CMPT 213 Assignment 5

Due: Dec 8, 11:59:59pm.

Overview

In case you forgot, a Tokimon is a rare creature found in the remote areas of Korea. Tokimons come in all shapes and sizes, have special abilities, and tend to fight with each other on occasion. Each Tokimon could have the ability to fly, throw fire, electrify, and freeze other Tokimons; their ability is measured by an integer between 0 and 100. Your task in this assignment is to create an application that will collect information about a set of Tokimons and store their attributes into a JSON file.
We will use a **client server model**. The server will be running a Spring application using Tomcat providing a REST API, while the client will be implemented using a JavaFX User Interface that allows users to look up and augment the existing data.

- Data: your application will keep all known Tokimons in a json file called tokimon.json, and in a folder called 'data'. From the root of your application, the file should be located in 'data/tokimon.json'. This file is used to collect information about a the Tokimons. The minimum attributes for the Tokimon are id (unique identifier for a Tokimon), name, weight, height, ability (i.e. fly, fire, water, electric, freeze (or ice)), strength, and color. You may add any other Tokimon attributes as you see fit.

- Features: your app should have the following features:
  1. The ability to add new Tokimon (with corresponding attributes).
  2. The ability to delete any of the Tokimons.
  3. The ability to display all Tokimons currently in the tokimon.json file. This could be a subset of the attributes. For example, you may only choose to display only the name and the picture of the Tokimon (see below), then let the use click on the image to reveal more information.
  4. The ability to display information about a specific Tokimon. A link should be provided from each of the Tokimons currently in the system.
  5. The ability to change a particular attribute about a Tokimon

In addition, you may also add or display any other attributes you wish. Please note that the requirement stated above is a minimum list, you may add to them as you see fit. Please be creative, part of your grade will be based on usability, and creativity.

Architecture

You will be creating two separate projects for this assignment.

- The first application will be a Server that provides a REST API. This application will listen for client requests on port 8080 and update the server accordingly, returning (responding with) the desired information to the client. This includes updating the tokimon.json file if needed.
- The second will be a client-side application written in JavaFX that is responsible for providing user interface and making client calls to the server.

Server (REST API)

General requirements

- All commands return HTTP 200 (OK) unless otherwise stated.
- Any end-point accepting an ID must return an HTTP 404 error with a meaningful

message if the ID does not exist.

The REST API for your application should have the following endpoints:

GET /api/tokimon/all

- Returns a list of all Tokimon objects in the tokimon.json file. This includes all attributes associated with each object.

GET /api/tokimon/{id}

- Returns an object corresponding to Tokimon with the specified id. For example /api/tokimon/3 would return the Tokimon object corresponding to id=3.

POST /api/tokimon/add

- Create a new Tokimon.
- Returns HTTP 201: Created.
- Expected body contents include all attributes of the new Tokimon.

POST /api/tokimon/change/{id}

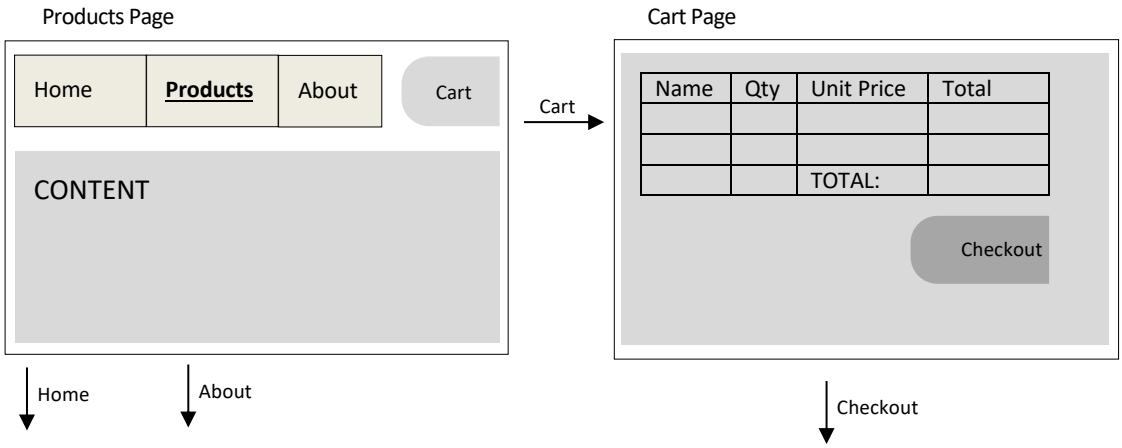
- Create a new Tokimon.
- Returns HTTP 201: Created.
- Expected body contents include attribute/value pairs to be altered in the Tokimon with specified id.

DELETE /api/tokimon/{id}

- Removes Tokimon with specified id.
- Returns HTTP 204: No content.

Client (JavaFX)

Begin with a UI mockup of your app. You do not need to submit the mockup, but it will be very helpful to program this in JavaFX. You are welcome to use as many (or as few) sub-windows as you'd like in the application. However, keep in mind the usability of your application. The UI mockup should contain your different scenes and how they are linked. As a small example, a UI mockup for part of a department store (buying) application could look like the following:



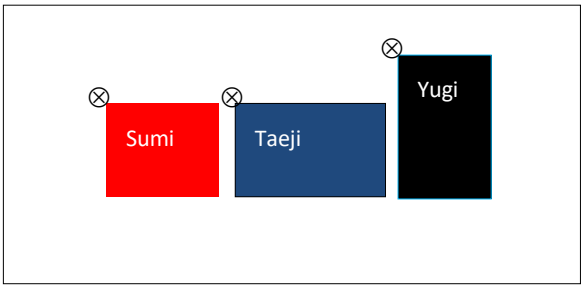
Requirements:

The only real requirement for the view is to display different types of Tokimon differently. One idea is to display Tokimon according to their weight and height. Let's say the current Tokimons in the json file looks like this:

	name	weight	height	fire	water	…	color
1	Sumi	150	30	100	0	…	red
2	Taeji	200	30	0	10	…	blue
3	Yugi	120	50	20	70	…	black

The display page *could* look something like this:

Display Page



Notice Sumi and Taeji have the same height but Taeji is proportionally heavier, Yugi weighs less and is taller than both Sumi and Taeji. You are free to use different shapes to represent different types of Tokimons as well. Generally, you are free to design your UI in any way as long as the functionality described earlier is present ☺

Note: any requirements not mentioned in this document is up to interpretation. :)

Marking Scheme:

Server – Total [25] Marks

- [20] Implementation of required REST endpoints, including proper storage/retrieval of information in/from the tokimon.json file
- [3] Proper error pages with meaningful descriptions.
- [2] Proper Response codes implemented

Client – Total [25] Marks

- [16] User Interface that allows users to access all functionality
- [5] Displaying all Tokimons in a meaningful manner using shapes
- [4] Usability, Effort, and Creativity

Programming General Requirements – Total [0] Marks

- Correctly follow coding style guide (up to -6 max in deductions)
- Code Design – follow the concepts of good design outlined in the class. For example, in the server application, dividing the responsibilities of the controller and models into individual classes/packages; on the client side, dividing responsibility of creating sub-windows, drawing, etc. to different classes. (up to -5 max in deductions)

Submission

Submit a zip file of your project (according to the directions outlined in the assignments link of the course website) to the coursys server. https://courses.cs.sfu.ca/. You do not need to generate JAR files for this assignment.

Please note: all submissions are automatically checked for similarities of all other submissions on the server.

THE END