**Document for Bright Edge Coding Assignment**

**1. Instruction for running the jar program:**

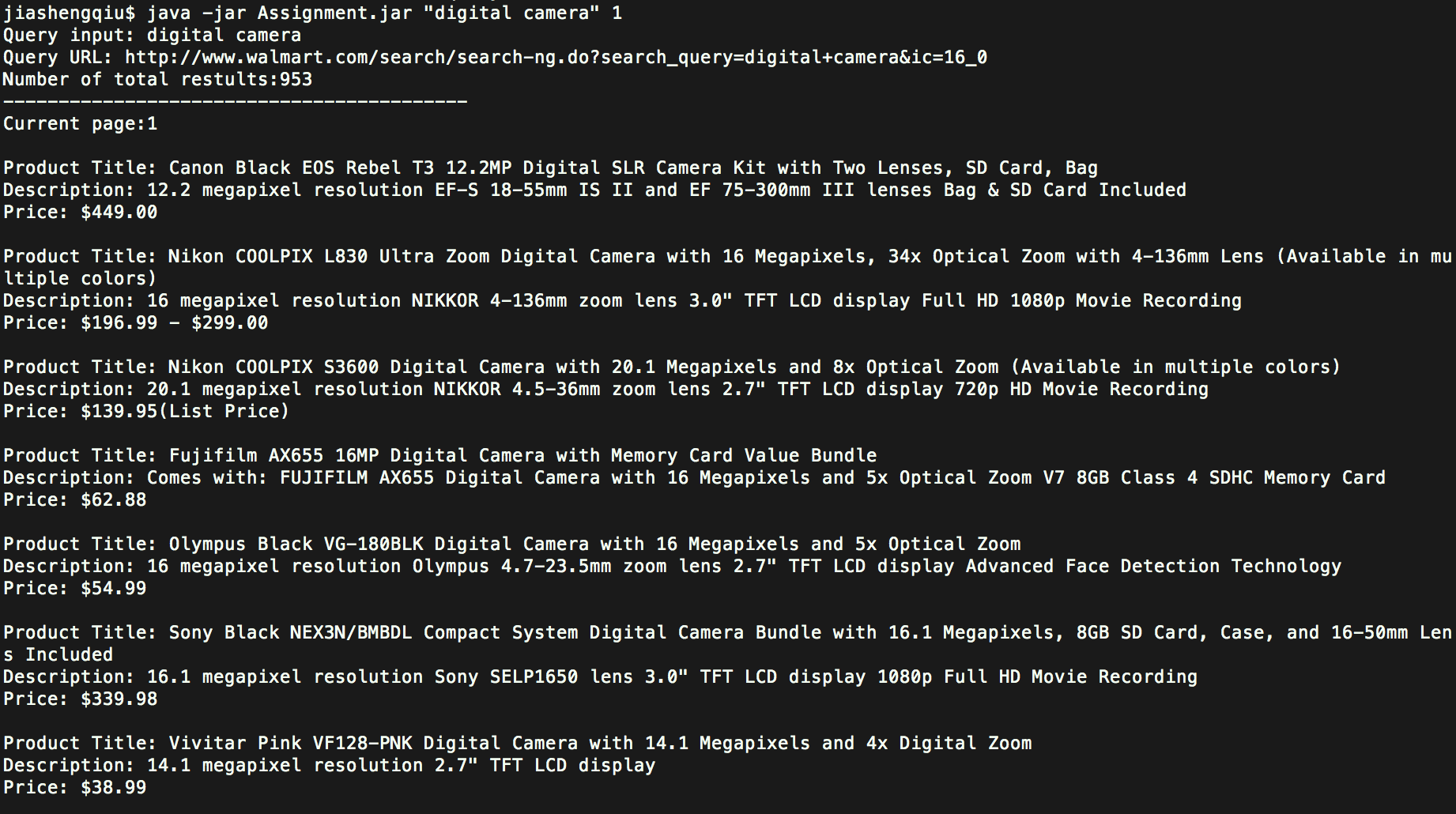
***Step 1: Extract the zip file and go to the directory of Assignment.jar***

***Step 2: With network connection, run the command to see the query result.***

**Query 1**: (requires a single argument) java -jar Assignment.jar <keyword> (e.g. java -jar Assignment.jar "baby strollers")

**Query 2**: (requires two arguments) java -jar Assignment.jar <keyword> <page number> (e.g. java -jar Assignment.jar "baby strollers" 2)

