**LearnToday**

**Spring Boot with REST Specification Document**

**Purpose of this document**

The purpose of this document is to specify a REST API for the LearnToday application.

**Definitions & Acronyms**

|  |  |
| --- | --- |
| **Definition / Acronym** | **Description** |
| REST | Representational State Transfer |

**Project Overview**

LearnToday is an online training application which allows students to view all the courses and allows them to enroll for the courses. It also features trainer registration and signin actions.

Scope

* Develop LearnToday SpringBoot application using Spring REST.
* Test the actions using Postman.

Intended Audience

* Application Architect
* Project Manager
* Test Manager
* Development Team
* Testing Team

Hardware and Software Requirement

**Hardware Requirement:**

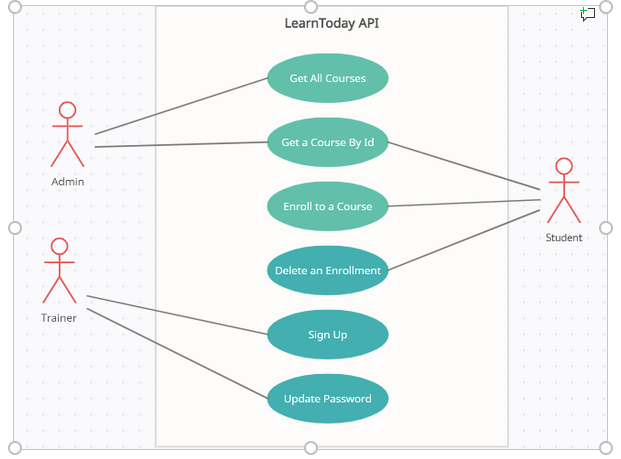
1. Developer Desktop PC with 8GB RAM

**Software Requirement:**

1. Eclipse/ Spring Tool Suite/ IntelliJ etc
2. Maven
3. JDK  8 or later

4. Postman tool

**Use Case Diagram**



**Create Spring Boot Application with REST in Eclipse**

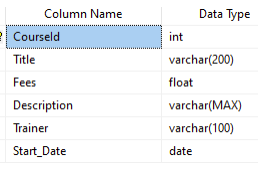
Create a Spring Boot Application using Spring initializer, <https://start.spring.io> or from IDE

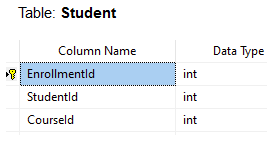
**Table Specification**

**The sample Database and Table will be given for your reference:**

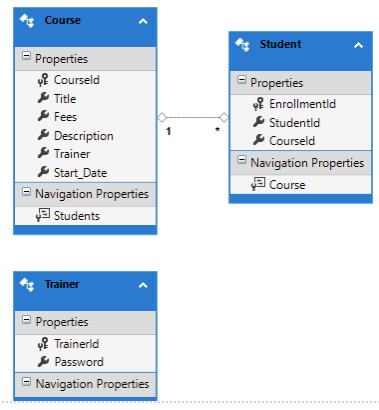
Database Name: **LearnTodayRESTAPIDb**

 Table: **Course**





**ER Diagram**

https://c1h-word-edit-15.cdn.office.net:443/we/s/hA3596C17DAD9A003_resources/1033/progress.gif 

**Design Considerations**

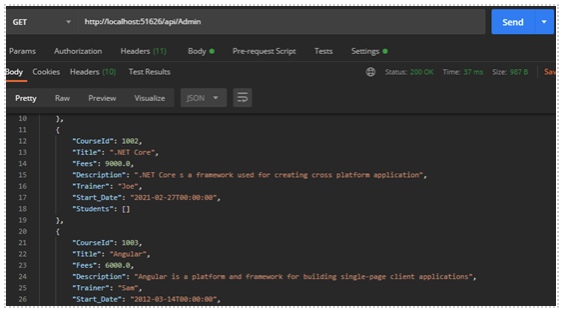
* Create the Database and the set of tables as per the table specifications.
* Create the model classes as per the requirement.
* Use Spring Data JPA for data access
* Business logic should be implemented in the Service classes.
* Controller classes should hold the mapping as per the given specification.
* Use DAO layer to interact with the Database.

