Hardening your Bookstore

Page and Site Robustness

Dr A

What is Hardening?

"to make rigid or unyielding; stiffen"

- The goal is to improve site resilience and perform basic testing.
- What is resilience?
  - To always show a page that offers a way to continue.

- (Background: When forging a knife, hardening involves quenching hot steel in oil. Any imperfections, warps or cracks in the blade show up during this process. Sometimes the blades shatter - but that won't happen to your site!)

What are we hardening our site against?

- Some of the activities we want to harden against:
  - Refresh each page, type in strange urls, hit the back button
  - Access the site using a variety of home page address variations - make sure it is accessible.
  - Have an empty cart and hit each page
  - Arrive on the confirmation page without order details

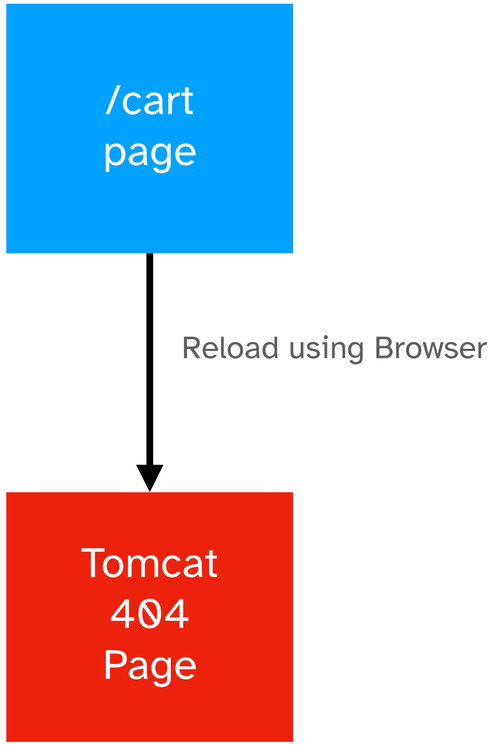
- Test with your production server (localhost:8080 or cs5244.cs.vt.edu)
  - Some deficiencies are masked by the development server

- Let's learn how to harden our site to handle these activities!

Reloading Pages

Page Reloading: The Issue

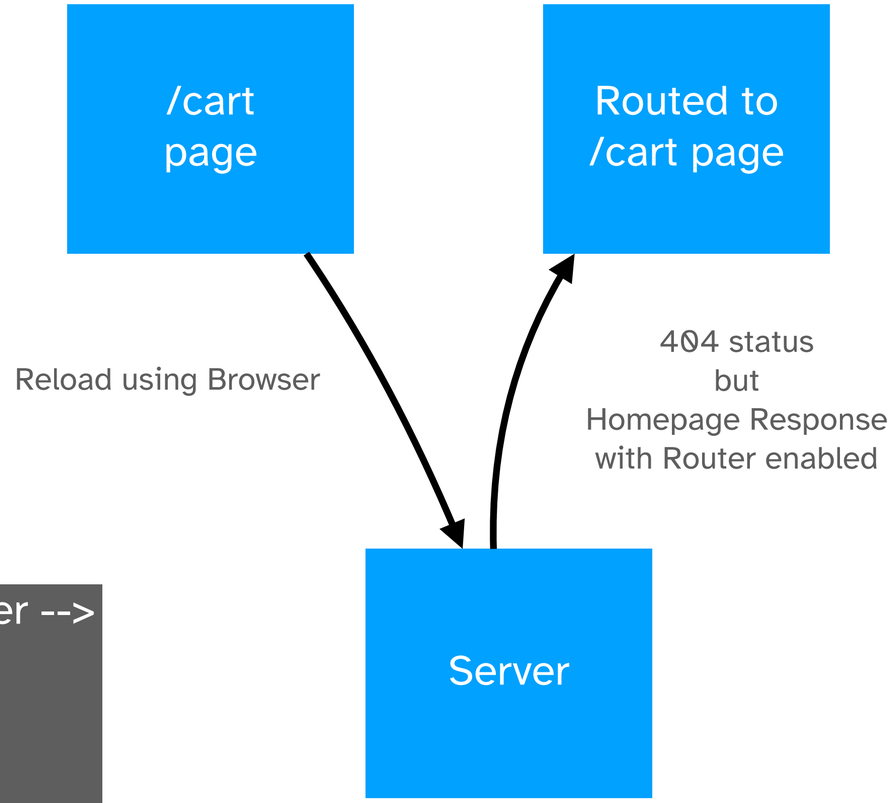
- Navigate to any non-home page.
- Reload the page.
- You will see a Server 404 page.
- Why? The server does not know about page urls like ***/cart***
- The server responds to /cart with a 404 (Not Found) response code and sends that that page by default.



