

Search By Category



Search for Products by Category

The screenshot shows a web application interface for 'luv2shop'. On the left, there's a sidebar with a red dashed border containing category links: Books, Coffee Mugs, Mouse Pads, and Luggage Tags. The main content area displays a grid of products. A green callout box points from the sidebar towards the grid, containing the text: 'Let's enhance the app to read categories from database via REST API'. Another green callout box points from the bottom left towards the grid, containing the text: 'Category names and IDs are static / hardcoded'. The products listed include:

Product	Price	Action
Crash Course in Python	\$14.99	Add to cart
Crash Course in Big Data	\$18.99	Add to cart
JavaScript Cookbook	\$23.99	Add to cart
Beginners Guide to SQL	\$14.99	Add to cart
Advanced Techniques in Big Data	\$13.99	Add to cart
Advanced Techniques in JavaScript	\$16.99	Add to cart

Development Process

Step-By-Step

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

```
{  
    "_embedded" : {  
        "productCategory" : [ {  
            "categoryName" : "Books",  
            "_links" : {  
                "self" : {  
                    "href" : "http://localhost:8080/api/product-category/1"  
                },  
                "productCategory" : {  
                    "href" : "http://localhost:8080/api/product-category/1"  
                },  
                "products" : {  
                    "href" : "http://localhost:8080/api/product-category/1/products"  
                }  
            }  
        },  
        {  
            "categoryName" : "Coffee Mugs",  
            "_links" : {  
                "self" : {  
                    "href" : "http://localhost:8080/api/product-category/2"  
                },  
                "productCategory" : {  
                    "href" : "http://localhost:8080/api/product-category/2"  
                },  
                "products" : {  
                    "href" : "http://localhost:8080/api/product-category/2/products"  
                }  
            }  
    }  
}
```

There is no entity id at the productCategory level

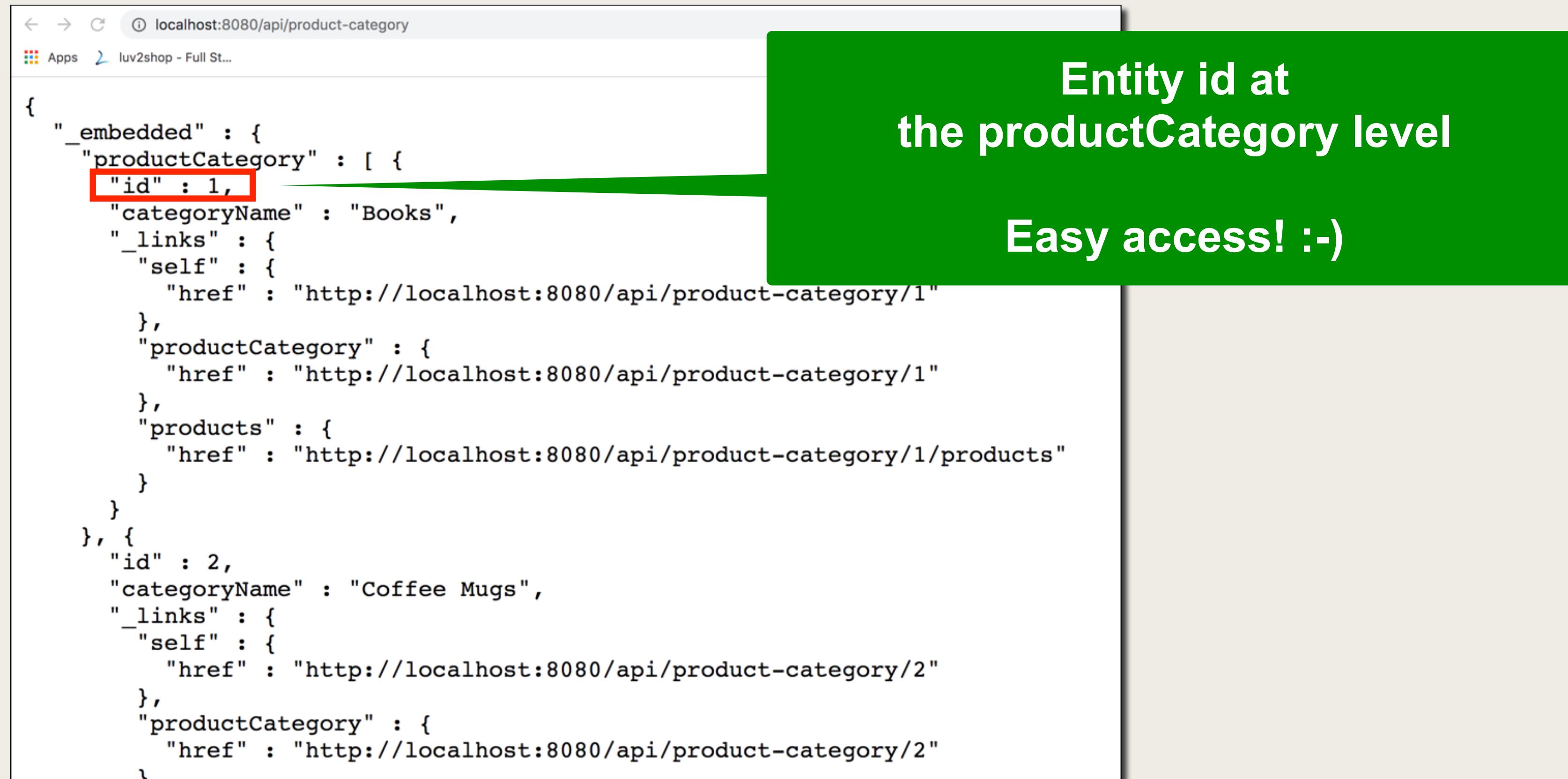
Actually, entity id is embedded in the HATEOS links.

