Create, Read, Update and Delete (CRUD)

- In this lesson we will…
  - Review how to access data in a table using our code
    - NamedParameterJdbcTemplate
    - MapSqlParameterSource
    - BeanPropertyRowMapper
  - Learn how to perform the other CRUD operations within Spring
  - Creating or inserting rows in a table
  - Updating existing rows
  - Deleting existing rows
  - Using PathVariables

Recall…

We will be using a few classes from

org.springframework.jdbc.core

- NamedParameterJdbcTemplate
- MapSqlParameterSource
- BeanPropertyRowMapper

- We will see how we use these three classes work together to retrieve data from the database.

NamedParameterJdbcTemplate

- Template class with a basic set of Java Database Connectivity (JDBC) operations.
- Allows use of name parameters instead of '?' placeholders.We'll use MapSqlParameterSource (next slide)
- Some methods we'll see (they are all overloaded)
  - query(…) for multiple row-returning queries
  - queryForObject(…) for retrieving a single entity
- update(…) for int-returning queries (i.e., insert, update)
- execute(…) for queries that don't return anything (i.e DDL statements)

MapSqlParameterSource

- A helper class that will hold a Map of parameters.
- Is intended for passing these parameters to an instance of NamedParameterJdbcTemplate that will ultimately execute the query.
- Notice we don't need the ' ' around strings.
- E.g.

```
27 MapSqlParameterSource namedParameters = new MapSqlParameterSource();
28
29 String query = "SELECT * FROM student WHERE name = :name AND age > :minAge";
30 namedParameters
31     .addValue("minAge", 19)
32     .addValue("name", "Smith"); // Notice the Builder pattern...
```

- Example only. Not part of the current application!

Reading from our table

```
MapSqlParameterSource namedParameters = new MapSqlParameterSource();

String query = "SELECT * FROM avengers WHERE age >:minAge";
namedParameters.addValue("minAge", 26);

// Will map a row coming in to an instance of Avenger
BeanPropertyRowMapper<Avenger> avengerMapper =
        new BeanPropertyRowMapper<Avenger>(Avenger.class);

List<Avenger> avengers =
        jdbc.query(query, namedParameters, avengerMapper);

return avengers;
```

Copy over the add_avenger.html

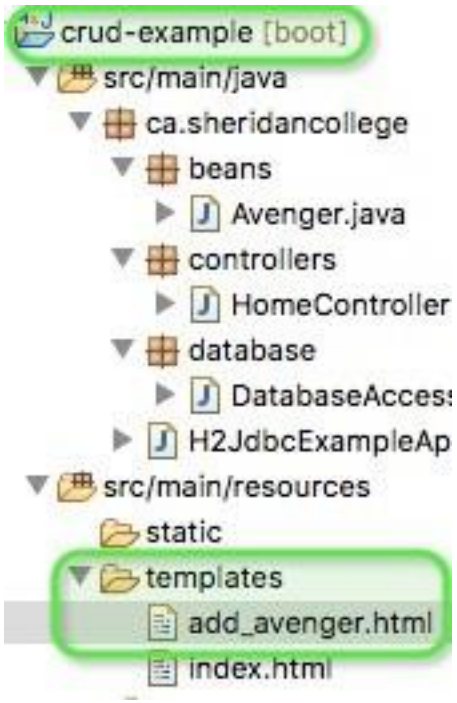
- We can also re-use most of the add_avenger form from our form-binding lesson.

add_avenger.html copied over

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4  <meta charset="UTF-8">
5  <title>Add an Avenger!</title>
6  </head>
7  <body>
8      <h1>Enter your hero's information</h1>
9
10     <form action="#" th:action="@{/addAvenger}" method= "post"
11         th:object="${avenger}">
12
13         <p>Name: <input type="text" th:field="*{name}"></p>
14         <p>Age: <input type="number" th:field="*{age}"></p>
15
16         <p>Power Source: <select th:field=*{powerSource}>
17             <option th:each="source : *{powerSources}"
18                 th:value="${source}" th:text="${source}">
19         </select></p>
20         <p><input type="submit" value="Add!"></p>
21     </form>
22 </body>
23 </html>
```

We will remove the Power Source functionality to concentrate on CRUD

add_avenger.html in crud-example




