Angular Environments

We have a problem

- Our Angular app currently has Spring Boot URL hard coded

```
export class ProductService {

  private baseUrl = 'http://localhost:8080/api/products';

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

  ...
```

- The Spring Boot backend is not even at the location anymore
  - New HTTPS location - **https://localhost:8443/api/products**

We have a problem

- Our Angular app currently has Spring Boot URL hard coded

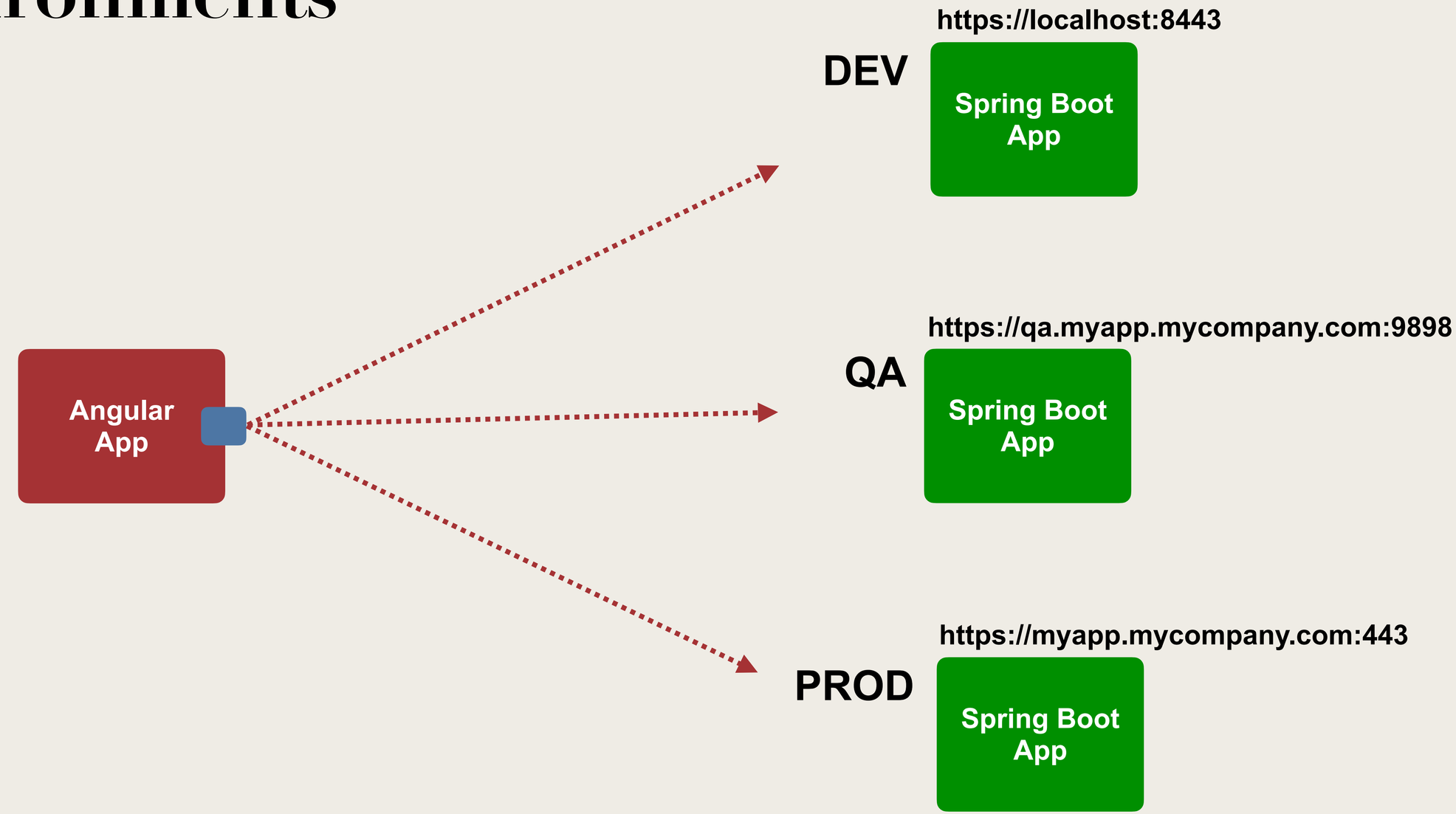
- The Spring Boot backend is not even at the location anymore
  - New HTTPS location - **https://localhost:8443/api/products**

Environments

- Instead of hard-coding URL in service class, place in a configuration
  - This will eliminate the need to change multiple references

- May need to use a different URL based on the environment
  - Allow app to easily run if deployed to different environment / server

Environments



Angular Environments

- Angular has support for **environments**
  - An environment is a named configuration for your app
  - You can add configurations for various environments

Angular Environments

- When you create a new Angular project usin
- By default, it creates the following files:
  - `src/environments`
    - `environment.ts`
    - `environment.prod.ts`

Default environment

Prod environment

Can add other environment files.
Can give any environment name.

```
environment.qa.ts
environment.betatest.ts
environment.westcoast.ts
...
```

environment.ts

```ts
export const environment = {
  production: false

  // add your custom environment configs
  // ...
};
```

environment.prod.ts

```ts
export const environment = {
  production: true

  // add your custom environment configs
  // ...
};
```

Development Process

1. Define configs in environment file
2. Use environment in your app
3. Run with environment configuration

Step 1: Define configs in environment file

environment.ts

```
export const environment = {
  production: false,

  // add your custom environment configs

  luv2shopApiUrl: "https://localhost:8443/api"

};
```

Define any name / value pairs

Step 2: Use environment in your app

Import the environment ... from Step 1

product.service.ts

```
import { environment } from 'src/environments/environment';


export class ProductService {

  private baseUrl = environment.luv2shopApiUrl + '/products';

  private categoryUrl = environment.luv2shopApiUrl + '/product-category';

  ...
}
```

Use the environment

environment.ts

```
export const environment = {
  production: false,

  // add your custom environment configs

  luv2shopApiUrl: "https://localhost:8443/api"

};
```

Step 3: Run with environment configuration

- By default, two environment configurations are defined: default and production

- Run with the default, just give command

```
$ npm start
```

Use environment.ts

- Run with production configuration

```
$ npm start -- --configuration=production
```

Use environment.prod.ts

Custom environments

- You can add custom environments and give any environment name

```
environment.qa.ts
environment.betatest.ts
environment.westcoast.ts
...
```

- Additional configuration steps are needed
- I'll cover this in upcoming videos :-)