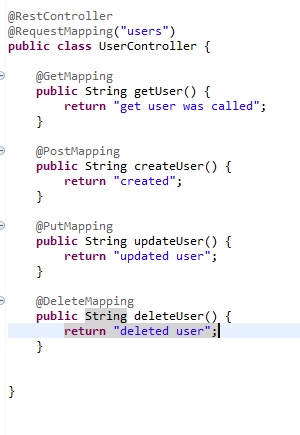
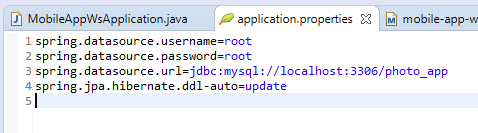
Spring Rest example

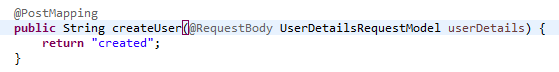


@RestController makes a class an endpoint for spring

@RequestMapping is like @Path allows to define a uri for the resource

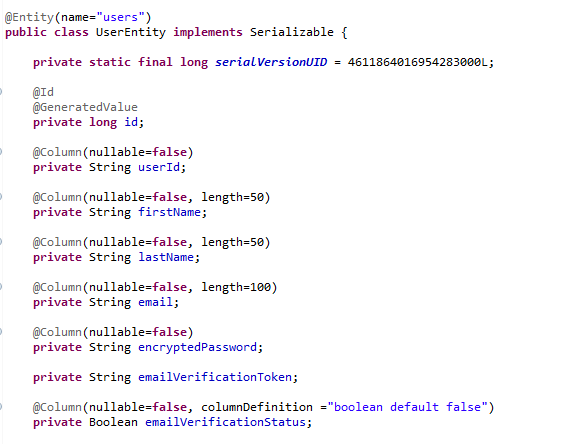


Database config with spring

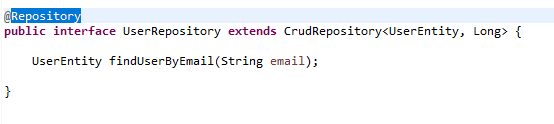


@RequestBody allows us to get objects and parse them to json (no annotations on the model class needed)





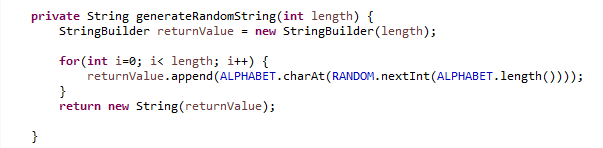
Persisting model into db



Inherits all methods to create read delete and update for a UserEntity, which has an id of type long

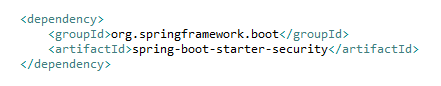
@Repository for spring to generate this bean at start up





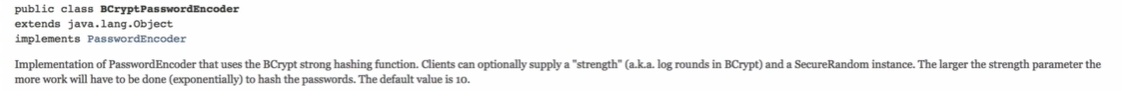
Random example

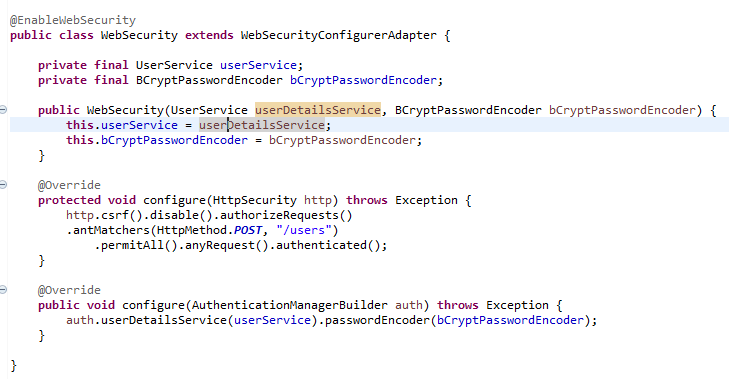
Security



Just adding the spring boot security dependency makes the ws secured, it generates an automatic token for use



