SPRING FRAMEWORK
REST CONTROLLER EXAMPLE

EW-008.STUD-ETF-IP.032_AeroflotUsersWorkerSA / src / main / java / yatospace / worker / services / rest / **ServicesPoint.java** / <> Jump to

```java
package yatospace.worker.services.rest;

import java.io.File;
import java.net.URLEncoder;
import java.nio.file.Files;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
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.RequestPart;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
```

```
25  import yatospace.email.data.controller.UserEmailController;
26  import yatospace.email.data.object.EmailDTO;
27  import yatospace.ip_messaging.controller.IPMessagingController;
28  import yatospace.ip_messaging.object.IPMessageDTO;
29  import yatospace.ip_messaging.util.UserMessageFormatEnviroment;
30  import yatospace.worker.services.bean.ExampleInfoBean;
31  import yatospace.worker.services.center.ControllerCenter;
32  import yatospace.worker.services.controller.FlightsController;
33  import yatospace.worker.services.controller.MessagesController;
34  import yatospace.worker.services.controller.ReservationsController;
35  import yatospace.worker.services.file.ReservationTransportSpecificationFileController;
36  import yatospace.worker.services.jpa.Flight;
37  import yatospace.worker.services.jpa.Reservation;
38  import yatospace.worker.web.registration.WorkerCredentialsTester;
```

```
40  /**
41   * Централни контролер за сервисирање функционалности
42   * за управљаање подацима летова, резервацијама, порукама.
43   * @author VM
44   * @version 1.0
45   */
46  @CrossOrigin("http://localhost:8080")
47  @RestController
48  @EnableAutoConfiguration
49  public class ServicesPoint {
50          private ExampleInfoBean        exampleInfoBean = new ExampleInfoBean();
51
52          private FlightsController      flightsController      = ControllerCenter.controllerCenter.getFlightsController();
53          private MessagesController     messagesController     = ControllerCenter.controllerCenter.getMessagesController();
54          private ReservationsController reservationsController = ControllerCenter.controllerCenter.getReservationsController();
55          private WorkerCredentialsTester workerCredentialsTester = ControllerCenter.controllerCenter.getWorkerCredentialsTester();
56          private ReservationTransportSpecificationFileController astController = ControllerCenter.controllerCenter.getAstController();
57          private UserEmailController userEmailController = ControllerCenter.controllerCenter.getUserEmailController();
58          private IPMessagingController ipMessagingController =  ControllerCenter.controllerCenter.getIpMessagingController();
59
```

```
public ExampleInfoBean getExampleInfoBean() {
            return exampleInfoBean;
    }

    public UserEmailController getUserEmailController() {
            return userEmailController;
    }

    public IPMessagingController getIpMessagingController() {
            return ipMessagingController;
    }

    public ReservationTransportSpecificationFileController getAstController() {
            return astController;
    }

    public FlightsController getFlightsController() {
            return flightsController;
    }

    public MessagesController getMessagesController() {
            return messagesController;
    }

    public ReservationsController getReservationsController() {
            return reservationsController;
    }

    public WorkerCredentialsTester getWorkerCredentialsTester() {
            return workerCredentialsTester;
    }
```

```java
@RequestMapping("/hello")
public String hello() {
    return "Hello Standard World";
}


@RequestMapping(value = "/ex_info/get", method = RequestMethod.GET, produces = "application/json")
public String getExample() {
    Gson gson = new Gson();
    return gson.toJson(exampleInfoBean);
}

@RequestMapping(value="/ex_info/set", method = RequestMethod.POST, produces = "application/json", consumes="text/plain")
public String setExample(@RequestBody String text) {
    try {
            Gson gson = new Gson();
            exampleInfoBean = gson.fromJson(text, ExampleInfoBean.class);
            JsonObject object = new JsonObject();
            object.addProperty("success", "true");
            object.addProperty("message", "");
            return object.toString();
    }catch(Exception ex) {
            JsonObject object = new JsonObject();
            object.addProperty("success", "false");
            object.addProperty("message", "");
            return object.toString();
    }
}
```

```java
    @RequestMapping(value="/flights/list", method = RequestMethod.GET, produces = "application/json")
    public String listFlights() {
                try {
                        Gson gson = new Gson();
                        JsonParser parser = new JsonParser();
                        JsonArray array = new JsonArray();
                        for(Flight f: flightsController.getFlightsDAO().list()) {
                                array.add(parser.parse(gson.toJson(f)));
                        }
                        return array.toString();
                }catch(Exception ex) {
                        return new JsonArray().toString();
                }
    }

    @RequestMapping(value="/flights/get", method = RequestMethod.POST, produces = "applicatin/json", consumes="text/plain")
