CST3130 Advanced Web Development with Big Data

Coursework1

Claudiu Busteaga

MDX-ID:M00812513

Project description

Price comparison websites are useful tools that compare the price of the same product at several retailers and can take off the pressure from finding the best deals for a product.

The products chosen for this project are caps and hats; therefore, the website name is Caps and Hats Compare, a price comparison website that helps users find the best prices for the desired product. The products chosen for this project are caps. The website contains two pages: the main page with the search option and the comparison page. The first page of the website contains a search bar and products displayed below once the customer search for them. When the user clicks on the desired product, the second page is landed with the product and different prices.

Web scraping

To extract data from different websites, developers use different scrapping methods. The tool chosen to scrape different websites for this project is Selenium. Selenium is a powerful tool that can be used to scrape data from complicated web pages that require authentication, making it one of the most well-liked tools for web scraping. It can also mimic user activities like selecting an option, clicking a button, or filling out a form.

The websites scrapped for this project are:

- https://www.sportsdirect.com/
- https://www.nike.com/
- "https://www.ftshp.co.uk/

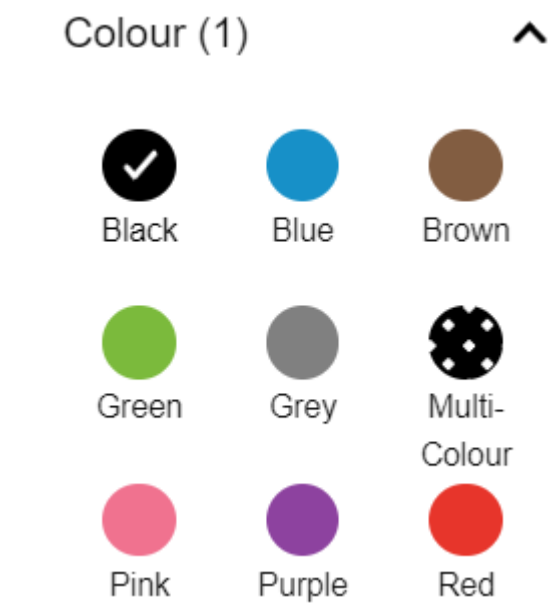
During the process of scrapping with the selenium for the Nike.com website, the following steps have been implemented:

Initialised the Web driver for scanning:

```
WebDriver driver;
```

Select the desired colour (in this example, black) with the help of class actions:

```
//find the cookie element and clicks on accept.
driver.findElement(By.cssSelector("#hf_cookie_text_cookieAccept")).click();
Actions actions = new Actions(driver);
WebElement colourB = driver.findElement(By.xpath( xpathExpression: "//*[@id=\"wallNavFG2\"]/button[1]/div"));
delayScrapping();
actions.moveToElement(colourB).click().perform();
```



After that, the Web Elements were identified and recorded with the help of a list.

```
List<WebElement> listProducts = driver.findElements(By.cssSelector("div[class*='product-card product-grid__card  css-c2ovjx']"));
 //list that save  the WebElements list
List<ProductWebScrap> nikeWebList = new ArrayList<>();
//get all the elements and save them
for (WebElement el : listProducts) {

    WebElement titleNike = el.findElement(By.className("product-card__link-overlay"));
    WebElement detailsNike = el.findElement(By.className("product-card__subtitle"));
    WebElement price = el.findElement(By.className("product-price"));
    WebElement url = el.findElement(By.className("product-card__img-link-overlay"));
    WebElement imgUrl = el.findElement(By.className("product-card__hero-image"));
    WebElement colour = el.findElement(By.xpath( xpathExpression: "/html/body/div[4]/div/div/div[3]/div[4]/div/div[5]/div[1]/div[1]/div[2]/div/div/div/div
    String priceNike = price.getText().substring( beginIndex: 1);
    String urlNike = url.getAttribute( name: "href");
    String imgUrlNike = imgUrl.getAttribute( name: "src");
    String colourN = colour.getText();
    String styleLink = url.getAttribute( name: "href");
    String styleKey = styleLink.substring( beginIndex: styleLink.lastIndexOf( str: "/") + 1);
    String details = detailsNike.getText().replaceAll( regex: "[^\\d]", replacement: "");
    ProductWebScrap nike = new ProductWebScrap();
    nike.setTitle(titleNike.getText() + " " + details);
    nike.setPrice(Double.parseDouble(priceNike));
    nike.setUrl(urlNike);
    nike.setImageUrl(imgUrlNike);
    nike.setColour(colourN);
    nike.setStyleKeyId(styleKey);
    nikeWebList.add(nike);


}
System.out.println(nikeWebList.size());
delayScrapping();
driver.close();
```

All the WebElements have been stored in a database using saveAndMerge method from Dao class that helps to record the dates in the database.

```
//send data to Caps class - colour Black

for (ProductWebScrap nikeScrap : nikeWebList) {
    Caps cap1 = new Caps();
    cap1.setName(nikeScrap.getTitle());
    cap1.setStyle_key(nikeScrap.getStyleKeyId());

    CapsInstance capIn1 = new CapsInstance();
    capIn1.setCaps(cap1);
    capIn1.setColour(nikeScrap.getColour());
    capIn1.setImage_url(nikeScrap.getImageUrl());


    Compare compare1 = new Compare();
    compare1.setUrl(nikeScrap.getUrl());
    compare1.setCapsInstance(capIn1);
    compare1.setPrice(nikeScrap.getPrice());
    try {
        dao.saveAndMerge(compare1);
    } catch (Exception e) {

        runThread = false;
    }


}
```

RESTful Api

Application programming interfaces (APIs) known as RESTful APIs use HTTP requests to GET, PUT, POST, and DELETE data. It is frequently used to access or push data into databases when making calls across various systems. RESTful APIs are made to be compact, user-friendly, and quick to implement.

The Restful APIs have also been implemented in this project to get data from the database.

To get all the caps and hats details such as id, name and the image link from the database, the following function with the specific SQL query have been used.

```
//return all products - id ,names and images
module.exports.getCaps = async () => {
    let sql ="SELECT c.id , c.name, ci.image_url    FROM caps c INNER JOIN caps_instance ci on c.id = ci.caps_id";

  return executeQuery(sql);

};
```

In the next step, “app. get “handler helped to get data and send it back in a object to be used for the front end.

```
// //Get the request for all products
app.get('/caps', async (request, response) => {
  let  caps = await db.getCaps();
  //send back to the client
  response.json(caps);
});
```

The Vue Js framework was chosen to display the data on the application website.

To do that, some specific methods have been used as the following :

The method that gets the object from the API and stores in the products array.

```
methods: {
  // Get All Products
  async getProducts() {
    try {
      const response = await axios.get('http://localhost:3030/caps');
      this.products = response.data;
    } catch (err) {

    }
```

To render on the website the v-for method and mustache syntax have been used as the following:

```
<div class="col-md">
  <div>
    <div class="row ">
      <div class="card my-2 mx-2" style="width: 18rem;" v-for="productDetail in productDetails">
        <div class="row">
          <div class="col">
            <div class="card-body">

              <p class="card-title">Product:{{productDetail.name}}</p>


              <p class="card-title">Colour:{{productDetail.colour}}</p>
              <p>Prices avalaible:</p>

            </div>
          </div>
```

Database design