# Application.java

* package com.cognizant.fsd.restfulapiassignment2;
* import org.springframework.boot.SpringApplication;
* import org.springframework.boot.autoconfigure.SpringBootApplication;
* /\*\*
* \*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* @SpringBootApplication
* public class Application {
* public static void main(String[] args) {
* SpringApplication.run(Application.class, args);
* }
* }

# BookStoreController.java

* package com.cognizant.fsd.restfulapiassignment2.controller;
* import java.net.URI;
* import java.util.ArrayList;
* import java.util.List;
* import java.util.Optional;
* import javax.validation.Valid;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.hateoas.Resource;
* import org.springframework.http.HttpStatus;
* import org.springframework.http.ResponseEntity;
* import org.springframework.web.bind.annotation.CrossOrigin;
* import org.springframework.web.bind.annotation.DeleteMapping;
* import org.springframework.web.bind.annotation.GetMapping;
* import org.springframework.web.bind.annotation.PathVariable;
* import org.springframework.web.bind.annotation.PostMapping;
* import org.springframework.web.bind.annotation.PutMapping;
* import org.springframework.web.bind.annotation.RequestBody;
* import org.springframework.web.bind.annotation.RequestMapping;
* import org.springframework.web.bind.annotation.RestController;
* import org.springframework.web.servlet.support.ServletUriComponentsBuilder;
* import org.springframework.web.util.UriComponentsBuilder;
* import com.cognizant.fsd.restfulapiassignment1.model.Book;
* import com.cognizant.fsd.restfulapiassignment1.model.BookNotFoundException;
* import com.cognizant.fsd.restfulapiassignment1.service.BookStoreService;
* /\*\*
* \*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* @RestController
* @RequestMapping(value="/bookstore/books")
* @CrossOrigin(origins = "http://localhost:3000")
* public class BookStoreController {
* @Autowired
* private BookStoreService bookStoreService;
* //1.Is a new book added?
* //2.What if the book is null for add book service?
* @PostMapping
* public ResponseEntity<?> addBook(@Valid @RequestBody Book book,UriComponentsBuilder ucBuilder) {
* Book savedBook = bookStoreService.addBook(book);
* URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{bookId}")
* .buildAndExpand(savedBook.getBookId()).toUri();
* return ResponseEntity.created(location).build();
* // return new ResponseEntity<>("Successfully Created",HttpStatus.CREATED);
* }
* //3.Delete book operation with existing id.
* //4.Delete book operation with non-exiting id
* @DeleteMapping("/{bookId}")
* public ResponseEntity<?> deleteBook(@PathVariable String bookId) {
* System.out.println("Delate ::"+bookId);
* boolean isDeleted = bookStoreService.deleteBook(Long.parseLong(bookId));
* if(isDeleted) {
* return new ResponseEntity<>("Successfully Deleted ",HttpStatus.OK);
* }else {
* return new ResponseEntity<>("Failure! Record is not Exist in DB, Not Deleted Successfully. ",HttpStatus.EXPECTATION\_FAILED);
* }
* }
* //7.Get a book operation with existing id
* //8.Get a book operation with non existing id
* @GetMapping("/{bookId}")
* public Resource<Book> getBookById(@PathVariable String bookId) {
* Optional<Book> bookDetails = bookStoreService.getBookById(Long.parseLong(bookId));
* if(!bookDetails.isPresent()) {
* throw new BookNotFoundException("bookId: "+bookId);
* }
* Resource<Book> resource = new Resource<Book>(bookDetails.get());
* return resource;
* }
* // 5.Update book with existing id
* // 6.Update book operation with non existing id.
* @PutMapping("/{bookId}")
* public Resource<Book> editBook(@RequestBody Book book,@PathVariable String bookId) {
* System.out.println("Book Id "+bookId);
* Book updatedBook = bookStoreService.editBook(book, Long.parseLong(bookId));
* System.out.println("updatedBook::"+updatedBook);
* Resource<Book> resource = new Resource<Book>(updatedBook);
* return resource;
* }
* @GetMapping
* public ResponseEntity<?> getAllBook(){
* Iterable<Book> allBookList = bookStoreService.getAllBooks();
* List<Book> returnBooks = new ArrayList<Book>();
* allBookList.forEach(returnBooks::add);
* return new ResponseEntity<>(returnBooks, HttpStatus.OK);
* }
* }

