



Spring Boot Project

Danyel James Harrison

22JuneEnable2

Topics

- ◆ MVP
- ◆ Product
- ◆ Documentation
- ◆ Code display
- ◆ Testing
- ◆ Lessons learned

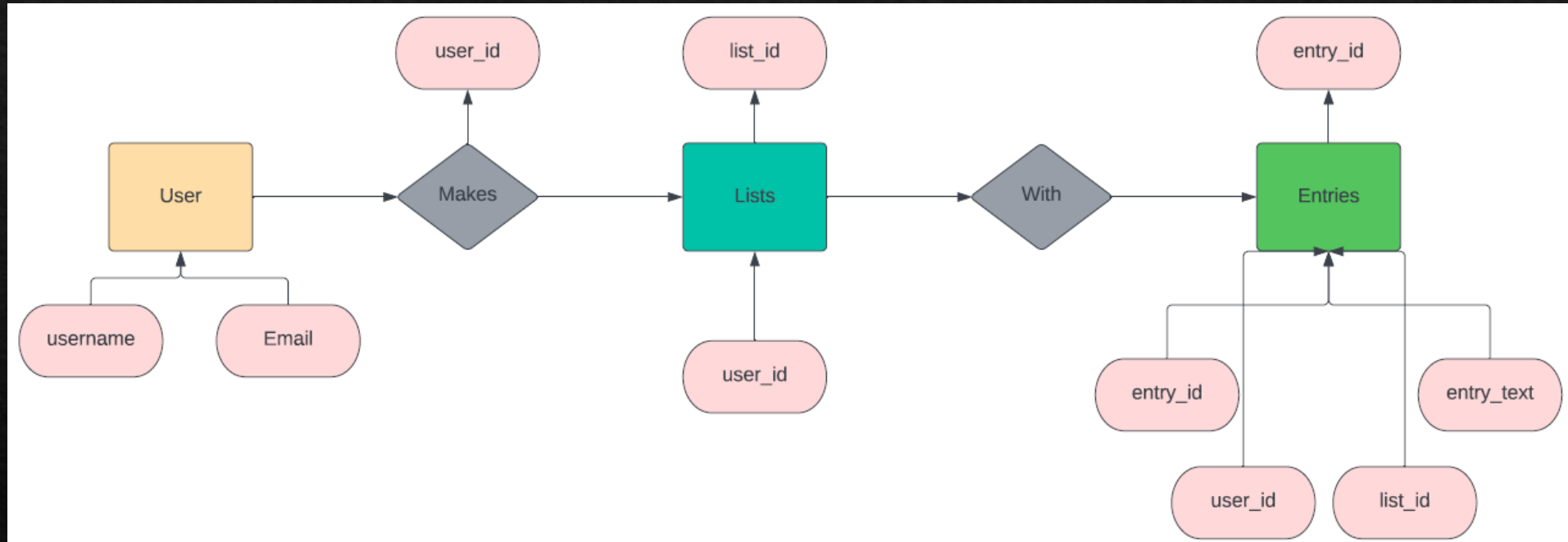
MVP

- ◊ Asked to create a basic Spring Boot Project with a HTML/CSS/JS front-end
- ◊ CRUD functionality
- ◊ Hit Testing targets