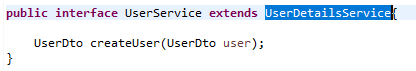


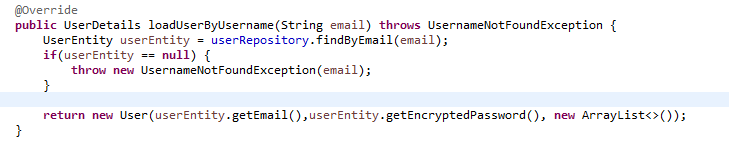


First configure is authorizing any post request to /users to pass without authentication, the rest other anyRequest needs to be authenticated

Second method configures an encoder for the passwords used in the web security

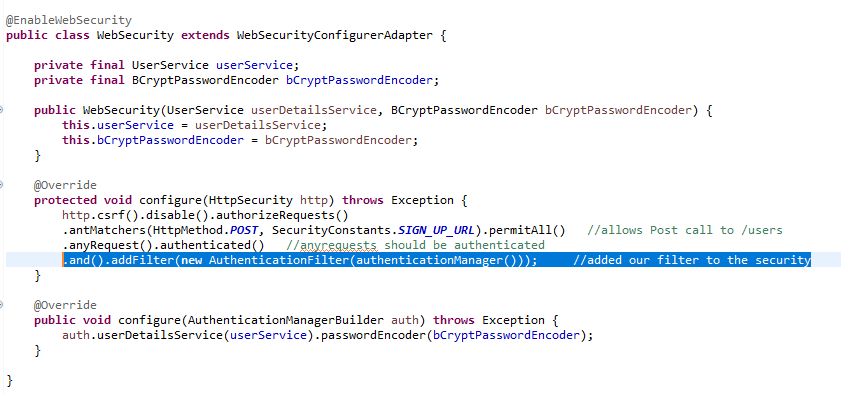
Our interface extends UserDetailsService, then this web security will apply for our code



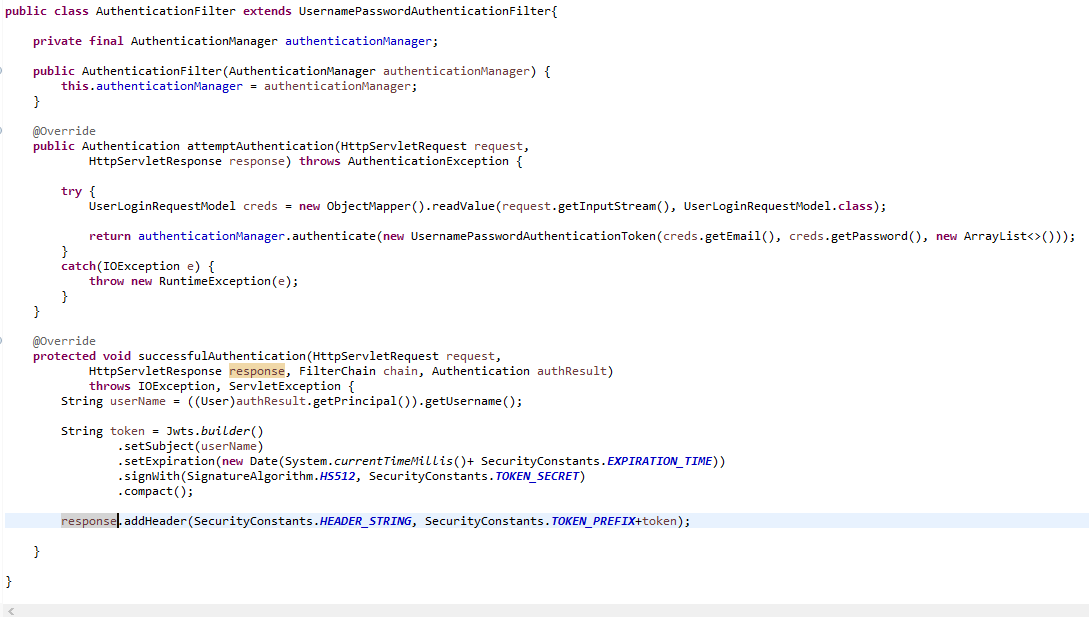


Method used by spring at

Authentication Filter used when an http request is sent for an user to sign in to the application

