Jegyzőkönyv SOA mérésről

A mérést végezte:	Szupera Zita
Neptun kód:	AWXUC6
Mérésvezető neve:	Bacsi Tibor
Mérés időpontja:	2014. 04. 23. 17.15
Mérés helyszíne:	HSZK N
Kapott feladatsor kódja:	23-SZORAK
Mellékelt forrás:	soa
On-line változat elérhető:	*http://localhost:20822/<...>*
A megoldott feladatok:	Kötelező 1,2,3 Választható: 4, 5
Elérhető pontszám (pluszpontok nélkül):	31

Kötelező feladatok

1. Feladat

Első rész:

A service.py fájl F1-1 kommenttel kezdődő résznél található a megoldás, a list_people függvényben.

A feladatot a mintaalkalmazás alapján oldottam meg, a kiinduló service.py megfelelő részét módosítottam csupán: egy lekérdezéssel elkérem a persons tábla összes elemét, majd a kapott eredményhalmazon végigiterálva az egyes rekordokat hozzáadom a HTTP válaszban visszaadott json-höz.

Teszteléshez a böngészőbe a következőt kell beírni (port forwarding használatával):

http://localhost:20822/persons.json

Második rész:

A service.py fájl F1-2 kommenttel kezdődő résznél található a megoldás, a show_person

függvényben.

Ezt a feladatot is a mintaalkalmazás alapján készítettem el.

Első lépésként lekérem a kapott ID-hoz tartozó személyt, ha az eredmény None, akkor a kért rekord nem létezik, 404-Not Found hibaüzenettel térek vissza.

Tesztelés: (tartalmazza a második feladat kimenetét is)

- Nem létező személy: http://localhost:20822/persons/10019.json
- Létező személy: http://localhost:20822/persons/10036.json

2. Feladat

A megoldás a service.py F2 kommentekkel jelzett részeinél található.

Ha a személy létezik, akkor le kell kérnem a lakhelyéhez (település) tartozó koordintátkat.

Ennek első lépése az address mezőből kinyerni a település nevét, ezt a '^\w*' reguláris kifejezés segítségével teszem meg, ennek jelentése: a mező első csak alfanumerikus vagy _ karaktereket tartalmazó karakterlánca, azaz az első szava.

Az így kapott sztring az alábbi kategóriák valamelyikébe esik:

- üres, tehát a cím nem volt megadva az adatbázisban
    - ekkor a koordináták helyén - jelenik meg
    - tesztelés: http://localhost:20822/persons/20001.json
- nem egy létező magyar városnevet kapunk
    - ekkor a koordináták helyén - jelenik meg
    - tesztelés: http://localhost:20822/persons/20004.json
- létező magyar városnevet kapunk
    - ekkor megjelennek a megfelelő koordináták
    - tesztelés: http://localhost:20822/persons/10222.json

3. Feladat

A megoldás a static mappa persons.html fájlában található.

Első lépésként a HTML fájl törzsében definiáltam egy üres listát persons ID-val. Az oldal fejrészében található kódban ezt a listát töltöm fel az oldal betöltődése után meghívódó ajax kérés eseménykezelőjében, ennek módja: a kérés eredményeként kapott json-ön végigiterálva szelektor segítségével hozzadok a listához egy-egy új elemet, aminek be is állítom a text attribútumát.

Tesztelés: http://localhost:20822/static/persons.html

4. Feladat

A feladat megoldása a service.py fájl F4 kommentel jelölt részeinél taláható.

A névre illetve a címre keresés megvalósítása logikailag megegyezik, ezért azokat együtt ismertetem.

A töredékekre kereséskor a _ és % karaktereket kell escape-elni. Ezt reguláris kifejezésekkel végeztem, amihez beimportáltam a re nevű python modult és a sub függvény segítségével elvégeztem a szükséges helyettesítéseket ügyelve a \ karakter megfelelő bevitelére.

Ezek után az SQL lekérdezésben = helyett LIKE-ot használtam és megadtam a escape karaktert is. Kis- és nagybetű között nem tesz küldönbséget a keresés.

Tesztelés:

- % karaktert tartalmazó név: http://localhost:20822/persons/by-name/%.json
- _ karaktert tartalmazó név: http://localhost:20822/persons/by-name/_.json
- % karaktert tartalmazó cím: http://localhost:20822/persons/by-address/%.json
- _ karaktert tartalmazó cím: http://localhost:20822/persons/by-address/_.json
- “Egyszerű” név: sütő g

  http://localhost:20822/persons/by-name/s%C3%BCt%C5%91%20g.json
- “Egyszerű” cím: göd http://localhost:20822/persons/by-address/g%C3%B6d.json

Választható feladatok

5. Feladat

A service.py F5 kommentekkel jelölt részeinél található a megoldás.