***Step 3: If the program executes well, you will see the screenshots as following:***



(The screenshot is for Query2 scenario)

**2. File Description:**

* Assignment.jar: Containing executable jar file
* Src: Containing the source code for the project.
* Jsoup.1.7.3.jar: the Jsoup library used in parsing the HTML

\* Others: generated Java doc

**3. Key components of the project**

The major functionality of the program includes taking the user query input, getting search result page from Walmart.com, parsing the content in the result page and displaying the results to console to users. Below is a class diagram to illustrate the program design:

Has a call

Produce

Calls call

Parser

Result

Product Scraper

Assignment

Has a

Displayer

Passed to

* **Assignment.java**: The main program to run the project.
* **Product Scraper.java**: The scraper object, user can set the parser and displayer for the scraper.
* **Parser.java**: Object used to parse the html documentation.
* **WalmartParser.java**: Implementation of the parser for Walmart.com
* **Result.java**: Object used to store the parsing result, it contains a list of products, query, query url, page number for the query. (If there is no page number passed into the query, the page number would be 0)
* **Product.java**: Object used to store a single piece of product information, including product name, product description and price.
* **Displayer.java**: Used to display the result to console.
* **ParserTest**: Unit Test program for Parser

**4. Parsing implementation**

Below is a very brief summary of what WalmartParser do.

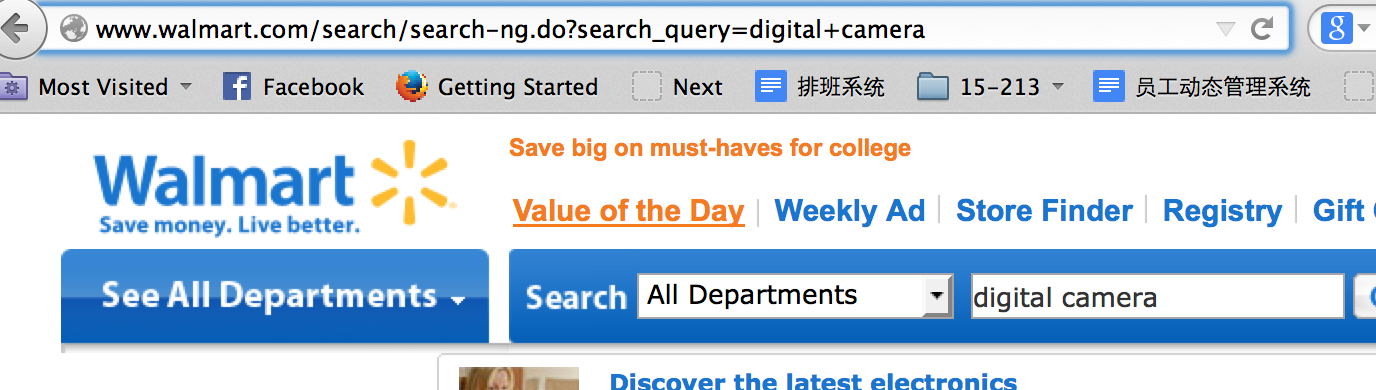
**4.1 Setting the base url.**

BaseURL: <http://www.walmart.com/search/search-ng.do?search_query>=

**4.2 Reformat the query String.**

When typing the query into search box in the web page, we can find that the query string is parsed into the URL in a different format.

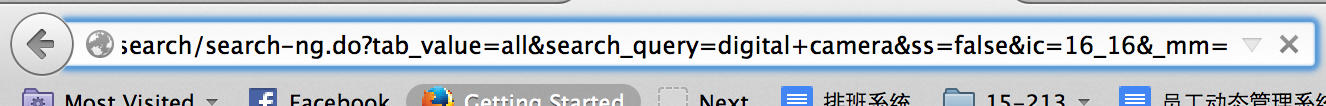
For example, changing “digital camera” to “digital+camera”.



So the modified version should be “http://www.walmart.com/search/search-ng.do?search\_query=digital+camera.

**4.3 Include page query parameter in the URL**

We can also easily get the following URL when we click page 2 in search result page.



It would not be difficult to find that “ic=16\_16” indicates number per page and page number. In fact ic = Number per page\_ (pagenumber – 1) \* Number per page for any page number greater than 0. We also need to append this query parameter in the url.

**4.4 Get the document object.**

Use Jsoup to send HTTP GET Request to the modified URL and get response as jsoup.nodes.Document model so that we can traverse the nodes easily.

