Assignment 5

Programming II / Enterprise Java Programming, Year 2019/2020, Semester 2

- *If asked to write Java code, submit source code files (filename ending .java). Please create a separate file for each of your Java classes.*
- *Put all files into a single zip-file named "Assignment5-YourName.zip" and submit via Blackboard. Please submit the entire Eclipse EE project folder (i.e. folder ct545rest and all subfolders containing .java files, .jsp files etc.). please also submit screenshots showing that you have tested the features of your RESTful API using Postman.*
- ***Deadline (strict): Friday, 03rd April, 23:59. Late submission only with a medical certificate.***
- *Please note that the graders are advised to check all submissions for plagiarism. Assignments need to be solved individually (no group work).*
- *If you have questions about the assignment or need help with it, the best approach is to ask a tutor during one of the lab sessions (it is recommended to visit all lab sessions).*
- *Use (Java) comments to explain your code. Missing or insufficient comments lead to mark deductions.*

In this assignment you will use the JAX-RS/Jersey API to create a simple RESTful Car Hire Booking System. Users should be able to interact with your service directly from another application (e.g. Postman). Your RESTful Car Hire Booking System should have full Create, Read, Update and Delete functionality for Car Hire Bookings. As a bonus (ungraded) extra feature you could also develop a browser-based GUI client using HTML/JSP pages.

Data Modelling (20%)

An appropriate Java data model will be required for all classes/entities which are part of the car hire booking (e.g. Customer, Vehicle, Booking, BookingList). Each Booking will be associated with one Customer and one Vehicle.

Customer details should include firstName, lastName and address.

Vehicle details should include registration, manufacturer, and colour.

Booking details should include id, customer, vehicle, startDate and endDate

Bookings is a wrapper class for a data structure that will hold the list of vehicle bookings.

N.B. The Customer, Vehicle and Booking classes all need to have default constructors. If you do not provide these, you will get errors when unmarshalling Bookings from JSON.

(Note: to simplify things, it is fine to store the startDate and endDate as strings e.g. 18/03/2020. N.B. This should **never** be done in a real application.)

RESTful Web Service (70%)

Design a RESTful Web Service **BookingService.java** using JAX-RS/Jersey which will act as the gateway for all clients that wish to use the Car Hire Booking System. Clients will be able to access CRUD functionality for car hire bookings using the GET, POST, PUT and DELETE methods.

BookingService.java will be responsible for marshalling/unmarshalling data to/from JSON for all requests/responses to/from the service using applications (e.g. Postman).

An appropriate data structure may be used to store the list of Booking objects within a singleton instance of **BookingService.java**, as per point 12 in the CT545_1920_REST_tutorial .pdf on Blackboard. You should create a few sample Booking objects and add them to the BookingList when **BookingService.java** is instantiated.

(Note: in a real application persistence would be handled by a database, accessed using e.g. Java Persistence API. To keep things relatively simple we will focus in this assignment only on the process of building a RESTful API. There is no need to connect your JAX-RS service to a database for this assignment!)

BookingService.java should be available at the URL http://localhost:8080/ct545rest/webapi/bookingservice and the API should have the following features:

1. When a GET request is received at the URL http://localhost:8080/ct545rest/webapi/bookingservice the service should return the list of details of all bookings in the system in JSON format.
2. When a GET request is received at the URL http://localhost:8080/ct545rest/webapi/bookingservice/{id} the service should return details of the booking with the specified id in JSON format.
3. When a POST request is received at the URL http://localhost:8080/ct545rest/webapi/bookingservice the service should create a new instance of Booking with the details in the POST request, and add it to the BookingList. This method should return details of the Booking that was just added in JSON format.
4. When a PUT request is received at the URL http://localhost:8080/ct545rest/webapi/bookingservice/{id} the service should update the details of the booking with the specified id using the JSON representation in the body of the PUT request. This method should return details of the Booking that was just updated in JSON format.
5. When a DELETE request is received at the URL http://localhost:8080/ct545rest/webapi/bookingservice/{id} the service should delete the Booking with the specified id

For items 1, 2, and 5 above, if the request is successful a HTTP response 200 OK should be returned to the client.

For item 3 above, if the request is successful a HTTP response 201 Created should be returned to the client.

For items 2, 4 and 5 above, if the Booking with the id specified does not exist, a HTTP 404 Not Found response code should be returned to the client.

API testing in Postman (10%)

Use the Postman app to test the functionality of each of items 1 – 5 above. Include screenshots showing the results of your tests along with your code.

BONUS: Browser-based Client

Note: it is not necessary to complete this section to get 100%. However, it will be good practice for you to attempt this part.

N.B. do not spend too much time on the styling and layout of the Web Client; basic unstyled HTML forms, tables and buttons etc. will suffice as this is not a client-side web development project. There is no need to use any css styling or client-side libraries like Bootstrap.

