```
 1 package org.example;
 2
 3 import org.apache.catalina.Context;
 4 import org.apache.catalina.LifecycleException;
 5 import org.apache.catalina.Wrapper;
 6 import org.apache.catalina.startup.Tomcat;
 7 import org.example.web.MyInvoiceServlet;
 8
 9 public class OldAppLauncher {
10
11     static final int defPort = 8080;
12
13     public static void main(String[] args) throws
   LifecycleException {
14         /** use this with java -Dport80xx -jar
   target/xy
15          if Port not provided as Parameter -> use
   default Port */
16
17         int port;
18         String sPort = (System.getProperty("port"
   ));
19
20         if ((sPort == null) || (sPort.isEmpty())) {
21             port = defPort;
22         } else {
23             port = Integer.parseInt(System.
   getProperty("port"));
24         }
25         Tomcat tomcat = new Tomcat();
26
27         tomcat.setPort(port);
28         tomcat.getConnector();
29
30         Context ctx = tomcat.addContext("", null);
31         Wrapper servlet = Tomcat.addServlet(ctx, "
   org.example.web.MyInvoiceServlet", new
   MyInvoiceServlet());
32         servlet.setLoadOnStartup(1);
33         servlet.addMapping("/*");
34
35         tomcat.start();
36         System.out.println("Tomcat started.....");
37         System.out.println("Listen at Port: " +
```

```
port);
    }
}
```

```
 1 package org.example;
 2
 3 import org.apache.catalina.Context;
 4 import org.apache.catalina.LifecycleException;
 5 import org.apache.catalina.Wrapper;
 6 import org.apache.catalina.startup.Tomcat;
 7 import org.example.context.ApplicationConfiguration
   ;
 8 import org.springframework.web.context.
   WebApplicationContext;
 9 import org.springframework.web.context.support.
   AnnotationConfigWebApplicationContext;
10 import org.springframework.web.servlet.
   DispatcherServlet;
11
12 import javax.servlet.ServletContext;
13
14 public class ApplicationLauncher {
15
16     static final int defPort = 8080;
17
18     public static void main(String[] args) throws
   LifecycleException {
19         /* use this with java -Dport80xx -jar
   target/xy
20          if Port not provided as Parameter -> use
   default Port */
21
22         int serverPort = 8080;
23         String portProperty = (System.getProperty("
   server.port"));
24         if (portProperty != null) {
25             serverPort = Integer.parseInt(
   portProperty);
26         }
27
28         Tomcat tomcat = new Tomcat();
29
30         tomcat.setPort(serverPort);
31         tomcat.getConnector();
32
33         Context tomcatCtx = tomcat.addContext("",
   null);
34         // Dispatcher Servlet
```

```
        WebApplicationContext appCtx = createApplicationContext(tomcatCtx.getServletContext());
        DispatcherServlet dispatcherServlet = new DispatcherServlet(appCtx);

        // Wrapper servlet = Tomcat.addServlet(ctx, "org.example.web.MyInvoiceServlet", new MyInvoiceServlet());
        Wrapper servlet = Tomcat.addServlet(tomcatCtx, "dispatcherServlet", dispatcherServlet);
        servlet.setLoadOnStartup(1);
        servlet.addMapping("/*");

        tomcat.start();
        System.out.println("Tomcat started.....");
        System.out.println("Listen at Port: " + serverPort);
    }

    // new - added for Dispatcher Servlet
    public static WebApplicationContext createApplicationContext(ServletContext servletContext) {
        AnnotationConfigWebApplicationContext ctx = new AnnotationConfigWebApplicationContext();
        ctx.register(ApplicationConfiguration.class);
        ctx.setServletContext(servletContext);
        ctx.refresh();
        ctx.registerShutdownHook();
        return ctx;
    }
}
```

```java
package org.example.dto;

import com.fasterxml.jackson.annotation.JsonProperty;

public class InvoiceDTO
{
    @JsonProperty("user_id")
    // @NotBlank -
    private String userId;

    // @Min(10)
    // @Max(100)
    private Integer amount;

    @JsonProperty("my_msg")
    private String myHello;

    public String getUserId() {
        return userId;
    }
    public void setUserId(String userId) {
        this.userId = userId;
    }

    public Integer getAmount() {
        return amount;
    }
    public void setAmount(Integer amount){
        this.amount = amount;
    }

    public String getMyHello() { return myHello; }
    public void setMyHello(String myHello){
        this.myHello = myHello;
    }
}