Page Reloading: The Solution

- The server needs to admit when it does not know about a page
- Rather than delegate to a default error page, delegate to the home page.
- This sends Router-ready JavaScript back to the browser to pick up /cart and route there.
- To make this happen, add this to the bottom of WEB-INF/web.xml inside the </webapp> tag:

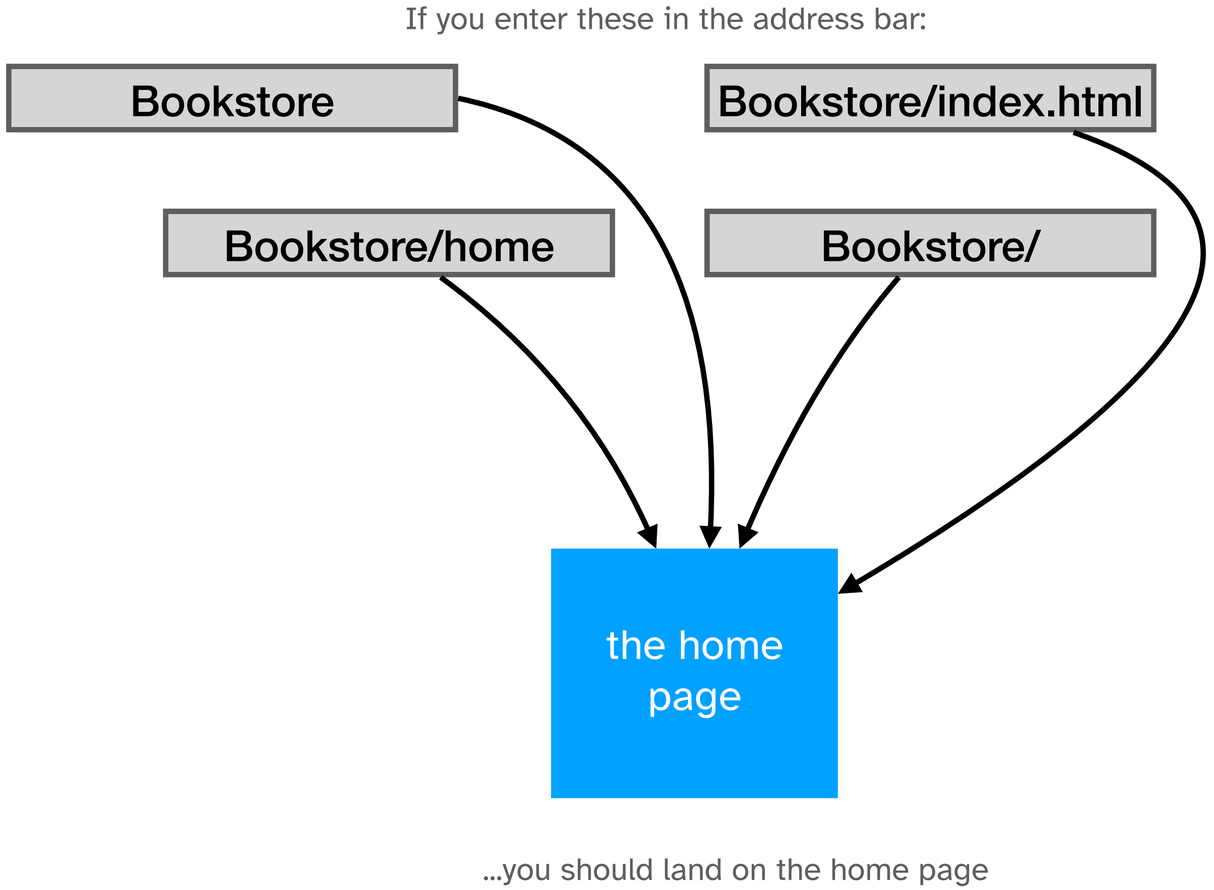
```
<!-- any unknown requests will be sent to the client application in the browser -->
<error-page>
  <error-code>404</error-code>
  <location>/index.html</location>
</error-page>
```