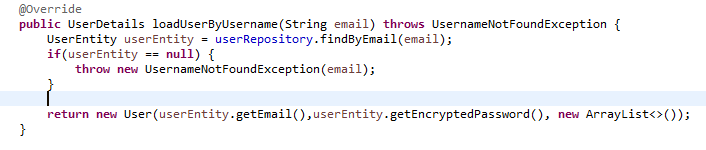


Add filter to the security of the application



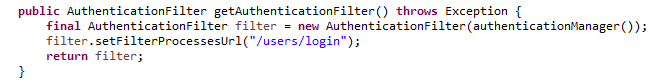
Filter

Will use the loadUserByUsername when trying to authenticate with this filter, that method we added on the service since it extends the interface UserDetailsService

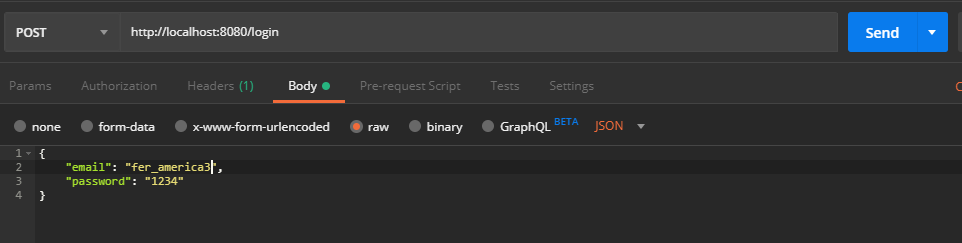


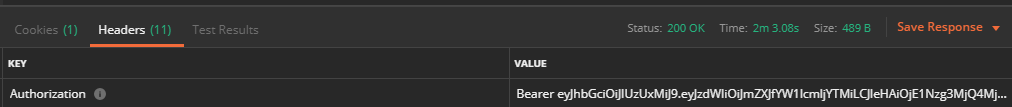


Authentication we get by default



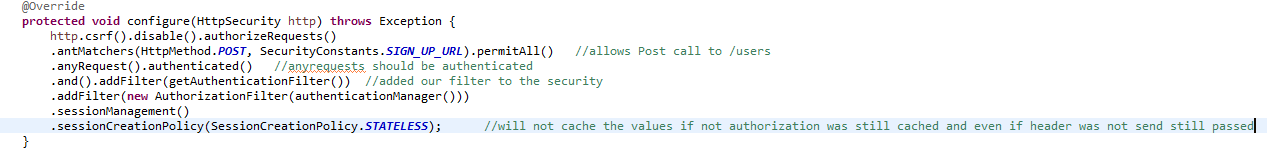
Changes default login path to out own customized path for login



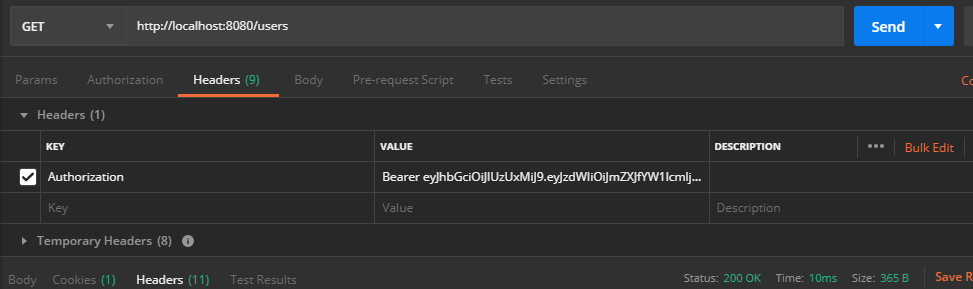


Entra al filtro en attempt authentication despues pasa por loadUserByUsername para obtener el valor de la base de datos, entonces pasa a successful authentication

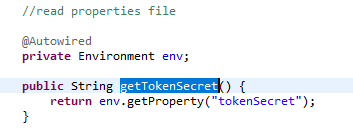
Reuse token





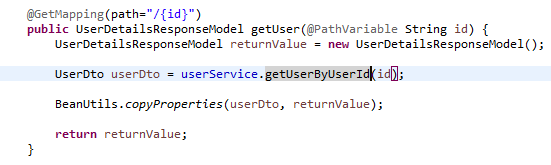


Get property from file

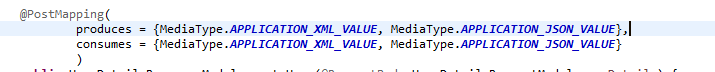


Path param

Adding path att to the get mapping and storing the value with @PathVariable



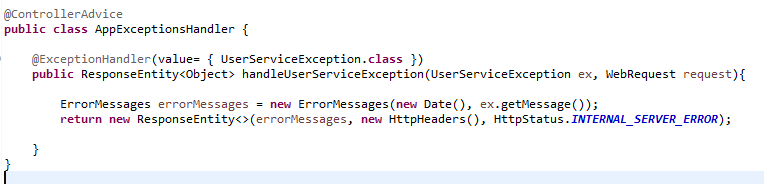
Consumes and Produces in spring



We can configure an Exception handler which will trigger when one of the configured exceptions is thrown and not handled, it allows you to do something different according to the type of exception



