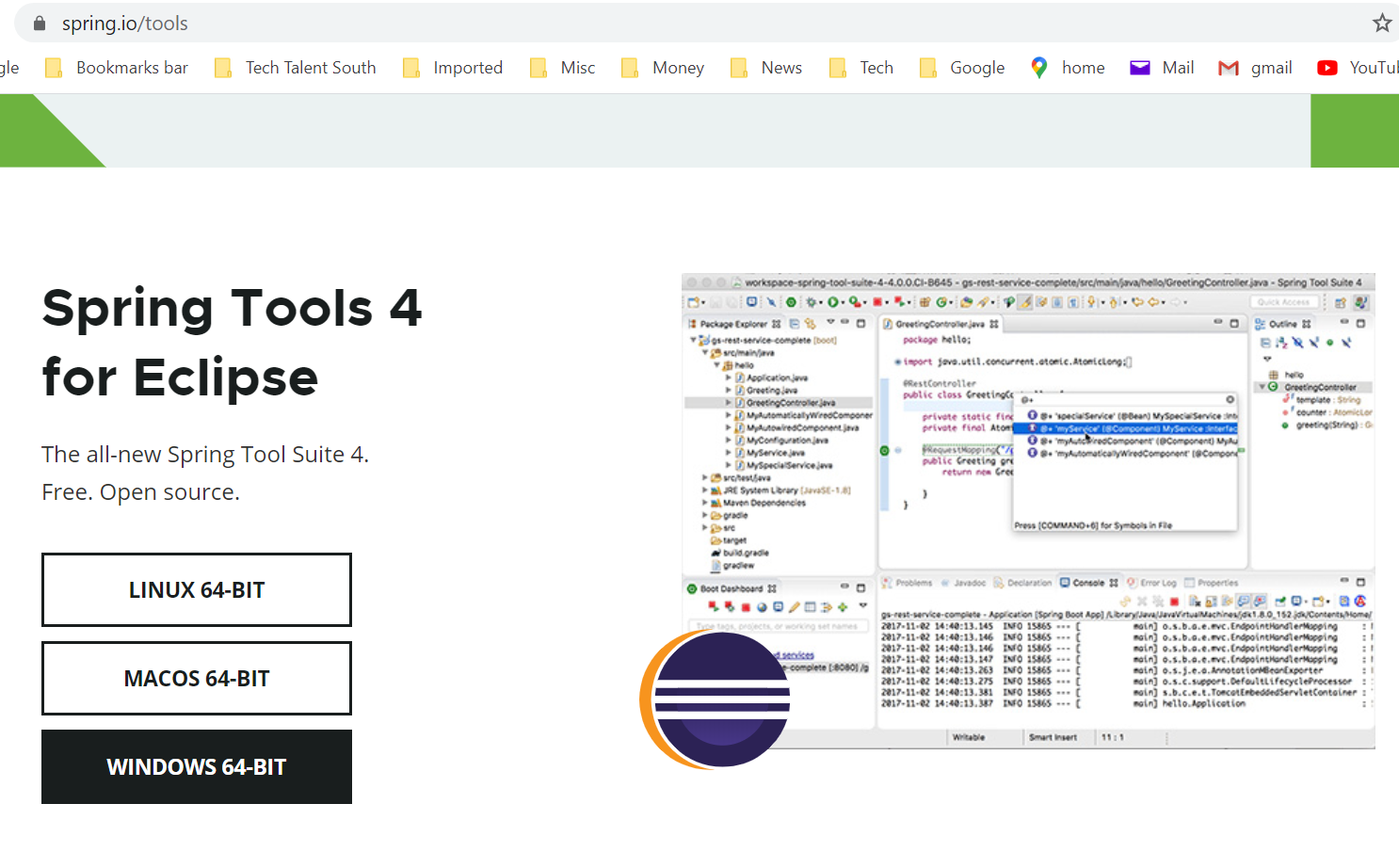
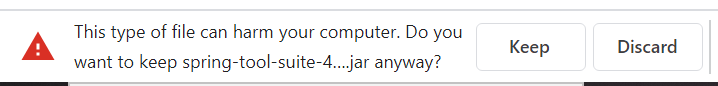
Day 11 Spring Boot Notes