Home Page Links

Home Page Links: The Issue

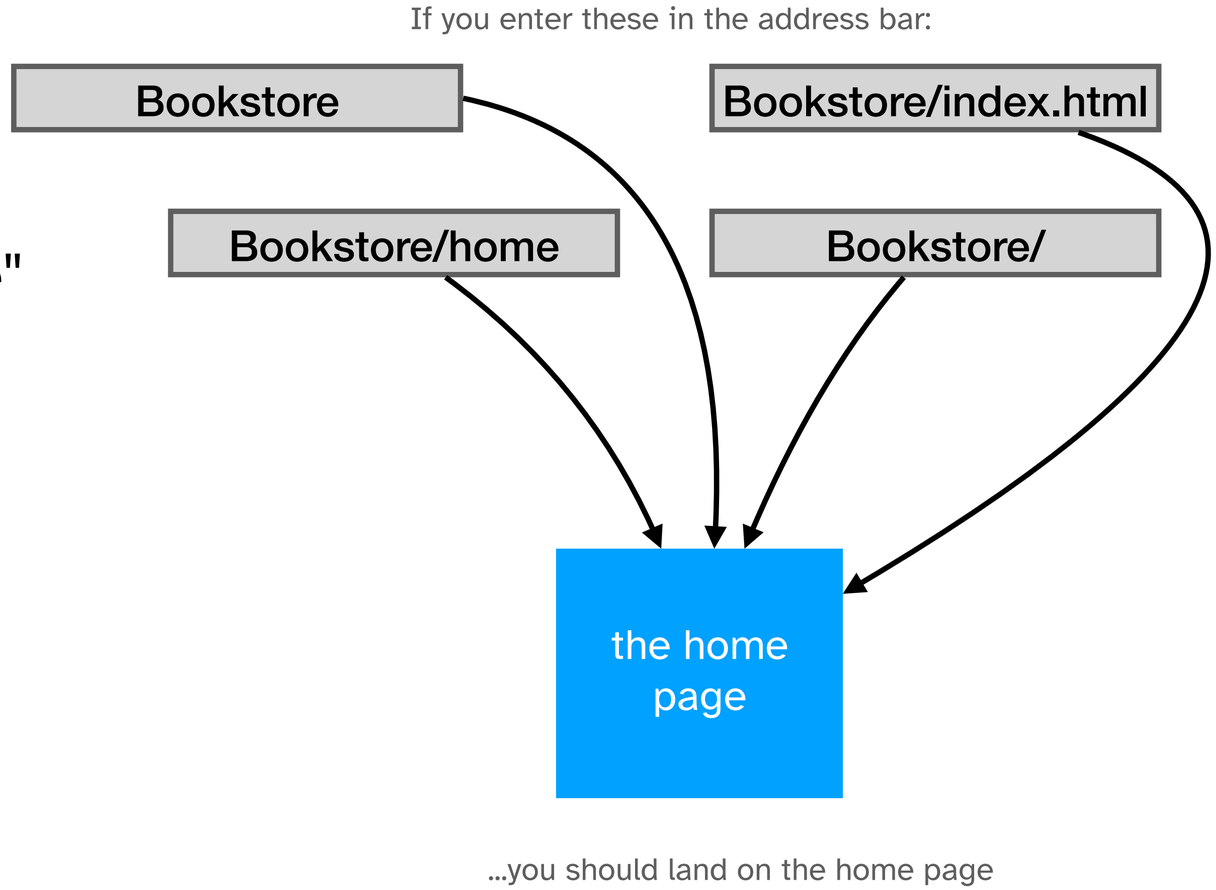
- If you navigate to https://cs5244.cs.vt.edu/YourBookstore using these variations, you should land on the home page.
- You may be surprised that some of these may not work.



Home Page Links: One Solution

- Let's tell the vue-router that "/", "/index.html" are really aliases for "/home"
- This way, the same route and component will be used for each variation.