**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



```
{  
    "_embedded" : {  
        "productCategory" : [ {  
            "id" : 1, // Entity id highlighted with a red box  
            "categoryName" : "Books",  
            "_links" : {  
                "self" : {  
                    "href" : "http://localhost:8080/api/product-category/1"  
                },  
                "productCategory" : {  
                    "href" : "http://localhost:8080/api/product-category/1"  
                },  
                "products" : {  
                    "href" : "http://localhost:8080/api/product-category/1/products"  
                }  
            }  
        }, {  
            "id" : 2,  
            "categoryName" : "Coffee Mugs",  
            "_links" : {  
                "self" : {  
                    "href" : "http://localhost:8080/api/product-category/2"  
                },  
                "productCategory" : {  
                    "href" : "http://localhost:8080/api/product-category/2"  
                }  
            }  
        }  
    }  
}
```

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();

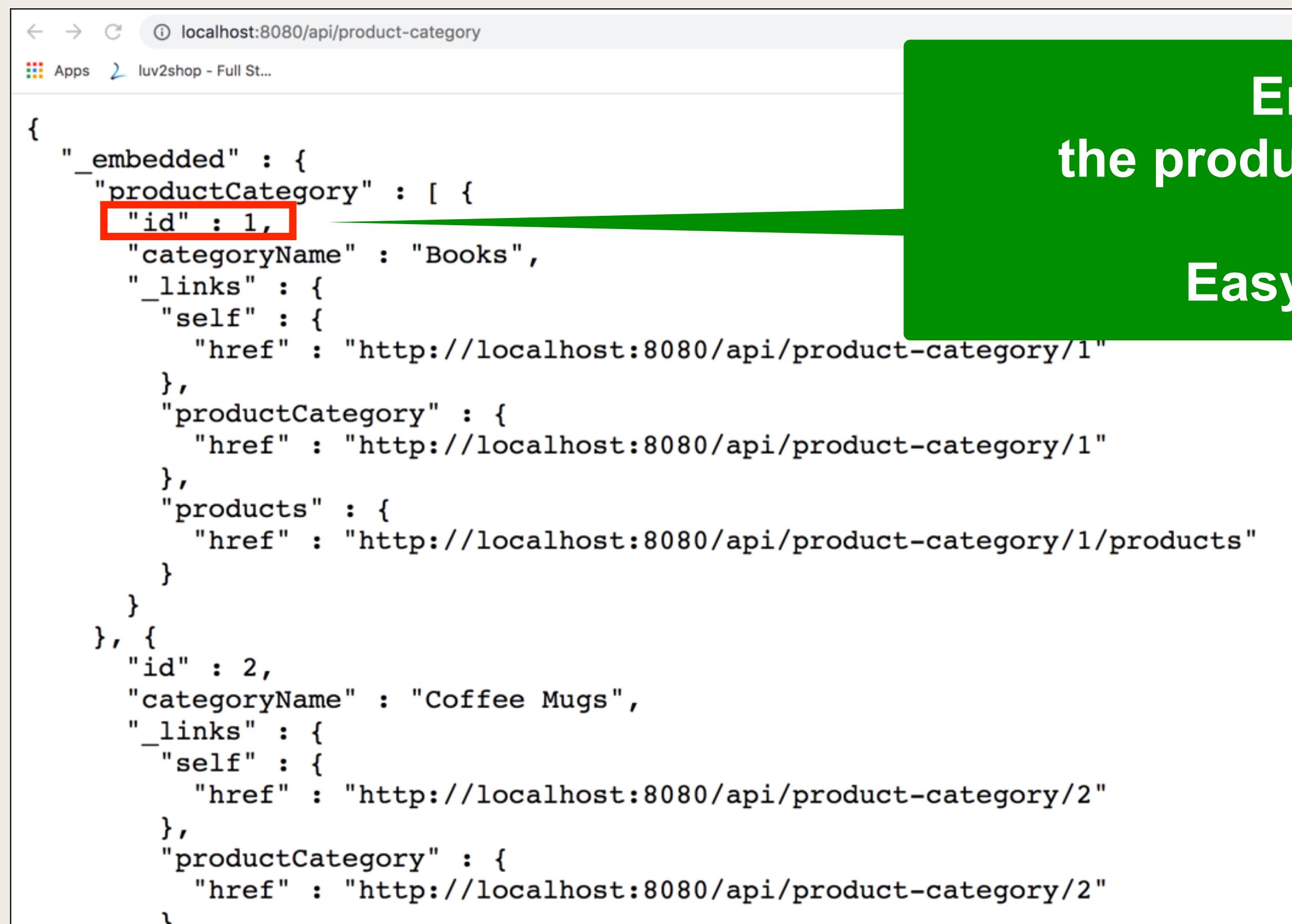
    // - create an array of the entity types
    List<Class> entityClasses = new ArrayList<>();

    // - 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



A screenshot of a web browser window titled "localhost:8080/api/product-category". The page displays a JSON array of product categories. One category, with id 1 and name "Books", is highlighted with a red box around its "id" field. A green callout box points to this field with the text "Entity id at the productCategory level". Another green callout box below it says "Easy access! :-)".

```
{
  "_embedded" : {
    "productCategory" : [ {
      "id" : 1,
      "categoryName" : "Books",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/product-category/1"
        },
        "productCategory" : {
          "href" : "http://localhost:8080/api/product-category/1"
        },
        "products" : {
          "href" : "http://localhost:8080/api/product-category/1/products"
        }
      }
    },
    {
      "id" : 2,
      "categoryName" : "Coffee Mugs",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/product-category/2"
        },
        "productCategory" : {
          "href" : "http://localhost:8080/api/product-category/2"
        }
      }
    }
  }
}
```