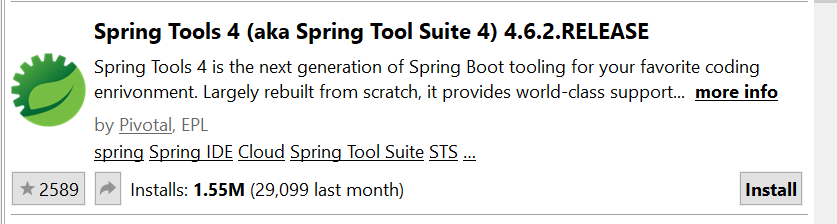
**Setup Options**

****

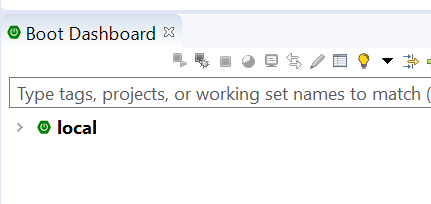


This just downloads a jar that you extract then run ??? to run the app. Talk to Jacob

**Eclipse martketplace help -> Eclipse Marketplace**



Should see boot dashoard after restarting eclipse

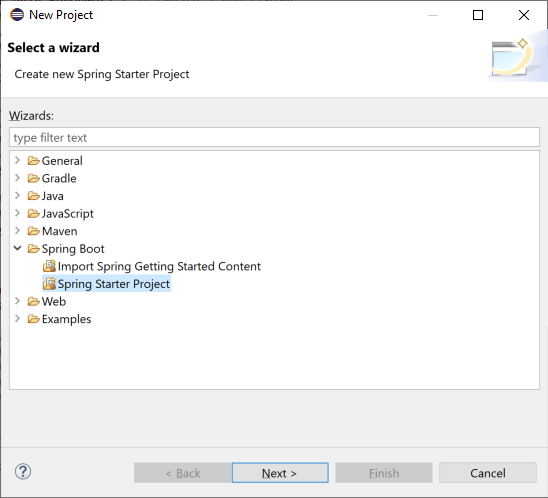


Have Ben walk through his fix.

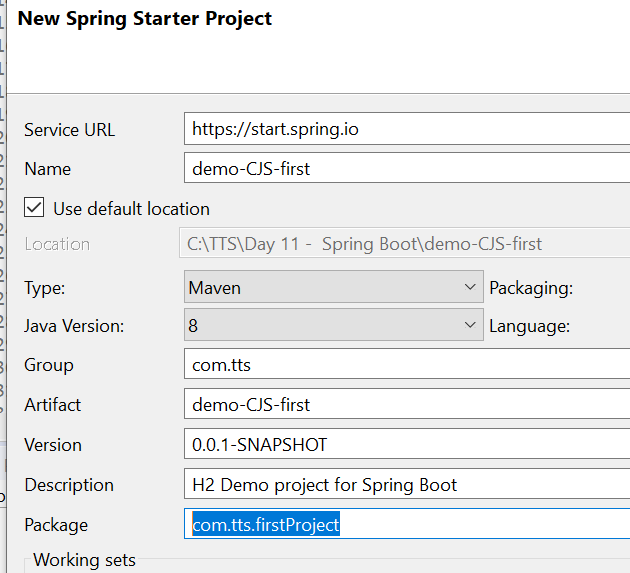
Create C:\TTS\Day 11 - Spring Boot

(Show slides 1-9 and do the following)

