

Product Search Design Document

By: Alice Zhou, June 13, 2014

For: Mr. Schattman, ICS 4UI

See <https://github.com/Alice-Z/productsearch> for code

Description

This program is a web application that allows users to conveniently compare products from retailers Amazon and Ebay.

The program starts at a search page that prompts the user to enter keywords and a postal code. Using these two parameters, the program executes a search and returns the top ten results from Amazon and Ebay, respectively. There is an option to sort these results by increasing price or by product name. To view more information about a particular item, the user can click on the item and the relevant information will appear in a pop-up.

This project is built with the Spring framework and uses Gradle as a build tool. Tomcat is used as the server on which the program is run, and Thymeleaf provides the view-layer services to resolve the views in the MVC (Model-View-Controller) framework. The Amazon and Ebay databases are accessed by making REST calls to the Amazon Product Advertising API and the Ebay Finding API.

An improvement to be made is to determine a method to search for more than ten items from each online retailer. Amazon can only handle ten items per request, so this would entail counting the current number of items and also keeping track of the page numbers.

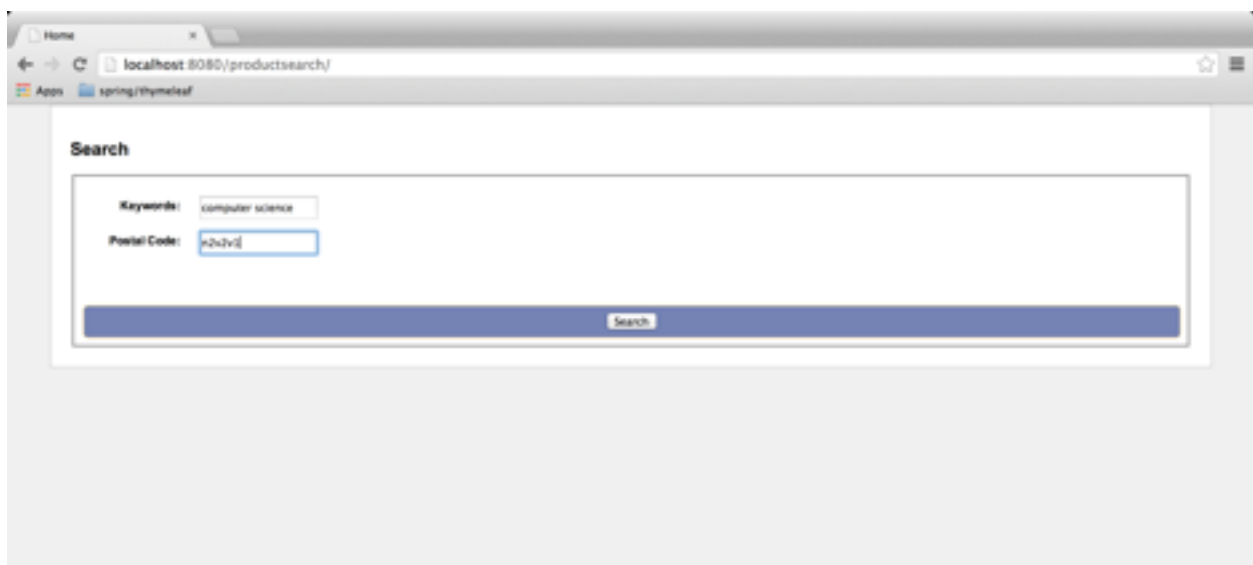
A screenshot of a web browser displaying a search form. The browser's address bar shows 'localhost:8080/productsearch/'. The page has a title 'Search'. Below the title, there is a form with two input fields: 'Keywords' with the text 'computer science' and 'Postal Code' with the text '92029'. A blue 'Search' button is located at the bottom right of the form.