Extend **BookingService.java** so that it can also handle requests/responses to/from a browser-based GUI, providing full CRUD functionality via a web browser like Google Chrome. Note that some pages for the browser-based GUI will be generated by appropriately annotated methods within **BookingService.java**, while others will have to be generated using JSP pages.

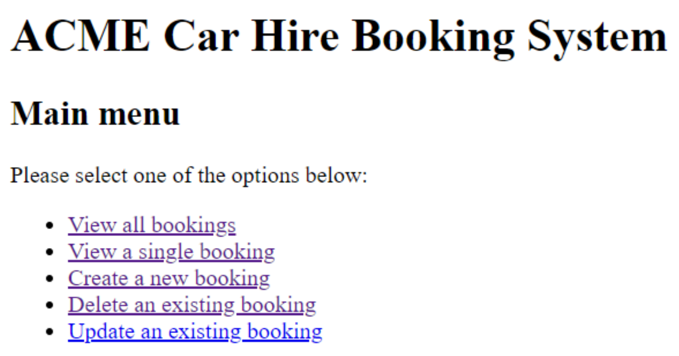
Note: To keep this simple I'd suggest using HTML forms to read in and send user data like the specified id number, details of a new booking etc. to the server. HTML forms unfortunately only support the GET and POST methods (DELETE and PUT are unsupported). Therefore, Update and Delete functionality from HTML forms will have to be handled via the POST method. Note the mismatch here between using the correct semantics for the HTTP verbs, and the functionality that is actually available to a client-side programmer using basic HTML. The JavaScript XMLHttpRequest object, and client-side JavaScript libraries like jQuery support sending requests using any HTTP method but covering those in detail is outside the scope of this module.

Your browser-based client should have the following functionality:

1. index.jsp should be the landing page/main menu of the Car Hire Booking Service. From this page, there should be links to view all bookings, view a single booking, delete a booking by id, add a new booking
2. All HTML/JSP pages displayed to the browser should have hyperlinks back to the main menu (index.jsp). All URLs should be relative rather than absolute. Note that from certain URLs you may need to go back one level to access the page that you want using the .. shorthand, i.e. "../pagetovisit.jsp"
3. The details of all bookings formatted in a HTML table should be returned when a web browser visits http://localhost:8080/ct545rest/webapi/bookingservice - note that you can write a HTML document with a table into a string and send it back as a HTTP response from an appropriately annotated method in **BookingService.java**. The format of the response that is sent should depend on the value of accept header field in the client's HTTP request. Responses should therefore be available in JSON and in the form of a HTML table.
4. The details of single booking with the specified id formatted in a HTML table should be returned when a web browser visits http://localhost:8080/ct545rest/webapi/bookingservice/{id}
5. A new booking should be created when the contents of a HTML form are sent to **BookingService.java** at the URL http://localhost:8080/ct545rest/webapi/bookingservice using a HTTP POST request. Use a HTML form on a JSP page called createbooking.jsp for this, then handle the form parameters using Jersey, as per point 8 in the CT545_1920_REST_tutorial .pdf
6. Delete a booking by id - use a simple HTML text input to read the id to be deleted, then POST the data to an appropriately annotated method in **BookingService.java**. The response should be a HTML page stating that the booking was successfully deleted if the request was made by a web browser.
7. Update the details of a booking with a specified id.

Note: all the sample images below relate only to the BONUS section. It is not necessary to create any of these pages to get full marks.

Sample index.jsp



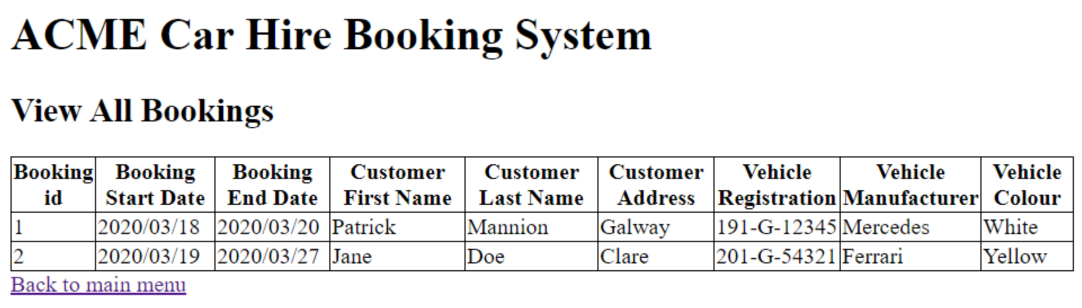
ACME Car Hire Booking System

Main menu

Please select one of the options below:

- View all bookings
- View a single booking
- Create a new booking
- Delete an existing booking
- Update an existing booking

Sample HTML output of GET request to http://localhost:8080/ct545rest/webapi/bookingservice using a browser (e.g. Chrome)