```

```
 1 package org.example.web;
 2
 3 import com.fasterxml.jackson.databind.ObjectMapper;
 4 import org.example.context.ApplicationConfiguration
   ;
 5 import org.example.model.Invoice;
 6 import org.example.service.InvoiceService;
 7 import org.example.service.UserService;
 8 import org.springframework.context.annotation.
   AnnotationConfigApplicationContext;
 9
10 import javax.servlet.ServletException;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14 import java.io.IOException;
15 import java.util.List;
16
17 public class MyInvoiceServlet extends HttpServlet {
18
19     private UserService userService;
20     private ObjectMapper objectMapper;
21     private InvoiceService invoiceService;
22
23     @Override
24     public void init() throws ServletException{
25         AnnotationConfigApplicationContext ctx
26                 = new
   AnnotationConfigApplicationContext(
   ApplicationConfiguration.class);
27
28         // is only needed that @PreDestroy works
   properly with IDE
29         ctx.registerShutdownHook();
30
31         this.userService = ctx.getBean(UserService.
   class);
32         this.objectMapper = ctx.getBean(
   ObjectMapper.class);
33         this.invoiceService = ctx.getBean(
   InvoiceService.class);
34
35         System.out.println("im servlet - init");
36     }
```

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException{
        if (request.getRequestURI().equalsIgnoreCase("/invoices")) {
            String userId = request.getParameter("user_id");
            Integer amount = Integer.valueOf(request.getParameter("amount"));
            String helloMsg = "Hello, My Dear";

            Invoice invoice = invoiceService.create(userId, amount, helloMsg);

            response.setContentType("application/json; charset=UTF-8");
            String json = new ObjectMapper().writeValueAsString(invoice);
            response.getWriter().print(json);
        }
        else {
            response.setStatus(HttpServletResponse.SC_NOT_FOUND);
        }
    }
    @Override
    protected void doGet(HttpServletRequest request,HttpServletResponse response) throws IOException {
        if (request.getRequestURI().equalsIgnoreCase("/")) {
            response.setContentType("text/html; charset=UTF-8");
            response.getWriter().print(
                    "<html>\n" +
                            "<body>\n" +
                            "<h1>Hello Barbara</h1>\n" +
                            "<p>You are working on your first Spring App</p>\n" +
                            "</body>\n" +
                            "</html>\n");
        }
        else if (request.getRequestURI().
```

```
equalsIgnoreCase("/invoices"))
        {
            response.setContentType("application/
json; charset=UTF-8");
            List<Invoice> invoices =
invoiceService.findAll();

           System.out.println(objectMapper.
writeValueAsString(invoices));
            // (2)
            response.getWriter().print(
objectMapper.writeValueAsString(invoices));
            // (3)
        }
    }
}
```

```
 1 package org.example.web;
 2
 3 import org.example.model.Invoice;
 4 import org.example.service.InvoiceService;
 5 import org.springframework.validation.annotation.
   Validated;
 6 import org.springframework.web.bind.annotation.*;
 7
 8 import javax.validation.constraints.NotBlank;
 9 import javax.validation.constraints.Max;
10 import javax.validation.constraints.Min;
11 import java.util.List;
12
13 @RestController
14 @Validated
15 public class InvoiceController {
16
17     private final InvoiceService invoiceService;
18
19     public InvoiceController(InvoiceService
   invoiceService) {
20         this.invoiceService = invoiceService;
21     }
22     // GET Method
23     @RequestMapping(value="/invoices", method =
   RequestMethod.GET)
24     public List<Invoice> invoices() {
25         return invoiceService.findAll();
26     }
27
28     // POST Method
29     /* Request Mapping Version
30    @RequestMapping(value="/invoices", method =
   RequestMethod.POST)
31     public Invoice createInvoice(@RequestParam("
   user_id") String userId,
32                                  @RequestParam
   Integer amount,
33                                  @RequestParam
   String myMsg) {
34         return invoiceService.create(userId,amount,
   myMsg);
35     }
36      */
```