Figure 1. The search form

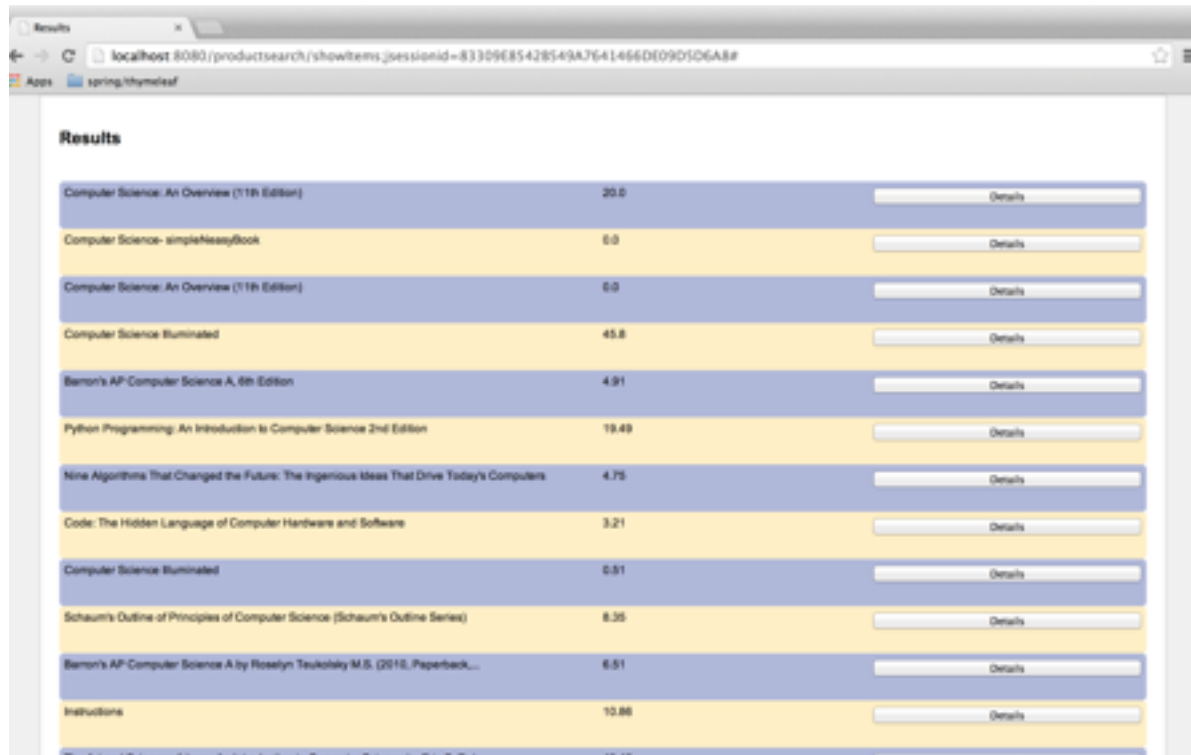


Figure 2. Results for the search