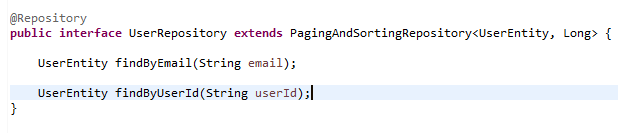




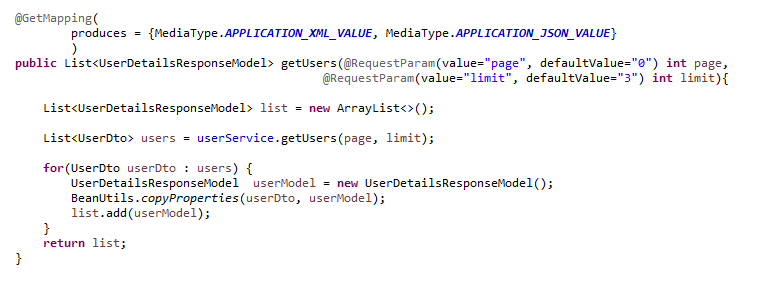
Custom error object returned

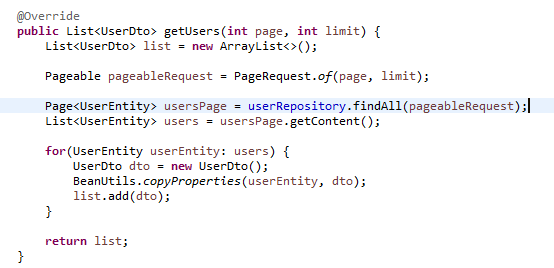


General Exception handler for all other kind of exceptions that might be thrown



Paging and sorting repository give some more functionality for paging and sorting, but keeping the use CrudRepository offered





Get paginated list

Run project outside sts

Open a command line and go into your folder at the root where pom.xml is located

Mvn install, build project and run tests

Mvn spring-boot:run



OneToMany ManyToOne

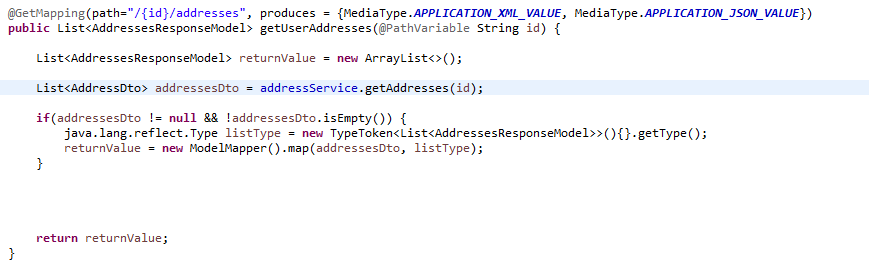
User Entity (who owns the addresses)