**4.5 Navigate the product information**



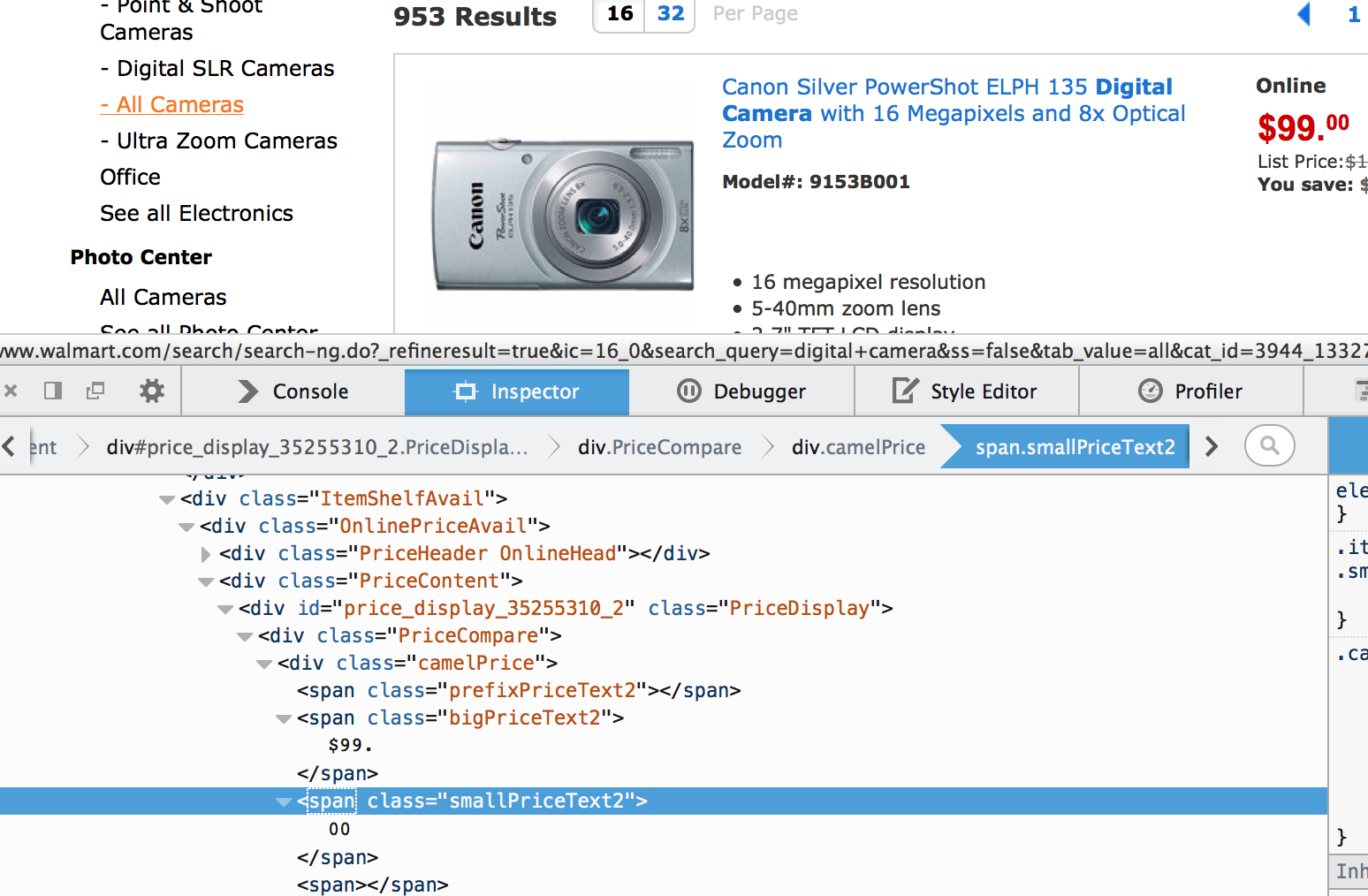
From the HTML source we can find that both product title and price are located within the <div class = “prodInfo”> tag. So we traverse through each prodInfo div and extract the product title, description and the price.