```
    /*
    @PostMapping("/invoices/{userId}/{amount}/{myMsg}")
    public Invoice createInvoice(@PathVariable String userId,
                                 @PathVariable Integer amount,
                                 @PathVariable String myMsg) {
        return invoiceService.create(userId,amount, myMsg);
    }
     */
    // Ergebnisse aus dem RequestBody werden als Parameter übergeben
    /*
    @PostMapping("invoices")
    public Invoice createInvoice(@RequestBody @Valid InvoiceDTO invoiceDTO) {
        return invoiceService.create(invoiceDTO.getUserId(),
                invoiceDTO.getAmount(), invoiceDTO.getMyHello());
    }
    */
    // Mit validation Bean in ApplicationLauncher Class
    @PostMapping("invoices")
    public Invoice createInvoice(@RequestParam("user_id") @NotBlank String userId,
                                 @RequestParam @Min(10) @Max(100) Integer amount,
                                 @RequestParam ("my_msg") String myMsg) {
        return invoiceService.create(userId, amount ,myMsg);
    }
}

```

```
 1 package org.example.web;
 2
 3 import org.springframework.http.HttpStatus;
 4 import org.springframework.web.bind.
   MethodArgumentNotValidException;
 5 import org.springframework.web.bind.annotation.
   ExceptionHandler;
 6 import org.springframework.web.bind.annotation.
   ResponseStatus;
 7 import org.springframework.web.bind.annotation.
   RestControllerAdvice;
 8
 9 import javax.validation.
   ConstraintViolationException;
10
11 // apply this class to all classes annotated with @
   Controllers or @RestControllers
12 // Information will be written directly to @
   ResponseBody in Json/XML
13 // alternative: @ControllerAdvice that will write
   HTML
14
15 @RestControllerAdvice
16 public class GlobalExceptionHandler {
17
18     @ResponseStatus(HttpStatus.BAD_REQUEST)
19     @ExceptionHandler(
   MethodArgumentNotValidException.class)
20     public String handlemethodArgumentNotValid(
   MethodArgumentNotValidException exception) {
21         //TODO choose to return custome object here
    - transformed to json/xml
22         return "Sorry, Bad Requests " + exception.
   getMessage();
23     }
24
25     @ResponseStatus(HttpStatus.BAD_REQUEST)
26     @ExceptionHandler(ConstraintViolationException.
   class)
27     public String handleConstraintViolation(
   ConstraintViolationException exception) {
28         //TODO choose to return custome object here
    - transformed to json/xml
29         return "Sorry, please enter correct values
```

```
29  " + exception.getMessage();
30     }
31 }
32
```

```java
package org.example.model;

public class User {

    private String id;
    private String name;

    public User(){
    }

    public User(String id, String name) {
        this.id = id;
        this.name =name;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}

```

```
 1 package org.example.model;
 2
 3 import com.fasterxml.jackson.annotation.
   JsonProperty;
 4 import java.util.UUID;
 5
 6 public class Invoice {
 7     private String id;
 8
 9     @JsonProperty("user_id")
10     private String userId;
11
12     @JsonProperty("pdf_url")
13     private    String pdfUrl;
14
15     private Integer amount;
16     // ist nur als Test hinzugefügt worden
17     @JsonProperty("my_hello")
18     private String myHello;
19
20     public Invoice(){
21     }
22
23     public Invoice(String userId, Integer amount,
   String pdfUrl,String myHello) {
24         this.id = UUID.randomUUID().toString();
25         this.userId = userId;
26         this.pdfUrl = pdfUrl;
27         this.amount = amount;
28         this.myHello = myHello;
29     }
30
31     public String getId() {
32         return id;
33     }
34     public void setId(String id) {
35         this.id = id;
36     }
37
38     public String getUserId() {
39         return userId;
40     }
41     public void setUserId(String userId) {
42         this.userId = userId;
```

