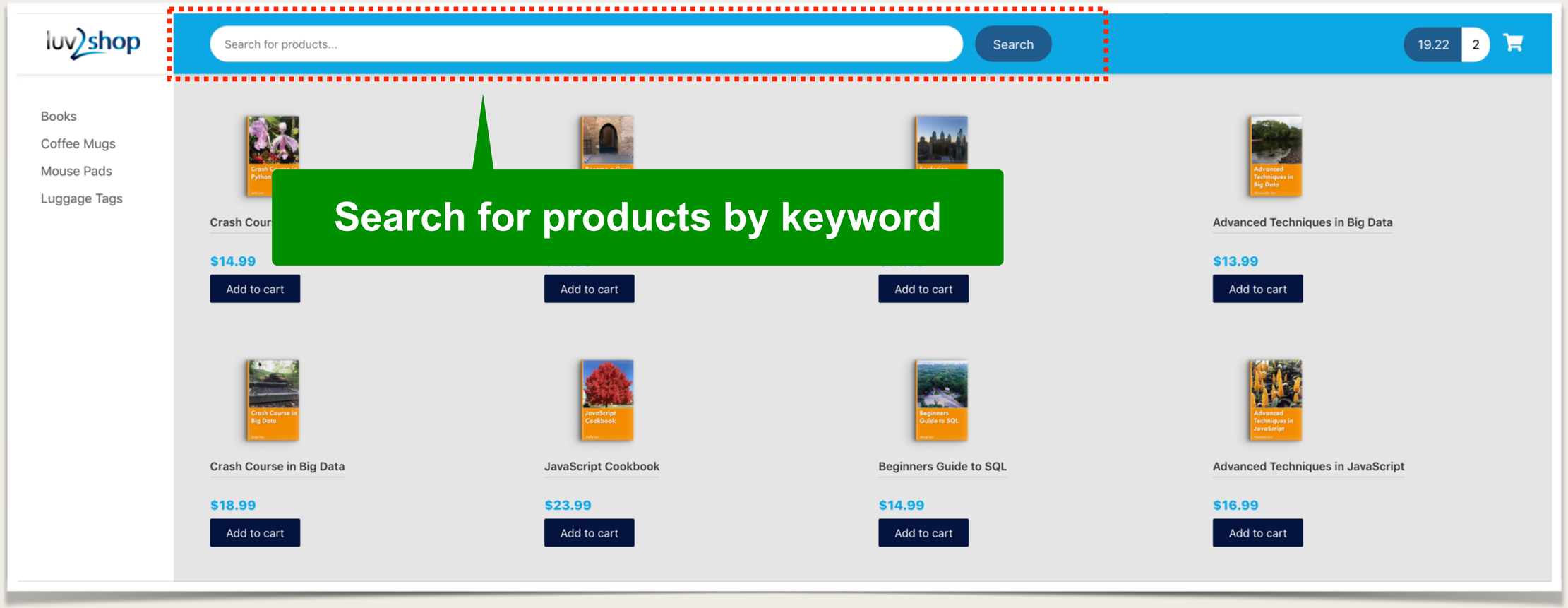
Search By Keyword

Search for Products by Keyword



Development Process

1. Modify Spring Boot app - Add a new search method
2. Create new component for search
3. Add new Angular route for searching
4. Update SearchComponent to send data to search route
5. Enhance ProductListComponent to search for products with ProductService
6. Update ProductService to call URL on Spring Boot app

Step 1: Modify Spring Boot app - Search Method

- Spring Data REST and Spring Data JPA supports "query methods"
- Spring will construct a query based on method naming conventions

Magic!

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...
}
```

Step 1: Modify Spring Boot app - Search Method

- Find products based on name

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...
}
```

"Containing" ... similar to SQL: "LIKE"

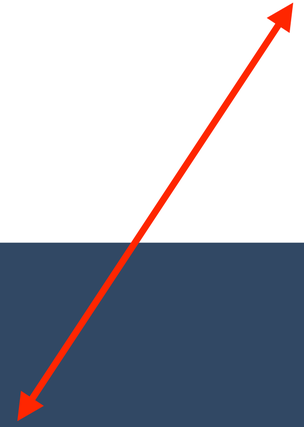
Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);
```

Behind the scenes, Spring will execute a query similar to this