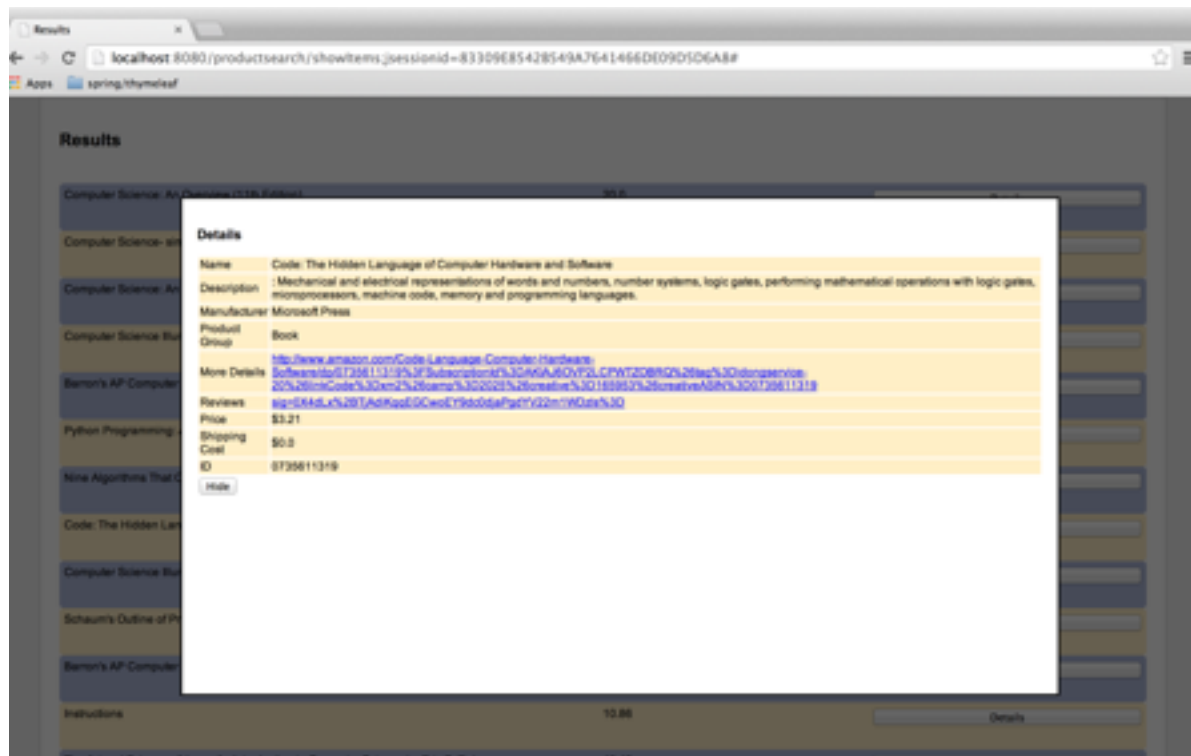


Figure 3. Details for an item