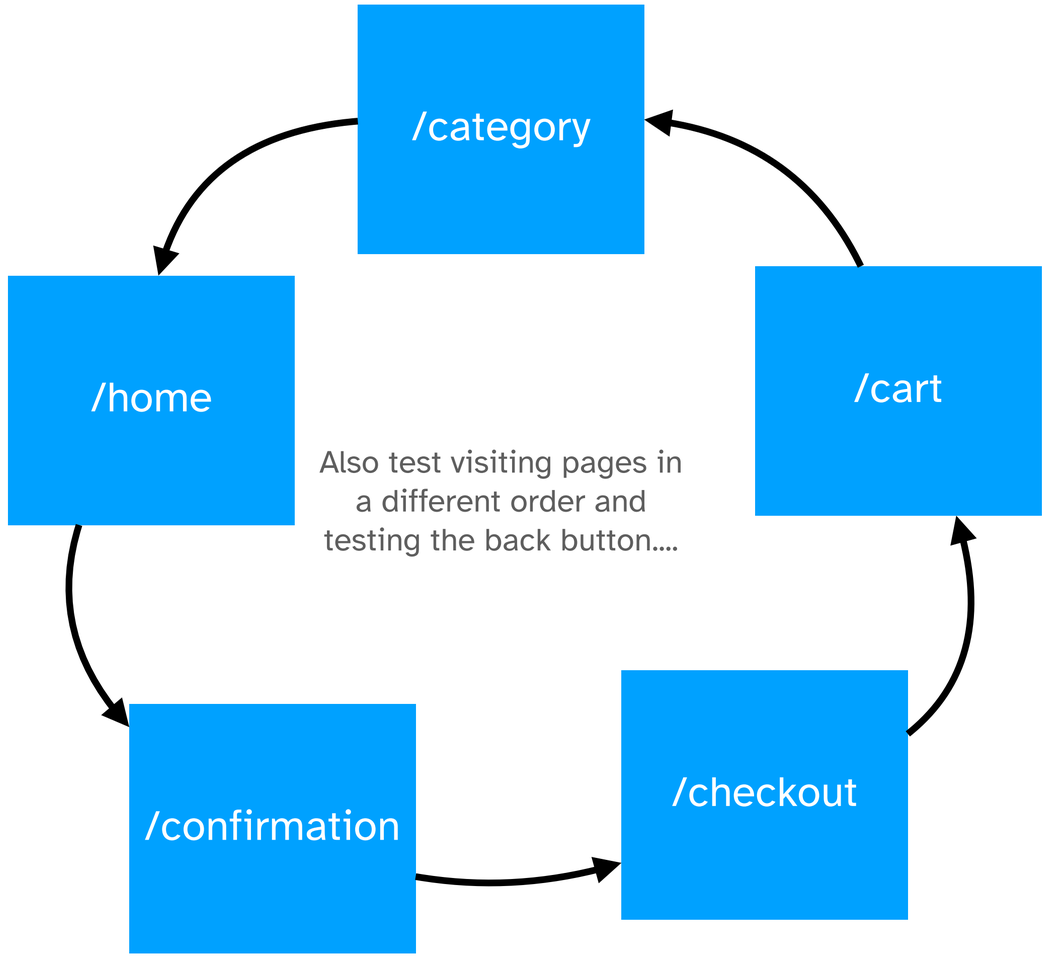
```
// In router/index.ts
{
  path: "/home",
  name: "home-view",
  component: HomeView,
  alias: ["/", "/index.html"],
},
```



Going Backwards: Back Button

Back Button Testing

- We need to test pressing the back button from each page
- We expect to see the prior page.
- If your cart was cleared placing an order, /cart and /checkout should at least display a message and not break
- Your selected category should be remembered when going between