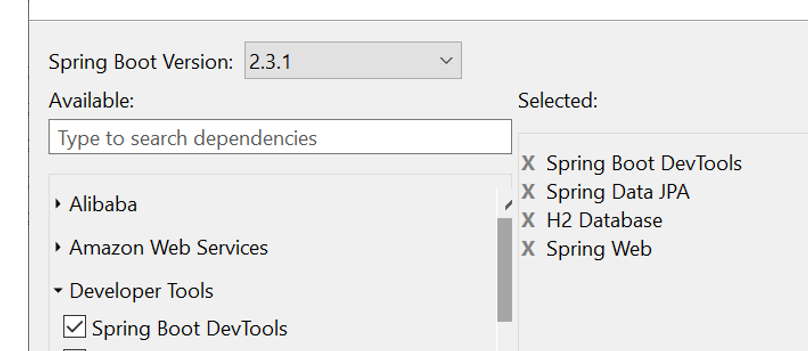
Create a new project File -> New -> Sprint Starter Project



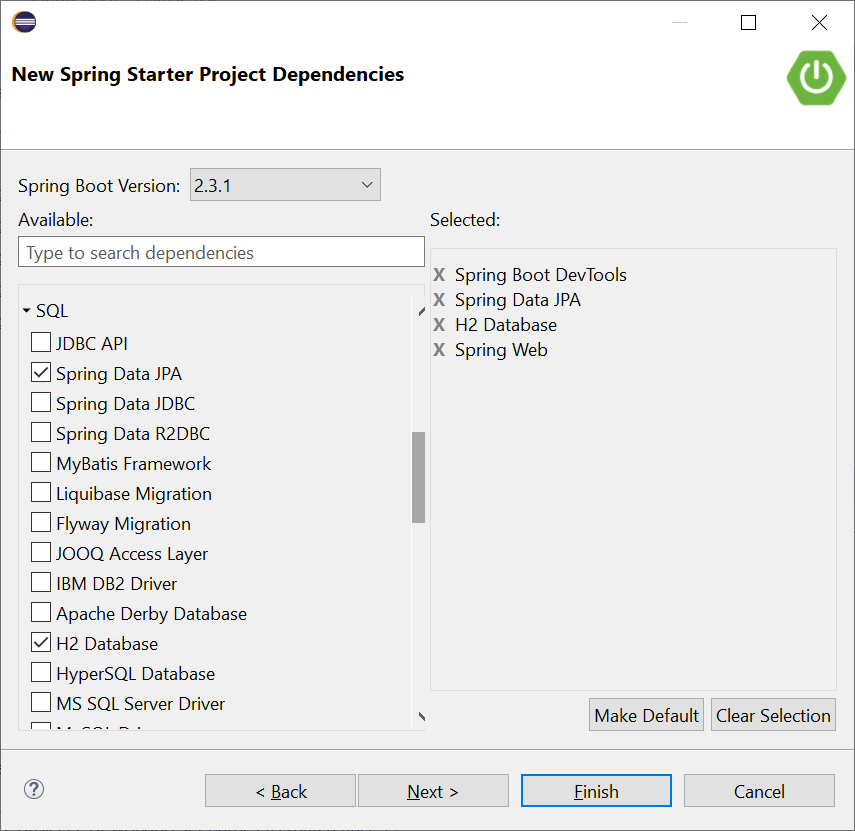
Fill out project details

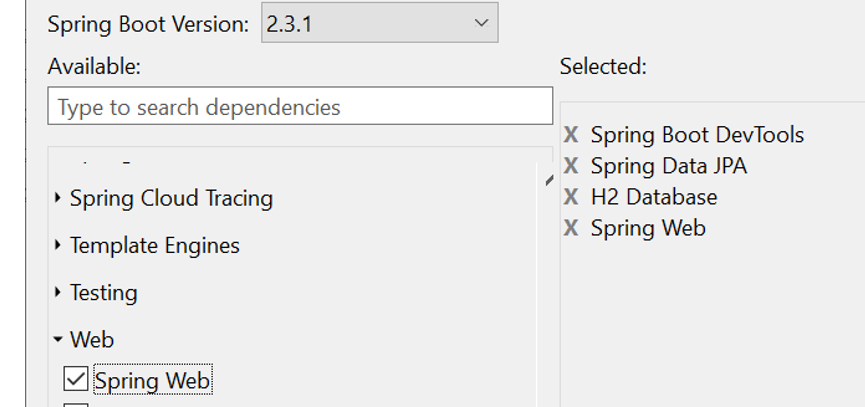


Hit Next



Pick dependencies





Select Finish

Review contents of spring boot app

Look at pom.xml

SpringBootFrstApplication

**Interact with Browser**

create a new folder called "com.tts.firstProject.controllers".