# Book.java

* package com.cognizant.fsd.restfulapiassignment1.model;
* import java.io.Serializable;
* import java.time.LocalDate;
* import javax.persistence.Column;
* import javax.persistence.Entity;
* import javax.persistence.GeneratedValue;
* import javax.persistence.GenerationType;
* import javax.persistence.Id;
* import javax.persistence.JoinColumn;
* import javax.persistence.ManyToOne;
* import javax.persistence.Table;
* import javax.persistence.Transient;
* import javax.validation.constraints.NotBlank;
* import javax.validation.constraints.NotEmpty;
* import org.hibernate.annotations.GeneratorType;
* /\*\*
* \*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* @Entity
* @Table(name="TBL\_BOOK")
* public class Book implements Serializable{
* @Transient
* private static final long serialVersionUID = 1L;
* @Id
* @GeneratedValue(strategy=GenerationType.AUTO)
* @Column(name="BOOK\_ID")
* private long bookId;
* @NotEmpty(message="Book Title should not be Empty!")
* @Column(name="BOOK\_TITLE")
* private String title;

* @Column(name="BOOK\_PRICE")
* private double price;

* @Column(name="BOOK\_VOLUME")
* private Integer volume;

* @Column(name="BOOK\_PUBLISH\_DATE")
* private LocalDate publishDate;
* @ManyToOne
* @JoinColumn(name="SUBJECT\_ID")
* private Subject subject;

* public Book() {
* super();
* // TODO Auto-generated constructor stub
* }
* public Book(long bookId, String title, double price, Integer volume, LocalDate publishDate) {
* super();
* this.bookId = bookId;
* this.title = title;
* this.price = price;
* this.volume = volume;
* this.publishDate = publishDate;
* }
* public long getBookId() {
* return bookId;
* }
* public void setBookId(long bookId) {
* this.bookId = bookId;
* }
* public String getTitle() {
* return title;
* }
* public void setTitle(String title) {
* this.title = title;
* }
* public double getPrice() {
* return price;
* }
* public void setPrice(double price) {
* this.price = price;
* }
* public Integer getVolume() {
* return volume;
* }
* public void setVolume(Integer volume) {
* this.volume = volume;
* }
* public LocalDate getPublishDate() {
* return publishDate;
* }
* public void setPublishDate(LocalDate publishDate) {
* this.publishDate = publishDate;
* }
* public Subject getSubject() {
* return subject;
* }
* public void setSubject(Subject subject) {
* this.subject = subject;
* }
* @Override
* public String toString() {
* return "Book [bookId=" + bookId + ", title=" + title + ", price=" + price + ", volume=" + volume
* + ", publishDate=" + publishDate + "]";
* }
* }

# Subject.java

* package com.cognizant.fsd.restfulapiassignment1.model;
* import java.io.Serializable;
* import java.util.HashSet;
* import java.util.Set;
* import javax.persistence.Column;
* import javax.persistence.Entity;
* import javax.persistence.GeneratedValue;
* import javax.persistence.GenerationType;
* import javax.persistence.Id;
* import javax.persistence.OneToMany;
* import javax.persistence.Table;
* import javax.persistence.Transient;
* /\*\*
* \*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* @Entity
* @Table(name="TBL\_SUBJECT")
* public class Subject implements Serializable {
* @Transient
* private static final long serialVersionUID = 1L;
* @Id
* @GeneratedValue(strategy=GenerationType.AUTO)
* @Column(name="SUBJECT\_ID")
* private long subjectId;
* @Column(name="SUBJECT\_TITLE")
* private String subTitle;
* @Column(name="SUBJECT\_DURATION")
* private int durationInHours;
* @OneToMany(mappedBy="subject")
* private Set<Book> references=new HashSet<Book>();
* public long getSubjectId() {
* return subjectId;
* }
* public void setSubjectId(long subjectId) {
* this.subjectId = subjectId;
* }
* public String getSubTitle() {
* return subTitle;
* }
* public void setSubTitle(String subTitle) {
* this.subTitle = subTitle;
* }
* public int getDurationInHours() {
* return durationInHours;
* }
* public void setDurationInHours(int durationInHours) {
* this.durationInHours = durationInHours;
* }
* public Set<Book> getReferences() {
* return references;
* }
* public void setReferences(Set<Book> references) {
* this.references = references;
* }
* @Override
* public String toString() {
* return "Subject [subjectId=" + subjectId + ", subTitle=" + subTitle + ", durationInHours=" + durationInHours
* + ", references=" + references + "]";
* }
* }

