1 Spring Boot Exception Handling

EProduct.java

```
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;}

}
```

ProductNotFoundException.java

```
package com.ecommerce.exceptions;

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

}
```

EProductExceptionController.java

```
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);
      }
}
```

MainController.java

```
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

Quote.java

```
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 +
                '}';
    }
}
```

Value.java

```
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 + '\'' +
                '}';
    }
}
```

MainController.java

```
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://type.fit/api/quotes", Quote.class);

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

3 Edge Server and Routing

SpringEdgeApplication.java

```
package com.ecommerce;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
public class SpringEdgeApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringEdgeApplication.class, args);
	}
}
```

4 File Handling

Index.html

```
<html>
<head><title>File Upload</title></head>
<body>

      <form  method="post" enctype="multipart/form-data" action="/upload">
       Upload file 
        <input type="file" name="fileToUpload" id="fileToUpload"><br><br>
   <input type="submit" value="Upload " name="submit">
      </form>
</body>
</html>
```

MainController.java

```
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()).contentType(
        MediaType.parseMediaType("application/txt")).body(resource);

      return responseEntity;
    }
}
```

5 Sending and Receiving Messages with Apache Kafka

KafkaProducerConfig.java

```
package com.ecommerce;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.common.serialization.StringSerializer;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.core.DefaultKafkaProducerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.core.ProducerFactory;

@Configuration
public class KafkaProducerConfig {
  @Bean
  public ProducerFactory<String, String> producerFactory() {
    Map<String, Object> configProps = new HashMap<>();
    configProps.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
    configProps.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
    configProps.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
    return new DefaultKafkaProducerFactory<>(configProps);
  }
  @Bean
  public KafkaTemplate<String, String> kafkaTemplate() {
    return new KafkaTemplate<>(producerFactory());
  }
}
```

KafkaConsumerConfig.java

```
package com.ecommerce;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.common.serialization.StringDeserializer;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.annotation.EnableKafka;
import org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;
import org.springframework.kafka.core.ConsumerFactory;
import org.springframework.kafka.core.DefaultKafkaConsumerFactory;

@EnableKafka
@Configuration
public class KafkaConsumerConfig {
  @Bean
  public ConsumerFactory<String, String> consumerFactory() {
    Map<String, Object> props = new HashMap<>();
    props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:2181");
    props.put(ConsumerConfig.GROUP_ID_CONFIG, "group-id");
    props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
    props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
    return new DefaultKafkaConsumerFactory<>(props);
  }
  @Bean
  public ConcurrentKafkaListenerContainerFactory<String, String> kafkaListenerContainerFactory() {
    ConcurrentKafkaListenerContainerFactory<String, String>
    factory = new ConcurrentKafkaListenerContainerFactory<>();
```

```
    factory.setConsumerFactory(consumerFactory());
    return factory;
  }
}
```

MainController.java

```
package com.commerce.controllers;

import java.util.Calendar;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.core.DefaultKafkaProducerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.core.ProducerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class MainController {

      @Autowired
        private KafkaTemplate<String, String> kafkaTemplate;

      @RequestMapping(value = "/")
   public String index() {
      this.sendMessage("This is a message sent at " + Calendar.getInstance().getTime());
      return "Check Eclipse console for kafka output";
   }
        private void sendMessage(String msg) {
                kafkaTemplate.send("ecommerce", msg);
      }

}
```

SpringBootKafkaApplication.java

```
package com.ecommerce;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.kafka.core.KafkaTemplate;

@SpringBootApplication
public class SpringBootKafkaApplication {

	@Autowired
	  private KafkaTemplate<String, String> kafkaTemplate;

	public static void main(String[] args) {
		SpringApplication.run(SpringBootKafkaApplication.class, args);

	}
	  @KafkaListener(topics = "ecommerce", groupId = "group-id")
	  public void listen(String message) {
	    System.out.println("Received Message in group - group-id: " + message);
	  }
}
```

6 HTTPS for Spring Boot

MainController.java

```
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”;
         }
}
```