create MainController and explain what it does.

create **public** **class** MainController

@RestController

**public** **class** MainController {

@RequestMapping("/")

**public** String home() {

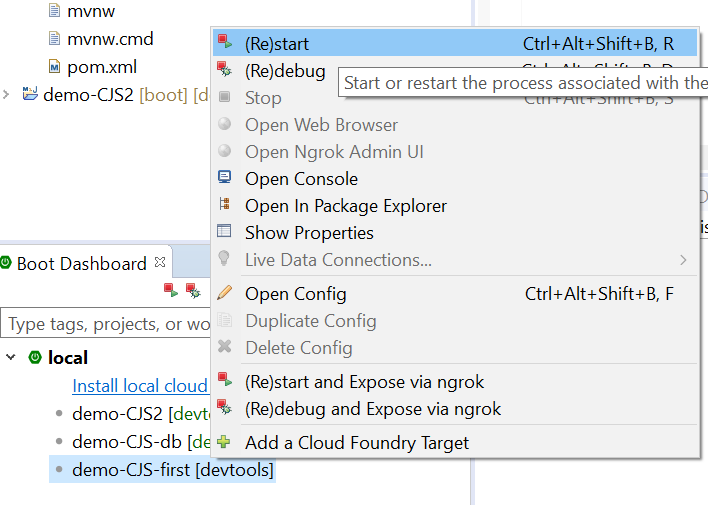
**return** "Hello World!";

}

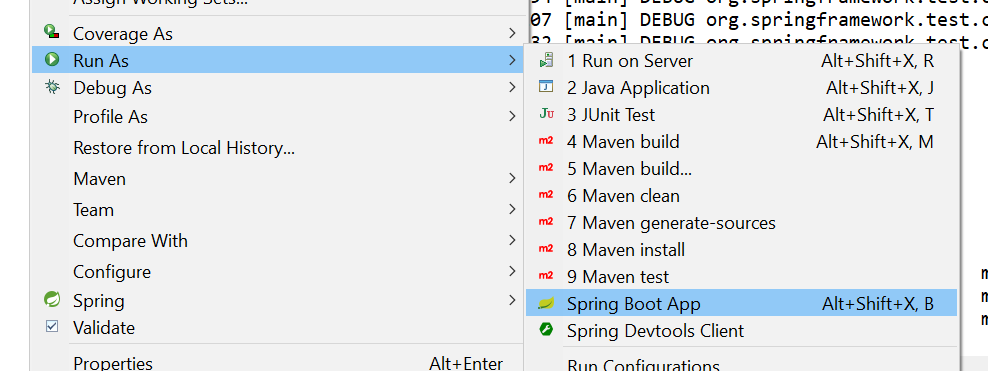
}

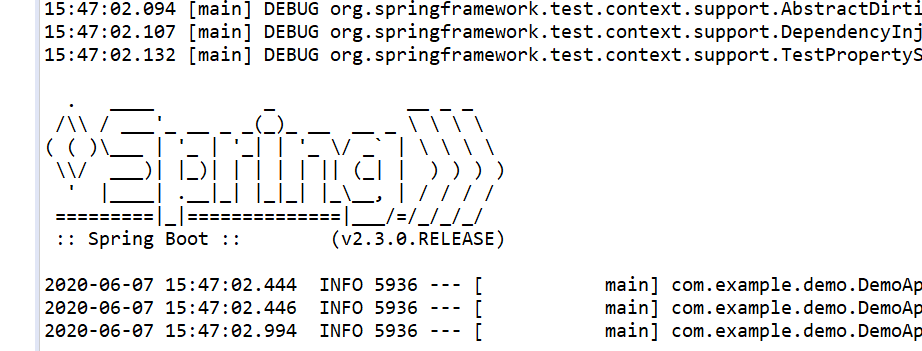
(Show and read slide 15)