    public String getFlights(@RequestBody String text) {
        try {
                Gson gson = new Gson();
                JsonParser parser = new JsonParser();
                JsonObject object = parser.parse(text).getAsJsonObject();
                String flightId = object.get("flight_id").getAsString();
                Flight flight = flightsController.getFlightsDAO().get(flightId);
                if(flight==null) throw new RuntimeException();
                else return gson.toJson(flight);
        }catch(Exception ex) {
                return new JsonObject().toString();
        }
    }
```

```java
@RequestMapping(value="/flights/put", method = RequestMethod.POST, produces = "application/json", consumes="text/plain")
public  String putFlight(@RequestBody String text) {
    try {
            JsonParser parser = new JsonParser();
            JsonObject object = parser.parse(text).getAsJsonObject();
            String username = object.get("username").getAsString();
            String password  = object.get("password").getAsString();
            String flyDate = object.get("fly_date").getAsString();
            String flyId   = object.get("fly_id").getAsString();
            String relation = object.get("relation").getAsString();

            if(!workerCredentialsTester.testWokrker(username, password))
                    throw new RuntimeException();

            Flight flight = new Flight();
            flight.setFlightDate(flyDate);
            flight.setFlightId(flyId);
            flight.setRelation(relation);

            flightsController.getFlightsDAO().put(flight);
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", true);
            return jsonObject.toString();
    }catch(RuntimeException ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }catch(Exception ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }
}
```

```java
190     @RequestMapping(value="/flights/delete", method = RequestMethod.POST, produces = "application/json", consumes="text/plain")
191     public  String deleteFlight(@RequestBody String text){
192         try {
193                 JsonParser parser = new JsonParser();
194                 JsonObject object = parser.parse(text).getAsJsonObject();
195                 String username = object.get("username").getAsString();
196                 String password  = object.get("password").getAsString();
197                 JsonObject jsonObject = new JsonObject();
198                 String flyId = object.get("fly_id").getAsString();
199                 if(!workerCredentialsTester.testWokrker(username, password))
200                         throw new RuntimeException();
201                 Flight flight = new Flight();
202                 flight.setFlightId(flyId);
203                 flightsController.getFlightsDAO().delete(flyId);
204                 jsonObject.addProperty("success", true);
205                 return jsonObject.toString();
206         }catch(RuntimeException ex) {
207                 JsonObject jsonObject = new JsonObject();
208                 jsonObject.addProperty("success", false);
209                 return jsonObject.toString();
210         }catch(Exception ex) {
211                 JsonObject jsonObject = new JsonObject();
212                 jsonObject.addProperty("success", false);
213                 return jsonObject.toString();
214         }
215     }
```

```
218     @RequestMapping(value="/flights/update", method = RequestMethod.POST, produces = "text/plain", consumes="text/plain")
219     public String updateFlight(@RequestBody String text){
220         try {
221                 JsonParser parser = new JsonParser();
222                 JsonObject object = parser.parse(text).getAsJsonObject();
223                 String username = object.get("username").getAsString();
224                 String password  = object.get("password").getAsString();
225                 String oldFlyId = object.get("old_fly_id").getAsString();
226                 String neoFlyId = object.get("neo_fly_id").getAsString();
227                 String flyDate    = object.get("fly_date").getAsString();
228                 String relation = object.get("relation").getAsString();
229 
230                 JsonObject jsonObject = new JsonObject();
231                 if(!workerCredentialsTester.testWokrker(username, password))
232                         throw new RuntimeException();
233 
234                 Flight flight = new Flight();
235                 flight.setFlightDate(flyDate);
236                 flight.setFlightId(neoFlyId);
237                 flight.setRelation(relation);
238 
239                 flightsController.getFlightsDAO().update(oldFlyId, flight);
240                 jsonObject.addProperty("success", true);
241                 return jsonObject.toString();
242         }catch(Exception ex) {
243                 JsonObject jsonObject = new JsonObject();
244                 jsonObject.addProperty("success", false);
245                 return jsonObject.toString();
246         }
247     }
```

```
249     @RequestMapping(value="/reservations/list", method = RequestMethod.GET, produces = "application/json")
250     public String listReservations() {
251         try {
252                         Gson gson = new Gson();
253                         JsonParser parser = new JsonParser();