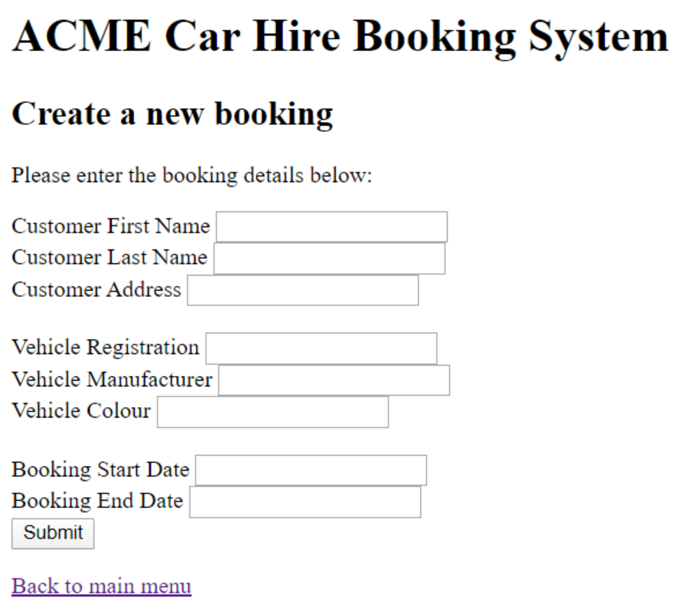
ACME Car Hire Booking System

View All Bookings

Booking id	Booking Start Date	Booking End Date	Customer First Name	Customer Last Name	Customer Address	Vehicle Registration	Vehicle Manufacturer	Vehicle Colour
1	2020/03/18	2020/03/20	Patrick	Mannion	Galway	191-G-12345	Mercedes	White
2	2020/03/19	2020/03/27	Jane	Doe	Clare	201-G-54321	Ferrari	Yellow

Back to main menu

Sample createbooking.jsp



ACME Car Hire Booking System

Create a new booking

Please enter the booking details below:

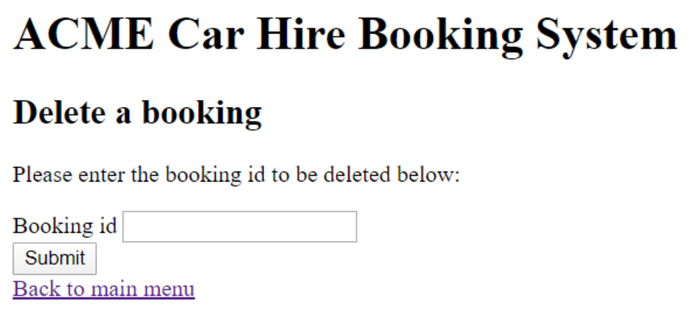
Customer First Name
Customer Last Name
Customer Address

Vehicle Registration
Vehicle Manufacturer
Vehicle Colour

Booking Start Date
Booking End Date

Back to main menu

Sample delete booking page



ACME Car Hire Booking System

Delete a booking

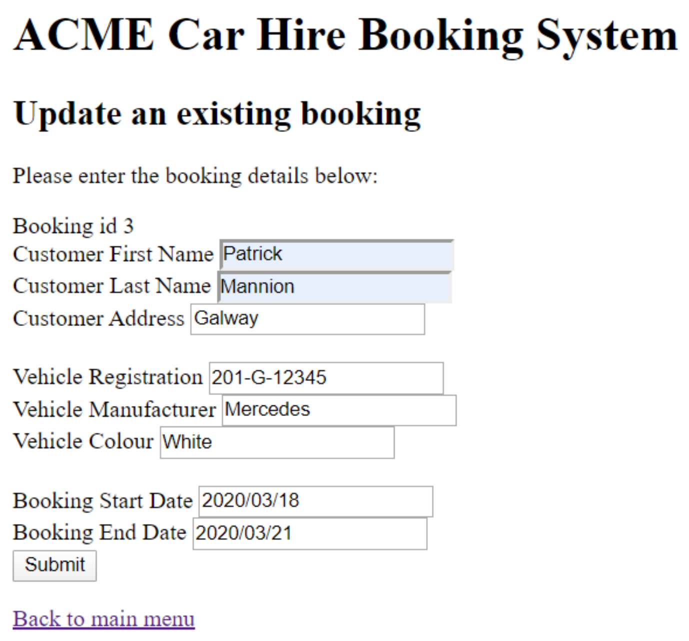
Please enter the booking id to be deleted below:

Booking id	

Submit

Back to main menu

Sample update booking page



ACME Car Hire Booking System

Update an existing booking

Please enter the booking details below:

Booking id	3
Customer First Name	Patrick
Customer Last Name	Mannion
Customer Address	Galway
Vehicle Registration	201-G-12345
Vehicle Manufacturer	Mercedes
Vehicle Colour	White
Booking Start Date	2020/03/18
Booking End Date	2020/03/21

Submit

Back to main menu