**REST Controllers and Postman Test Specification**

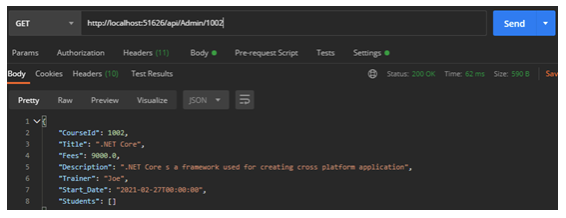
Create a new AdminController which will contain 2 action methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Type** | **Method Name** | **Parameter** | **Return Type** | **Description** |
| Method: GET | getAllCourses | No Parameter | ResponseEntity<Object> | This method will return all the courses. |
| Method: GET | getCourseById | int id | ResponseEntity<Object> | Here the course will be searched by course id. If searched course is found it will return HttpStatus.OK and the searched course data. Otherwise it will return HttpStatus.NOT\_FOUND with a message "Searched Data not Found". |

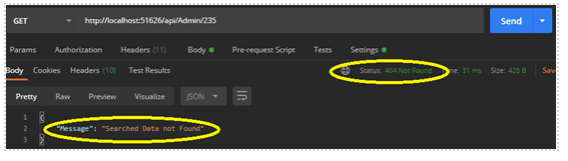
**Below is POSTMAN screenshot for getAllCourses action:**

https://c1h-word-edit-15.cdn.office.net:443/we/s/hA3596C17DAD9A003_resources/1033/progress.gif 

**Below is POSTMAN screenshot for getCourseById action:**



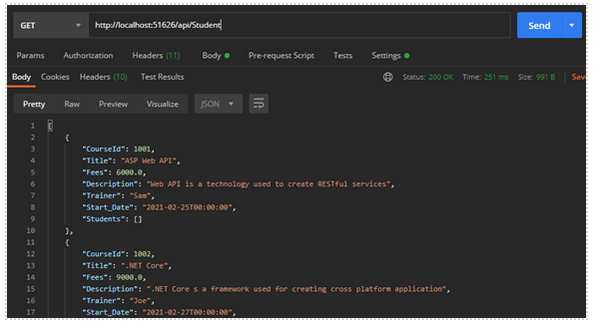
**If Course is not found by searched Course Id below is the Postman result:**

https://c1h-word-edit-15.cdn.office.net:443/we/s/hA3596C17DAD9A003_resources/1033/progress.gif

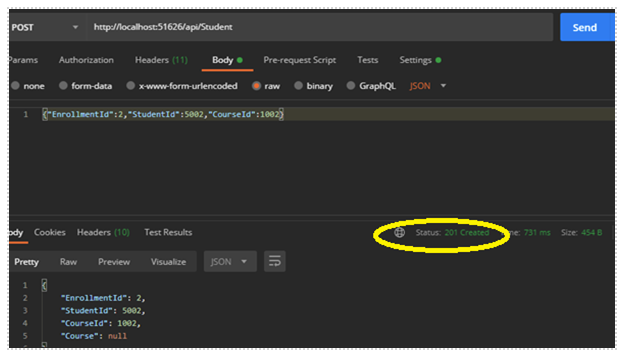
 Create a new StudentController with the below 3 action methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Type** | **Method Name** | **Parameter** | **Return Type** | **Description** |
| Method: GET | getAllCourses | No Parameter | ResponseEntity<Object> | This method will return all the courses in ascending order by start date |
| Method: Post | postStudent | Student model | ResponseEntity<Object> | This method is used to method to enroll student for the course. Input student information thru the request body. Use try-catch block. Return HttpStatus.CREATED with the model object if enrollment is successful; otherwise return HttpStatus.BAD\_REQUEST with exception object in catch block. |
| Method:@DELETE | deleteStudentEnrollment | int id | ResponseEntity<Object> | This method is used to remove the student information by the enrollment id Use a try catch block. If deletion successful then return HttpStatus.OK. If enrollement information is not found by searched id then return HttpStatus.NOT\_FOUND with a message "No enrollement information found". Under catch block return HttpStatus.BAD\_REQUEST with exception object if some exceptions occur in the program. |