Start the application



Or start it this way





Try <http://localhost:8080/> in a browser

Add another request mapping

@RequestMapping("/cjs")

**public** String chuck() {

**return** "Hello Chuck!";

}

@RequestMapping("/pi")

**public** String pi() {

**return** "<h1>I like " + Math.***PI*** + "</h1>" +

"<br><a href='/fudge'> go to fudge</a>";

}

@RequestMapping("/fudge")

**public** String fudge() {

**return** "<h1>I like fudge </h1>" +

"<br><a href='/pi'> go to pi </a>";

}

Try <http://localhost:8080/>pi in a browser

**Start a new project for database H2 (Slide 19) --------------------------------------------------------------------------------**

In STS, select File->New -> Spring Starter Project

Name: h2Explore

Group: com.tts

Artifact: h2Explore

Description: Exploring the h2 database

Package name: com.tts.h2Explore

Select these 4



To get database to run use the following in application.properties

spring.h2.console.enabled=true

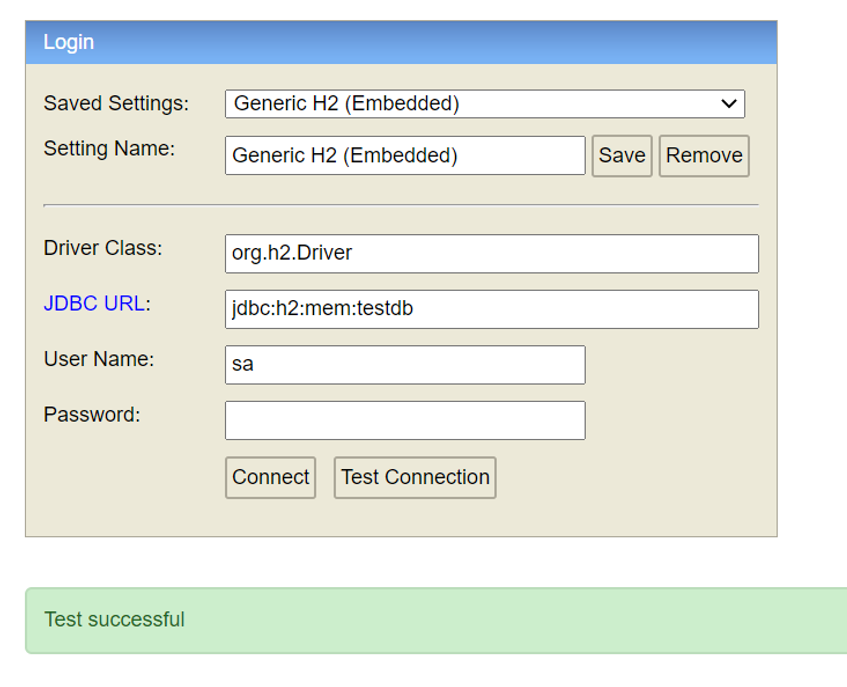
spring.h2.console.path=/console

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.url=jdbc:h2:mem:testdb

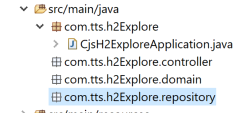
Start the application

And enter this in browser >> localhost:8080/console



Show slide 22, create 3 new directories

Controller, domain, repository



Slide 23 create customer class in domain folder, add instance variables

**public** **class** Customer {

**private** Long id;

**private** String firstName;

**private** String lastName;

}

**Slides 23 – 33 all set up the customer class**

**package** com.tts.h2Explore.domain;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

@Entity

**public** **class** Customer {

@Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

**private** Long id;

**private** String firstName;

**private** String lastName;

**public** Customer(String firstName, String lastName) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

}

**public** Customer() {

**super**();

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

@Override

**public** String toString() {

**return** "Customer [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + "]";

}

}

Go to console and click “Connect” button > <http://localhost:8080/console>

(Show slide 34 – 35)