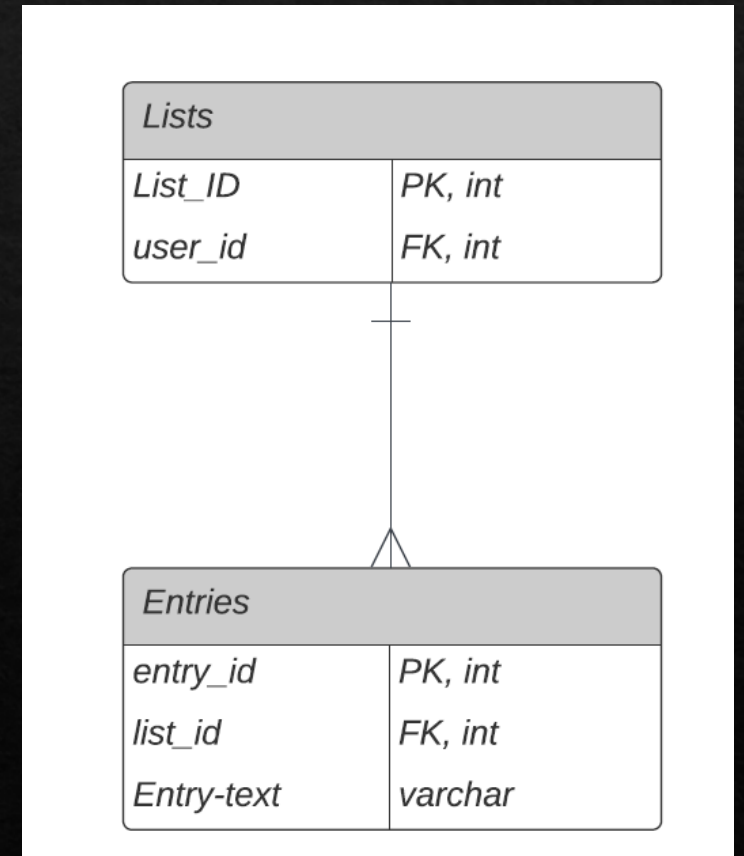
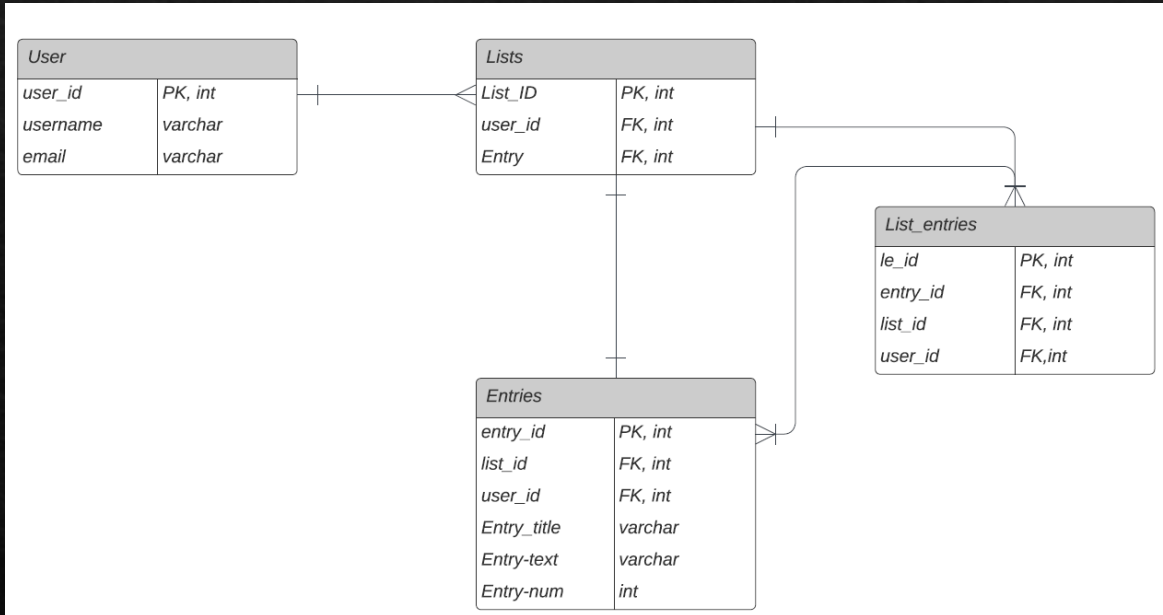
The Product

- ◇ Basic ranking application
- ◇ Select a topic, choose Top 10 options from a given list
- ◇ Store the list on a server
- ◇ Retrieve, update, or delete your list afterwards

