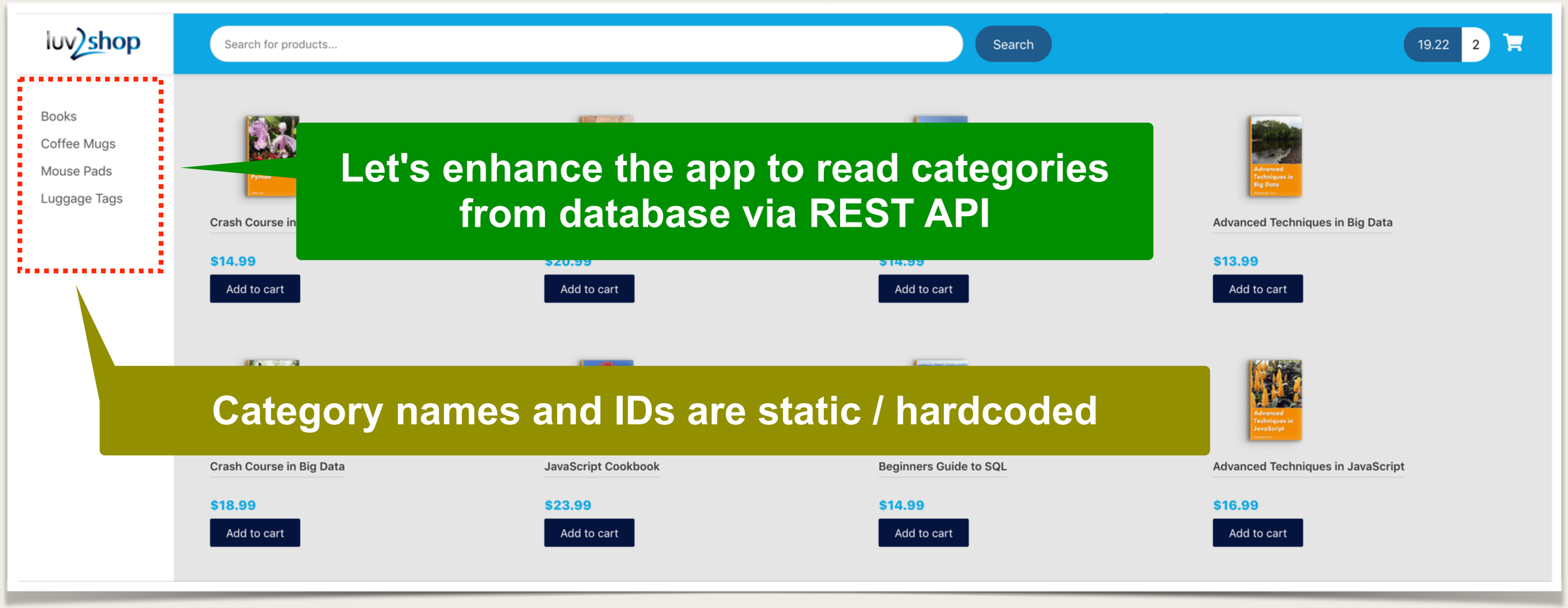
Search By Category

Search for Products by Category



Development Process

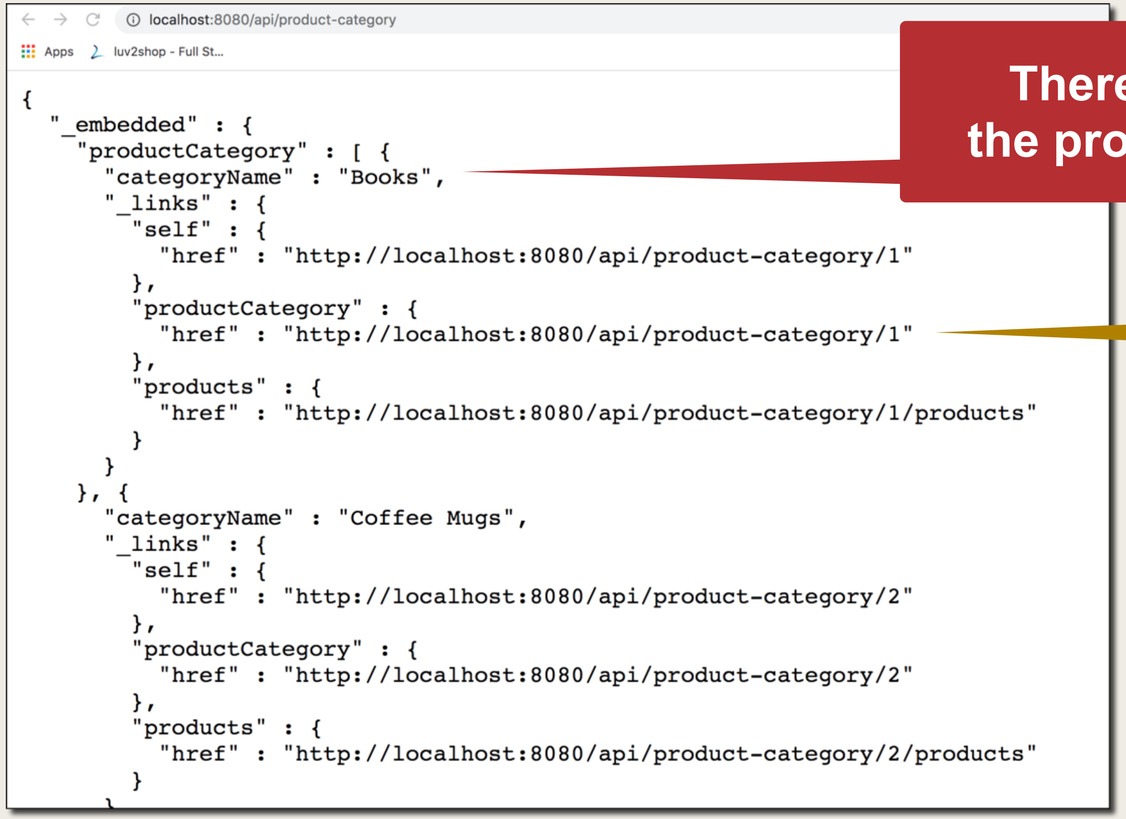
1. Modify Spring Boot app - Expose entity ids
2. Create a class: ProductCategory
3. Create new component for menu
4. Enhance menu component to read data from product service
5. Update product service to call URL on Spring Boot app
6. In HTML, replace hard-coded links with menu component

Step 1: Modify Spring Boot app - Expose entity ids

- By default, Spring Data REST does not expose entity ids
- We need entity IDs for a number of use cases
  - Get a list of product categories by id
  - Master/detail view ... get a product by id

Step 1: Modify Spring Boot app - Expose entity ids

- By default, Spring Data REST does not expose entity ids



There is no entity id at the productCategory level

Actually, entity id is embedded in the HATEOS links.

But no easy access.
Requires parsing URL string

This approach is cumbersome and brittle :-(

Step 1: Modify Spring Boot app - Expose entity ids

- This is what we need

Entity id at the productCategory level

Easy access! :-)

Step 1: Modify Spring Boot app - Expose entity ids

- Update Spring Data REST config to expose entity ids

```
// expose entity ids
//
// - gets a list of all entity classes from the entity manager
// - create an array of the entity types
// - get the entity types for the entities
// - expose the entity ids for the array of entity/domain types
```

Step 1: Modify Spring Boot app - Expose entity ids

- Update Spring Data REST config to expose entity ids

File: MyDataRestConfig.java

```
@Configuration
public class MyDataRestConfig implements RepositoryRestConfigurer {

 private EntityManager entityManager;

 @Autowired
 public MyDataRestConfig(EntityManager theEntityManager) {
     entityManager =  theEntityManager;
 }

 ...
}
```