```
SELECT * FROM Product p
WHERE
p.name LIKE CONCAT('%', :name ,'%')
```



Step 1: Modify Spring Boot app - Search Method

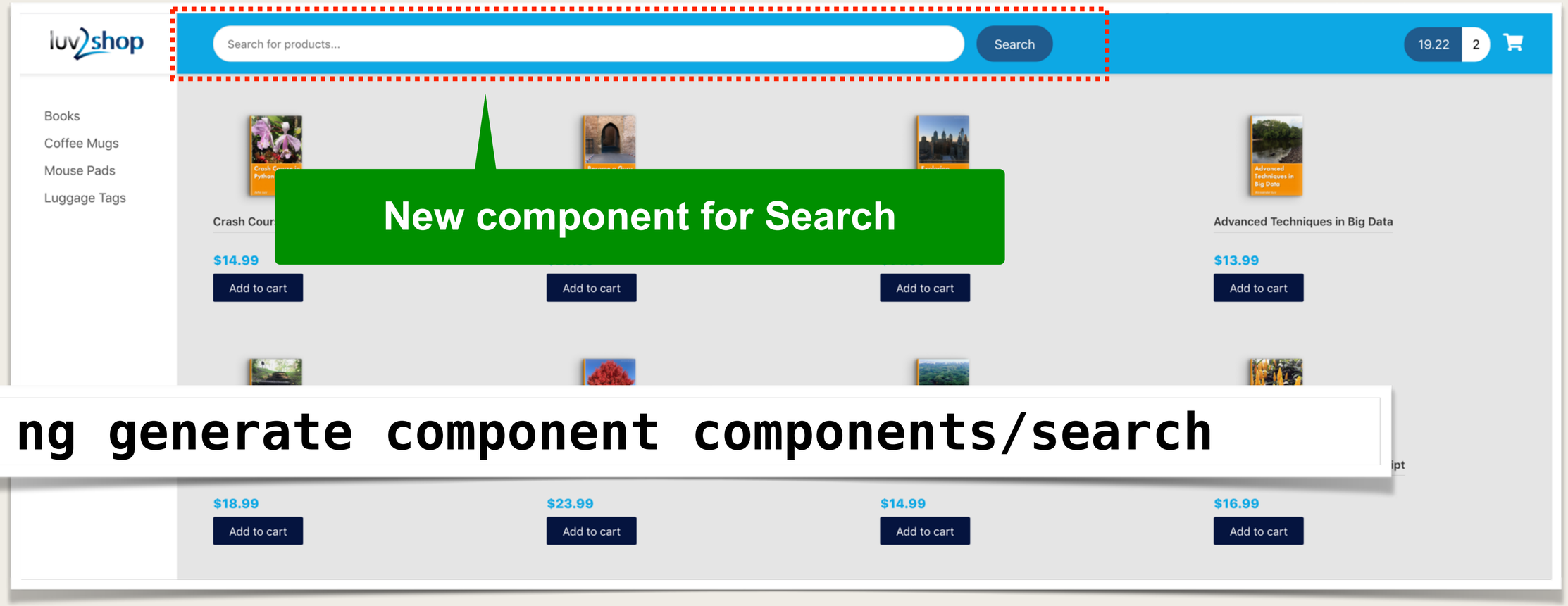
File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```

http://localhost:8080/api/products/search/findByNameContaining?name=Python

To pass data to REST API

Step 2: Create new component for Search

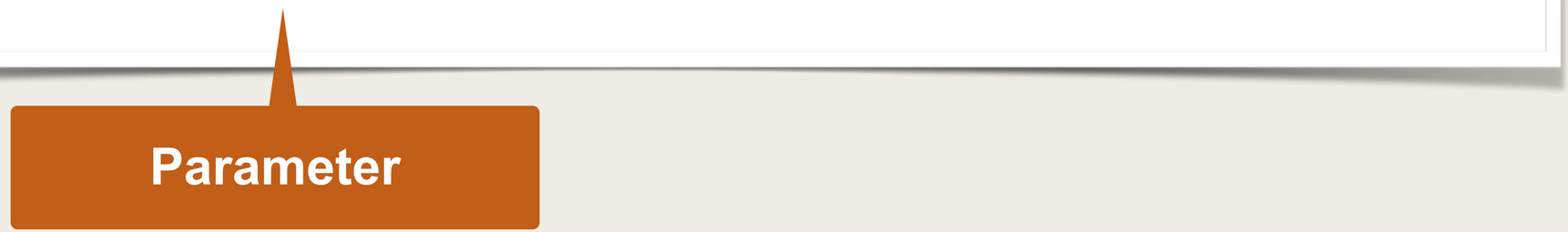