```
43      }
44
45      public String getPdfUrl(){
46          return pdfUrl;
47      }
48      public void setPdfUrl(String pdfUrl){
49          this.pdfUrl = pdfUrl;
50      }
51
52      public Integer getAmount() {
53          return amount;
54      }
55      public void setAmount(Integer amount){
56          this.amount = amount;
57      }
58
59      public String getMyHello() { return myHello; }
60      public void setMyHello(String myHello){
61          this.myHello = myHello;
62      }
63  }
64
```

```java
package org.example.context;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.example.ApplicationLauncher;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.validation.beanvalidation.MethodValidationPostProcessor;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration
@ComponentScan(basePackageClasses = ApplicationLauncher.class)
// Injection of Property-Fields in a property file
// Reihenfolge ist wichtig (overwrite) & falls File nicht vorhanden ignore
@PropertySource("classpath:/application.properties")
@PropertySource(value="classpath:/application-${spring.profiles.active}.properties"
    , ignoreResourceNotFound = true)
@EnableWebMvc
public class ApplicationConfiguration {

    // für Validation @ReuestParam -> MyInvoiceDTO
    @Bean
    public MethodValidationPostProcessor methodValidationPostProcessor() {
        return new MethodValidationPostProcessor();
    }
    @Bean
    public ObjectMapper objectMapper(){
        return new ObjectMapper();
    }
}

```

```java
package org.example.service;

import org.example.model.User;
import org.springframework.stereotype.Component;

import java.util.UUID;

@Component
public class UserService {

    public User findById(String id) {
        String randomName = UUID.randomUUID().toString();
        // always "finds" the user, every user has a random name
        return new User(id,randomName);
    }
}

```

```java
package org.example.service;

import org.example.model.Invoice;
import org.example.model.User;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.List;
import java.util.concurrent.CopyOnWriteArrayList;

@Component
public class InvoiceService {

    private final UserService userService;
    private final List<Invoice> invoices = new CopyOnWriteArrayList<>();
    private final String cdnUrl;
    private final String myMsg;

    public InvoiceService(UserService userService, @Value("${cdn.url}") String cdnUrl, @Value("${my.hello}") String myMsg) {
        this.userService = userService;
        this.cdnUrl = cdnUrl;
        this.myMsg = myMsg;
        System.out.println(this.myMsg);
    }
    @PostConstruct
    public void init() {
        System.out.println("Fetch PDF Template from S3...");
        // TODO download from s3 and save locally
    }
    @PreDestroy
    // works only when application is really shutdown
    public void shutdown(){
        System.out.println("Deleting downloaded template");
    }

```

```
    // -------- Start of the methods --------
    public List<Invoice> findAll() {
        // Debugging : System.out.println("in findAll");
        return invoices;
    }
    public Invoice create(String userId, Integer amount, String myMsg){
        User user = userService.findById(userId);
        if (user == null) {
            throw new IllegalStateException();
        }
    //  TODO real pdf creation and storing it on network server
        Invoice invoice = new Invoice(userId,amount, cdnUrl+"/images/default/sample.pdf", myMsg);

        invoices.add(invoice);
        return invoice;
    }
}
```

```java
package org.example.service;

import org.springframework.context.annotation.Profile;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;

@Service
@Profile("dev")

// This needs to run with following command: java -Dspring.profiles.active=dev -jar target/MyI*
public class DummyInvoiceServiceLoader {
    private final InvoiceService invoiceService;

    public DummyInvoiceServiceLoader(InvoiceService invoiceService) {
        this.invoiceService = invoiceService;
    }

    @PostConstruct
    public void setup() {
        System.out.println("Creating dev Invoices ...");
        invoiceService.create("Sibylle", 50, "Hello, my Dear");
        invoiceService.create("Barbara", 20, "Hello, my Dear");
    }
}

```