# BookStoreService.java

* package com.cognizant.fsd.restfulapiassignment1.service;
* import java.util.NoSuchElementException;
* import java.util.Optional;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.dao.EmptyResultDataAccessException;
* import org.springframework.stereotype.Component;
* import com.cognizant.fsd.restfulapiassignment1.model.Book;
* import com.cognizant.fsd.restfulapiassignment2.repository.BookRepository;
* /\*\*
* \*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* @Component
* public class BookStoreService implements IBookStoreService {
* @Autowired
* BookRepository bookRepository;
* @Override
* public Book addBook(Book book) {
* Book savedBook = bookRepository.save(book);
* return savedBook;
* }
* @Override
* public Book editBook(Book book,long bookId) {
* book.setBookId(bookId);
* Book updatedBook = bookRepository.save(book);
* return updatedBook;
* }
* @Override
* public boolean deleteBook(long bookId) {
* boolean isSuccess = Boolean.TRUE;
* try {
* bookRepository.deleteById(bookId);
* }catch(EmptyResultDataAccessException e) {
* isSuccess=Boolean.FALSE;
* e.printStackTrace();
* }
* return isSuccess;
* }
* @Override
* public Optional<Book> getBookById(long bookId) {
* Book record = null;
* Optional<Book> returnValue = bookRepository.findById(bookId);
* try {
* record = returnValue.get();
* }catch(NoSuchElementException e) {
* System.out.println(e.getMessage());
* }
* return returnValue;
* }
* @Override
* public Iterable<Book> getAllBooks() {
* Iterable<Book> bookList = bookRepository.findAll();
* return bookList;
* }
* }

# IBookStoreService.java

* package com.cognizant.fsd.restfulapiassignment1.service;
* import java.util.Optional;
* import com.cognizant.fsd.restfulapiassignment1.model.Book;
* public interface IBookStoreService {
* public Book addBook(Book book);
* public Book editBook(Book book,long bookId);
* public boolean deleteBook(long bookId);
* public Optional<Book> getBookById(long bookId);
* public Iterable<Book> getAllBooks();
* }

# BookNotFoundException.java

* package com.cognizant.fsd.restfulapiassignment1.model;
* import org.springframework.http.HttpStatus;
* import org.springframework.web.bind.annotation.ResponseStatus;
* @ResponseStatus(HttpStatus.NOT\_FOUND)
* public class BookNotFoundException extends RuntimeException{
* public BookNotFoundException(String exception) {
* super(exception);
* }
* }

# CustomizedResponseExceptionHandler.java

* package com.cognizant.fsd.restfulapiassignment2.exception;
* import java.util.Date;
* import org.springframework.http.HttpHeaders;
* import org.springframework.http.HttpStatus;
* import org.springframework.http.ResponseEntity;
* import org.springframework.web.bind.MethodArgumentNotValidException;
* import org.springframework.web.bind.annotation.ExceptionHandler;
* import org.springframework.web.context.request.WebRequest;
* import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
* import com.cognizant.fsd.restfulapiassignment1.model.BookNotFoundException;
* /\*\*
* \*
* \* @author Sudhir Kumar Thakur
* \*
* \*/
* public class CustomizedResponseExceptionHandler extends ResponseEntityExceptionHandler{
* @ExceptionHandler(Exception.class)
* public final ResponseEntity<Object> handleAllExceptions(Exception ex, WebRequest request) {
* CustomizeErrorDetails errorDetails = new CustomizeErrorDetails(new Date(), ex.getMessage(),
* request.getDescription(false));
* return new ResponseEntity(errorDetails, HttpStatus.INTERNAL\_SERVER\_ERROR);
* }
* @ExceptionHandler(BookNotFoundException.class)
* public final ResponseEntity<Object> handleUserNotFoundException(BookNotFoundException ex, WebRequest request) {
* CustomizeErrorDetails errorDetails = new CustomizeErrorDetails(new Date(), ex.getMessage(),
* request.getDescription(false));
* return new ResponseEntity(errorDetails, HttpStatus.NOT\_FOUND);
* }
* @Override
* protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
* HttpHeaders headers, HttpStatus status, WebRequest request) {
* CustomizeErrorDetails errorDetails = new CustomizeErrorDetails(new Date(), "Validation Failed",
* ex.getBindingResult().toString());
* return new ResponseEntity(errorDetails, HttpStatus.BAD\_REQUEST);
* }
* }