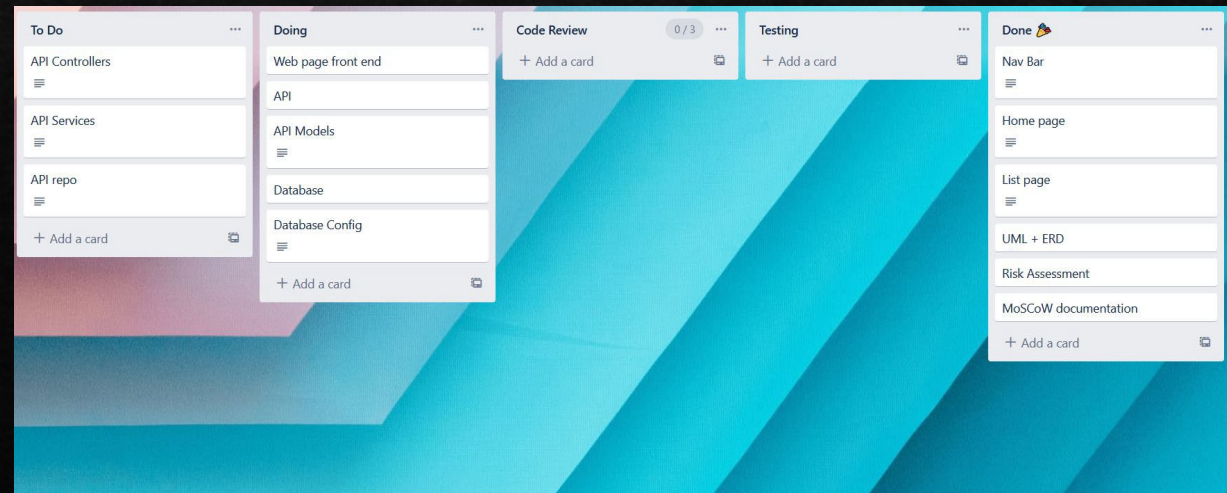
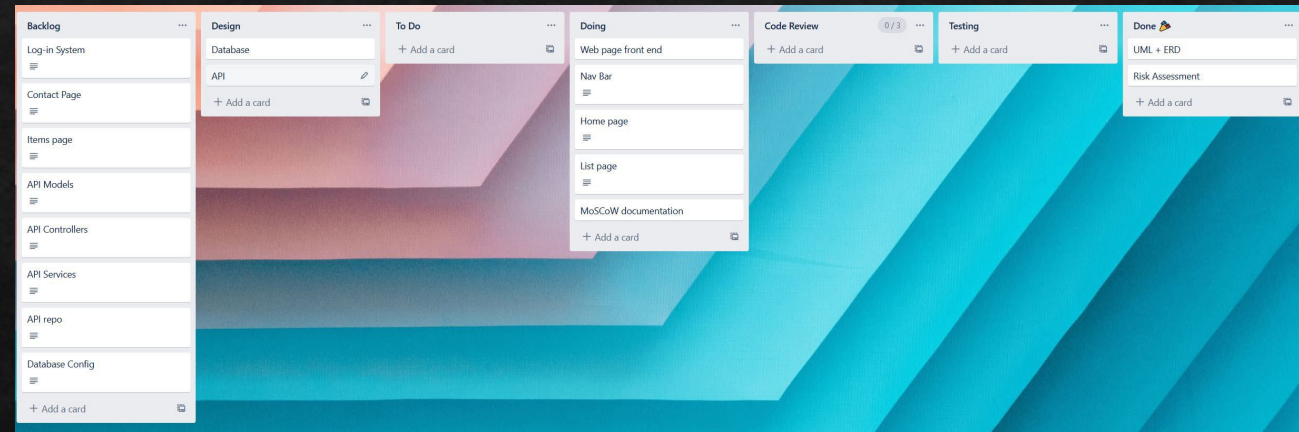
