TESZTAUTOMATIZÁLÁS

5. forduló

<epam>

A kategória támogatója: EPAM

Ismertető a feladathoz

Felhasznált idő: 00:00/30:00 **Elért pontszám:** 0/65

1. feladat 0/5 pont

Mi mondható el az újratesztelésről (Re-testing) és a regressziós (Regression) tesztelésről?

Válasz

- (●) A regressziós tesztelés célja, hogy újrateszteljük a program különböző részeit, olyan hibákat keresve, ami a már meglevő működést befolyásolná. A regressziós tesztelést megéri automatizálni.

  Az újratesztelés célja, hogy az egyszer már elbukott teszteseteket újra lefuttassuk, hogy validáljuk, hogy a hibák ténylegesen kijavításra kerültek. Az újratesztelést nem éri meg automatizálni.

  Ez a válasz helyes, de nem jelölted meg.

- ( ) A regressziós tesztelés célja, hogy az egyszer már elbukott teszteseteket újra lefuttassuk, hogy validáljuk, hogy a hibák ténylegesen kijavításra kerültek. Az regressziós tesztelést nem éri meg automatizálni.

  Az újratesztelés célja, hogy újrateszteljük a program különböző részeit, olyan hibákat keresve, ami a már meglevő működést befolyásolná. Az újratesztelést megéri automatizálni.

- ( ) A regressziós tesztelés célja, hogy újrateszteljük a program különböző részeit, olyan hibákat keresve, ami a már meglevő működést befolyásolná. A regressziós tesztelést nem éri meg automatizálni.

  Az újratesztelés célja, hogy az egyszer már elbukott teszteseteket újra lefuttassuk, hogy validáljuk, hogy a hibák ténylegesen kijavításra kerültek. Az újratesztelést megéri automatizálni.

- ( ) A regressziós tesztelés célja, hogy újrateszteljük a program különböző részeit, olyan hibákat keresve, ami a már meglevő működést befolyásolná. A regressziós tesztelést nem éri meg automatizálni.

  Az újratesztelés célja, hogy az egyszer már elbukott teszteseteket újra lefuttassuk, hogy validáljuk, hogy a hibák ténylegesen kijavításra kerültek. Az újratesztelést nem éri meg automatizálni.

Magyarázat

A regressziós tesztelés olyan részeket tesztel, amik nem feltétlen az éppen változtatott kódrészlethez tartoznak. Ezeket sűrűn futtatjuk, ezért érdemes automatizálni.

Az újratesztelés újra letesztel egy már egyszer elbukott tesztesetet. Ezt csak akkor futtatjuk, amikor feltételezzük, hogy a feature megjavult, ezért ennek az automatizálási költsége (ROI) nem térülne meg.

2. feladat 0/10 pont

Adott a következő DOM struktúra. Mely XPath selectorokkal tudjuk kiválasztani a Password szöveggel rendelkező label elemet?

```
<form action="action_page.php" method="post">
        <div class="imgcontainer">
                <img src="https://www.w3schools.com/howto/img_avatar.png" alt="Avatar" class="avatar" width="
        </div>
        <div class="container">
                <label for="uname">Username</label>
                <input type="text" placeholder="Enter Username" name="uname" required=""/>
                <div>
                        <label for="psw">Password</label>
                        <input type="password" placeholder="Enter Password" name="psw" required=""/>
                </div>
                <button type="submit">Login</button>
                <label>
                        <input type="checkbox" checked="checked" name="remember">Remember me</input>
                </label>
        </div>
        <div class="container" style="background-color:#f1f1f1">
                <button type="button" class="cancelbtn">Cancel</button>
                <span label="psw">Forgot password?</span>
        </div>
</form>
```

Válaszok

- [x] /form/div[2]/div/label
  Ez a válasz helyes, de nem jelölted meg.
- [x] //div/label[@for="psw"]
  Ez a válasz helyes, de nem jelölted meg.
- [ ] //div/span[@label="psw"]
- [ ] label[for=psw]

Magyarázat

A /form/div[2]/div/label selector helyes, Absolute XPath tehát a struktúrára épít (ha a struktúra megváltozik, akkor a selector már nem biztos, hogy a megfelelő elemet fogja meg).

A //div/label[@for="psw"] selector helyes, Relative XPath tehát egy jól meghatározott részletet keres (ha a struktúra meg is változik körülötte, akkor jó eséllyel továbbra is a megfelelő elemet fogja meg).