```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3  <head>
4  <meta charset="UTF-8">
5  <title>Add an Avenger!</title>
6  </head>
7  <body>
8      <h1>Enter your hero's information</h1>
9
10     <form action="#" th:action="@{/addAvenger}" method= "post"
11         th:object="${avenger}">
12
13         <p>Name: <input type="text" th:field="*{name}"></p>
14         <p>Age: <input type="number" th:field="*{age}"></p>
15
16         <p><input type="submit" value="Add!"></p>
17     </form>
18 </body>
19 </html>
```

- Make sure you are modifying the correct file! When I do these types of things (copy/paste resources between projects), I always close the previous project.
- Otherwise, you may scratch your head for a loooong time!

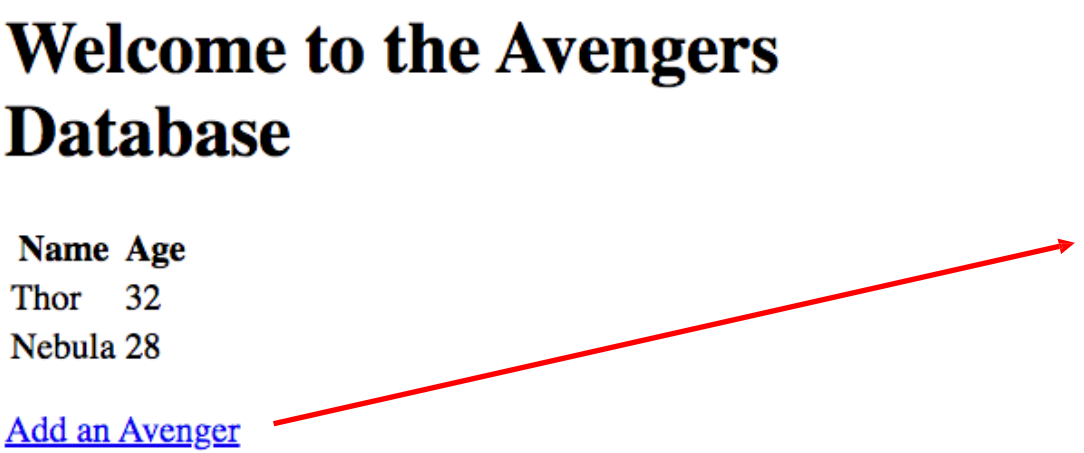
Add to DatabaseAccess class

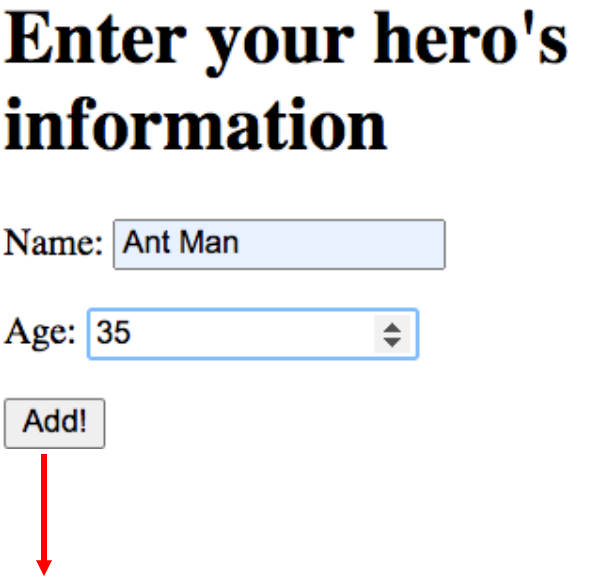
```
45    /**
46     * Adds an Avenger to the database
47     * @param avenger: the Avenger to add
48     * @return the number of rows affected; 1 - successful, 0 - not.
49     */
50    public int addAvenger(Avenger avenger) {
51
52        // create a new instance of MapSqlParameterSource for our use
53        MapSqlParameterSource namedParameters =
54                new MapSqlParameterSource();
55
56        String query =
57                "INSERT INTO avengers (name, age) VALUES (:name, :age)";
58
59
60        // add the parameters to our map
61        namedParameters
62            .addValue("name", avenger.getName())
63            .addValue("age", avenger.getAge());
64
65        int returnValue = jdbc.update(query, namedParameters);
66
67        return returnValue;
68    }
```

Add to HomeController class

```
39    /**
40     * @param model Model object supplied by Spring MVC.
41     * @return "add_avenger" the form page
42     */
43    @GetMapping("/addPage")
44    public String goToAdd(Model model) {
45
46        // Recall form binding...
47        model.addAttribute("avenger", new Avenger());
48        return "add_avenger";
49    }
50
51
52    /**
53     * @param avenger is a ModelAttribute. A special instance that
54     *        was populated with values the user entered on the form
55     * @return "redirect:/" Redirects the request to the '/' resource
56     */
57    @PostMapping("/addAvenger")
58    public String addAvenger(@ModelAttribute Avenger avenger) {
59
60        // call the addAvenger method of the DatabaseAccess class
61        int returnValue = database.addAvenger(avenger);
62
63        // we're not doing anything with this now, but we could
64        // send a message back through the model, use Elvis ...
65        System.out.println("return value is: " + returnValue);
66
67        // redirect to '/', so we don't have to add to the model...
68        return "redirect:/";
69    }
```

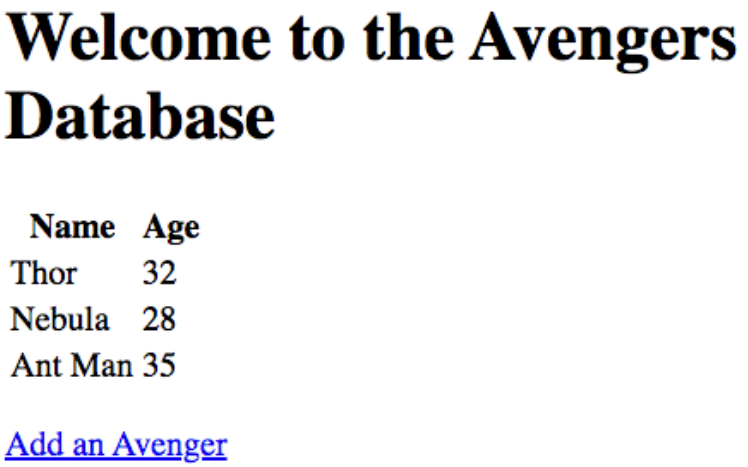
Run the application!





Not working?

- 404/405 errors are usually Mapping errors. Check your links and mappings.
- 500 errors are usually runtime errors in your code or Thymeleaf markup. Look at the console for hints!
- Make sure it isn't a silly typo 



General References

1. Notes from Prof. Jonathan Penava, Sheridan College
2. Notes from Prof. Simon Hood, Sheridan College
3. Slides from Prof. Paul Bonenfant
4. https://www.thymeleaf.org/
5. https://www.baeldung.com/
6. https://docs.spring.io/
7. https://www.baeldung.com/spring-pathvariable