Πανεπιστήμιο Κρήτης
ΗΥ359 – Διαδικτυακός Προγραμματισμός (Web Programming)
Χειμερινό Εξάμηνο 2023/2024
Διδάσκων: **Μιχαήλ Μουνταντωνάκης**
Υπεύθυνος Βοηθός: **Αντώνιος Τόσιος** (Συντονιστής) και οι υπόλοιποι βοηθοί

3η Σειρά Ασκήσεων (Ατομική)

Διάρκεια: Δευτέρα 20/11 – Κυριακή 10/12
Αξία: 11% του τελικού σας βαθμού
(Συνολο ασκήσεων: **105 μονάδες**)
Θεματική ενότητα : **Servlets - AJAX- REST API**

Άσκηση 1. Εγκατάσταση Tomcat Και Βάσης [15 μονάδες]

Θα πρέπει να εγκαταστήσετε τον Tomcat και να τρέξετε παραδείγματα του μαθήματος, αλλά και τη βάση του μαθήματος. **Σαν παραδοτέο θα έχετε screenshots από την εκτέλεση των παραδειγμάτων,** ενώ και στην εξέταση του μαθήματος θα πρέπει να τρέξετε κάποιο παράδειγμα για να δείξετε ότι δουλεύει ο tomcat και η βάση του μαθήματος, ακόμα και αν δεν έχετε υλοποιήσει κάποια άσκηση από τις υπόλοιπες.

Άσκηση 2. [Pet-Care] Εγγραφή χρηστών [30 μονάδες]

Καλείστε να καλύψετε τις ανάγκες εγγραφής ενός χρήστη για την πλατφόρμα φιλοξενίας κατοικιδίων που θα υλοποιήσετε. Η φόρμα εγγραφής θα περιέχει τα πεδία εγγραφής που υλοποιήσατε στην πρώτη και στην δεύτερη σειρά ασκήσεων. Πλέον όμως η εγγραφή θα ολοκληρώνεται με τη δημιουργία ενός χρήστη στη μεριά του server και την αποθήκευση των στοιχείων του στην βάση δεδομένων.

- Μπορείτε να χρησιμοποιήσετε είτε HTML είτε **JSON απαντήσεις από τη μεριά του server (προτείνεται το δεύτερο αφού είναι πιο ευέλικτο).**
- Κώδικας για τη βάση και οδηγίες σας δίνονται στο elearn
  - **Προσαρμόστε τις φόρμες σας να έχουν τα ίδια ονόματα με τη βάση**
  - **Στέλνετε json δεδομένα από τον client**

- Σε περίπτωση που ο χρήστης δηλώσει ένα username το οποίο υπάρχει ήδη στη βάση, θα πρέπει να τυπωθεί ότι το συγκεκριμένο username χρησιμοποιείται ήδη. Αντίστοιχα και για το email.
  - **Pet Owners και Pet Keepers θα αποθηκεύονται σε διαφορετικά tables στη βάση**
  - **Δεν επιτρέπεται να έχουν το ίδιο username ή email**
    - **ένας pet owner με έναν pet keeper**
  - **Αυτά θα γίνονται τη στιγμή που συμπληρώνεται το αντίστοιχο πεδίο μέσω AJAX. Άρα εδώ θα χρειαστούν διαφορετικά AJAX requests για να το κάνετε αυτό.**

Σημαντικά

- Θα πρέπει να αποθηκεύονται στη βάση και οι συντεταγμένες **(lat lon)** της **διεύθυνσης** (από την άσκηση 2), οπότε στέλνετε και αυτά τα έξτρα πεδία.
- Θα πρέπει να δουλεύουν όλοι οι έλεγχοι που έχετε κάνει στις προηγούμενες ασκήσεις. Αν για οποιοδήποτε λόγο δε δουλεύουν, μπορείτε να κάνετε τους ελέγχους αυτούς στο server.
- Η μικροϋπηρεσία (servlet) που θα φτιάξετε θα πρέπει σε περίπτωση που υπάρχει κάποιο λάθος από τη μεριά του client θα πρέπει ο **server να επιστρέφει το κατάλληλο status code** και τα κατάλληλα μηνύματα λάθους (π.χ. Status code 403 - το username υπάρχει).
- Εφόσον έχουν συμπληρωθεί όλα τα υποχρεωτικά πεδία με σωστό τρόπο και ο χρήστης κάνει click στο κουμπί "Εγγραφή", **η διαδικασία εγγραφής πρέπει να ολοκληρωθεί επιτυχώς, αποθηκεύοντας το συγκεκριμένο λογαριασμό στη βάση.**
  - Αν ακολουθήσετε τον κώδικα της βάσης, θα γίνει αυτόματα
- Το servlet θα πρέπει να επιστρέψει όλα τα στοιχεία που έδωσε ο χρήστης και το μήνυμα
  - «Η εγγραφή σας πραγματοποιήθηκε επιτυχώς»
  - Μπορείτε να χρησιμοποιήσετε όποια λύση θέλετε, ενδεικτικά να ανανεώνεται η σελίδα μετά την εγγραφή
    - είτε μέσω JAVA Servlets/AJAX (Single Page Application)
    - είτε να μεταφέρεται σε μία καινούρια σελίδα (πχ μέσω JAVA Servlets.)