# BookRepository.java

* package com.cognizant.fsd.restfulapiassignment2.repository;
* import org.springframework.data.repository.CrudRepository;
* import com.cognizant.fsd.restfulapiassignment1.model.Book;
* public interface BookRepository extends CrudRepository<Book , Long>{
* }

# IBookStoreService.java

* **package** com.cognizant.fsd.restfulapiassignment2.exception;
* **import** java.util.Date;
* /\*\*
* \*
* \* **@author** Sudhir Kumar Thakur
* \*
* \*/
* **public** **class** CustomizeErrorDetails {
* **private** Date timestamp;
* **private** String message;
* **private** String details;
* **public** CustomizeErrorDetails(Date timestamp, String message, String details) {
* **super**();
* **this**.timestamp = timestamp;
* **this**.message = message;
* **this**.details = details;
* }
* **public** Date getTimestamp() {
* **return** timestamp;
* }
* **public** **void** setTimestamp(Date timestamp) {
* **this**.timestamp = timestamp;
* }
* **public** String getMessage() {
* **return** message;
* }
* **public** **void** setMessage(String message) {
* **this**.message = message;
* }
* **public** String getDetails() {
* **return** details;
* }
* **public** **void** setDetails(String details) {
* **this**.details = details;
* }
* }

# SubjectRepository.java

* package com.cognizant.fsd.restfulapiassignment2.repository;
* import org.springframework.data.repository.CrudRepository;
* import com.cognizant.fsd.restfulapiassignment1.model.Subject;
* public interface SubjectRepository extends CrudRepository<Subject,Integer> {
* }

# application.properties

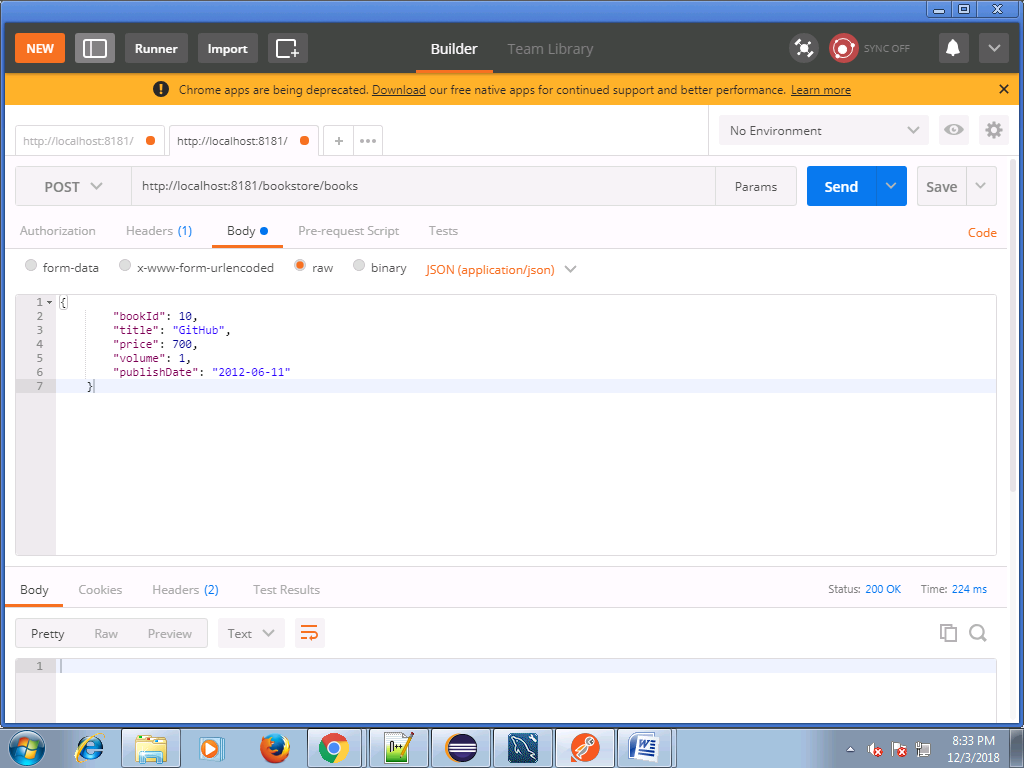
* server.port = 8181
* ## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
* spring.datasource.url = jdbc:mysql://localhost:3306/hibernateassignment?useSSL=false
* spring.datasource.username = root
* spring.datasource.password = pass@word1
* ## Hibernate Properties
* # The SQL dialect makes Hibernate generate better SQL for the chosen database
* spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
* # Hibernate ddl auto (create, create-drop, validate, update)
* spring.jpa.hibernate.ddl-auto = update