```
> ng generate component components/search
```

Step 3: Add new Angular route for searching

File: app.module.ts

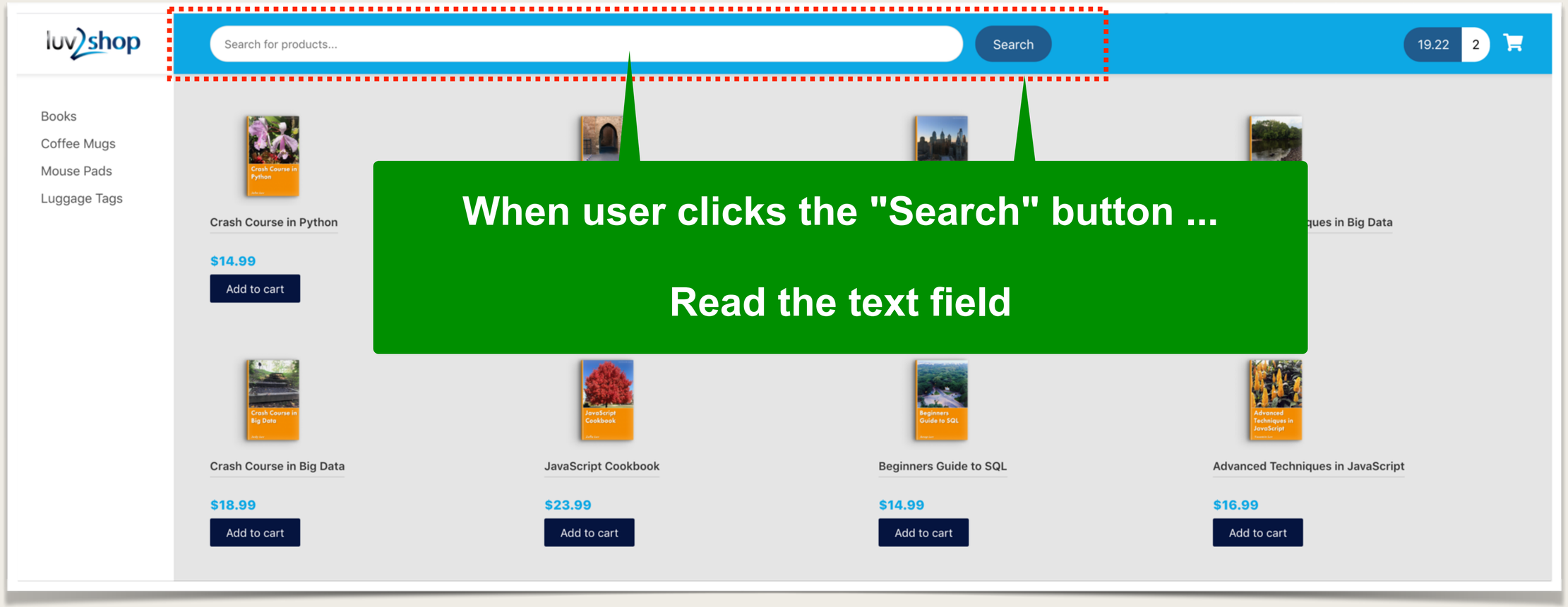
```
...
  {path: 'search/:keyword', component: ProductListComponent},
...
```



Quick Discussion

Event Binding

Quick Discussion - Event Binding



Event Binding

- In Angular, you can listen for events with "event binding"
- In other languages / frameworks, also know

```
<button (click)="doMyCustomWork()">Search</button>
```

Event handler

Listen for "click" event

Call a method in our Angular component code

Can be any method name we define

Event Binding - Example

Call a method in our Angular component code

```
<button (click)="doMyCustomWork()">Search</button>
```

Listen for "click" event

```
export class SearchComponent implements OnInit {

  doMyCustomWork() {
    console.log(`Hey! You pushed my button!`);
 }

  ...
}
```

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

Listen for the "enter" key

The text the user typed in

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />

<button (click)="doMyCustomWork(myInput.value)">Search</button>
```