Βαθμολόγηση

- AJAX requests και έλεγχος για διπλότυπα (username, email) **(10 μονάδες)**
- Σωστό registration και σωστή αποθήκευση στη βάση δεδομένων **(15 μονάδες)**
- Κατάλληλα μηνύματα και status codes για επιτυχία ή αποτυχία των requests και της εγγραφής **(5 μονάδες)**

Άσκηση 3. [Pet-Care] Λειτουργίες Pet Keeper [25 μονάδες]

Εδώ θα υλοποιήσετε κάποιες λειτουργίες ενός pet keeper, ιδανικά μέσω ενός single page application και μέσω **HTTPsession API ή Cookies.**
Ένας εγγεγραμμένος χρήστης θα πρέπει να μπορεί: **(13 μονάδες)**

- να κάνει login, αλλιώς το σύστημα να του πετάει κατάλληλο μήνυμα μη επιτυχημένου login
- να κάνει logout
- να παραμένει συνδεδεμένος μέχρι να τελειώσει η συνεδρία - session (να μη χρειάζεται δηλαδή να κάνει login ξανά μέχρι να κάνει log-out).

Επίσης σε μία μοναδική σελίδα (σε ένα single page application) ο συνδεδεμένος χρήστης πρέπει να δει τα στοιχεία του συγκεντρωτικά σε μια σελίδα και να μπορεί να τα αλλάξει όλα εκτός από τα username και email **(12 μονάδες)**

Άσκηση 4. [Pet-Care] Καταχώρηση Κατοικιδίου – Ανεξάρτητο REST API [35%]

**** Αν δε μπορεί κάποιος να κάνει το REST API και το κάνει μέσω Servlets, θα πάρει max 25/35 για την άσκηση 4**

Μπορείτε να το υλοποιήσετε με όποια βιβλιοθήκη θέλετε, πχ με την Java Spark που είδαμε στο μάθημα

α) Δημιουργία REST API (25 μονάδες)

Υποθέστε ότι υπάρχει ένα καθολικό API, στο οποίο κάποιος βάζει πληροφορίες για τα κατοικίδια που έχει. Το REST API, θα πρέπει να έχει τις εξής δυνατότητες (μπορείτε και να τις επεκτείνεται αν σας βολεύει), χωρίς να χρειάζεται κάποιος να συνδεθεί σε αυτό με username/password (θεωρούμε στα πλαίσια της άσκησης ότι είναι προσβάσιμο από όλους):

- **1. Καταχώρηση κατοικιδίου** (POST) **(9 μονάδες)**
  - /pet/
  - Να τοποθετεί ένα καινούριο κατοικίδιο μέσω ενός JSON.
    - Να έχει μέσα το JSON τα στοιχεία που χρειάζονται
      - pet_id
      - owner_id
      - name (όνομα)
      - type (Ειδός)
      - breed (Ράτσα)
      - gender (Φύλο)
      - birthyear (Έτος γεννήσεως)
      - weight (Κιλά)
      - description (Περιγραφή)
      - photo (Φωτογραφία)
  - **Έλεγχος**
    - το id του κατοικιδιου να είναι 10 ψηφία
    - το id του pet owner να υπάρχει στη βάση στο σχετικό πίνακα
      - Για λόγους απλότητας θα θεωρήσουμε ότι ο κάθε owner έχει ένα κατοικίδιο και γι αυτό ελέγχουμε αν έχει ήδη κατοικίδιο (σε αυτήν την περίπτωση επιστρέφουμε λάθος)
    - τα κιλά του να είναι >0 και το birthyear>2000
    - Η photo να ξεκινά με http
    - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
  - **Αποτέλεσμα**
    - Να έχει τα κατάλληλα response codes και μηνύματα ανάλογα με την επιτυχία ή όχι του request
  - Το πρόγραμμα να αποθηκεύει το κατοικίδιο στη βάση
  - **Παράδειγμα**

- /pet/
  - JSON → {
    {"pet_id":"1234554321","
    ""owner_id":"1","
    ""name":"Max","
    ""type":"cat","
    ""breed":"aegean","
    ""gender":"male","
    ""birthyear":"2019","
    ""weight":"4.1","
    ""description":"Agrios gatos, thelei trofi 3 fores ti mera","

    ""photo":"https://i.ytimg.com/vi/pM7RDz9BhUY/hqdefault.jpg"}";