# pom.xml

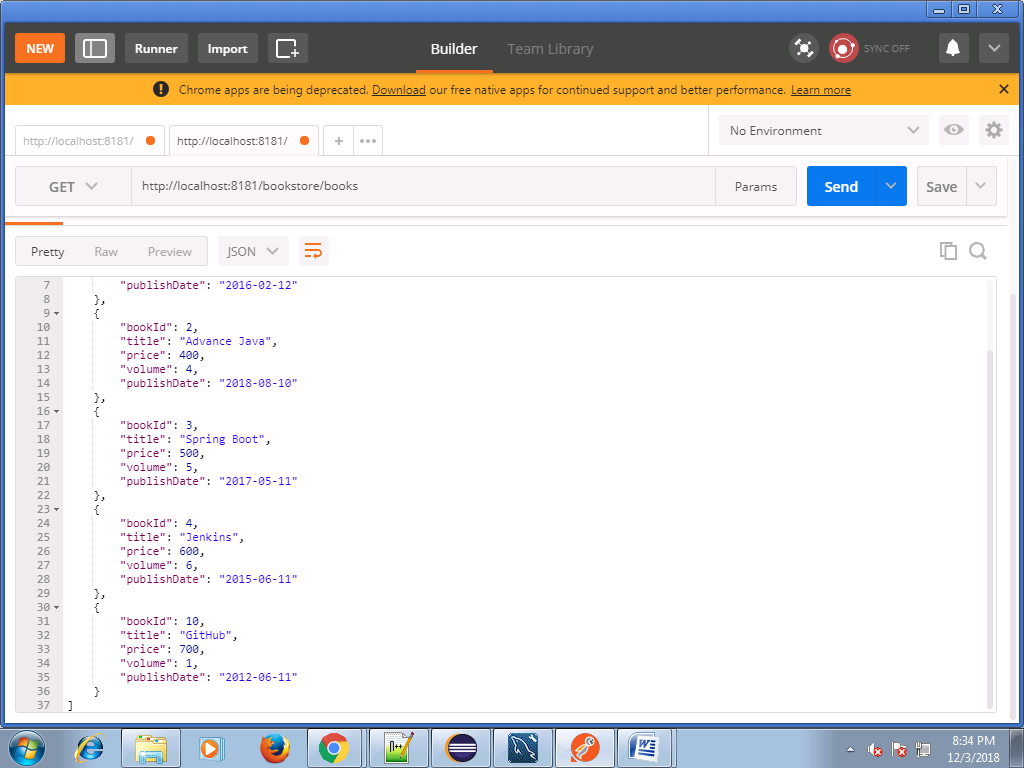
* <?xml version=*"1.0"* encoding=*"UTF-8"*?>
* <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>
* <modelVersion>4.0.0</modelVersion>
* <groupId>com.cognizant.fsd.restfulapiassignment1</groupId>
* <artifactId>11\_RestfulAPI\_Assignmet2</artifactId>
* <version>0.0.1-SNAPSHOT</version>
* <packaging>jar</packaging>
* <name>11\_RestfulAPI\_Assignmet2</name>
* <description>Demo project for Spring Boot</description>
* <parent>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-parent</artifactId>
* <version>2.1.0.RELEASE</version>
* <relativePath/> <!-- lookup parent from repository -->
* </parent>
* <properties>
* <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
* <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
* <java.version>1.8</java.version>
* </properties>
* <dependencies>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-data-jpa</artifactId>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-web</artifactId>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-devtools</artifactId>
* <scope>runtime</scope>
* </dependency>
* <dependency>
* <groupId>mysql</groupId>
* <artifactId>mysql-connector-java</artifactId>
* <scope>runtime</scope>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-test</artifactId>
* <scope>test</scope>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-hateoas</artifactId>
* </dependency>
* </dependencies>
* <build>
* <plugins>
* <plugin>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-maven-plugin</artifactId>
* </plugin>
* </plugins>
* </build>
* </project>