A //div/span[@label="psw"] selector nem a megfelelő elemet fogja meg, mert spant talál meg label attribútummal, így a "Forgot password?" kerül elkapásra.

A label[for=psw] selector ugyan a megfelelő elemet fogja meg, de ez CSS selector, így a feladat szempontjából helytelen.

3. feladat 0/10 pont

Egy alkalmazásban azt szeretnénk validálni, hogy a feltöltött kiadványok megfelelő időintervallumban kerültek kiadásra. Az elfogadott tartomány a 01.01.1800-12.12.2022, az elfogadott formátum pedig "NN.HH.ÉÉÉÉ". A felsorolt reguláris kifejezések közül melyik a **legszigorúbb** az illeszkedők közül?

Válasz

- (?:0[1-9]|[12][0-9]|3[01])[-/.](?:0[1-9]|1[012])[-/.](?:18\d{2}|19\d{2}|20[01][0-9]|202[012])
- (0[1-9]|[12]\d|3[01])\.(0[1-9]|1[012])\.(18|19|20)\d\d
- (0[1-9]|[12]\d|3[01]).(0[1-9]|1[012]).(18\d{2}|19\d{2}|20[01]\d|202[012])
- (0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[012])\.(18[0-9]{2}|19[0-9]{2}|20[01][0-9]|202[012])

  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

"(?:0[1-9]|[12][0-9]|3[01])[-/.](?:0[1-9]|1[012])[-/.](?:18\d{2}|19\d{2}|20[01][0-9]|202[012])" : elég szigorú a napok, hónapok és évek számát tekintően, azonban megengedi, hogy más karakterekkel válasszuk el a napokat hónapokat és éveket egymástól (- és /).

"(0[1-9]|[12]\d|3[01])\.(0[1-9]|1[012])\.(18|19|20)\d\d": hónapokat és napokat megfelelően kezeli, jó az elválasztó karakterre vonatkozó megszorítás, azonban évek esetén engedi a 2022 utáni számokat is, így vele a 2099 is validnak minősülne.

"(0[1-9]|[12]\d|3[01]).(0[1-9]|1[012]).(18\d{2}|19\d{2}|20[01]\d|202[012])": a napok 01-től 31-ig mehetnek, hónapok 01-től 12-ig, évek pedig 1800-tól 2022-ig, megfelelő elválasztó karakterrel. "\d"egyenértékű a "[0-9]" használatával, így elég szigorú opció, azonban az elválasztó karaktereknél a "." elé nem került "\" karakter (nincs escape-elve), ezért bármely karakterre illeszkedik.

"(0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[012])\.(18[0-9]{2}|19[0-9]{2}|20[01][0-9]|202[012])": legszigorúbb opció, a napok 01-től 31-ig mehetnek, hónapok 01-től 12-ig, évek pedig 1800-tól 2022-ig, megfelelő elválasztó karakterrel.

4. feladat 0/10 pont

Adott az alábbi JSON schema. Válaszd ki mely JSON objektumra illeszkedik a lehetőségek közül.

```
{
    "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "email": {
      "type": "string",
      "format":"email"
    },
    "favourites": {
      "type": "object",
      "properties": {
        "colors": {
          "type": "array",
          "items": {
            "type": "string"
          }
        },
        "animals": {
          "type": "array",
          "items": {
            "type": "string"
```

```
          }
        }
      },
      "additionalProperties": false,
      "required": [
        "colors",
        "animals"
      ]
    }
  },
  "additionalProperties": false,
  "required": [
    "name",
    "email",
    "favourites"
  ]
}
```

Válasz

- [x]
```
{
  "name":"John Doe",
  "email":"john.doe@example.com",
  "favourites": {
     "colors":["blue","green"],
     "animals":["dog","cat"]
   }
}
```
Ez a válasz helyes, de nem jelölted meg.

