**Rest**

It is representation of object based of urls.

@RestController will enable the controller to send an object back to client in converted form like json or xml. But converters like Jackson and xml are not switched on by default.

To activate them, we need to add a tag called <mvc: annotation-driven>. This tag switches on validations, message converters etc. Default message converter is httpmessageconverter.

For converting an entity into xml, we should use annotation @XmlRootElement over the entity else the converter will fail and throw 406 errors.

We can access the rest resources of the server via a rest client. Spring has inbuilt rest client (RestTemplate) but most common and widely used is Jersey.

While calling a service and asking for particular data type (content negotiation), there are two ways

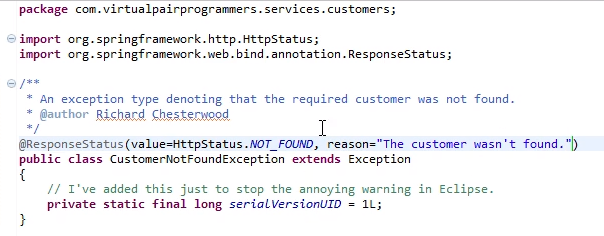
* Setting the Url extensions with <http://localhost/app/xyz.json> (not advised)
* Setting the url with <http://localhost/app?format=json> (by default off)
* Setting AcceptHeader properties in request. So if we have set the first option, the accept header way will be ignored irrespective of what readers are set.

HTTPStatus Codes – 4XX related errors are generally client error. These errors are generated because of client inputs.

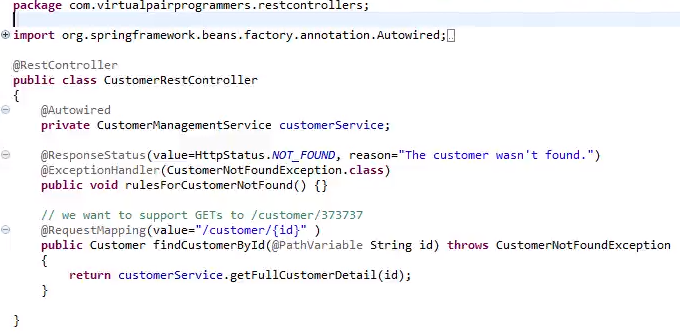
Whatever errors are thrown by database and hibernate, we should catch the exception and throw a more meaningful exception to be shown to user.

There are three approaches –

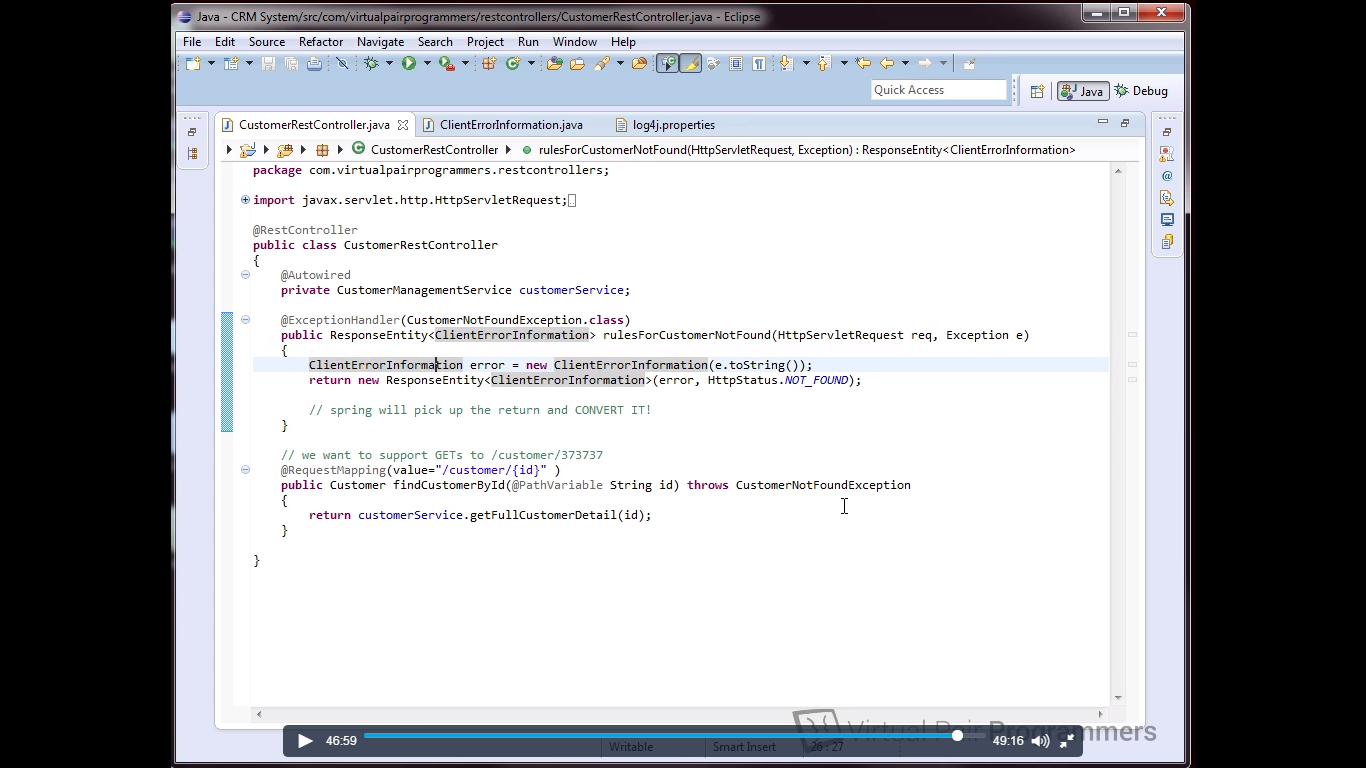
* Returning ResponseEntity from the controller with status code as argument. (not adviced)
* Annotating the custom error class with @ResponseStatus.



* The third approach is to annotate an empty function



But in all the approaches, the return error is in a usual 404 error page supplied by the server. This is html error page and it should not be like that as client cannot always be browsers. We should always send representational way of error either in xml or json but not html representation.



We can add a body to the empty exceptionHandler annotated function to return spring object ResponseEntity.

**Collections and Ranges –**

**HTTP Verbs** –

Put is idempotent so we can repeat the operation. But POST is not so we should use a POST operation for creating a customer on server. HTTP 201 status code = created. For updates, we can use PUT verb usually as update is idempotent.