luv2code

Refactoring Backend Configs

Development Process

1. Fix deprecated method for Spring Data REST
2. Configure CORS mapping for Spring Data REST
3. Configure CORS mapping for @RestController
4. Disable HTTP PATCH method
5. Modify Spring Data REST Detection Strategy

Step 1: Fix deprecated method for Spring Data REST

File: MyDataRestConfig.java

```
public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config) {

   ...

}
```

Before - deprecated

File: MyDataRestConfig.java

```
public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

   ...

}
```

After - current

Step 2: Configure CORS mapping for Spring Data REST

File: MyDataRestConfig.java

```
public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

  ...

  // configure cors mapping
  cors.addMapping("/api/**").allowedOrigins("http://localhost:4200");

}
```

File: application.properties

```
...
spring.data.rest.base-path=/api
```

Step 2: Configure CORS mapping for Spring Data REST

File: MyDataRestConfig.java

```
public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

  ...

  // configure cors mapping
  cors.addMapping("/api/**").allowedOrigins("http://localhost:4200");

}
```

Now we can remove the @CrossOrigin from JpaRepositories

File: StateRepository.java

```
@CrossOrigin("http://localhost:4200")
@RepositoryRestResource
public interface StateRepository extends JpaRepository<State, Integer> {

    ... ...

}
```

Step 2: Configure CORS mapping for Spring Data REST

Move configuration for "allowed origins" to application.properties

File: MyDataRestConfig.java

```
@Configuration
public class MyDataRestConfig implements RepositoryRestConfigurer {


    @Value("${allowed.origins}")
    private String[] theAllowedOrigins;


    @Override
    public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

      ...
      // configure cors mapping
      cors.addMapping("/api/**").allowedOrigins(theAllowedOrigins);

    }
}
```

1

2

File: application.properties

```
allowed.origins=http://localhost:4200
...
```

If you have multiple origins, then give a comma-delimited list

Step 2: Configure CORS mapping for Spring Data REST

File: MyDataRestConfig.java

```
@Configuration
public class MyDataRestConfig implements RepositoryRestConfigurer {

    @Value("${allowed.origins}")
    private String[] theAllowedOrigins;

    @Override
    public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

      ...
      // configure cors mapping
      cors.addMapping(config.getBasePath() + "/**").allowedOrigins(theAllowedOrigins);

    }
}
```

Use existing property:
`spring.data.rest.base-path`

Available in the `config` object

File: application.properties

```
spring.data.rest.base-path=/api
...
```

Step 3: Configure CORS mapping for @RestController

@RestController configuration is separate from Spring Data REST configuration

File: MyA

```
@Con
public class MyAppConfig implements WebMvcConfigurer {

    @Value("${spring.data.rest.base-path}")
    private String basePath;

    @Value("${allowed.origins}")
    private String[] theAllowedOrigins;

    @Override
    public void addCorsMappings(CorsRegistry cors) {

        // configure cors mapping
        cors.addMapping(basePath + "/**").allowedOrigins(theAllowedOrigins);

    }
}
```

File: application.properties

```
spring.data.rest.base-path=/api
allowed.origins=http://localhost:4200
...
```

Step 3: Configure CORS mapping for @RestController

Now we can remove the @CrossOrigin from @RestController

File: CheckoutController.java

```
@CrossOrigin("http://localhost:4200")
@RestController
@RequestMapping("/api/checkout")
public class CheckoutController {

   ...

}
```