Type this into SQL tool

*insert into customer(id,first\_name, last\_name) values (1,'John','Smith');*

*insert into customer(id,first\_name, last\_name) values (2,'Joe','Smith');*

*select \* from customer;*

(Slide 38-40 Create repository interface)

**package** com.tts.h2Explore.repository;

**import** java.util.List;

**import** org.springframework.data.repository.CrudRepository;

**import** com.tts.h2Explore.domain.Customer;

**public** **interface** CustomerRepository **extends** CrudRepository<Customer, Long>{

List<Customer> findByLastName(String lastName);

}

(Slide 41 Update H2ExploreApplication.java)

**package** com.tts.h2Explore;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.Bean;

**import** com.tts.h2Explore.domain.Customer;

**import** com.tts.h2Explore.repository.CustomerRepository;

@SpringBootApplication

**public** **class** CjsH2ExploreApplication {

**private** **static** **final** Logger ***log*** = LoggerFactory.*getLogger*(CjsH2ExploreApplication.**class**);

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(CjsH2ExploreApplication.**class**, args);

}

@Bean

**public** CommandLineRunner demo(CustomerRepository repository) {

**return** (args) -> {

// save a couple of customers

repository.save(**new** Customer("Michael", "Smith"));

repository.save(**new** Customer("Aaron", "Moon"));

repository.save(**new** Customer("Kim", "Lassiter"));

repository.save(**new** Customer("Joan", "Daniels"));

repository.save(**new** Customer("Eric", "Patterson"));

// read all customers

***log***.info("Customers found with findAll():");

***log***.info("-------------------------------");

**for** (Customer customer : repository.findAll()) {

***log***.info(customer.toString());

}

***log***.info("");

// read an individual customer by ID

repository.findById(1L)

.ifPresent(customer -> {

***log***.info("Customer found with findById(1L):");

***log***.info("--------------------------------");

***log***.info(customer.toString());

***log***.info("");

});

// read customers by last name

***log***.info("Customer found with findByLastName('Patterson'):");

***log***.info("--------------------------------------------");

repository.findByLastName("Patterson").forEach(bauer -> {

***log***.info(bauer.toString());

});

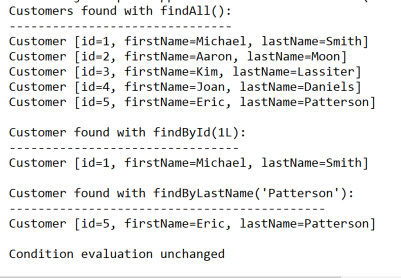
***log***.info("");

};

}

}

Should see this in the console



**Show second slide deck Spring Boot: Adding and Accessing Data---------------------------------------------------------**

(Show slides 1-3)

Create a new project

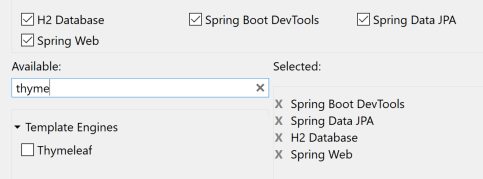
Name: SubscriberList

Group: com.tts

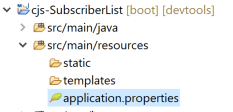
Artifact: SubscriberList

Description: Subscriber Sign up using JPA, H2, and Thymeleaf

Package name: com.tts.subscriberlist



Edit application.properties



*#allows us to interact with the database*

*spring.h2.console.enabled=true*

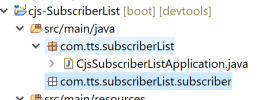
*spring.h2.console.path=/console*

*#allows us to bring in data from an external sql resource to add to the database.*

*spring.datasource.platform=h2*

*spring.datasource.url=jdbc:h2:mem:testdb*

Create subscriber directory (Slide 9)

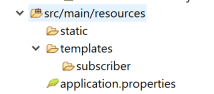


Create Subscriber class (Copy from slide 18)

Create SubscriberController (Copy slide 22)

Create HTML templates (See slide 23)