254                         JsonArray array = new JsonArray();
255                         for(Reservation r: reservationsController.getReservationsDAO().listReservations()) {
256                                 JsonObject object = parser.parse(gson.toJson(r)).getAsJsonObject();
257                                 JsonObject fmt = new JsonObject();
258                                 String rId = r.getReservationId();
259                                 String encodeRID = URLEncoder.encode(rId, "UTF-8");
260                                 String astf = r.getArticleTransportFile();
261                                 if(astf==null) astf = "";
262                                 if(astf.startsWith(encodeRID+"_"))
263                                         astf = astf.substring(encodeRID.length()+1);
264                                 fmt.addProperty("astf_name", astf);
265                                 object.add("fmt", fmt);
266                                 array.add(object);
267                         }
268                         return array.toString();
269                 }catch(Exception ex) {
270                         return new JsonArray().toString();
271                 }
272     }
```

```
@RequestMapping(value="/reservations/get", method = RequestMethod.POST, produces = "applicatin/json", consumes="text/plain")
public String getReservations(@RequestBody String text) {
    try {
            Gson gson = new Gson();
            JsonParser parser = new JsonParser();
            JsonObject object = parser.parse(text).getAsJsonObject();
            String reservationId = object.get("reservation_id").getAsString();
            Reservation reservation = reservationsController.getReservationsDAO().get(reservationId);
            if(reservation==null) throw new RuntimeException();
            else {
                    JsonObject obj = parser.parse(gson.toJson(reservation)).getAsJsonObject();
                    JsonObject fmt = new JsonObject();
                            String rId = reservation.getReservationId();
                            String encodeRID = URLEncoder.encode(rId, "UTF-8");
                            String astf = reservation.getArticleTransportFile();
                            if(astf==null) astf = "";
                            if(astf.startsWith(encodeRID+"_"))
                                    astf = astf.substring(encodeRID.length()+1);
                            fmt.addProperty("astf_name", astf);
                            obj.add("fmt", fmt);
                            return obj.toString();
            }
    }catch(Exception ex) {
            return new JsonObject().toString();
    }
}
```

```
@RequestMapping(value="/reservations/put", method = RequestMethod.POST, produces = "application/json", consumes="text/plain")
public  String putReservation(@RequestBody String text) {
    try {
            JsonParser parser = new JsonParser();
            JsonObject object = parser.parse(text).getAsJsonObject();
            String username = object.get("username").getAsString();
            String password  = object.get("password").getAsString();

            String targetUsername = object.get("target_username").getAsString();
            String flyId          = object.get("fly_id").getAsString();
            String reservationId  = object.get("reservation_id").getAsString();

            int placeCount        = -1;
            String articleDescription = "";
            String articleTranspofrtFile = "";

            try{articleDescription = object.get("article_description").getAsString();}catch(Exception ex) {}
            try{articleDescription = object.get("articleTransportFile").getAsString();}catch(Exception ex) {}
            try{placeCount = object.get("place_count").getAsInt();}catch(Exception ex) {}

            if(!workerCredentialsTester.testWokrker(username, password))
                    throw new RuntimeException();

            if(!workerCredentialsTester.existsUser(targetUsername))
                    throw new RuntimeException();
```

```
@RequestMapping(value="/reservations/delete", method = RequestMethod.POST, produces = "application/json", consumes="text/plain")
public  String deleteReservation(@RequestBody String text){
    try {
            JsonParser parser = new JsonParser();
            JsonObject object = parser.parse(text).getAsJsonObject();
            String username = object.get("username").getAsString();
            String password  = object.get("password").getAsString();
            JsonObject jsonObject = new JsonObject();
            String reservationId = object.get("reservation_id").getAsString();
            if(!workerCredentialsTester.testWokrker(username, password))
                    throw new RuntimeException();
            reservationsController.getReservationsDAO().delete(reservationId);
            jsonObject.addProperty("success", true);
            return jsonObject.toString();
    }catch(RuntimeException ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }catch(Exception ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }
}
```

```
@RequestMapping(value="/reservations/update", method = RequestMethod.POST, produces = "text/plain", consumes="text/plain")
public String updateReservation(@RequestBody String text){
    try {
            JsonParser parser = new JsonParser();