**Screenshots**

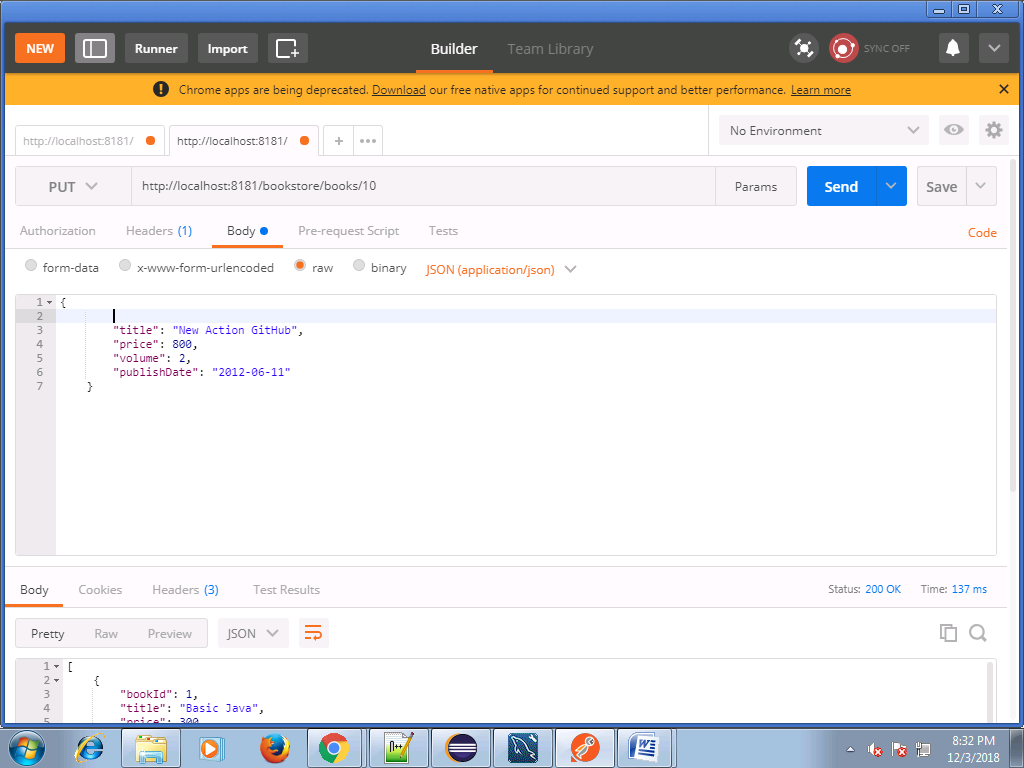
**a) POST Book(create a new book):**

****

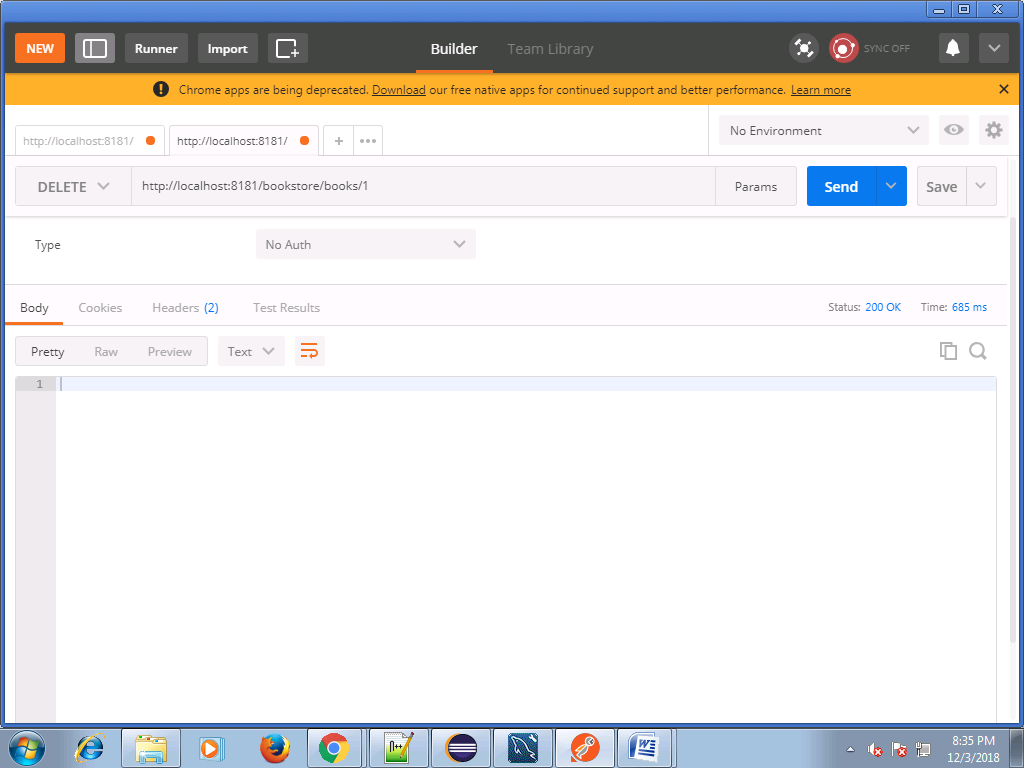