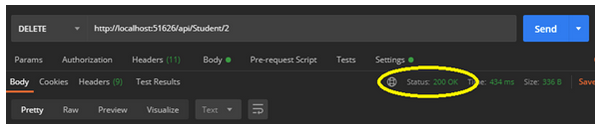
**Below is POSTMAN screenshot for getAllCourses action:**

https://c1h-word-edit-15.cdn.office.net:443/we/s/hA3596C17DAD9A003_resources/1033/progress.gif 

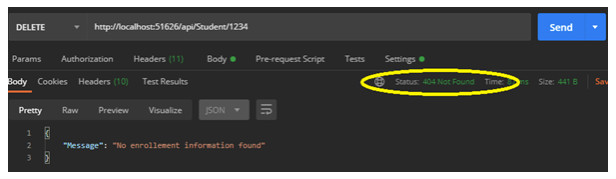
**Below is POSTMAN screenshot for postStudent action:**



**Below is POSTMAN screenshot for deleteStudentEnrollment action:**



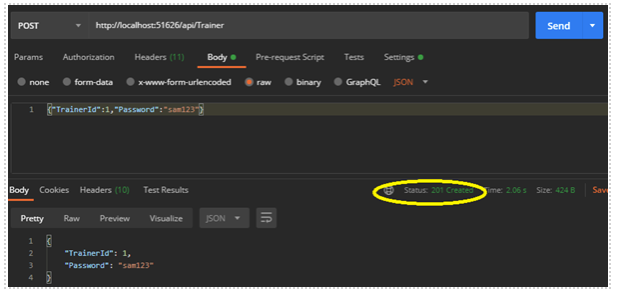
**Below is POSTMAN screenshot for deleteStudentEnrollment action if searched    enrolment id is not found:**

https://c1h-word-edit-15.cdn.office.net:443/we/s/hA3596C17DAD9A003_resources/1033/progress.gif

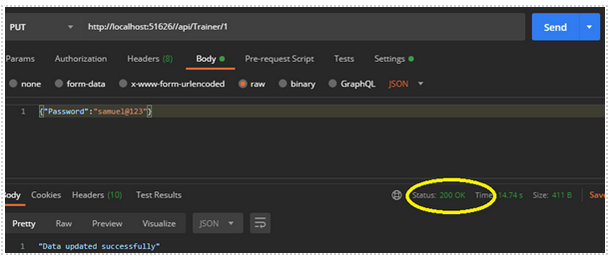
Create a new TrainerController with below 2 action methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Type** | **Method Name** | **Parameter** | **Return Type** | **Description** |
| Method:Post | trainerSignUp | Trainer model | ResponseEntity<Object> | This method is used to register new trainer. Input trainer information by using @RequestBody annotation. Use try-catch block. Return HttpStatus.CREATED with the model object if sign up is successful; otherwise return HttpStatus.BAD\_REQUEST with exception object in catch block. |
| Method: PUT | updatePassword | int id, Trainer model | ResponseEntity<Object> | This method will update trainer password by accepting trainer id from url and Trainer object from message body. If password update is successul then provide HttpStatus.OK response output with a message “Data updated successfully”; otherwise provide HttpStatus.NOT\_FOUND response output with message "Searched Data not Found". Use try catch block. If there is any exception raised catch it and provide an HttpStatus.BAD\_REQUEST response along with exception object. |

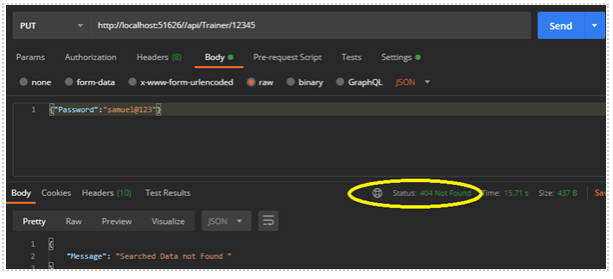
**Below is POSTMAN screenshot for trainerSignUp action:**



**Below is POSTMAN screenshot for updatePassword action:**



**Below is POSTMAN screenshot for updatePassword action if trainer id not found:**



https://c1h-word-edit-15.cdn.office.net:443/we/s/hA3596C17DAD9A003_resources/1033/progress.gif

**Note: Integrate JWT for Security and integrate Swagger**