1.Spring Boot Exception Handling

1. Creating a EProduct entity class

```
package com.ecommerce.entity;


import java.math.BigDecimal;

import java.util.Collection;

import java.util.Date;

import java.util.List;

import java.util.Set;

import java.util.Map;



public class EProduct {
        private long ID;
        private String name;
        private BigDecimal price;
        private Date dateAdded;



        public EProduct() {


        }


        public long getID() {return this.ID; }
        public String getName() { return this.name;}
        public BigDecimal getPrice() { return this.price;}
        public Date getDateAdded() { return this.dateAdded;}
```

```
    public void setID(long id) { this.ID = id;}
    public void setName(String name) { this.name = name;}
    public void setPrice(BigDecimal price) { this.price = price;}
    public void setDateAdded(Date date) { this.dateAdded = date;}

}
```

2.Creating a ProductNotFoundException class

```
package com.ecommerce.exceptions;

public class ProductNotFoundException extends RuntimeException {
      private static final long serialVersionUID = 1L;

}
```

3.Creating a EProductExceptionController class

```
package com.ecommerce.controllers;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

import com.ecommerce.exceptions.ProductNotFoundException;

@ControllerAdvice
public class EProductExceptionController {
    @ExceptionHandler(value = ProductNotFoundException.class)
      public ResponseEntity<Object> exception(ProductNotFoundException
exception) {
        return new ResponseEntity<>("Product not found",
HttpStatus.NOT_FOUND);
      }
}
```

4.Creating MainController to throw ProductNotFoundException

```
package com.ecommerce.controllers;


import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;


import com.ecommerce.entity.EProduct;
import com.ecommerce.exceptions.ProductNotFoundException;


@Controller
public class MainController {
        @RequestMapping(value = "/product/{id}", method =
RequestMethod.GET)
         @ResponseBody
          public String getProduct(@PathVariable("id") String id) {

                      if (id.contentEquals("0"))
                             throw new ProductNotFoundException();
             return "Product was found";
          }
}
```

2.Consuming RESTful Web Services

1.Creating a class Quote to work with the public REST service

```
package com.ecommerce.beans;
import com.fasterxml.jackson.annotation.*;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)
public class Quote {

    private String type;
    private Value value;

    public Quote() {
    }
    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public Value getValue() {
        return value;
    }

    public void setValue(Value value) {
        this.value = value;
    }

    @Override
    public String toString() {
        return "Quote{" +
                "type='" + type + '\'' +
                ", value=" + value +
                '}';
    }
}
```

2.Creating a class Value to act as a wrapper for the REST data

```
package com.ecommerce.beans;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)
public class Value {

    private Long id;
    private String quote;

    public Value() {
    }

    public Long getId() {
        return this.id;
    }

    public String getQuote() {
        return this.quote;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public void setQuote(String quote) {
        this.quote = quote;
    }

    @Override
    public String toString() {
        return "Value{" +
                "id=" + id +
                ", quote='" + quote + '\'' +
                '}';
    }
}
```

3.Creating MainController to consume the REST service

```
package com.ecommerce.controllers;


import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.client.RestTemplate;


import com.ecommerce.beans.Quote;


@Controller
public class MainController {


        @RequestMapping("/")
         @ResponseBody
         public String index() {


               RestTemplate restTemplate = new RestTemplate();
            Quote quote = restTemplate.getForObject("https://gturnquist-
quoters.cfapps.io/api/random", Quote.class);


           return quote.toString();
         }
}
```

3.File Handling

1.Creating an HTML file that will show a form of uploading a file

```
<html>

<head><title>File Upload</title></head>

<body>


        <form  method="post" enctype="multipart/form-data"
action="/upload">

         Upload file&nbsp;

           <input type="file" name="fileToUpload" id="fileToUpload"><br><br>

    <input type="submit" value="Upload " name="submit">

        </form>

</body>

</html>
```

2.Creating MainController for handling file upload and download

```
package com.ecommerce.controllers;


import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.InputStreamResource;
import org.springframework.core.io.Resource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.util.ResourceUtils;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;


@Controller
public class MainController {

    @RequestMapping(value = "/")
  public String index() {
    return "index.html";
  }

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

```
        public String fileUpload(@RequestParam("file") MultipartFile file) {
            String result = "File was uploaded successfully";


            try {
             File convertFile = new File("/var/tmp/"+file.getOriginalFilename());
           convertFile.createNewFile();
           FileOutputStream fout = new FileOutputStream(convertFile);
           fout.write(file.getBytes());
           fout.close();


            } catch (IOException iex) {
                result = "Error " + iex.getMessage();


            } finally {
                return result;
            }
        }


        @RequestMapping(value = "/download", method =
    RequestMethod.GET)
        public ResponseEntity<Object> downloadFile() throws IOException  {
            String fileName = "static/dump.txt";
        ClassLoader classLoader = new
    MainController().getClass().getClassLoader();


        File file = new File(classLoader.getResource(fileName).getFile());
        InputStreamResource resource = new InputStreamResource(new
FileInputStream(file));
        HttpHeaders headers = new HttpHeaders();

        headers.add("Content-Disposition", String.format("attachment;
filename=\"%s\"", file.getName()));
        headers.add("Cache-Control", "no-cache, no-store, must-revalidate");
        headers.add("Pragma", "no-cache");
        headers.add("Expires", "0");

        ResponseEntity<Object>
        responseEntity =
ResponseEntity.ok().headers(headers).contentLength(file.length()).contentTy
pe(
          MediaType.parseMediaType("application/txt")).body(resource);

        return responseEntity;
    }
}
```

4.HTTPS for Spring Boot

1.Creating MainController for showing a page in the browser under SSL

```
package com.ecommerce.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;




@Controller
public class MainController {

    @Autowired
    private ProductRepository repository;

    @RequestMapping("/")
     @ResponseBody
     public String index() {



       return “This is running under SSL”;
     }
```

2.Configuring application.properties to run the site in SSL

```
server.port=8443
server.ssl.key-alias=selfsigned_localhost_sslserver
server.ssl.key-password=changeit
server.ssl.key-store=classpath:ssl-server.jks
server.ssl.key-store-provider=SUN

server.ssl.key-store-type=JKS
```