**Annotations:**

* @EnableConfigServer: enables spring cloud config server.
* @SpringBootApplication : @EnableAutoConfiguration ,@ComponentScan ( enables @Component which scan all the packages where application is located),@Configuration(allow to register extra beans in the context or import additional configuration classes)
* @RestController :used to create rest controller ,it adds @ controller and @response body which return json or xml.
* @ConfigurationProperties : to read value from application .properties file.

Eg. @ConfigurationProperties(name mentioned in app.properties) and create var getter and setter for value defined in app.properties .

* @PathVariable: takes {} value from uri
* *@Service* and *@Repository* are special cases of *@Component*. They are technically the same, but we use them for the different purposes.
* @ Component: ***We can use @Component across the application to mark the beans as Spring's managed components****. Spring only pick up and registers beans with*@Component *and doesn't look for*@Service*and*@Repository*in general.*
* *@RequestBody* ***:*  @RequestBody*annotation maps the*HttpRequest*body to a transfer or domain object, enabling automatic deserialization of the inbound*HttpRequest*body onto a Java object.***
* *ResponseBody* : ***@ResponseBody* annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the *HttpResponse* object**
* *@Scope: by default, its singleton*
* ***@Transactional : to avoid code for commit rollback ,(calls entity manager) write to handle spring transaction***
* ***@synchronized:*** ***@Synchronized is a safer variant of the synchronized method modifier. Like synchronized, the annotation can be used on static and instance methods only***
* ***@GetMapping: The @GetMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).***
* ***@ConditionalOnMissingBean annotation: Tell an Auto-Configuration to Back Away When a Bean Exists***