Autowire JPA entity manager

Step 1: Modify Spring Boot app - Expose entity ids

File: MyDataRestConfig.java

```
@Override
public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config) {

    // ...

    // call an internal helper method
    exposeIds(config);
}

private void exposeIds(RepositoryRestConfiguration config) {

    // expose entity ids
    //

    // - gets a list of all entity classes from the entity manager
    Set<EntityType<?>> entities = entityManager.getMetamodel().getEntities();

    // - create an array of the entity types
    List<Class> entityClasses = new ArrayList<>();

    // - get the entity types for the entities
    for (EntityType tempEntityType : entities) {
        entityClasses.add(tempEntityType.getJavaType());
    }

    // - expose the entity ids for the array of entity/domain types
    Class[] domainTypes = entityClasses.toArray(new Class[0]);
    config.exposeIdsFor(domainTypes);
}
```

Step 1: Modify Spring Boot app - Expose entity ids

- The configuration gives us the desired output

Entity id at the productCategory level

Easy access! :-)

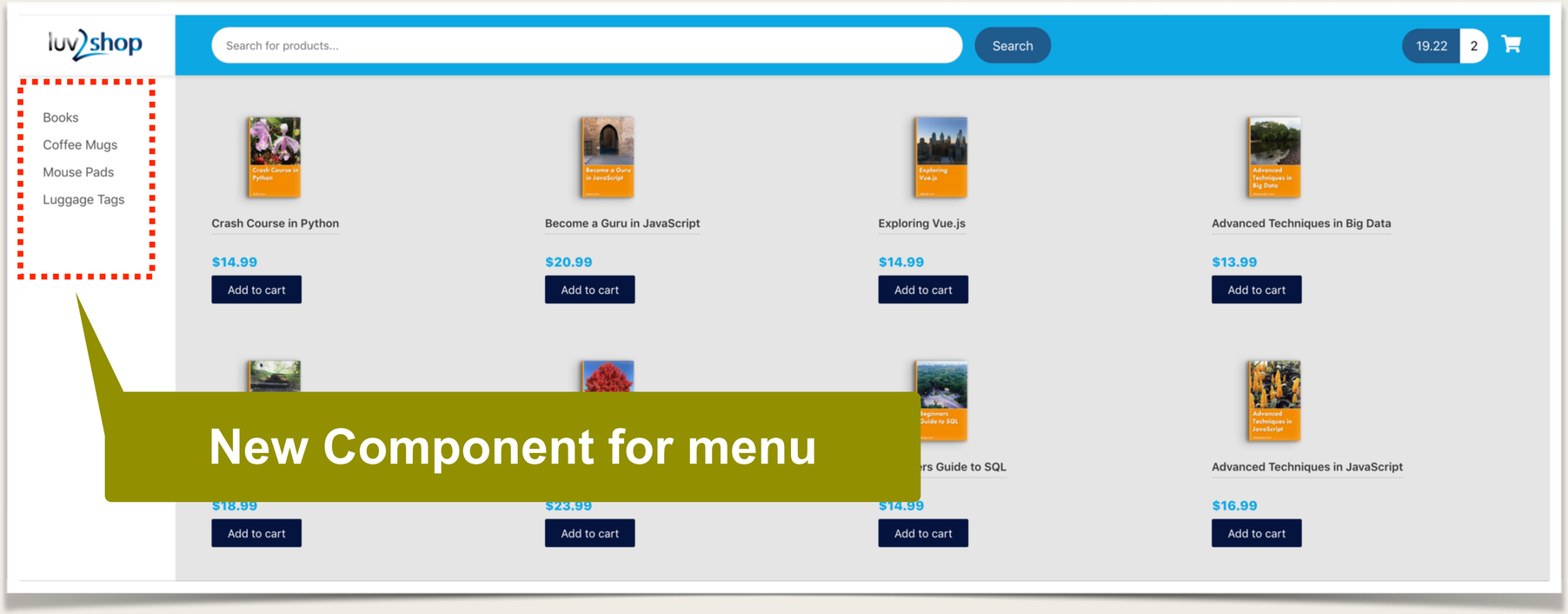
Step 2: Create class: ProductCategory

```
> ng generate class common/product-category
```

File: product-category.ts

```
export class ProductCategory {
    id: number;
    categoryName: string;
}
```

Step 3: Create new component for menu




```
> ng generate component components/product-category-menu
```