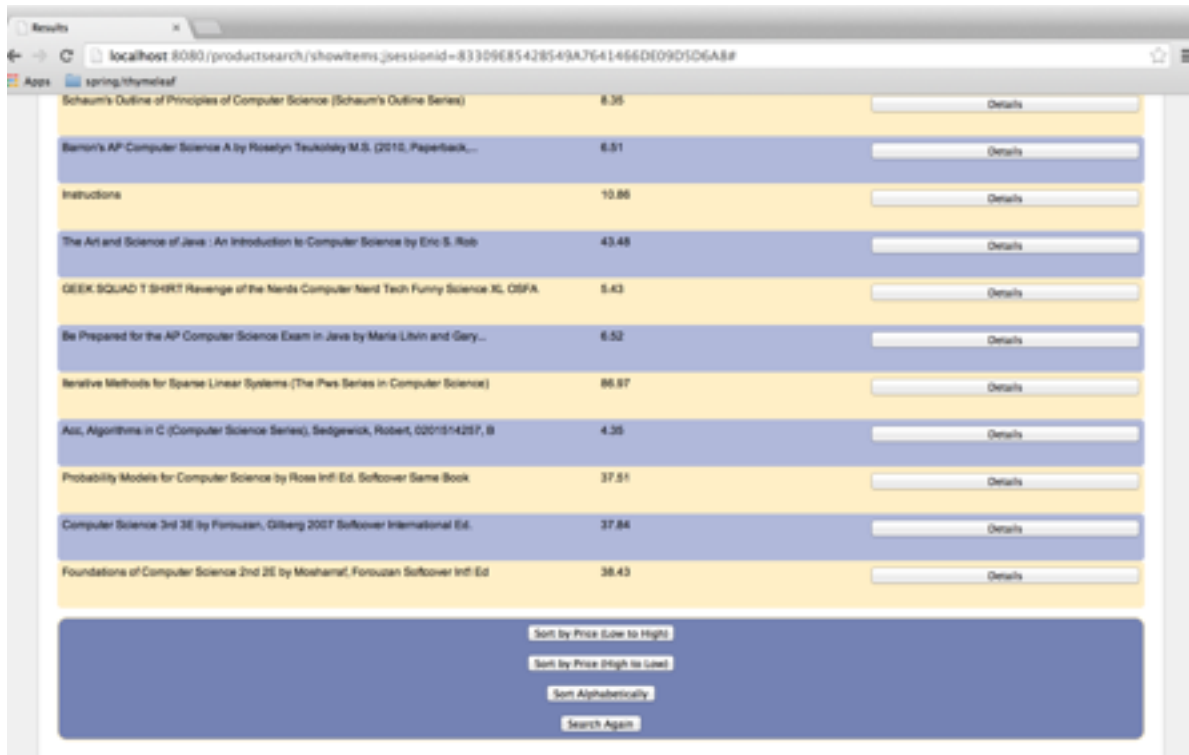


Figure 4. Buttons below for searching again and sorting by increasing price