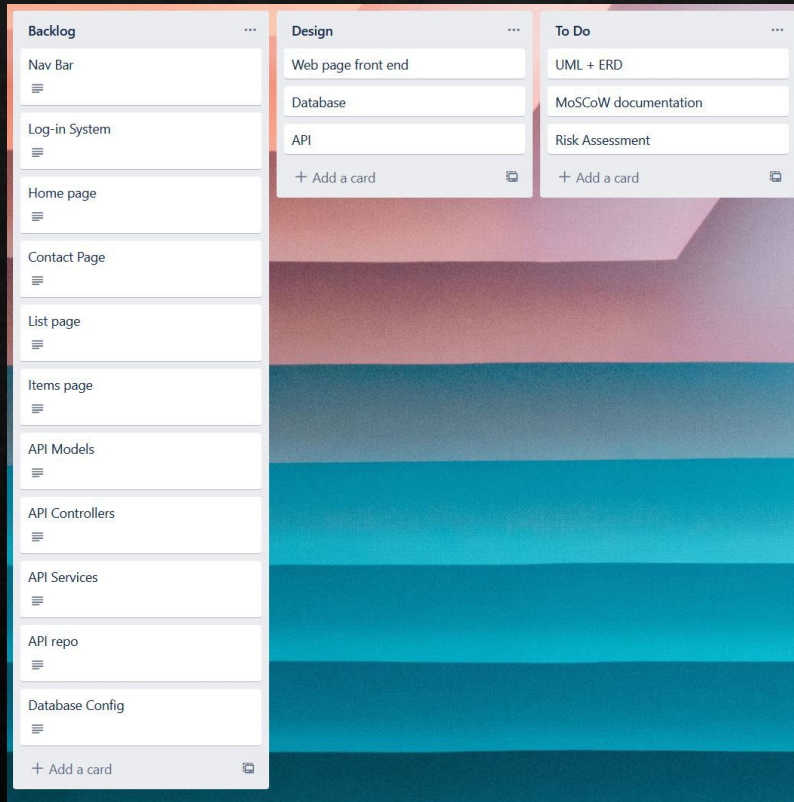
Documentation - UML