Step 4: Enhance component to read categories from service

File: product-category-menu.component.ts

```
export class ProductCategoryMenuComponent implements OnInit {

  productCategories: ProductCategory[];

  constructor(private productService: ProductService) { }

  ngOnInit() {
    this.listProductCategories();
  }

  listProductCategories() {

    this.productService.getProductCategories().subscribe(
      data => {
        console.log('Product Categories=' + JSON.stringify(data));
        this.productCategories = data;
      }
    )
  }

}
```

Define our property

Inject the service

Invoke the service

Log data returned from service

Assign data to our property

Step 5: Update product service to call URL on Spring Boot app

File: product.service.ts

```
export class ProductService {

  private categoryUrl = 'http://localhost:8080/api/product-category';
  ...

  getProductCategories(): Observable<ProductCategory[]> {

    return this.httpClient.get<GetResponseProductCategory>(this.categoryUrl).pipe(
      map(response => response._embedded.productCategory)
    );
  }

}

interface GetResponseProductCategory {
  _embedded: {
    productCategory: ProductCategory[];
  }
}
```

URL for product categories

Call REST API

Returns an observable

Map the JSON data from Spring Data REST to ProductCategory array

Unwraps the JSON from Spring Data REST _embedded entry

Step 6: Replace hard-coded links with menu component

Old Version

File: app.component.html

```
<!-- MENU SIDEBAR-->
<aside class="menu-sidebar d-none d-lg-block">
  <div class="logo">
    <a href="#">
      <img src="assets/images/logo.png" alt="luv2shop" class="img-responsive">
    </a>
  </div>
  <div class="menu-sidebar-content js-scrollbar1">
    <nav class="navbar-sidebar">
      <ul class="list-unstyled navbar-list">
        <li>
          <a routerLink="/category/1" routerLinkActive="active-link">Books</a>
        </li>
        <li>
          <a routerLink="/category/2" routerLinkActive="active-link">Coffee Mugs</a>
        </li>
      </ul>
    </nav>
  </div>
</aside>
<!-- END MENU SIDEBAR-->
```

New Version

File: app.component.html

```
<!-- MENU SIDEBAR-->
<aside class="menu-sidebar d-none d-lg-block">
  <div class="logo">
    <a href="#">
      <img src="assets/images/logo.png" alt="luv2shop" class="img-responsive">
    </a>
  </div>

  <app-product-category-menu></app-product-category-menu>

</aside>
<!-- END MENU SIDEBAR-->
```

Our new menu component

Step 6: Replace hard-coded links with menu component

File: product-category-menu.component.html

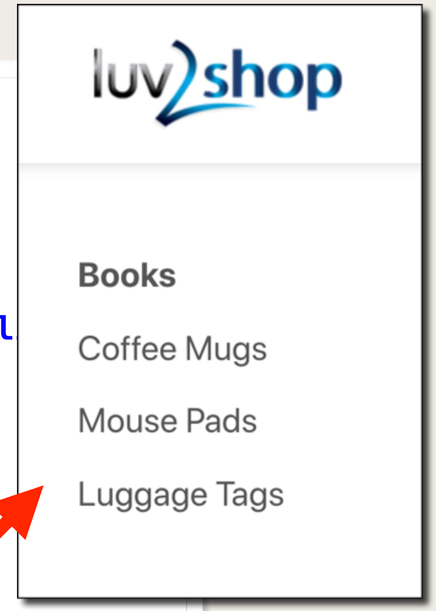
```
<div class="menu-sidebar-content js-scrollbar1">
    <nav class="navbar-sidebar">
        <ul class="list-unstyled navbar-list">

            <li *ngFor="let tempProductCategory of productCategories">

                <a routerLink="/category/{{ tempProductCategory.id }}" routerLinkActive="active-l
                    {{ tempProductCategory.categoryName }}
                </a>

            </li>

        </ul>
    </nav>
</div>
```

Loop over categories and build links dynamically

Add category id to link

Display category name

From REST API



id	category_name
1	Books
2	Coffee Mugs
3	Mouse Pads
4	Luggage Tags