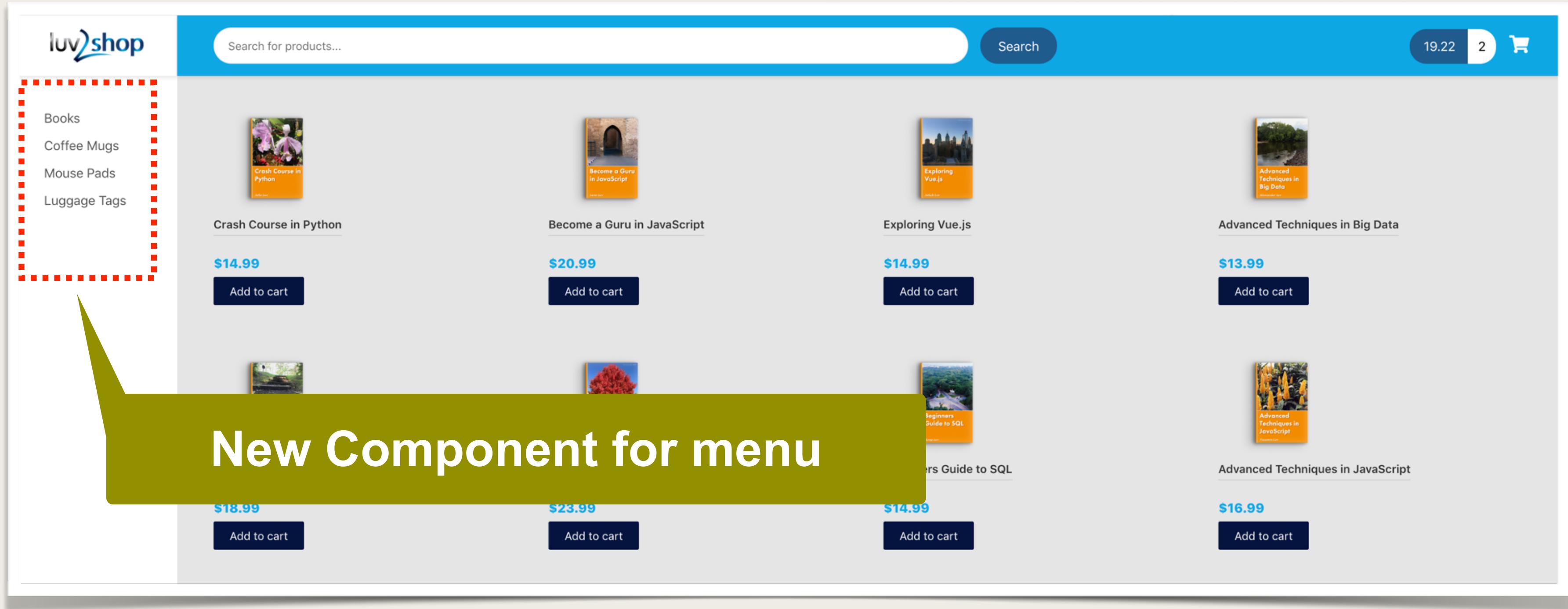
Step 2: Create class: ProductCategory

```
> 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[];
```

Define our property

```
    constructor(private productService: ProductService) { }
```

```
    ngOnInit() {
        this.listProductCategories();
    }
```

Inject the service

```
    listProductCategories() {
```

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

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

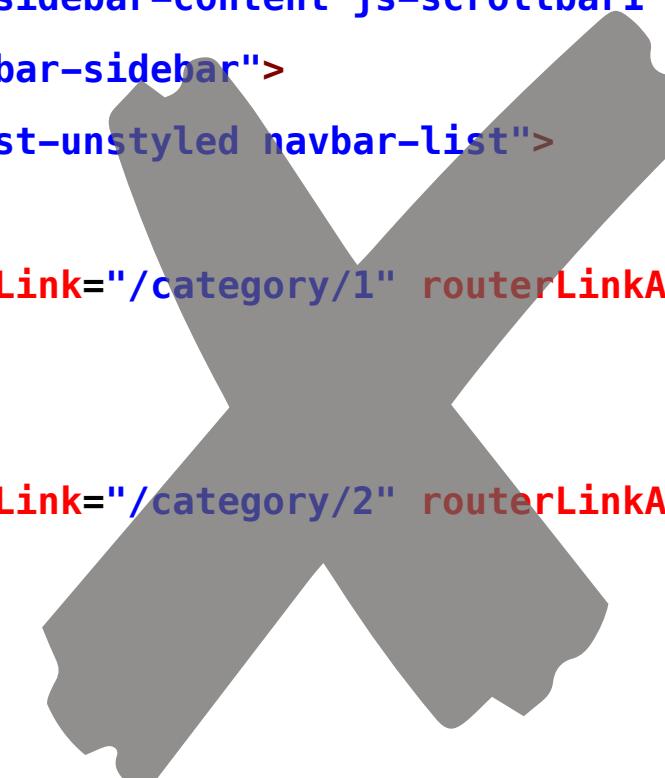
Unwraps the JSON from
Spring Data REST
_embedded entry