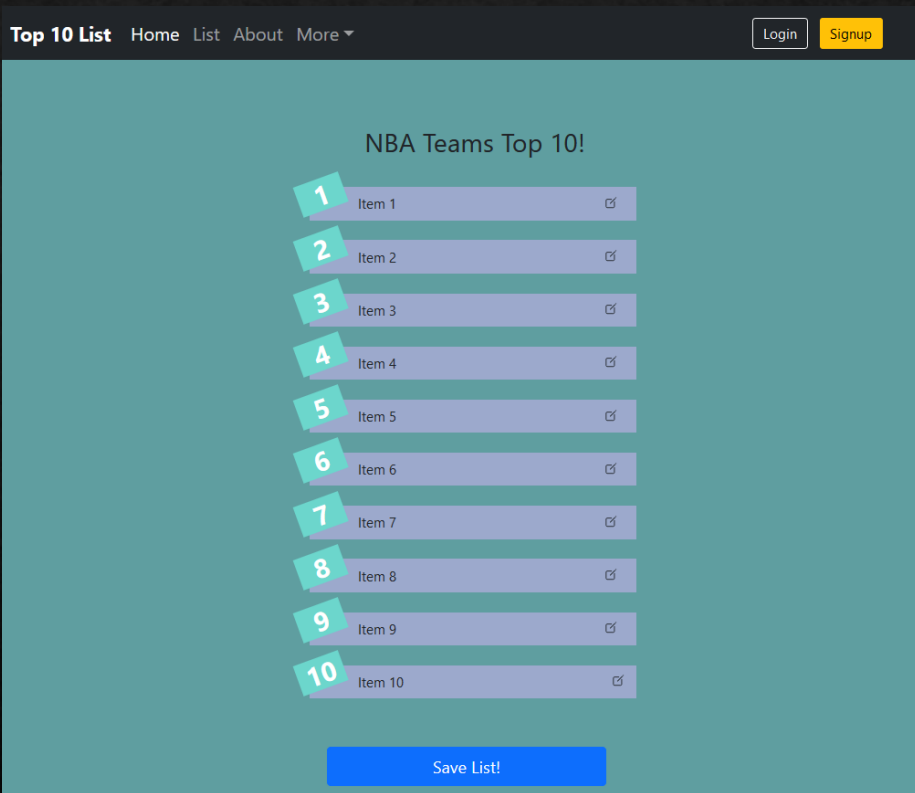
Documentation - ERD



Documentation - Trello



HTML Front-end



```
<div class="list-type5" id="listy" style="display: none;">
  <h2 id="listName">Title</h2>
  <ol class="bigList">
    <li class="dropzone" id='0' draggable="true">Item 1 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="0"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='1' draggable="true">Item 2 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="1"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='2' draggable="true">Item 3 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="2"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='3' draggable="true">Item 4 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="3"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='4' draggable="true">Item 5 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="4"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='5' draggable="true">Item 6 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="5"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='6' draggable="true">Item 7 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="6"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='7' draggable="true">Item 8 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="7"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='8' draggable="true">Item 9 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="8"><ion-icon name="create-outline"></ion-icon></button></li>
    <li class="dropzone" id='9' draggable="true">Item 10 <button class="bb" id="listyButton" data-bs-toggle="modal" type="button" value="9"><ion-icon name="create-outline"></ion-icon></button></li>
  </ol>
</div>
```


CSS Styling

```
/* CSS for list page */
```

```
.list-type5{
  width:410px;
  margin:0 auto;
  padding-top: 5%;
}

.list-type5 ol {
  list-style-type: none;
  margin: 0;
  margin-left: 1em;
  padding: 0;
  counter-reset: li-counter;
}

.list-type5 ol li{
  position: relative;
  margin-bottom: 1.5em;
  padding: 0.5em;
  background-color: #9ca9cc;
  padding-left: 58px;
}

.list-type5 a{
  text-decoration:none;
  color:black;
  font-size:15px;
  font-family: 'Raleway', sans-serif;
}

.list-type5 li:hover{
  box-shadow:inset -1em 0 #6CD6CC;
  -webkit-transition: box-shadow 0.5s; /* For Safari 3.1 to 6.0 */
  transition: box-shadow 0.5s;
}
```

```
.list-type5 ol li:before {
  position: absolute;
  top: -0.3em;
  left: -0.5em;
  width: 1.8em;
  height: 1.2em;
  font-size: 2em;
  line-height: 1.2;
  font-weight: bold;
  text-align: center;
  color: white;
  background-color: #6CD6CC;
  transform: rotate(-20deg);
  -ms-transform: rotate(-20deg);
  -webkit-transform: rotate(-20deg);
  z-index: 99;
  overflow: hidden;
  content: counter(li-counter);
  counter-increment: li-counter;
}

/*draggable list elements */

.dropzone{
  cursor:move;
  user-select: none;
}

#saveButton{
  width: 30%;
}

#createButton{
  width: 30%;
}

#createbt{
  padding-top: 2%;
  text-align: center;
}

#savebt{
  padding-top: 3%;
  text-align: center;
}
```

```
/* Home Page specifics */
```

```
#into-text{
  padding-top: 3%;
  padding-left: 2%;
  width: 40%;
  position: absolute;
  left:0%;
}

#right-text{
  padding-top: 3%;
  padding-right: 2%;
  width: 40%;
  position: absolute;
  right: 0%;
}

Footer{
  position:absolute;
  bottom: 0;
  background-color: #212529;
  padding-top: 11%;
}

#listy button{
  all: unset;
  cursor: pointer;
  margin-left: 75%;
}

#listy button:hover{
  background-color: #a3b6da;
}

#listName{
  padding-bottom: 7%;
  padding-left: 20%;
}
```