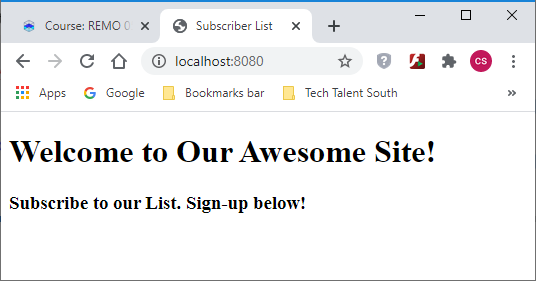
Create folder subscriber



Create index.html

Copy html from slide 25

**Start the server / test connection Slide 26-27**



**Setting up the template (Slide 28)**

Update index.html with slide 29 html code

<html xmlns:th=*"http://www.thymeleaf.org"*>

Add index.html body from slide 30 (Explain this)

*<p>Please use the form below to sign up as a subscriber:</p>*

*<form action="#" th:action="@{/}" th:object="${subscriber}"*

*method="post">*

*<table>*

*<tr>*

*<td>First Name:</td>*

*<td><input type="text" th:field="\*{firstName}" /></td>*

*</tr>*

*<tr>*

*<td>Last Name:</td>*

*<td><input type="text" th:field="\*{lastName}" /></td>*

*</tr>*

*<tr>*

*<td>Username:</td>*

*<td><input type="text" th:field="\*{userName}" /></td>*

*</tr>*

*<tr>*

*<td><button type="submit">Submit Your Name</button></td>*

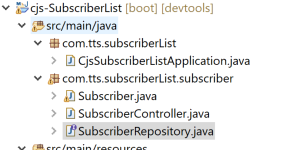
*</tr>*

*</table>*

*</form>*

Add a repository (Slide 31)

SubscriberRepository



(Copy from slide 32 to new repository)

**Add repository to SubscriberController (See slide 33)**

@Autowired

**private** SubscriberRepository subscriberRepository;

**Add Post method to SubscriberController (See slide 34)**

@Controller

**public** **class** SubscriberController {

@Autowired

**private** SubscriberRepository subscriberRepository;

@GetMapping

**public** String index(Subscriber subscriber) {

**return** "subscriber/index";

}

@PostMapping(value = "/")

**public** String addNewSubscriber(Subscriber subscriber, Model model) {

subscriberRepository.save(**new** Subscriber(subscriber.getFirstName(),

subscriber.getLastName(), subscriber.getUserName(), subscriber.getSignedUp()));

model.addAttribute("firstName", subscriber.getFirstName());

model.addAttribute("lastName", subscriber.getLastName());

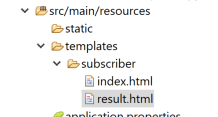
model.addAttribute("userName", subscriber.getUserName());

**return** "subscriber/result";

} // point out that subscriber/result points to result.html

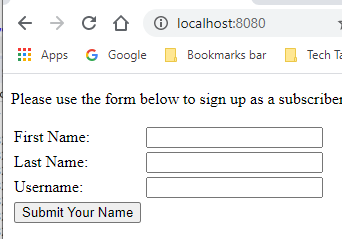
}

**Create result template copy from slide 36**



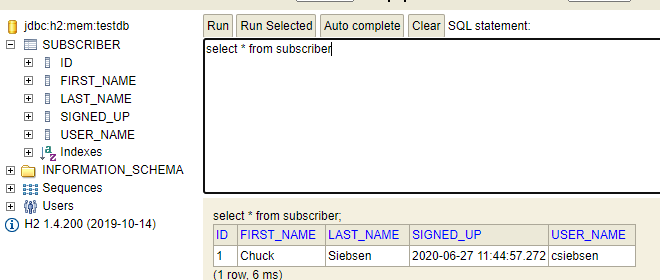


Start the server and access localhost:8080 (Slide 37)



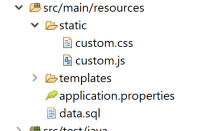
Submit the form Slides 38-39

Connect to database to view new entry <http://localhost:8080/console>



**Review homework**

**Adding css?**



**Time permitting download chinook.db and test**