Address Entity



mappedBy should match the nam of the userDetails in the Many to one relationship

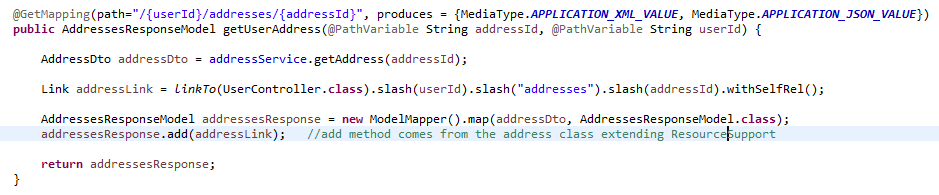


Map properties from collections

Add hateoas



**Generate link and add it to the response**

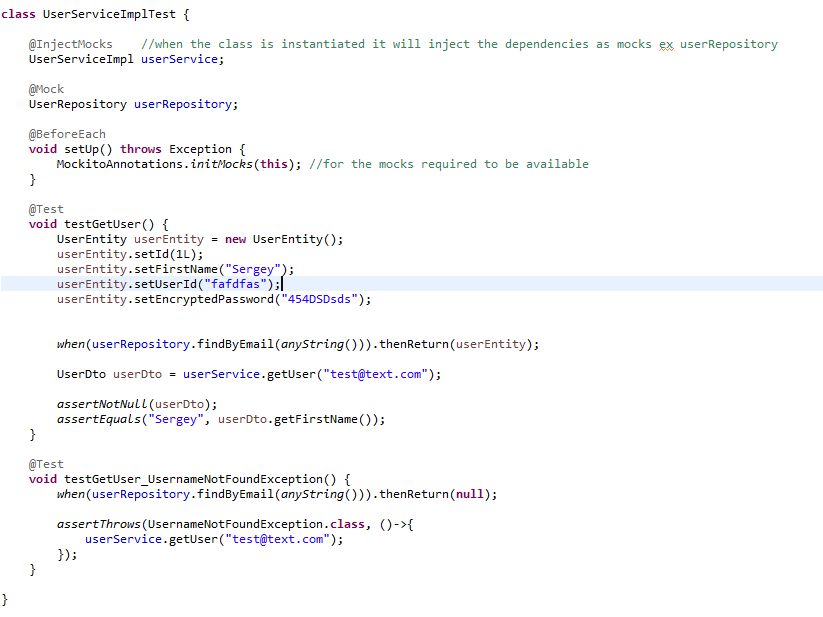




With methodOn avoid harcoding

Junit checks the code methods normally not communicating with external entities

Rest Assured tests the whole rest endpoint, sending the request validating the response





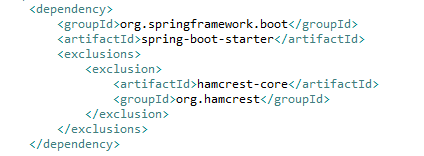
@InsertMocks class which will be used to test and which will require some mocks to be injected on it

@Mock mock object which will be created for mockito, using when allows to mock a response on the wanted method

Rest Assured

Http requests are real, request and reposnse objects are real

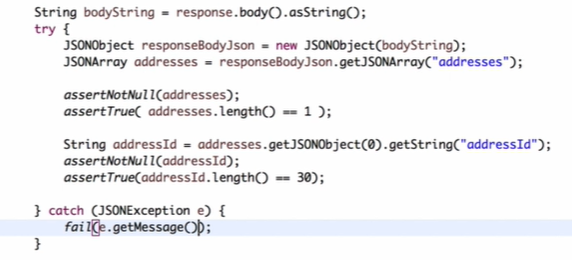
Exclude library maven



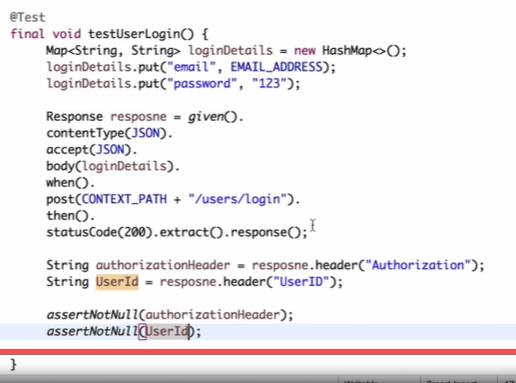
Rest assured example (end point must be up and running since it is a real http request)

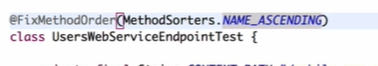


Working with the response body



Working with the headers from the response



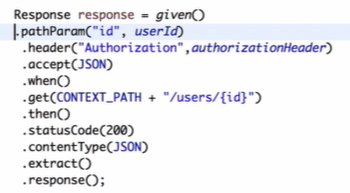


Allows to execute the tests in order, by the name of the method in the NAME\_ASCENDING case.

Use from the result var from 1 test in a next test, assigning the result value to a static var



Path param



CORS

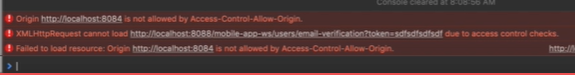
Allow cros requests like ajax on your rest service

Allow asynchronows requests

It is by default unavailable

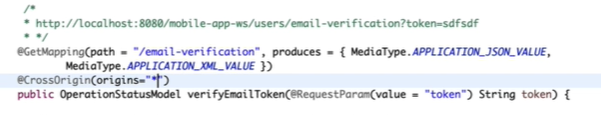
you configure for the rest service to allow those

same origin only if port, protocol and host is the same, you configure origins to be able to do cross origen request