**get Request:**

****

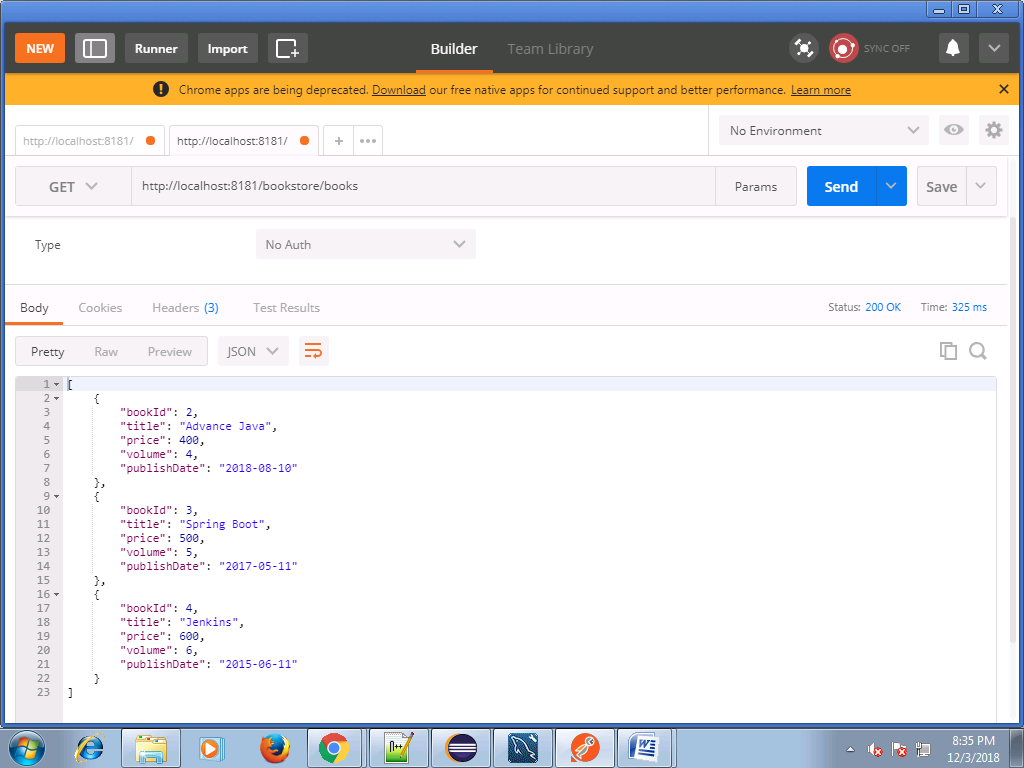
**b) PUT book (modify an existing book entry)**