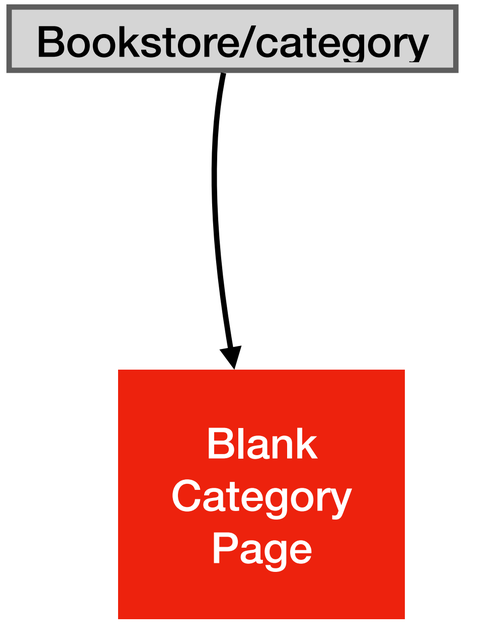
Going to /category

/category: The Issue

- Navigate to https://cs5244.cs.vt.edu/YourBookstore/category with no category name.
- You may be surprised to see a blank or empty page. We'd rather see the default category page.
- Why? /category does not match the /category/:name route

```
// In router/index.ts
{
  path: '/category/:name',
  name: 'category',
  component: Category,
  props: true
}
```

If you enter this in the address bar:

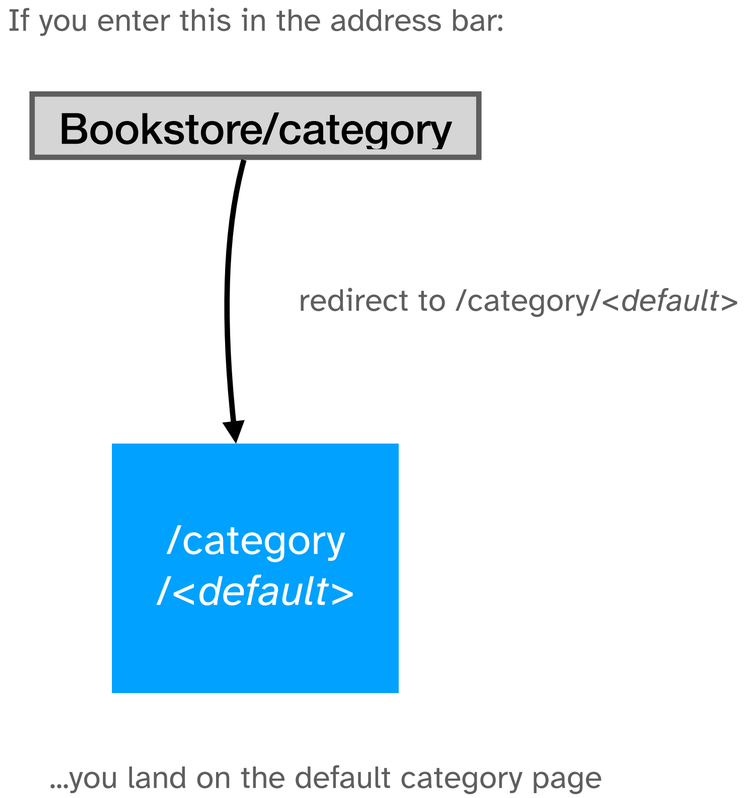


...you land on a blank page

/category: One Possible Solution

- There are many ways to solve this issue.
- Evaluate solutions using the amount of change, readability and maintainability.
- Many other solutions involve changing the code to support a missing category name.
- Instead, let's add a /category route to index.ts; we can specify to redirect to the default category!

