REST API Security
Basic Security Concepts

Guadalupe Ortiz Bellot

Department of Computer Science and Engineering

UCA
Universidad de Cádiz

Contents

1. **Use of Response**
2. Logger

1. Use of Response (i)

- We use a Response when we want to return an error code and information when something goes wrong in the invocation.
- We can use runtime information.
- If invocations is ok, we return the result of invoking the method and a 200 status code.

1. Use of Response (ii)

```
@POST
@Path("/date2015to2016JSONResponse")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response getDate_JSONResponse(MyDate myDate) {
  if (myDate.getYear()!=2015) {
     return Response.status(Status.NOT_FOUND.getStatusCode()).entity((String)"Year is not correct").build();}
  else{
     myDate.setYear(2016);
     return Response.status(200).entity(myDate).build();}   }
```

Contents

1. Use of Response
2. **Logger**

2. Logger (i)

- We can use a logger file to debug the behabiour of our service
- We need to:
  - Create **log4j.properties** in Java Resources/resources (Botón derecho en el Proyecto → New –→ Source Folders , seleccionamos el proyecto y ponemos el nombre *resources* a la carpeta.
  - Include such folder in the classpath of the project: Right click on the project→Build Path→ Configure build path→ source→ Add folder
  - Create the local text file for storing the logs (i.e. C:\\Development\log4j-application.log)
  - Include the log4j library
  - Create the logger instance
  - Write in the logger whatever we need

Note that you cannot change the name to the file log4j.properties, but you can use any other name for the local file where you store the logs as long it is a text file.

2. Logger (ii) - Create log4j.properties **in resources**

```
log4j.rootLogger=DEBUG, stdout, file

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

# Redirect log messages to a log file, support file rolling.
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=C:\\Development\\log4j-application.log
log4j.appender.file.MaxFileSize=5MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
```

2. Logger (iii)

Create the file where the logs will be stored:

C:\\Development\log4j-application.log

You have to include log4j-1.2.17.jar in the project libraries (we already included it)

Create the logger instance in you hello class: inside the class, outside the methods *(use the Apache import)* and replace *NameofTheClass* by the real name of your service class:

```
static Logger logger = Logger.getLogger(NameOfTheClass.class);
```

Write in the log in any of your methods, for instance when the year is not found in getDate_JSONResponse

```
logger.info("error "+Status.NOT_FOUND.getStatusCode());
```

Test it in Postman and check what is written in the log file

Support Bibliography and References

- Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON. By: Masoud Kalali; Bhakti Mehta. Publisher: Packt Publishing Pub. Date: October 15, 2013. Print ISBN-13: 978-1-78217-812-5
- REST with Java (JAX-RS) using Jersey - Tutorial Lars Voguel https://www.vogella.com/tutorials/REST/article.html
- Postman API Client - Postman Inc. https://www.postman.com/product/api-client/