DELETE

A kapott ID-t beillesztem egy paraméteres törlést végrehajtó SQL lekérdezésbe. Ha az ID nem szerepel a persons táblában, akkor a kurzor rowcount attribútuma 0 lesz, ekkor 404-NotFound hibaüzenettel térek vissza.

Ha a kapott bemenet formátuma nem megfelelő, tehát nem csak számokat tartalmaz, akkor 400-as-Bad Request hibaüzenetet küldök válaszként, ennek ellenőrzéséhez elkapnom a cx_Oracle.DatabaseError típusú kivételt, és a hibaüzenet alapján döntöm el, hogy milyen hiba miatt volt sikeretlen a művelet. (Itt a kivétel messages attribútumát használom, mert a code változóra hibát kaptam, holott a cxOracle dokumentációja szerint létezik ilyen attribútum.) (Negatív illetve tört számok esetén nem kapok ilyen hibát, tehát ekkor 404-es hibakódot jelez az alkalmazás-)

Ha másfajta hiba történik, akkor 500-Internal Server Error hibával térek vissza.

Ha a törlés sikeres volt, akkor egy JSON-ben visszaadom a törölt ID-t.

Tesztelés:

- Sikeres törlés: `curl http://localhost:20822/persons/30012 --request DELETE`
- Nem létező rekord törlése: `curl http://localhost:20822/persons/1 --request DELETE`
- Nem megfelelő bemenet: `curl http://localhost:20822/persons/-1dd --request DELETE`

POST

Itt egy olyan insert parancsot írtam, amely a persons minden attribútumának értéket ad. Az adott érték a HTTP kérés törzsében kapott JSON megfelelő kulcsának értéke.

A felhasználónak azonban nem kell minden attribútumhoz kulcsot definiálnia a bmenetben, ha egy kulcs hiányzik, akkor az hozzáadom én a JSON-hoz üres sztring értékkel.

A következő felhasználói hibák esetén térek vissza 400-Bad Request hibakóddal:

- a kérés törzsében nincs JSON
- ID nem szám típusú
- Unique kényszer megsértése

- Kötelező mező értéke null

Egyéb hibák esetén 500-Internal Server Error hibaüzenetet küldök.

Ha a beszúrás sikeres volt, akkor a válasz törzsében visszaadom az új ID-t, és 201-es állapotkóddal térek vissza.

Tesztelés:

- Hiányzó parameter: `curl http://localhost:20822/persons.json --request POST --header "Content-Type: application/json" --data '{"name" : "valami", address" : "valami"}'`
- Sikeres beszúrás: `curl http://localhost:20822/persons.json --request POST --header "Content-Type: application/json" --data '{"person_id" : "3", "name" : "valami", "address" : "valami"}'`

PUT

Ebben a feladatban az update utasítást dinamikusan állítom elő sztringek összefüzéséből. Egy tömbben eltárolom a persons lehetséges mezőit, ezen végigiterálok, és ha a kapott JSON-ben van az adott elemmel megegyező kulcs, akkor az utasítást ennek megfelelően bővítem.

A paramétereket egy JSON-ben tárolom, ezt is dinamikusan készítem el.

Annak érdekében, hogy az ID is módosítható legyen, a SET és a WHERE után álló ID paramétereknek más-más nevet adtam.

Hibakezelést az alábbiak szerint végzek:

- Hiányzó bemenet: 400-Bad Request

- A módisítani kívánt rekord nem létezik: 404-Not Found

- ID nem csak számokat tartalmaz, unique kényszer megsértése, kKötelező mező értéke null: 400-Bad Request

- Egyéb esetben: 500-Internal Server Error

Tesztelés:

- Nem létező személy módosítása: `curl http://localhost:20822/persons/0 --request PUT --header "Content-Type: application/json" --data '{"person_id" : "2", "name" : "valami", "address" : "valami"}'`

- ID sikeres módosítása, és annak tesztelése, hogy csak a módisítani kívánt mezők változnak:

  http://localhost:20822/persons/20005.json

  curl http://localhost:20822/persons/20005 --request PUT --header "Content-Type: application/json" --data '{"person_id" : "101"}'

  http://localhost:20822/persons/101.json

Vélemény a mérésről

Nagyon tetszett a mérés, viszont az gondolom, hogy a kivételkezelésről több szó is lehetett volna, én például próbáltam a felhasználói hibákat alaposan lekezelni, viszont csak utólag döbbentem rá, hogy kell legyen egyszerűbb módja annak, mint az általam hibának gondolt kivételek elkapása. Örültem volna, ha írtok arról, hogy milyen gyakorlatok vannak erre, a különböző adatbázis hibák típusairól, mert szerintem ez később is hasznos lehet.