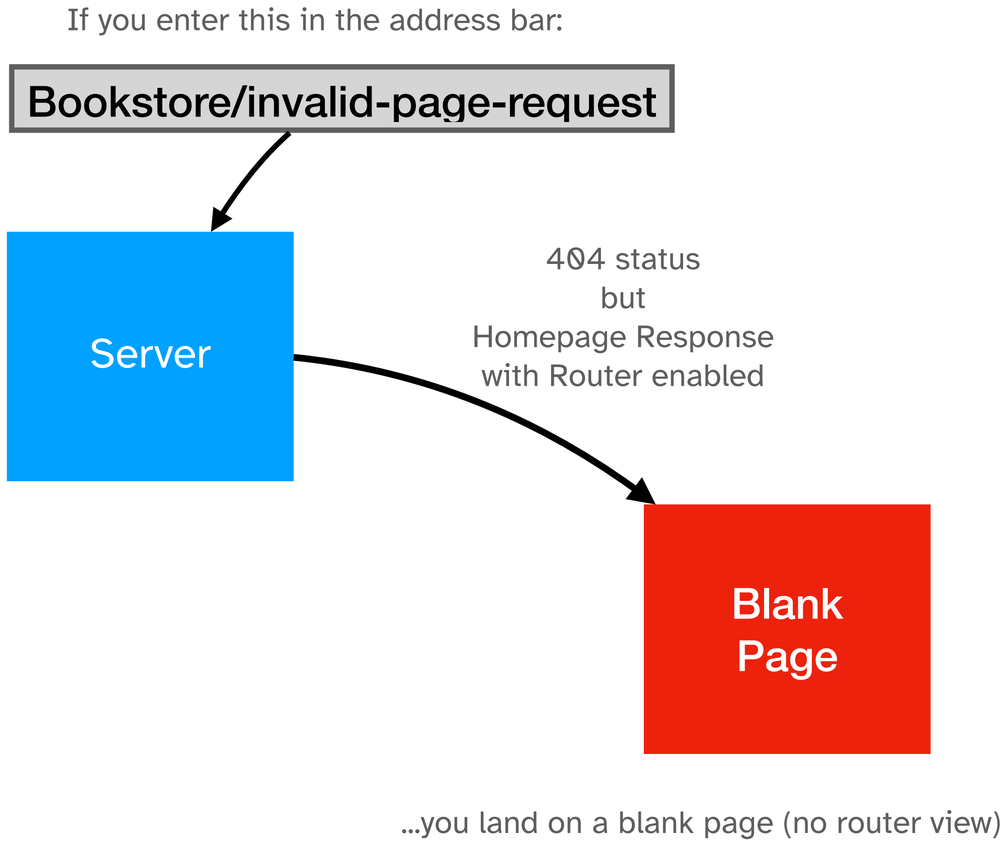
```
// In router/index.ts
{
    path: '/category',
    redirect: '/category/Classics'
},
```



Unknown Pages

Handling Unknown Pages: The Issue

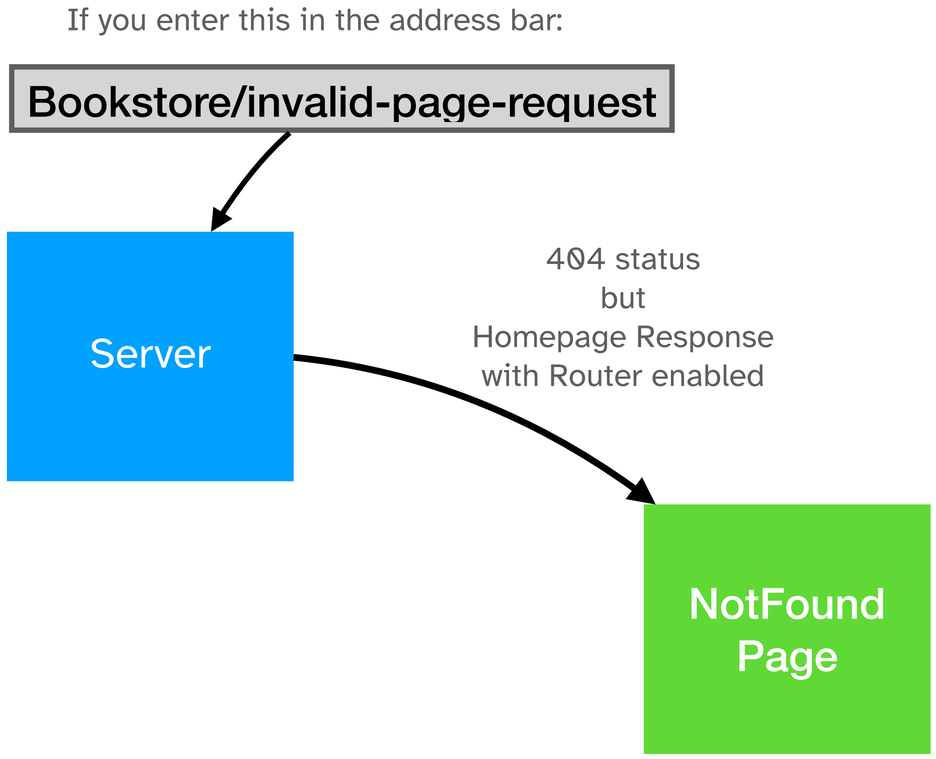
- Navigate to https://cs5244.cs.vt.edu/YourBookstore/invalid-page-request.
- You may be surprised to see a blank or empty page.
- Our page reloading solution sends us to the browser but no route matches '/invalid-page-request' so the router view does not render.
- What should we do instead?