            JsonObject object = parser.parse(text).getAsJsonObject();
            String username = object.get("username").getAsString();
            String password  = object.get("password").getAsString();

            String targetUsername = object.get("target_username").getAsString();
            String flyId          = object.get("fly_id").getAsString();
            String reservationId  = object.get("reservation_id").getAsString();
            String newReservationId = object.get("neo_reservation_id").getAsString();

            int placeCount        = -1;
            String articleDescription = "";
            String articleTranspofrtFile = "";

            try{articleDescription = object.get("article_description").getAsString();}catch(Exception ex) {}
            try{articleDescription = object.get("articleTransportFile").getAsString();}catch(Exception ex) {}
            try{placeCount = object.get("place_count").getAsInt();}catch(Exception ex) {}

            JsonObject jsonObject = new JsonObject();
            if(!workerCredentialsTester.testWokrker(username, password))
                    throw new RuntimeException();

            Reservation reservation = new Reservation();
            reservation.setArticleDescription(articleDescription);
            reservation.setArticleTransportFile(articleTranspofrtFile);
            reservation.setFlyId(flyId);
            reservation.setPlaceCount(placeCount);
            reservation.setReservationId(newReservationId);
            reservation.setUsername(targetUsername);
```

```
            Reservation reservation = new Reservation();
            reservation.setArticleDescription(articleDescription);
            reservation.setArticleTransportFile(articleTranspofrtFile);
            reservation.setFlyId(flyId);
            reservation.setPlaceCount(placeCount);
            reservation.setReservationId(newReservationId);
            reservation.setUsername(targetUsername);

            reservationsController.getReservationsDAO().update(reservationId, reservation);
            jsonObject.addProperty("success", true);

            return jsonObject.toString();
    }catch(Exception ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }
}
```

```
@RequestMapping(value="/reservations/upload_ast", method = RequestMethod.POST, produces = "text/plain", consumes = { "multipart/mixed", "multipart/form-data" })
public String uploadFile(@RequestPart("file")        MultipartFile file,
                         @RequestPart("username")    String username,
                         @RequestPart("password")    String password,
                         @RequestPart("file_name")    String fileName,
                         @RequestPart("reservation_id") String rId) {
    try {
            byte[] bytes = file.getBytes();

            if(!workerCredentialsTester.testWokrker(username, password))
                    throw new RuntimeException();

            if(reservationsController.getReservationsDAO().get(rId)==null)
                    throw new RuntimeException("Resservation not found");

            fileName = URLEncoder.encode(rId,"UTF-8")+"_"+fileName;
            astController.put(rId, fileName, bytes);

            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", true);
            return jsonObject.toString();
    }catch(RuntimeException ex) {
            ex.printStackTrace();
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }catch(Exception ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }
}
```

```java
@RequestMapping(value="/reservations/reset_ast", method = RequestMethod.POST, produces = "text/plain", consumes="text/plain")
public String resetFile(@RequestBody String text) {
    try {
            JsonParser parser = new JsonParser();
            JsonObject object = parser.parse(text).getAsJsonObject();

            String username = object.get("username").getAsString();
            String password  = object.get("password").getAsString();

            if(!workerCredentialsTester.testWokrker(username, password))
                    throw new RuntimeException();

            String rId = object.get("reservation_id").getAsString();
            if(rId==null) throw new RuntimeException();
            Reservation r = this.reservationsController.getReservationsDAO().get(rId);
            if(r==null) throw new RuntimeException();
            String name = r.getArticleTransportFile();
            this.astController.remove(rId, name);

            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", true);
            return jsonObject.toString();
    }catch(RuntimeException ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }catch(Exception ex) {
            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", false);
            return jsonObject.toString();
    }
}
```

```java
@RequestMapping(value="/messages/list/read", method = RequestMethod.GET, produces = "application/json")
public String listReadMessages() {
    try {
            Gson gson = new Gson();
            JsonParser parser = new JsonParser();
            JsonArray array = new JsonArray();
            for(IPMessageDTO messageDTO: ipMessagingController.getMessageDAO().listReviewedMessage()) {
                    if(messageDTO==null) continue;
                    if(messageDTO.getMessage()==null) continue;
                    JsonObject dataObject = parser.parse(gson.toJson(messageDTO.getMessage())).getAsJsonObject();