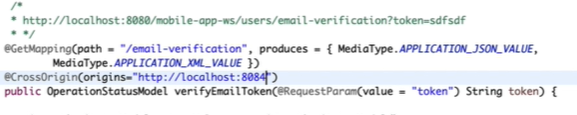


Default error when asynchronous request to a rest service is sent

All origins authorized



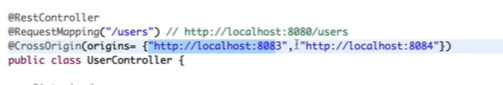
Allow specific origin



Multiple origins allowed



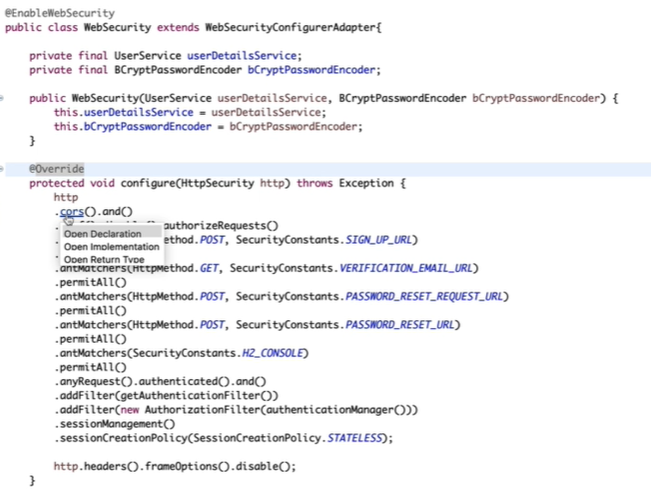
All methods allowed specifying @CrossOrigin at level class}

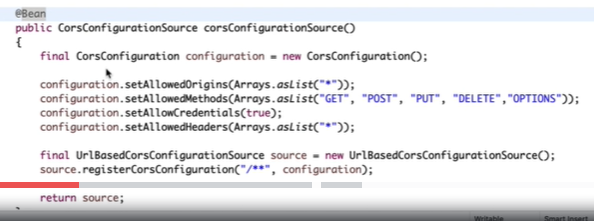


Configure to allow all rest controllers in the project



Spring Security Configuration for cors (if we use spring security in the project)





Swagger

1 Configure

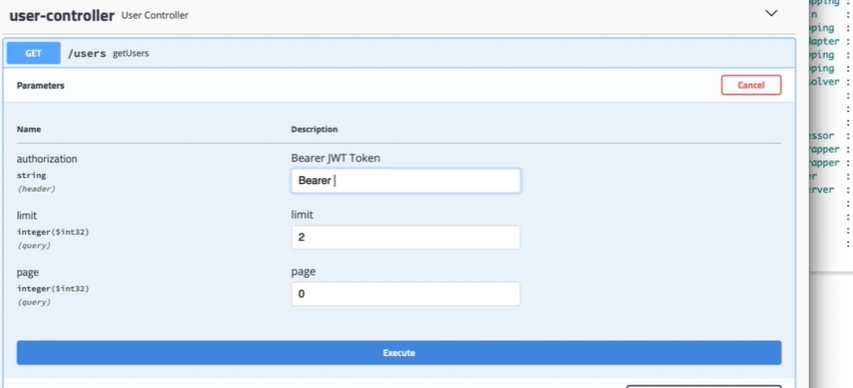


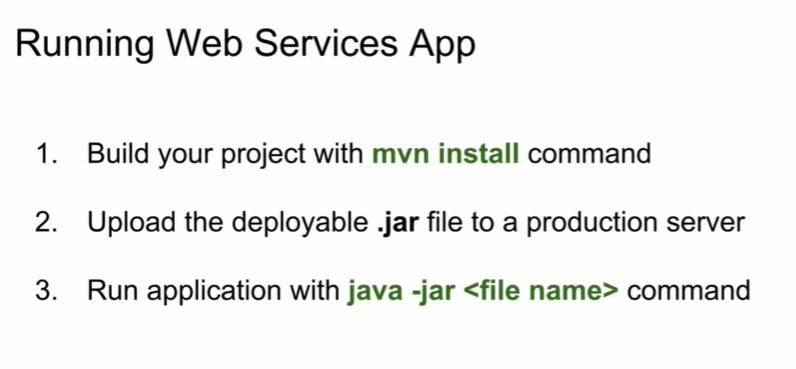
2 If Spring security enabled, configure it to allow swagger



Add headers for the swagger methods with ApiImplicitParams







Mvn clean removes target folder