Making API Call

```
public int getNews() {
     @Value( "${api.url}" )
      private String uri;
     @Value( "${api.key}" )
      private String key;
    ObjectMapper mapper = new ObjectMapper();
    RestTemplate restTemplate = new RestTempl-
ate();
    List<Articles> result = restTemplate.getForOb-
ject(uri + key,
   InboundNewsResponse.class).getArticles();
    return 1
}
```

Rest Template

```
@Bean
public int getRest(RestTemplateBuilder restTempl-
ateBuilder) {
        String baseUrl = <URL>
        RestTemplate restTemplate = restTemplate-
Builder
                .setConnectTimeout(ncoaConnectTim-
eout)
                .setReadTimeout(ncoaReadTimeout)
                .build();
        return new NCOAClient(baseUrl, restTempl-
ate);
    }
```

Misc.

Random Number

```
public static double getRandomNumber(){
    double x = Math.random();
    return x;
}
```

Recieving API Call

Basic Get

```
@GetMapping(path = "/enp"
protected boolean get() {
      return true;
}
```

Info From Request Body

```
(@RequestBody String blah)
```

Info From Request Header

```
(@RequestHeader ("dp-token") String dpToken)
```

Receive File

```
@RequestMapping(value = "/upload", method =
RequestMethod.POST,
 consumes = MediaType.MULTIPART_FORM_DATA_VALUE)

public String fileUpload(@RequestParam("file")
MultipartFile file) throws IOException {
      File convertFile = new File("/var/tmp/"+fil-
e.getOriginalFilename());
      convertFile.createNewFile();
      FileOutputStream fout = new FileOutputStrea-
m(convertFile);
      fout.write(file.getBytes());
      fout.close();
      return "File is upload successfully";
   }
```

General

toString

```
@Override
public String toString() {
  return "MailFwdRequest{" +
     "cfpsid='" + cfpsId +
     ", btid=" + btId +
     '}';
}
```

Read file

```
import java.io.*;
public class ReadingFromFile
{
  public static void main(String[] args) throws
Exception
  {
    // pass the path to the file as a parameter
    FileReader fr =
      new FileReader("C:\\Users\\pankaj\\Desktop\-
\test.txt");

    int i;
    while ((i=fr.read()) != -1)
      System.out.print((char) i);
  }
}
```

Run Method on Start

```
    @PostConstruct
    private void init() {
        log.info("InitDemoApplication initializ-
ation logic ...");
        // ...
    }
```

Lists, Maps, Trees

Trees

Trees Article

Maps

```
HashMap<String, Integer> map = new HashMap<>();
map.put("vishal", 10);
map.get("vishal");
```

Lists

```
List a = new ArrayList();
List b = new LinkedList();
List c = new Vector();
List d = new Stack();
List<Integer> l1 = new ArrayList<Integer>();
```

Lists, Maps, Trees (cont)

```
l1.add(0, 1); // adds 1 at 0 index
l1.add(1, 2); // adds 2 at 1 index
```

Jooq

```
private final DSLContext jooq;

@Transactional
public boolean selectExists(Long <var) {
   return jooq.fetchExists(
        select()
         .from(COA_INFORMED_DELIVERY)
         .where(<table>.<column>.eq(<var>))
      );
}
@Transactional(readOnly = true)
public List<MailFwdExtension> select(String
cfpsId) {
    List<MailFwdExtensionsRecord> records = jooq
         .selectFrom(MAIL_FWD_EXTENSIONS)
         .where(MAIL_FWD_EXTENSIONS.CFPSID.eq(cf-
psId))
         .fetchInto(MailFwdExtensionsRecord.c-
lass);
}
```

OAuth / Auth

h2

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=false
spring.h2.console.enabled=true
http://localhost:8080/h2-console
@Entity
public class Person {
    @Id
    @GeneratedValue
```

h2 (cont)

```
    private int id;
    private String name;
    private int age;
    private String emailId;
    //getters and setters
}
public interface PersonRepository extends CrudRe-
pository<Person, Integer> {}
@Service
public class PersonService {
    @Autowired
    PersonRepository personRepository;
    public List<Person> getAllPersons() {
        List<Person> persons = new ArrayList<Pe-
rson>();
        personRepository.findAll().forEach(person
-> persons.add(person));
        return persons;
    }
    public Person getPersonById(int id) {
        return personRepository.findById(id).g-
et();
    }
    public void saveOrUpdate(Person person) {
        personRepository.save(person);
    }
    public void delete(int id) {
        personRepository.deleteById(id);
    }
}
```