                    String username = messageDTO.getMessage().getUsername();
                    JsonObject emailObject = new JsonObject();
                    EmailDTO emailDTO = userEmailController.getEmailDataSource().getEmailByUsername(username);
                    if(emailDTO==null) emailObject.addProperty("address", "");
                    else if(emailDTO.getUserEmail()==null) emailObject.addProperty("address", "");
                    else emailObject.addProperty("address", emailDTO.getUserEmail().getEmailAddress());
                    JsonObject messageRowInfo = new JsonObject();
                    UserMessageFormatEnviroment umfEnviroment = new UserMessageFormatEnviroment(messageDTO.getMessage());
                    JsonObject fmt = new JsonObject();
                    fmt.addProperty("created", umfEnviroment.createdTimestampStandard());
                    fmt.addProperty("changed", umfEnviroment.modifiedTimestampStandard());
                    messageRowInfo.add("message", dataObject);
                    messageRowInfo.add("email", emailObject);
                    messageRowInfo.add("fmt", fmt);
                    array.add(messageRowInfo);
            }
            return array.toString();
    }catch(RuntimeException ex) {
            JsonArray json = new JsonArray();
            return json.toString();
    }catch(Exception ex) {
            JsonArray json = new JsonArray();
            return json.toString();
    }
}
```

```java
@RequestMapping(value="/messages/list/non_read", method = RequestMethod.GET, produces = "application/json")
public String listNonReadMessages() {
    try {
            Gson gson = new Gson();
            JsonParser parser = new JsonParser();
            JsonArray array = new JsonArray();
            for(IPMessageDTO messageDTO: ipMessagingController.getMessageDAO().listUnviewedMessage()) {
                    if(messageDTO==null) continue;
                    if(messageDTO.getMessage()==null) continue;
                    JsonObject dataObject = parser.parse(gson.toJson(messageDTO.getMessage())).getAsJsonObject();
                    String username = messageDTO.getMessage().getUsername();
                    JsonObject emailObject = new JsonObject();
                    EmailDTO emailDTO = userEmailController.getEmailDataSource().getEmailByUsername(username);
                    if(emailDTO==null) emailObject.addProperty("address", "");
                    else if(emailDTO.getUserEmail()==null) emailObject.addProperty("address", "");
                    else emailObject.addProperty("address", emailDTO.getUserEmail().getEmailAddress());
                    JsonObject messageRowInfo = new JsonObject();
                    UserMessageFormatEnviroment umfEnviroment = new UserMessageFormatEnviroment(messageDTO.getMessage());
                    JsonObject fmt = new JsonObject();
                    fmt.addProperty("created", umfEnviroment.createdTimestampStandard());
                    fmt.addProperty("changed", umfEnviroment.modifiedTimestampStandard());
                    messageRowInfo.add("message", dataObject);
                    messageRowInfo.add("email", emailObject);
                    messageRowInfo.add("fmt", fmt);
                    array.add(messageRowInfo);
            }
            return array.toString();
    }catch(RuntimeException ex) {
            JsonArray json = new JsonArray();
            return json.toString();
    }catch(Exception ex) {
            JsonArray json = new JsonArray();
            return json.toString();
    }
}
```

```
@RequestMapping(value="/messages/mark_read", method = RequestMethod.POST, produces = "application/json")
public String messageMarkRead(@RequestBody String text) {
    try {
            JsonParser parser = new JsonParser();
            JsonObject request = parser.parse(text).getAsJsonObject();
            String messageId = request.get("message_id").getAsString();
            if(messageId==null) messageId = "";

            String username = request.get("username").getAsString();
            String password  = request.get("password").getAsString();

            if(!workerCredentialsTester.testWokrker(username, password))
                    throw new RuntimeException();

            if(ipMessagingController.getMessageDAO().get(messageId)==null)
                    throw new RuntimeException("Message not found");

            ipMessagingController.getMessageDAO().markRead(messageId);

            JsonObject object = new JsonObject();
            object.addProperty("success", true);
            return object.toString();
    }catch(RuntimeException ex) {
            JsonObject object = new JsonObject();
            object.addProperty("success", false);
            return object.toString();
    }catch(Exception ex) {
            JsonObject object = new JsonObject();
            object.addProperty("success", false);
            return object.toString();
    }
}
```

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

Jersey REST Services API Example

```
@Path("/myResource")
@Produces("text/plain")
public class SomeResource {
        @GET
        public String doGetAsPlainText() {
                ...
        }

        @GET
        @Produces("text/html")
        public String doGetAsHtml() {
                ...
        }
}
```