JavaScript functionality

- ◆ CRUD functionality within JS
- ◆ Button Functions
- ◆ Combining multiple functions into one action

```
//create new list function

const createNewList = () => {
  let titleInput = document.getElementById("listTitle").value;
  let cList = document.querySelector("#listy").value;
  console.log(cList);

  const obj = {
    "listName": titleInput
  }

  fetch("http://localhost:8080/ListModel" , {
    method: "POST",
    headers: {
      "Content-Type": "application/json"
    },
    body: JSON.stringify(obj)
  })
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.err(err))
}

// adding on click functionality for createNewList
const makeList = document.querySelector("#saveButton");
makeList.addEventListener("click", (event) => {
  event.preventDefault();
  createNewList();
  listEntryCreate();
});
```


Java API back-end

- ◆ Multiple Entities
- ◆ Controllers/Services/Repos
- ◆ Examples of Entity classes, using Spring-Boot Annotations

```
@Entity
@Table(name = "Lists")
public class ListModel {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "list_id")
    private long ListId;

    @Column(name = "list_name")
    private String listName;

    @OneToMany(mappedBy = "listModel", fetch = FetchType.LAZY, orphanRemoval = true)
    @OnDelete(action = OnDeleteAction.CASCADE)
    private List<ListEntries> listEntries;
```

```
@Entity
public class ListEntries {

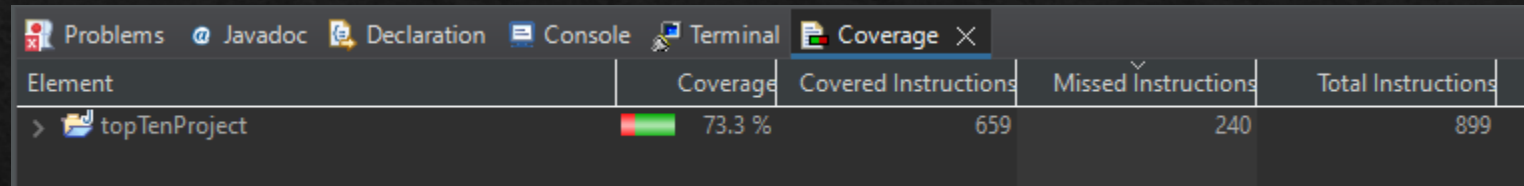
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long entryId;

    @ManyToOne(targetEntity = ListModel.class, fetch = FetchType.LAZY,
    @JoinColumn(name = "fk_list_id")
    private ListModel listModel;

    @Column(name = "list_entry", nullable = false)
    private String list_entry;
```

Testing

- ◇ 73.3% testing coverage
- ◇ Attempting TDD



The screenshot shows the 'Coverage' tab in an IDE. It displays a table with the following data:

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
> topTenProject	73.3 %	659	240	899

```
@Test
public void readListByIdTest() {
    Long id = 1L;
    ListModel testList = new ListModel(1L, "hello", null);
    Optional<ListModel> testOpt = Optional.of(testList);
    ListModel testModel = testOpt.get();

    Mockito.when(repo.findById(id)).thenReturn(testOpt);
    assertEquals(testModel, this.service.getListById(id).get());
    Mockito.verify(repo, Mockito.times(1)).findById(id);
}
```

```
@Test
public void constructorTest() {
    ListModel list0 = new ListModel();
    ListModel list1 = new ListModel("test list 1");
    ListModel list2 = new ListModel(2L, "test list 2");
    ListModel list3 = new ListModel(3L, "test list 3", null);

    assertTrue(list0 instanceof ListModel == true);
    assertTrue(list1 instanceof ListModel == true);
    assertTrue(list2 instanceof ListModel == true);
    assertTrue(list3 instanceof ListModel == true);
}
```


What I learned/Conclusion

- ◇ Hit MVP as simply as possible
- ◇ Test as you go
- ◇ Focus on functionality, before appearance