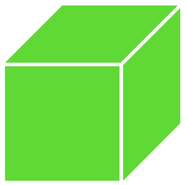

```
// In App.vue our <router-view> is empty

<template>
  <div id="app">
    <app-header></app-header>
    <router-view :key="$route.fullPath"></router-view>
    <app-footer></app-footer>
  </div>
</template>
```

Handling Unknown Pages: The Solution

- We need to detect whenever we try to navigate to a route that does not exist.
- If that happens we will present a view where we say "The page you were looking for was not found." with a home-page button.




```
// In router/index.ts below all other routes
{
  path: "/:pathMatch(.*)*",
  name: "not-found",
  component: NotFound,
},
```



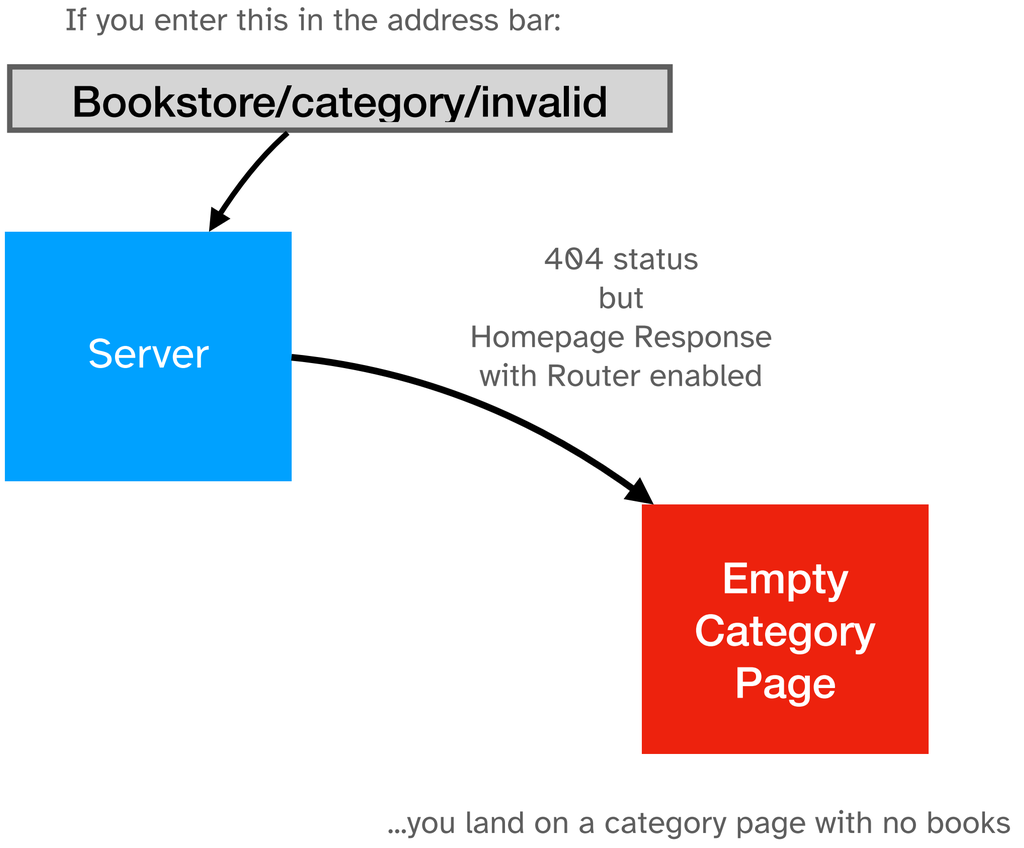
Add a new NotFound
to your views/ folder

