FORECAST CONTROLLER

1. **@RestController**: This is a class-level annotation indicating that this class serves as a RESTful web service controller.
2. **@RequestMapping("/forecasts")**: This is a class-level annotation indicating that all the methods in this class will be mapped to URLs that start with "/forecasts".
3. **private final ForecastService forecastService**: This is a private variable that holds an instance of the ForecastService interface. It is annotated with **@Autowired** to have Spring automatically inject the instance at runtime.
4. **public ForecastController(ForecastService forecastService)**: This is the constructor that sets the value of the **forecastService** variable to the passed in **forecastService** argument.
5. **@GetMapping("/{id}")**: This is a method-level annotation indicating that this method will handle HTTP GET requests with URLs that end with "/{id}". It also defines a path variable named **id** to extract the value of the id from the URL.
6. **public Forecast findById(@PathVariable Long id)**: This method returns a Forecast object by calling the **findById** method of the **forecastService** and passing in the **id** argument.
7. **@GetMapping**: This is a method-level annotation indicating that this method will handle HTTP GET requests with URLs that match the class-level **@RequestMapping** URL.
8. **public List<Forecast> findAll()**: This method returns a list of Forecast objects by calling the **findAll** method of the **forecastService**.
9. **@PostMapping**: This is a method-level annotation indicating that this method will handle HTTP POST requests with URLs that match the class-level **@RequestMapping** URL.
10. **public Forecast save(@RequestBody Forecast forecast)**: This method saves a Forecast object by calling the **save** method of the **forecastService** and passing in the **forecast** argument that is extracted from the request body.
11. **@PutMapping**: This is a method-level annotation indicating that this method will handle HTTP PUT requests with URLs that match the class-level **@RequestMapping** URL.
12. **public Forecast update(@RequestBody Forecast forecast)**: This method updates a Forecast object by calling the **update** method of the **forecastService** and passing in the **forecast** argument that is extracted from the request body.
13. **@DeleteMapping("/{id}")**: This is a method-level annotation indicating that this method will handle HTTP DELETE requests with URLs that end with "/{id}". It also defines a path variable named **id** to extract the value of the id from the URL.
14. **public void deleteById(@PathVariable Long id)**: This method deletes a Forecast object by calling the **deleteById** method of the **forecastService** and passing in the **id** argument.

LOCATION CONTROLLER

1. **@RestController** - this is a Spring annotation that indicates that this class serves as a REST API endpoint.
2. **@RequestMapping("/api/location")** - this maps the class level endpoint to **/api/location**. All the HTTP request methods defined in this class will have the base URL of **/api/location**.
3. **private final LocationService locationService;** - this is a private variable that holds a reference to the **LocationService** object.
4. **@Autowired** - this is a Spring annotation that is used to wire the **LocationService** object in the constructor.
5. **public LocationController(LocationService locationService) {** - this is the constructor that takes **LocationService** as an argument and wires it to the private variable **locationService**.
6. **@GetMapping** - this maps the HTTP GET request to the method **getAllLocations()**.
7. **public List<Location> getAllLocations()** - this is a method that returns a list of all **Location** objects.
8. **@PostMapping** - this maps the HTTP POST request to the method **createLocation(@RequestBody Location location)**.
9. **public Location createLocation(@RequestBody Location location)** - this is a method that creates a new **Location** object. The **Location** object is passed as the request body with a POST request.
10. **@GetMapping("/{id}")** - this maps the HTTP GET request to the method **getLocationById(@PathVariable Long id)** with a path variable **id**.
11. **public Location getLocationById(@PathVariable Long id)** - this is a method that retrieves a **Location** object based on the **id** passed in the URL.
12. **@PutMapping("/{id}")** - this maps the HTTP PUT request to the method **updateLocation(@PathVariable Long id, @RequestBody Location location)** with a path variable **id**.
13. **public Location updateLocation(@PathVariable Long id, @RequestBody Location location)** - this is a method that updates an existing **Location** object based on the **id** passed in the URL. The updated **Location** object is passed as the request body with a PUT request.
14. **@DeleteMapping("/{id}")** - this maps the HTTP DELETE request to the method **deleteLocation(@PathVariable Long id)** with a path variable **id**.
15. **public void deleteLocation(@PathVariable Long id)** - this is a method that deletes an existing **Location** object based on the **id** passed in the URL.