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="#">
      
    </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="#">
      
    </a>
  </div>
  <div>
    <app-product-category-menu></app-product-category-menu>
  </div>
</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">

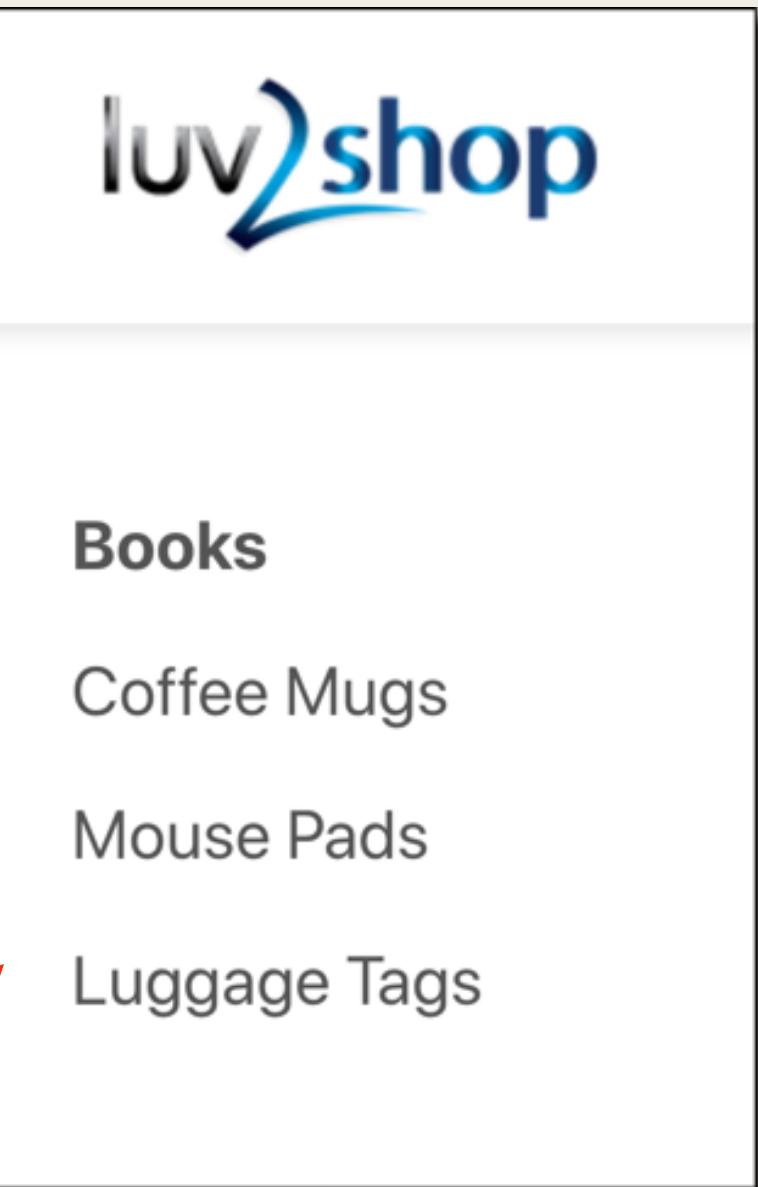
      <li *ngFor="let tempProductCategory of productCategories">
        <a routerLink="/category/{{ tempProductCategory.id }}" routerLinkActive="active-link">
          {{ tempProductCategory.categoryName }}
        </a>
      </li>
    </ul>
  </nav>
</div>
```

Display category name

Loop over categories
and build links
dynamically

Add category id to link

From REST API



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