Handling Invalid Categories

Handling Invalid Categories: The Issue

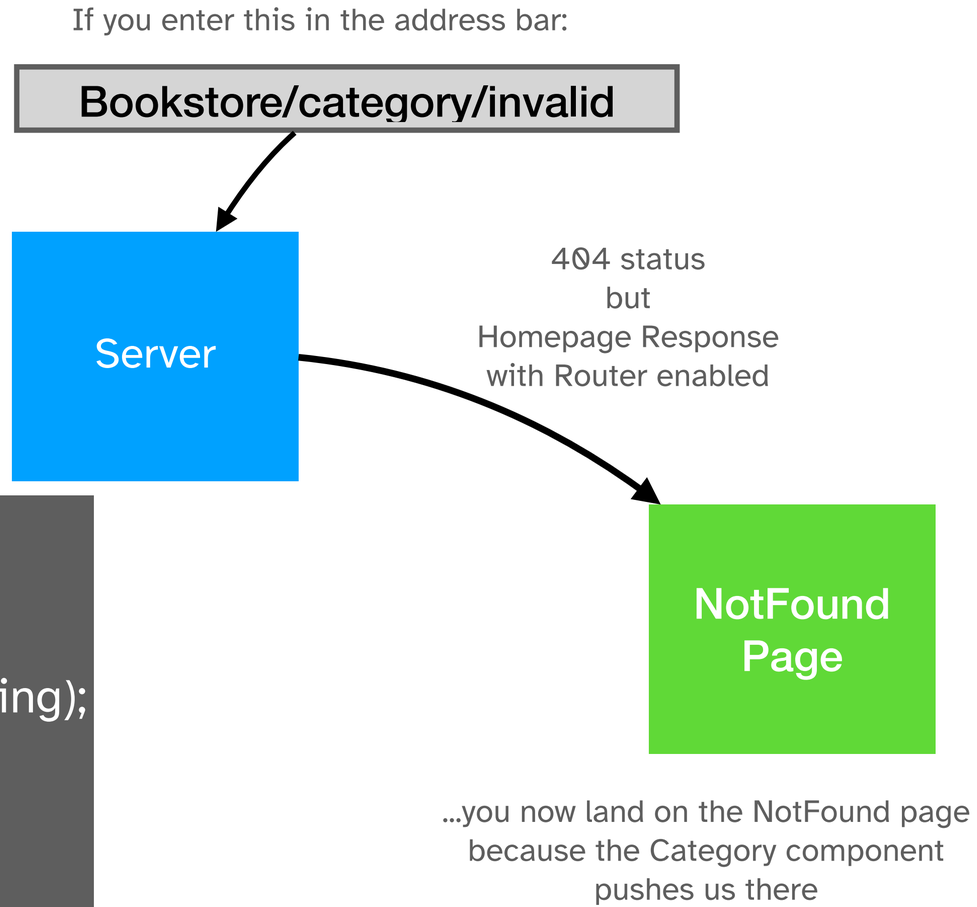
- Navigate to https://cs5244.cs.vt.edu/YourBookstore/category/invalid.
- You may be surprised to see a empty category page, or the default category page.
- Shouldn't we expect the "NotFound" experience?
- ...the category component is handling this response, and it's failing to load 'invalid' books

```
// In router/index.ts

{
  path: '/category/:name',
  name: 'category',
  component: Category,
  props: true
}
```



Handling Invalid Categories: One Solution

- We are not catching an error from the BookStore.fetchBooks function.
- The error can now flow from that call, up to the Category component
- The Category component takes us to the "NotFound" experience using the router.




```
watch(
  () => route.params.name,
  (newName) => {
    categoryStore.setSelectedCategoryName(newName as string);
    bookStore.fetchBooks(newName as string).catch(() => {
      router.push("/not-found");
    });
  },
  { immediate: true }
);
```

Test Pages against Store States

Pages vs Store States

- Here are our pages
  - Home
  - Category
  - Cart
  - Checkout
  - Confirmation
- Here are some interesting store states
  - Your cart is empty or not empty
  - You have selected a category or not
  - You have previously placed an order or not
- Make sure you test that your pages can handle those states!

Try all the combinations!

Final Testing

Final Thoughts

- Test with your production server (localhost:8080 or cs5244.cs.vt.edu)
- Let's get tough!

Toughen up!