The text the user typed in

Listen for "click" event

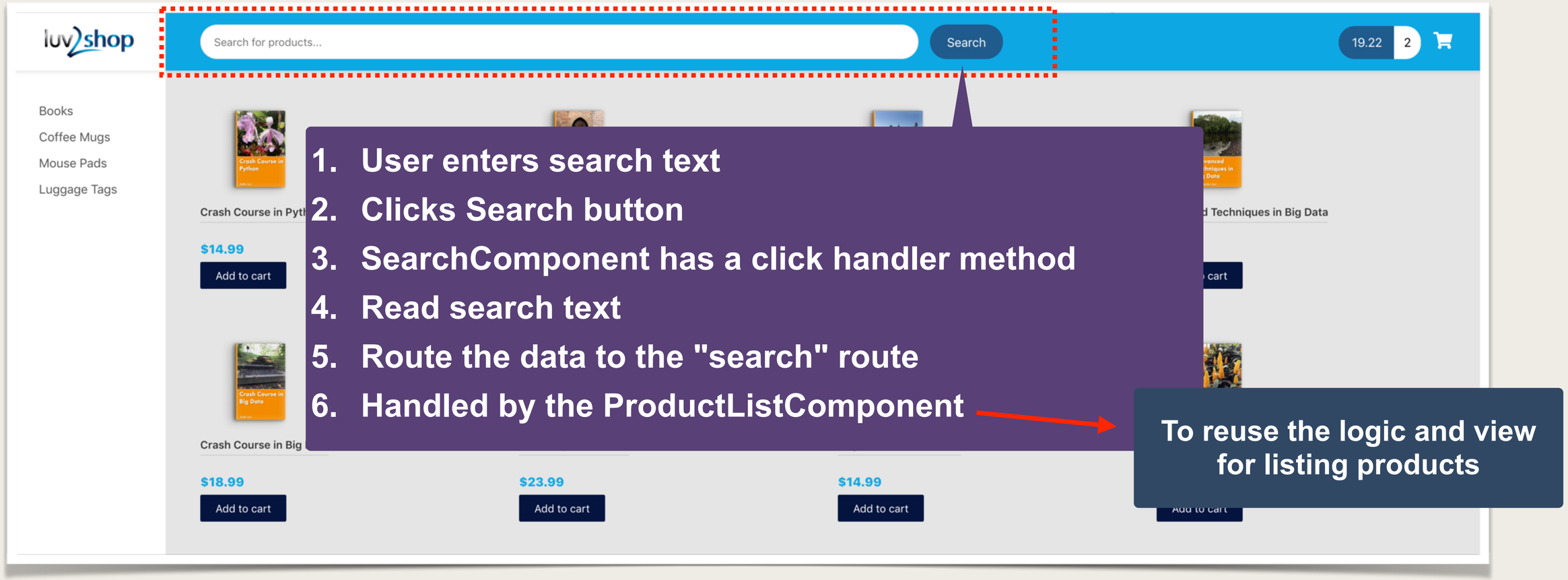
```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```

Other Events

Name	Description
focus	An element has received focus
blur	An element has lost focus
keyup	Any key is released. For a specific key, the ***enter*** key, use: **keyup.*enter***
keydown	Any key is pressed
dblclick	The mouse is clicked twice on an element
more ….	

https://developer.mozilla.org/en-US/docs/Web/Events

Step 4: Update SearchComponent to send data to search route



Step 4: Update SearchComponent to send data to search route

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```

Step 4: Update SearchComponent to send data to search route

Method defined in our component

We can give it any name

File: search.component.ts

```
export class SearchComponent implements OnInit {

  constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```

Route the data to our "search" route

It will be handled by the ProductListComponent

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```

To reuse the logic and view for listing products

Step 5: Enhance component to search for products with product service

File: app.module.ts

```
...
  {path: 'search/:keyword', component: ProductListComponent},
...
```

File: product-list.component.ts

```
...
    const theKeyword: string = this.route.snapshot.paramMap.get('keyword');

    // now search for the products using keyword
    this.productService.searchProducts(theKeyword).subscribe(
      data => {
        this.products = data;
      }
    )
...
```

File: search.component.ts

```
export class SearchComponent implements OnInit {

 constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```

Passed in from SearchComponent

Step 6: Update product service to call URL on Spring Boot app

File: product.service.ts

```
...

  searchProducts(theKeyword: string): Observable<Product[]> {

    // need to build URL based on the keyword
    const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;

    return this.httpClient.get<GetResponseProducts>(searchUrl)
                          .pipe(
                            map(response => response._embedded.products));
  }

...

interface GetResponseProducts {
  _embedded: {
    products: Product[];
  }
}
```

URL for searching products

Call REST API

Returns an observable

Map the JSON data from Spring Data REST to Product array

Unwraps the JSON from Spring Data REST _embedded entry