- [ ]
```
{
  "name": "John Doe",
  "favourites": {
    "colors": ["blue","green"],
    "animals": ["dog","cat"]
  }
}
```

- [ ]
```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "favourites":{
    "colors": "blue",
    "animals": ["dog","cat"]
  }
}
```

- [ ]
```
{
  "name":"John Doe",
  "email":"john.doeaexample.com",
  "favourites":{
"colors":["blue","green"],
 "animals":["dog","cat"]
  }
 }
```

Magyarázat

A helyes válasz:

```
"{
  "name":"John Doe",
  "email":"john.doe@example.com",
  "favourites": {
     "colors":["blue","green"],
     "animals":["dog","cat"]
   }
}"
```

A további válaszlehetőségekben az alábbi hibák fordulnak elő:

hiányzó property, nem megfelelő típusú property érték, nem megfelelő formátumú property érték

5. feladat 0/15 pont

Egy könyveket árusító webshop tesztelésére kell automata teszteseteket írnod. Az alkalmazásnak az alábbi követelményeknek kell megfelelnie:

- a felhasználók csak bejelentkezés után tudnak vásároni,
- a kosárba helyezést követően tudják
- a megrendelést módosítani (törlés, darabszám növelés),
- kuponkódot érvényesíteni,
- fizetési módot kiválasztani,
- szállítási módot kiválasztani,
- ezt követően a megrendelést véglegesíteni.

Az alábbiak közül melyik tesztesetet automatizálnád elsőként?

Válaszok

- [x] A felhasználó megpróbál regisztrálni egy olyan email címmel, amellyel már van regisztrált felhasználói fiók
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Belépést követően a felhasználó kiválaszt egy könyvet, berakja a kosárba, kiválasztja a bankkártyás fizetési módot és a futárszolgáltatással való szállítási módot. Kuponkód érvényesítés nélkül nem véglegesíti a megrendelést és törli a terméket a kosárból.
- [x] Belépést követően a felhasználó kiválaszt egy könyvet, berakja a kosárba, kiválasztja az utánvétes fizetési módot és a futárszolgáltatással való szállítási módot. Kuponkód érvényesítés nélkül véglegesíti a megrendelést.
  Ez a válasz helyes, de nem jelölted meg.
- [ ] A felhasználó megrendelés során érvényesít egy kuponkódot, majd megpróbálja ismételten felhasználni egy újabb vásárlás alkalmával.

Magyarázat

Mind a négy eset jó jelölt automatizálásra, de a teljes, sikeres megrendelési folyamatot tesztelő esetet célszerű automatizálni elsőként, mivel ez a teszteset a legfontosabb üzleti szempontból és a rendszer legtöbb funkcióját érinti.

6. feladat 0/15 pont

Nézzük az alábbi kódrészletet:

```
class TestUser {
    constructor(public userName: string, public password: string) {};
}

class PowerUser {
    constructor(public userName: string, public password: string) {};
}

class ClassifiedTestUser {
    constructor(public userName: string, public password: string, public classification: string[]) {};
}
```

Ha a három osztályt használjuk, akkor egy TypeScript környezetben melyik kódsorra fogunk hibát kapni?

Válasz

- ```
  const user: TestUser = new TestUser('john123', 'verystrongpassword');
  ```
- ```
  const user2: TestUser = new ClassifiedTestUser('joe123', "verystrongpassword!", ["software engineer", "j
  ```
- ```
  const user3: ClassifiedTestUser = new TestUser('jane123', 'verySTRONGpass!23');
  ```

  Ez a válasz helyes, de nem jelölted meg.
- ```
  const user4: PowerUser = new TestUser('power111', 'poweruserpassword123!A');
  ```

Magyarázat

A user, user2, és user4 létrehozhatóak ilyen módon, ugyanis a TS "duck typingot" használ, vagyis két objektumot az alapján hasonlít össze, hogy ugyanazokat a kulcsokat tartalmazzák-e.

Az user konstans simán létrehozható. Egyszerű osztálypéldányosítás.

Mivel a TestUser osztály egyforma a PowerUser osztállyal, így a TS a tipusrendszere miatt engedélyezi a user4 konstans létrehozását.

A user2 konstans létrehozása picit trükkösebb, mivel a ClassifiedTestUsernek van egy 'classification' fieldje, ami nincs jelen a TestUserben. Viszont a TS ezt engedélyezi, hiába nincs köztük alkalmazott öröklődés, a duck typing miatt ez valid TS kód.

A user3 létrehozása viszont nem lehetséges. A user3 egy ClassifiedTestUser, és a változó mögé pedig egy TestUsert szeretnénk példányosítani, ami nem lehetséges, mivel nincs neki classification fieldje. Ez nem egy valid eset TypeScriptben.

KÉSZÍTETTE cone

Megjelenés

Világos