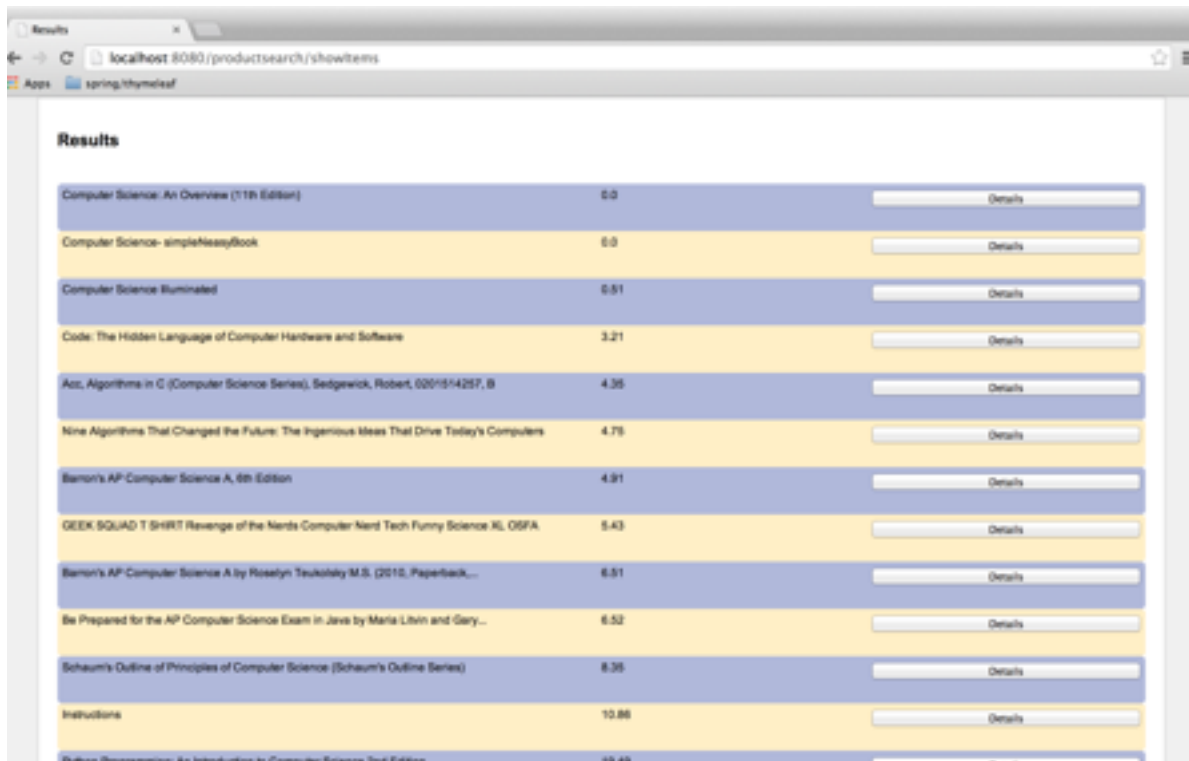
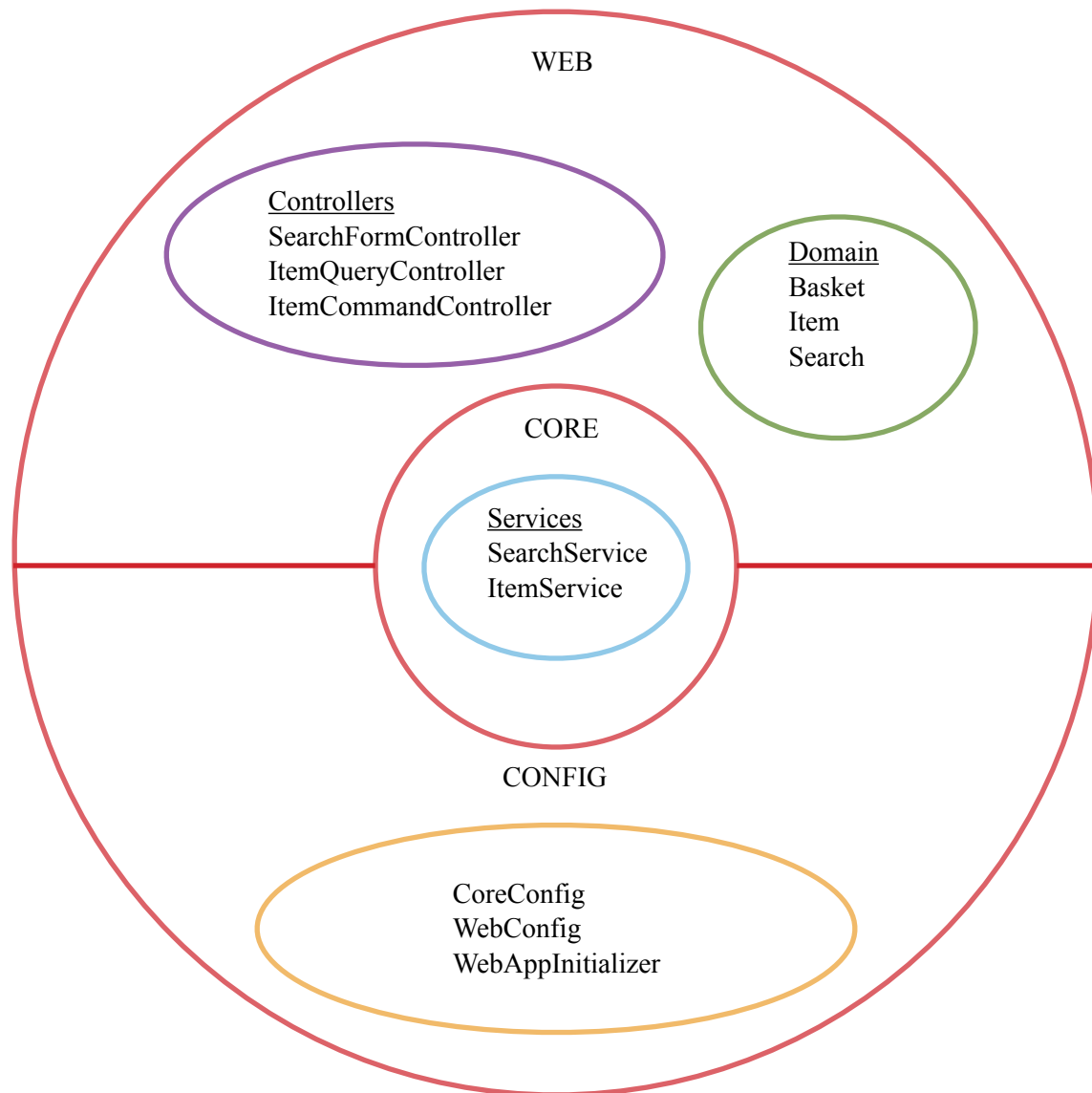