It is easier to get the title since it locates within the <a class = “prodLink ListItemLink>. We can just strip the text within this tag to get the title. There is no tag that can help us simplify the title. I have tried one simple way, which is to extract the text before the last highlight text. It works for query like “digital camera” or “baby stroller” but not for query like “cat”, “dog” since text ending with these queries cannot become a short title. So in the implementation I keep the original title.

For price, it has **THREE** scenarios we need to take into consideration

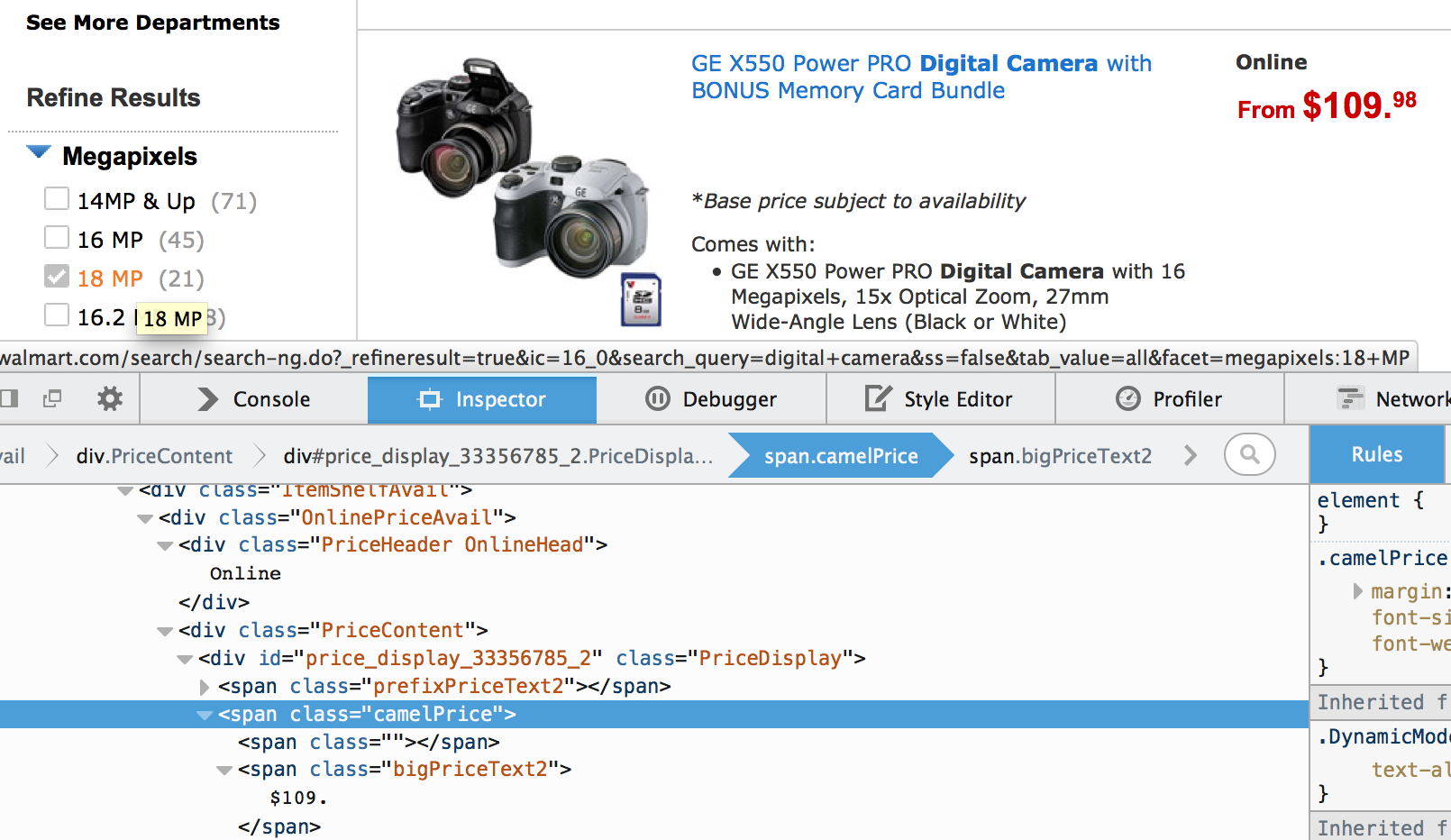
**Case1: Regular online price**

When online price is an exact price we can just get the text from the div with class name “camelPrice”.



