InstaPark Documentation

How to Use InstaPark

* Red Car = Parker is currently defaulted to that parking spot
* Yellow Car = Parker is temporarily reserving that spot
* Green Car = Nobody is currently parked there, empty spot
* Changing the date on the top bar will display the current reservations for both East and West wings for the selected date. It will also set the default date to the selected date when making a new reservation
* Clicking on a default or empty car (red or green) will display current information on the parking spot and present an option to create a new reservation for that selected spot. Once a valid reservation is made, the reservation will be saved and the car will change to yellow to reflect that change
* Clicking on a reserved car (yellow) will display current information on that reservation and present an option to delete the reservation. Once it is confirmed, which is done by accepting the dialog that pops up, the reservation will be deleted and the car will return to either a default red color or empty green color
* A valid reservation is one that includes the name of the parker and a correctly desired begin and end reservation dates; the reason for the reservation is optional

How the Code Works

Index.html

* This is the main html file used to display everything on the screen
* Contains 3 main components: east wing (separate html file), west wing (separate html file) and the date picker
* Loads and runs all the Javascript files in the head, including the controllers; when garage-controller is run, it initially loads the cars
* Loads all the CSS files in the head; some are designed to only load depending on screen size, i.e. mobile or desktop

East.html

* This is the html file that contains all the parking spots for the east wing. There are a bunch of individual parking spots separated into two divs (one for northeast and one for southeast). Each car has an ng-click attribute that when clicked, will run the showEastData() function in garage-controller and display the data for that current spot
* Controlled by garage-controller

West.html

* This is the html file that contains all the parking spots for the west wing. There are a bunch of individual parking spots separated into two divs (one for northeast and one for southeast). Each car has an ng-click attribute that when clicked, will run the showWestData() function in garage-controller and display the data for that current spot
* Controlled by garage-controller

Date Picker on line 57 in index.html

* This is the date-picker element that’s on top of the screen. When the date is changed with this element, it runs three functions. From date-controller, it runs setDate(), which sets the currentDate field in the parkingFactory to whichever date was selected. From garage-controller, it runs clearText(), which clears any previously clicked descriptions from the screen. More importantly, it runs changeDate(), which in turn triggers a chain reaction. It first calls removeAddReservations(), then it gets the new car data for both the east and west side and updates the view accordingly
* Controlled by date-controller

Garage-controller.js

* Main controller for the entire app; contains all the logic, car arrays, http requests, directive definitions, and factory updates.
* Garage.EastObjectArray is the array of cars that is used for logic in the controller for the east section
* Garage.WestObjectArray is the array of cars that is used for logic in the controller for the west section
* $scope.EastCarArray is the array of cars that are shown on the screen for the east section; it really just stores a bunch of image links that are referenced on the front-end
* $scope.WestCarArray is the array of cars that are shown on the screen for the west section; it really just stores a bunch of image links that are referenced on the front-end
* Garage.data is the text that is shown on the screen when the user clicks on a car
* Garage.displayNewReservationButton toggles when the “create new reservation” button is shown (only when a default or empty spot it clicked)
* Garage.displayDeleteButton toggles when the “delete reservation” button is shown (only when a reserved spot is clicked)
* *garage.getParkingData(section)*

This method takes in a section as a parameter, i.e. EAST or WEST, and runs an http request to the server for the current array of cars in that section. Once that array of car objects is returned from the server, **then** it saves it as a local variable in the controller by setting it as Garage.EastObjectArray and it sets that same array to the variable EastCars in the parkingfactory, **then** it goes through each car object in the previous array, and adds a new image link to a temporary array based on its type property, and saves the temporary array by setting it as $scope.EastCarArray. Same thing happens for the West side.

* *garage.getDefaultData()*

This method runs an http request to the server which returns all the data from the table parker, which I then save in the parkingFactory. This is used to load all the data on the screen when the user clicks on a default car

* *garage.removeAddReservations()*

This method runs an http request to the server, with the current date on the calendar up top as query data, which will set the reservation\_parker and default\_parker field

For example, if the selected date was tomorrow, and there the reservation for spot#12 expires after today, this method will set the default\_parker field in the parkingspot table to a 1 and the reservation\_parker to a 0.