Figure 5. The sorted items, in $O(N\log(N))$ time!

Integration



Classes

Core

Services

AmazonDriver

Fields	
String keyWords	Keywords for search
private static final String SECRET_KEY, AWS_KEY, ASSOCIATE_TAG	Needed for authenticating the search
Methods	
public AmazonDriver()	Constructor
public List<Item> search()	Create URL using Amazon's SignedRequestsHelper, then gives created URL to XMLtoItem, then returning the acquired list of Items.

EbayDriver

Fields	
String keyWords	Keywords for search
String postalCode	Postal code for search
public final static String EBAY_APP_ID, EBAY_FINDING_SERVICE_URI, SERVICE_VERSION, OPERATION_NAME, GLOBAL_ID	Needed for authenticating the search and provides basic URL framework
public final static int REQUEST_DELAY = 3000;	Delay time
private int numItems;	Number of items returned
Methods	
public EbayDriver()	Constructor
public List<Item> search()	Calls createURL method, giving the created URL to XMLtoItem, and then returning the acquired list of Items.
public String createURL()	Creates URL with keywords and postal code

XMLtoItem

Fields	
String keyWords	Keywords for search
Classes	
class SAXHandler extends DefaultHandler	methods: startElement, which creates a new Item upon finding an XML “Item” endElement, which populates the attributes of the item
class EbayHandler extends SAXHandler	methods: startElement, which creates a new Item upon finding an XML “item” endElement, which populates the attributes of the item
class AmazonHandler extends SAXHandler	fields: ArrayList<Item> itemList; Item item; String content;
Methods	
public static ArrayList<Item> parseItems(String sourceUrl, int i) throws ParserConfigurationException, SAXException, MalformedURLException, IOException {	Searches given a sourceUrl and an integer ‘i’, which denotes whether the search will be for Amazon or Ebay. Takes the XML results of the search and converts into a list of Items.

SearchServiceHandler class implements SearchService

Fields	
private Search search;	The Search
private AmazonDriver amazonDriver;	AmazonDriver to search
private EbayDriver ebayDriver;	EbayDriver to search
Methods	
public SearchServiceHandler();	Constructor for SearchServiceHandler class
public List<Item> Searcher()	Uses AmazonDriver and EbayDriver to search
getters and setters for search	Gets and returns search

ItemServiceHandler class implements ItemService

Methods	
<code>public ItemServiceHandler();</code>	Constructor for ItemServiceHandler class
<code>public List<Item> sort(List<Item> items, String type)</code>	Receives a list of Items and a string which indicates the type of search requested. When this calls merge sort, it passes the type along.
<code>private List<Item> mergesort(List<Item> items, String type)</code>	Basic recursive function for mergesort, splitting array in half, then merging. The type of merge used is determined by the type given to this function.
<code>private List<Item> mergePriceUp(List<Item> a, List<Item> a)</code>	Compares the price when merging
<code>private List<Item> mergePriceDown(List<Item> a, List<Item> a)</code>	Compares the price when merging
<code>private List<Item> mergeAlpha(List<Item> a, List<Item> a)</code>	Compares alphabetical order when merging

Web Domain

Item class

Fields	
String name, description, manufacturer, detailPageURL, reviewURL, condition	The attributes of the item
double price, shippingCost	The prices involved with the item
Methods	
getters and setters for the above fields	Sets and retrieves the above fields.

Search class

Fields	
String keyWords, postalCode	The keywords and postal code used in the search.
Methods	
getters and setters for the above fields	Sets and retrieves the above fields.

Basket class

Fields	
List<Item> items	The List of items in the basket
Methods	
public Basket();	Constructor for basket class
public Basket(List<Item> items);	Constructor for basket class given a list of items
public Item add(Item item)	Adds item into the list
public List<Item> getItems()	Returns the list of items
public int getSize()	Returns the size of the list

<code>public void clear()</code>	Empties the basket
<code>public void reset(List<Item> items)</code>	Clears the current basket, then sets the new one
<code>public void print()</code>	Prints out content of the basket

Controller

SearchFormController

Fields	
SearchService searchService	Takes care of the searching. See searchService class under Core.services
Search search	Search object for this instance
Basket basket	Basket of returned items
Methods	
@RequestMapping(value = "/", method = RequestMethod.GET) public ModelAndView show()	Maps the html GET request "/" to the main view, which shows a search form
@RequestMapping(method = RequestMethod.POST) public String submit(@ModelAttribute("search") Search search, @ModelAttribute("basket") Basket basket, BindingResult bindingResult)	Takes the html POST request "/" and transfers the inputs to a search object, then searches for items, returns the items and enters into the basket, and redirects to a "/showItems", which will handle the view.

ItemQueryController

Fields	
Basket basket	Basket of returned items
Methods	
@RequestMapping(value = "/showItems", method = RequestMethod.GET) public String show(@ModelAttribute("basket") Basket basket)	Takes html GET request "/showItems" and returns a view that displays the items in the basket.