****

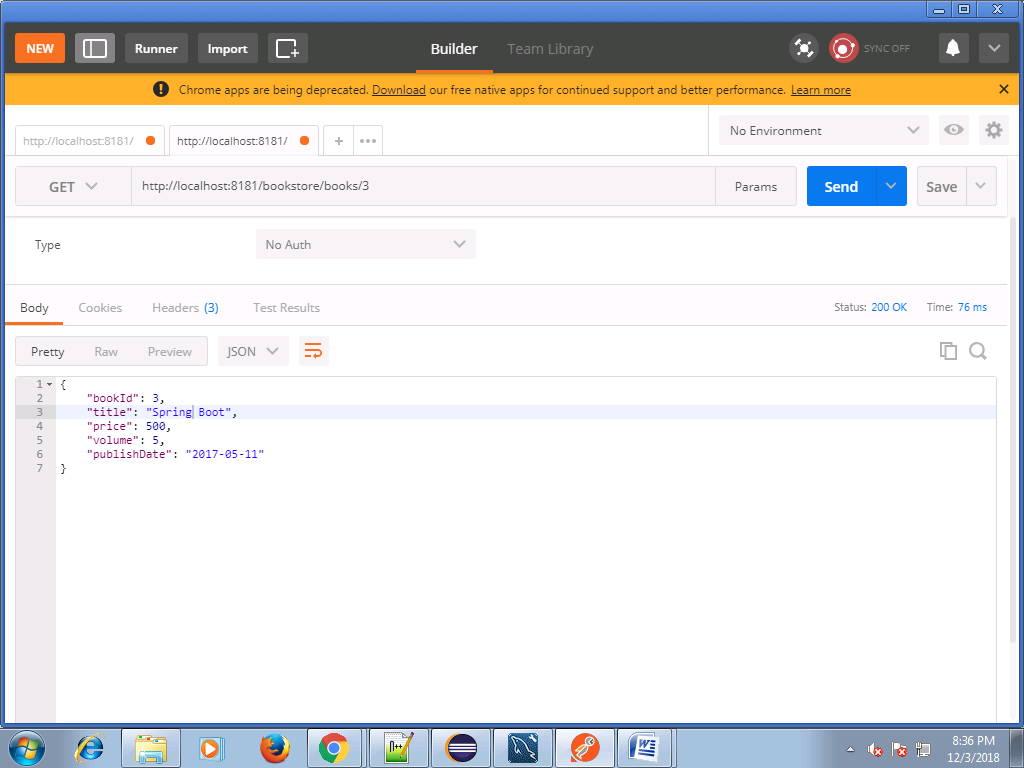
**c) DELETE Book**

****

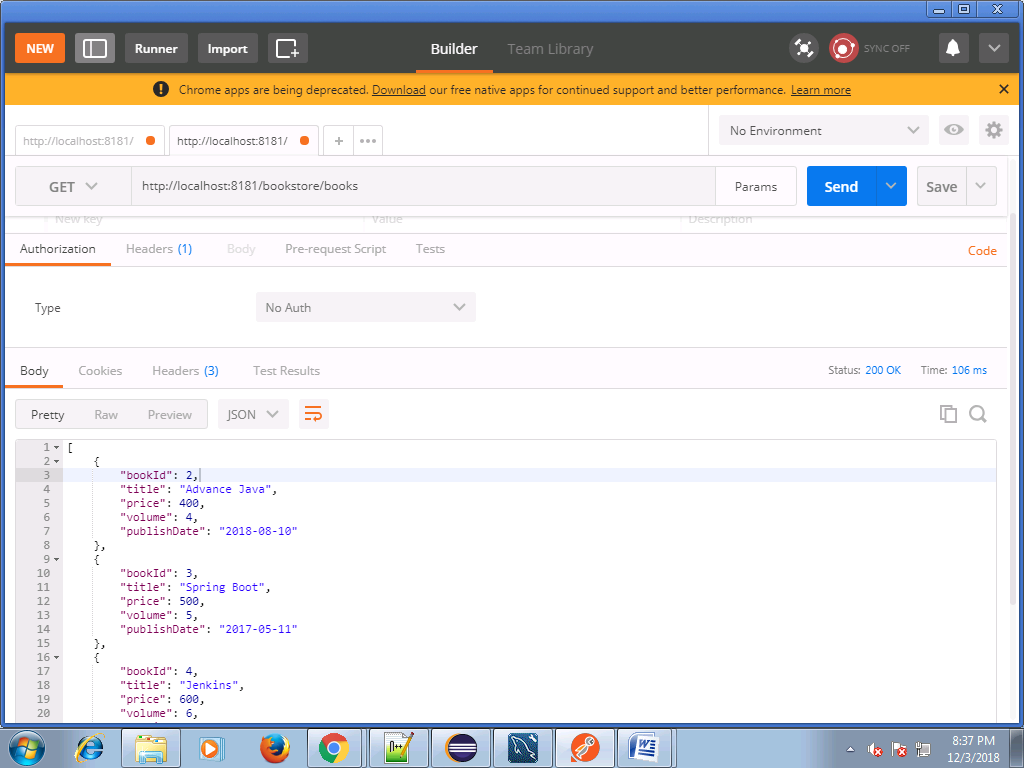
**get All Books**

****

**d) GET Book by id**

****

**e) GET all books**

****