Cookie nach setzen von Session

Nach dem setzen der Sessionvariable wird ein Cookie generiert und im Browser des Benutzers gespeichert.

Wird der Cookie gelöscht und die Seite wieder vom selben Browser besucht, dann können die Logininformationen nicht mehr beibehalten werden

Auch wenn man den Cookie auf einem anderen Browser überträgt kann man wieder auf die Logininformationen Zugreifen (Sicherheitslücke)

Zustandsverwaltung

https://docs.microsoft.com/de-de/aspnet/core/fundamentals/app-state?view=aspnetcore-2.2#session-state

Zustände können mithilfe mehrerer Ansätze gespeichert werden. Die Ansätze werden im Verlauf dieses Artikels beschrieben.

Speicheransatz	Speichermechanismus
Cookies	HTTP-Cookies (schließt möglicherweise Daten ein, die mit serverseitigem App-Code gespeichert wurden)
Sitzungszustand	HTTP-Cookies und serverseitiger App-Code
TempData	HTTP-Cookies oder Sitzungszustand
Abfragezeichenfolgen	HTTP-Abfragezeichenfolgen
Verborgene Felder	HTTP-Formularfelder
HttpContext.Items	Serverseitiger App-Code
Cache	Serverseitiger App-Code
Abhängigkeitsinjektion	Serverseitiger App-Code

Beschreiben was die Information sha1:64000:18 bedeuten?

- `PBKDF2_HASH_ALGORITHM`: The hash function PBKDF2 uses. By default, it is SHA1 for compatibility across implementations, but you may change it to SHA256 if you don't care about compatibility. Although SHA1 has been cryptographically broken as a collision-resistant function, it is still perfectly safe for password storage with PBKDF2.
- `PBKDF2_ITERATIONS`: The number of PBKDF2 iterations. By default, it is 32,000. To provide greater protection of passwords, at the expense of needing more processing power to validate passwords, increase the number of iterations. The number of iterations should not be decreased.
- `PBKDF2_SALT_BYTES`: The number of bytes of salt. By default, 24 bytes, which is 192 bits. This is more than enough. This constant should not be changed.
- `PBKDF2_HASH_BYTES`: The number of PBKDF2 output bytes. By default, 18 bytes, which is 144 bits. While it may seem useful to increase the number of output bytes, doing so can actually give an advantage to the attacker, as it introduces unnecessary (avoidable) slowness to the PBKDF2 computation. 144 bits was chosen because it is (1) Less than SHA1's 160-bit output (to avoid unnecessary PBKDF2 overhead), and (2) A multiple of 6 bits, so that the base64 encoding is optimal.

Wofür ist das wichtig?

Password hashing algorithms such as PBKDF2, bcrypt, and scrypt are meant for use with passwords and are purposefully slow. Cryptographic hashing algorithms are fast. Fast is good in most situations, but not here. Slowing down the algorithm (usually by iteration) make the attacker's job much harder. Password hashes also add a salt value to each hash to make it unique so that an attacker can not attack multiple hashes at the same time.

Attackers will try to recover passwords by performing dictionary and brute-force attacks where they guess passwords by hashing them and comparing them to the stored password to determine if they match. With regular cryptographic hash functions (e.g. MD5, SHA256), an attacker can guess billions of passwords per second. With PBKDF2, bcrypt, or scrypt, the attacker can only make a few thousand guesses per second (or less, depending on the configuration).

This means that every password is much stronger if PBKDF2, bcrypt, or scrypt are used instead of a regular hash function.

In addition, PBKDF2, bcrypt, and scrypt all use large random "salt" values to make sure that each user's password is hashed uniquely. Attacking 100 password hashes will take 100 times longer than attacking one hash. Attacking a million will take a million times longer, etc. With SHA256, the attacker can try to crack thousands or millions of hashes at the same time with very little slow down.

You should always use a password hash or "key derivation formula" for passwords rather than an ordinary cryptographic hash. It's very hard to select passwords that are strong enough to withstand a dedicated cracking effort if they are hashed with something like SHA or MD5. With PBKDF2, bcrypt,

or scrypt, passwords can be as short as 7 or 8 characters but with MD5 or SHA, they need to be at least 13-14 characters.

6.1 Trigger Situationen

Bei insert wird der Vorrat reduziert. Wird der Vorrat unter 0 reduziert, dann greift das Contraint Vorrat > -1.

Bei Update in der Datenbanktabelle wird kein Trigger ausgelöst.

6.2 HTTP Header

Gibt es andere HTTP Header, die für eine Authorisierung bereits vorgesehen sind?

Keyed-Hash Message Authentication Code
https://de.wikipedia.org/wiki/Keyed-Hash_Message_Authentication_Code

https://demo.hitbtc.com/api#authenticationrestful

OAUTH
https://de.wikipedia.org/wiki/OAuth

JSON Web Token
https://de.wikipedia.org/wiki/JSON_Web_Token

OpenID
https://de.wikipedia.org/wiki/OpenID#Kritik

Token (Rechnernetz)
https://de.wikipedia.org/wiki/Token_(Rechnernetz)

Welche Möglichkeiten sind besser geeignet, Anfragen von anderen Diensten zu authentifizieren, als per festgelegtem HTTP Header?

PK Verfahren, API Tokens
https://developer.twitter.com/en/docs/basics/authentication/guides/authorizing-a-request

Wie sieht ein Beispielaufruf über diese Methoden aus?

```
curl -u "publicKey:secretKey" https://api.hitbtc.com/api/2/trading/balance
```

```
import requests
session = requests.session()
session.auth = ("publicKey", "secretKey")
```

```
curl -X GET -u
"ff20f250a7b3a414781d1abe11cd8cee:fb453577d11294359058a9ae13c94713" \
     "https://api.hitbtc.com/api/2/trading/balance"
```

```
import requests
session = requests.session()
session.auth = ("ff20f250a7b3a414781d1abe11cd8cee",
"fb453577d11294359058a9ae13c94713")
b = session.get('https://api.hitbtc.com/api/2/trading/balance').json()
print(b)
```

https://api.hitbtc.com/#authentication