ItemCommandController

Fields	
ItemService itemService	Service that takes care of the sorting.
Basket basket	Basket of returned items
Methods	
@RequestMapping(value = "/searchAgain", method = RequestMethod.POST) public String searchAgain()	Takes html GET request “/searchAgain” with name “searchAgain” and redirects to “/”, which will show the form.
@RequestMapping(value = "/sortPriceUp", method = RequestMethod.POST) public String sortPrice(@ModelAttribute("basket") Basket basket)	Takes html GET request “/sortPriceUp” and uses itemService to sort the items by increasing price, then resets basket with the newly-ordered items.
@RequestMapping(value = "/sortPriceDown", method = RequestMethod.POST) public String sortPrice(@ModelAttribute("basket") Basket basket)	Takes html GET request “/sortPriceDown” and uses itemService to sort the items by decreasing price, then resets basket with the newly-ordered items.
@RequestMapping(value = "/sortAlpha", method = RequestMethod.POST) public String sortPrice(@ModelAttribute("basket") Basket basket)	Takes html GET request “/sortAlpha” and uses itemService to sort the items alphabetically, then resets basket with the newly-ordered items.

Configuration

CoreConfig

Methods	
@Bean public SearchService searchService()	Used by Spring framework to make a SearchService; returns the implementation SearchServiceHandler
@Bean public ItemService itemService()	Used by Spring framework to make an ItemService; returns the implementation ItemServiceHandler
@Bean public Search search()	Used by Spring framework to make a Search class
@Bean public AmazonDriver amazonDriver()	Used by Spring framework to make an AmazonDriver for the SearchService
@Bean public EbayDriver ebayDriver()	Used by Spring framework to make an EbayDriver for the SearchService

WebAppInitializer extends

AbstractAnnotationConfigDispatcherServletInitializer

Methods	
protected Class<?>[] getRootConfigClasses() {	Gives configuration class
protected Class<?>[] getServletConfigClasses() {	Gives configuration class needed for servlet
protected String[] getServletMappings() {	Prefaces servlet mapping with "/"
protected Filter[] getServletFilters() {	Encodes characters with "UTF-8"

WebConfig extends WebMvcConfigurerAdapter

Methods	
<code>public void addResourceHandlers(ResourceHandlerRegistry registry) {</code>	Adds resource handlers
<code>public void addResourceHandlers(ResourceHandlerRegistry registry) {</code>	Adds resource handlers to handle the views
<code>public void addInterceptors(InterceptorRegistry registry) {</code>	Adds interceptors for different languages
<code>public LocaleResolver localeResolver() {</code>	Resolves the languages
<code>public ServletContextTemplateResolver templateResolver() {</code>	Resolves for templates - makes program search for templates in the WEB-INF/views package
<code>public SpringTemplateEngine templateEngine() {</code>	Selects HTML5 as the template mode

Run-time analysis of the program

Let N represent the number of items returned for one of the retailers.

Run-time analysis for searching

(1) Setup

This takes constant time $O(1)$ because it does not depend on N .

(2) Search

```
stuff1 //initialize drivers
stuff2 //make amazon url
for i=0 to N
    stuff3 //parse each amazon item
stuff4 //make ebay url
for i=0 to N
    stuff5 //parse each ebay item
for i = 0 to N
    stuff6 //add ebay items to amazon item list
 $T(N) = \text{stuff1} + \text{stuff2} + \text{stuff3} + \text{stuff4} + N\text{stuff5} + N\text{stuff6}$ 
 $O(N) = N$ 
```

Therefore this takes linear time $O(N)$ because the dominant term is linear.

(3) Display

```
for i = 0 to 2N //for all items (N from Amazon and N from Ebay)
    stuff1 //show item in <div> container, along with button
    stuff2 //prepare pop-up
```

This takes linear time $O(N)$, as the coefficient is insignificant.

(4) Total

$\text{Total} = O(1) + O(N) + O(N) = O(N)$

Therefore, searching takes linear time $O(N)$.

Run-time analysis for sorting items

This takes $O(N \log N)$ time, because the items are sorted by mergeSort, which, from the master theorem, is $O(N \log N)$ because the time taken to split the array is linear.