LOCATION FORECAST CONTROLLER

1. **@RestController** - This annotation is used to mark the class as a RESTful controller class in Spring Boot, which provides several default behavior for handling HTTP requests.
2. **@RequestMapping("/location-forecast")** - This annotation maps the HTTP requests that start with **/location-forecast** to the methods in the class.
3. **private final LocationForecastService locationForecastService;** - This is a private final variable that holds a reference to the **LocationForecastService** instance, which is used to perform the business logic operations.
4. **@Autowired** - This annotation is used to automatically wire the **LocationForecastService** instance from the Spring application context, into the **locationForecastService** variable.
5. **public LocationForecastController(LocationForecastService locationForecastService)** - This is a constructor of the **LocationForecastController** class, which is used to initialize the **locationForecastService** variable.
6. **@GetMapping("/all")** - This annotation maps the HTTP GET requests to the **findAll** method, which returns a list of all the **LocationForecast** objects.
7. **@GetMapping("/{id}")** - This annotation maps the HTTP GET requests to the **findById** method, which returns a single **LocationForecast** object, based on the id passed in the URL.
8. **@PostMapping("/save")** - This annotation maps the HTTP POST requests to the **save** method, which is used to save a new **LocationForecast** object in the database.
9. **@DeleteMapping("/{id}")** - This annotation maps the HTTP DELETE requests to the **delete** method, which is used to delete a **LocationForecast** object from the database, based on the id passed in the URL.
10. **return locationForecastService.findAll();** - This line returns the list of all the **LocationForecast** objects, by calling the **findAll** method on the **locationForecastService** instance.
11. **return locationForecastService.findById(id);** - This line returns the **LocationForecast** object, by calling the **findById** method on the **locationForecastService** instance, passing the id as an argument.
12. **return locationForecastService.save(locationForecast);** - This line saves the **LocationForecast** object and returns it, by calling the **save** method on the **locationForecastService** instance.
13. **LocationForecast locationForecast = locationForecastService.findById(id);** - This line finds the **LocationForecast** object, by calling the **findById** method on the **locationForecastService** instance, passing the id as an argument.
14. **locationForecastService.delete(locationForecast);** - This line deletes the **LocationForecast** object, by calling the **delete** method on the **locationForecastService** instance.

WEATHERDATA CONTROLLER

1. **@RestController** is a class-level annotation that marks the class as a RESTful web service controller class.
2. **@RequestMapping("/api/weatherdata")** maps the incoming HTTP requests with the URL path **/api/weatherdata** to the methods in this controller class.
3. The class **WeatherDataController** defines a RESTful web service controller class.
4. The class has a private final field **weatherDataService** of type **WeatherDataService** which is being autowired.
5. The class has a constructor that accepts **WeatherDataService** and assigns it to the **weatherDataService** field.
6. The method **getAllWeatherData** is annotated with **@GetMapping** which means it maps to HTTP GET requests and returns a list of all **WeatherData** entities.
7. The method **createWeatherData** is annotated with **@PostMapping** which maps to HTTP POST requests and creates a new **WeatherData** entity.
8. The method **getWeatherDataById** is annotated with **@GetMapping("/{id}")** which maps to HTTP GET requests and returns a specific **WeatherData** entity with the given **id**.
9. The method **updateWeatherData** is annotated with **@PutMapping("/{id}")** which maps to HTTP PUT requests and updates a specific **WeatherData** entity with the given **id**.
10. The method **deleteWeatherData** is annotated with **@DeleteMapping("/{deleteId}")** which maps to HTTP DELETE requests and deletes a specific **WeatherData** entity with the given **deleteId**.

FORECAST SERVICE

1. **Forecast findById(Long id)**: This method is used to find a single Forecast object by its id. It returns a Forecast object if found, or **null** if not found.
2. **List<Forecast> findAll()**: This method is used to find all Forecast objects stored in the repository. It returns a list of Forecast objects.
3. **Forecast save(Forecast forecast)**: This method is used to save a Forecast object to the repository. It returns the saved Forecast object.
4. **Forecast update(Forecast forecast)**: This method is used to update an existing Forecast object in the repository. It returns the updated Forecast object.
5. **void deleteById(Long id)**: This method is used to delete a Forecast object from the repository by its id. It takes the id of the Forecast object to be deleted as its argument.

LOCATIONFORECAST CONTROLLER TEST – JUNIT

* **@ExtendWith(MockitoExtension.class)**: This annotation tells JUnit to run tests using the Mockito framework.
* **@Mock**: This annotation is used to create mock objects of the specified type, in this case **LocationForecastService**.
* **@InjectMocks**: This annotation is used to instantiate the class being tested, in this case **LocationForecastController**, and automatically injects the mock objects annotated with **@Mock** into it.

The following methods are defined in this class:

* **findAllTest**: This method tests the **findAll** method in the **LocationForecastController** class. It creates a list of **LocationForecast** objects and stubs the **findAll** method of the mock **LocationForecastService** object to return the list. The method then calls the **findAll** method of the **LocationForecastController** class and compares the result to the stubbed list.
* **findByIdTest**: This method tests the **findById** method in the **LocationForecastController** class. It creates a **LocationForecast** object, sets its ID, and stubs the **findById** method of the mock **LocationForecastService** object to return the object. The method then calls the **findById** method of the **LocationForecastController** class and compares the result to the stubbed object.
* **saveTest**: This method tests the **save** method in the **LocationForecastController** class. It creates a **LocationForecast** object and stubs the **save** method of the mock **LocationForecastService** object to return the object. The method then calls the **save** method of the **LocationForecastController** class and compares the result to the stubbed object.
* **deleteTest**: This method tests the **delete** method in the **LocationForecastController** class. It creates a **LocationForecast** object, sets its ID, and stubs the **findById** method of the mock **LocationForecastService** object to return the object. The method then calls the **delete** method of the **LocationForecastController** class.