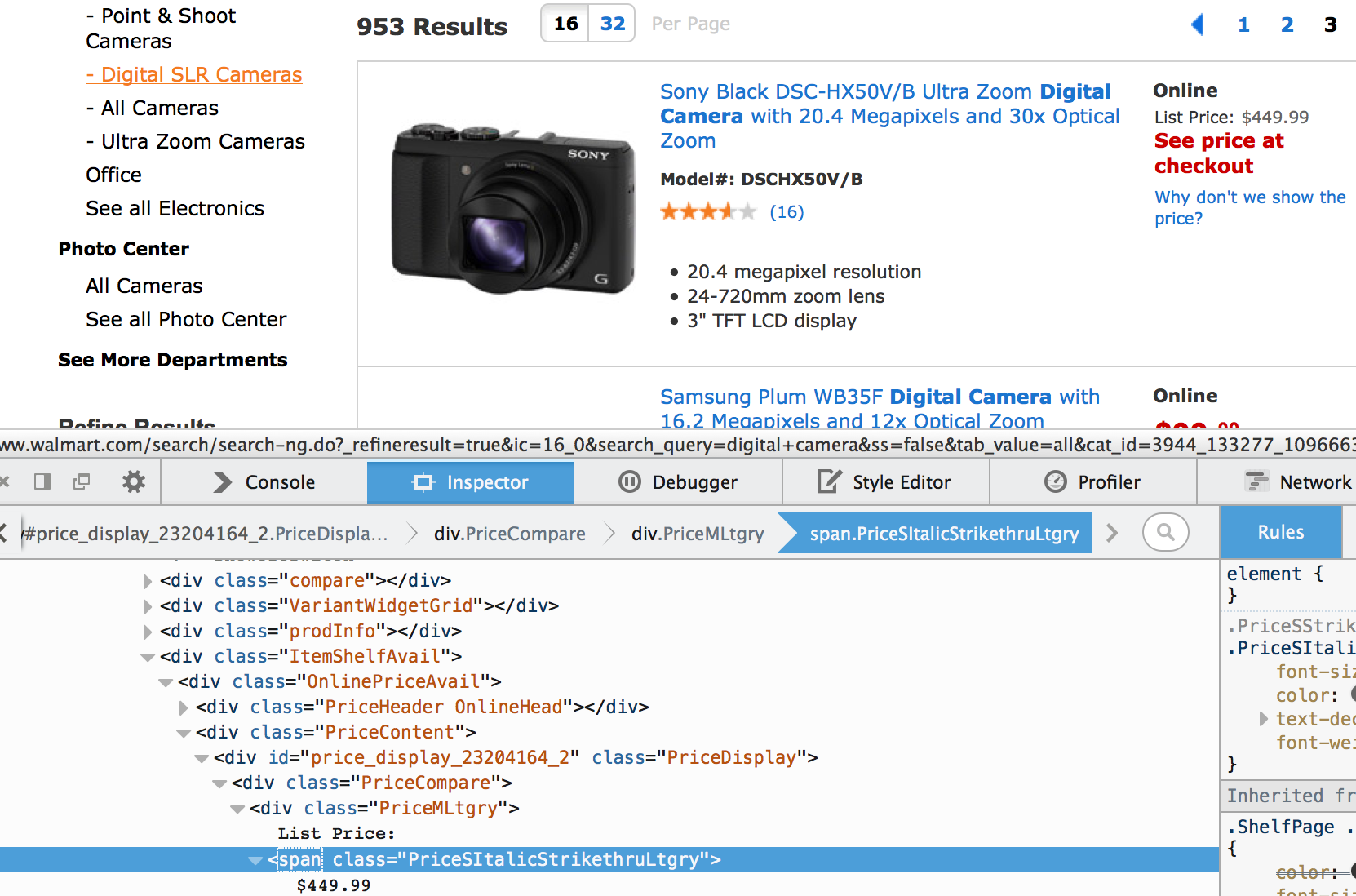
**Case 2: Price is a range**

When online price is a range like “From $109.98”, there is no div with class name “camel price”, we need to combine the text of span prefixPriceText and span camel price to get the price range.



**Case 3: Price at checkout**

When online price is not available, we need to get its list price.



In the implementation first we will check if we can find regular price, and then price range, if we cannot find previous two then we extract the list price.

**5 Store the result**

The last step is to create the product object and store each piece of the product information in the result set. After we collect all the product information we return the result object with product list, query term, query URL, page number, results in the current page and the total number of the query result. The result object will then be passed to the displayer.