}

- **2. Ανάκτηση κατοικιδίων συγκεκριμένου είδους** (GET) **(8 μονάδες)**
  - /pets/:type/:breed
  - **Path parameter:** type, breed
  - **Optional Parameters**: fromWeight, toWeight
  - **Έλεγχος**
    - Μπορείτε να θεωρήσετε ότι αν δώσει breed=all, λαμβάνει υπόψη όλες τις ράτσες
    - Το fromWeight να είναι μικρότερο από το toWeight
      - Να έχουν τιμή μεγαλύτερη από 0
  - **Αποτέλεσμα**: Ένα JSON που περιέχει όλα τα κατοικίδια.
  - **Παραδείγματα**:
    - /pets/cats/all (όλες οι γάτες)
    - /pets/dogs/labrador (όλα τα labrador)
    - /pets/cats/all?fromWeight=3&toWeight=4 (όλες οι γάτες από 3 έως 4 κιλά)
    - /pets/cats/all?fromWeight=5 (όλες οι γάτες με βάρος πάνω από 5 κιλά)
    - /pets/dogs/all?toWeight=25 (όλοι οι σκύλοι με βάρος λιγότερο από 25 κιλά)

- **3. Ανανέωση κιλών κατοικιδίου** (PUT) **(4 μονάδες)**
  - petWeight/:pet_id/:weight
  - **Path parameter:** pet_id, weight
  - **Έλεγχος**
    - για σωστές μεταβλητές (να υπάρχει το pet id)
      - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
    - να είναι ο αριθμός των κιλων>0
  - **Αποτέλεσμα**
    - Αν όλα πάνε καλά να επιστρέφει status 200 και σχετικό μήνυμα
  - **Παράδειγμα**
    - petWeight/1234554321/4

- **4. Διαγραφή Κατοικιδίου** (DELETE) **(4 μονάδες)**
  - /petDeletion/:pet_id

- **Path parameter:** pet_id
- **Έλεγχος**
  - για σωστές μεταβλητές (να υπάρχει το pet id)
    - Να επιστρέφει σχετικό μήνυμα σε περίπτωση λάθους
- **Αποτέλεσμα**
  - Αν όλα πάνε καλά να επιστρέφει status 200 και σχετικό μήνυμα
- **Παράδειγμα**
  - /petDeletion/1234554321

Η υλοποίησή σας ιδανικά θα παίζει με JSON και θα κάνει σωστή διαχείριση λαθών (σκεφτείτε τη χρήση των status codes). Για παράδειγμα

- 200 όταν όλα είναι καλά
- 403 forbidden (όταν είναι λάθος κάποιο id)
- 406 Not Acceptable (πχ αν δοθούν λάθος τιμές, πχ στο weight)

β) Τεστάρισμα REST API (10 μονάδες)

Θα πρέπει να τεστάρετε το REST API **με έναν από τους παρακάτω τρόπους (όπως είδαμε/θα δούμε στο μάθημα).** Δε χρειάζεται να κάνετε όλους τους τρόπους.

- **1**) **(Plain JAVA)** Με έναν java client για να τεστάρετε όλα τα σενάρια, χωρίς τη χρήση κάποιου γραφικού περιβάλλοντος
- **2) Μέσω Postman,** όπου θα πρέπει να δείξετε screenshots, αλλά και να το χρησιμοποιήσετε στην εξέταση της άσκησης
- **3) (AJAX+HTML).** Μέσω απλού γραφικού HTML και AJAX requests στο REST API, που να μπορεί να τεστάρει όλα τα σενάρια
  - βοηθητικό για το project
  - καλύτερα να το ανοίξετε από browser με απενεργοποιημένο το CORS

Βοηθητικά

Δείτε τις διαφάνειες και τα παραδείγματα. Για οτιδήποτε ζητάει η άσκηση, έχουμε δείξει έστω ένα μικρό παράδειγμα. Μη ξεχνάτε τη χρήση του "use strict"; για την JavaScript.
Το parse ενός JSON string γίνεται με χρήση της JSON.parse(str) που επιστρέφει το javascript object που αντιστοιχεί στο str, δεδομένου ότι το str είναι μια σωστή αναπαράσταση JSON .
Μπορείτε να χρησιμοποιήσετε κάποιον online linter όπως ο jshint http://jshint.com/ για να βελτιώσετε την ποιότητα του js κώδικά σας.
Προτείνεται η χρήση του netbeans 12.4 σαν IDE και για τα servlets και για το Spark REST API. Επίσης προσπαθήστε να οργανώσετε με όμορφο τρόπο τα servlets που θα χρησιμοποιήσετε.

Τρόπος Παράδοσης

Η Παράδοση θα γίνεται μέσω elearn. Θα πρέπει να παραδώσετε ένα zip που να περιέχει ένα φάκελο A3_AM, όπου AM ο αριθμός μητρώου σας.
Μπορείτε να έχετε ένα φάκελο για τις 2 πρώτες ασκήσεις (είτε και για τις 3 πρώτες), και ένα φάκελο για τις υπόλοιπες.

Εκπρόθεσμες ασκήσεις **δεν θα γίνονται δεκτές**.

Για σοβαρά θέματα που δεν επιτρέπουν την εμπρόθεσμη παράδοση των ασκήσεων, στείλτε email στον διδάσκοντα.

Αντιγραφή
Σε περίπτωση αντιγραφής θα μηδενίζονται άμεσα οι εργασίες όλων των εμπλεκόμενων.

Καλή εργασία