Test Document

Online Learning Platform

AUTHOR: ÅSA WEGELIUS, CLOVIS LEBRET, TUDOR STOICA

OWNER: ÅSA WEGELIUS

CLIENT: JARL TUXEN

VERSION: 1.2.0

1. Test Document History

1.2 Revision History

version	Revision	date	Implemented by
1.0	Added Junit tests	14-05-16	Åsa Wegelius
1.2	Load, Stress, Spike, Soak Test	25-05-16	Åsa Wegelius

1.3 Approvals

version	Name	title	Date

1.4 Distribution

version	Name	title	Date

1.5 Confidentiality Rating

Rating	
Company Confidential	x
Non Confidential	

2. Content Table

3. General

1. Purpose

The Test Document presents the test scenario proposed to verify and validate the application functionalities according to the business requirements. The test cases are designed having as starting point the User Stories. These test cases cover the applications functionality area and assure that the user stories are covered. Another set of test cases cover the non-functional part of the application. These tests assure that the application is working under load and stress conditions. Due to dimension of the actual version of the application and the small number of users, the test scenarios will not address tests to validate the infrastructure reliability and redundancy or crash and disaster recovery.

Tests were split in four main areas thus they cover the main areas of the software testing process: - Unit testing during the development process to assure that the main code sections are functional during the development sprints and in the deployment phases;; - System Testing executed locally on the development environment to assure that the application is running in accordance with the development exit criteria; - User Acceptance Testing (UAT) executed by third party team to assure that the application meets the business requirements; - Performance Testing was executed to assure that the applications assure functionality in conditions of high usability and to determine what the conditions are when scalability should be implemented.

2. Reasons

The Tests series are mandatory to assure that the applications satisfy the customers need to use an online platform to improve and test their software knowledge.

4. Load, Stress, Spike, Soak Test

Test objectives

Our goal is to find out if the web site meets the performance requirements as specified below.

The site needs to be able to handle 20000 unique visitors per day providing the industry standard level of service in terms of response time and error rate. Average response time should be less than 7 seconds and error rate should be less than 1%.

The assumtions for the load test is that qe expect the average number of simultaneous users will be about 100. And that the average number of simultaneous users during peak hours will increase to 300.

General test conditions

These general test conditions are appropriate for an average user:

1. Ask for all branches
2. Ask for all courses
3. Select a course
4. Watch the course

Tests performed

The folowing tests have been designed and executed #### Load tests #####OLPStudentHandler load test:

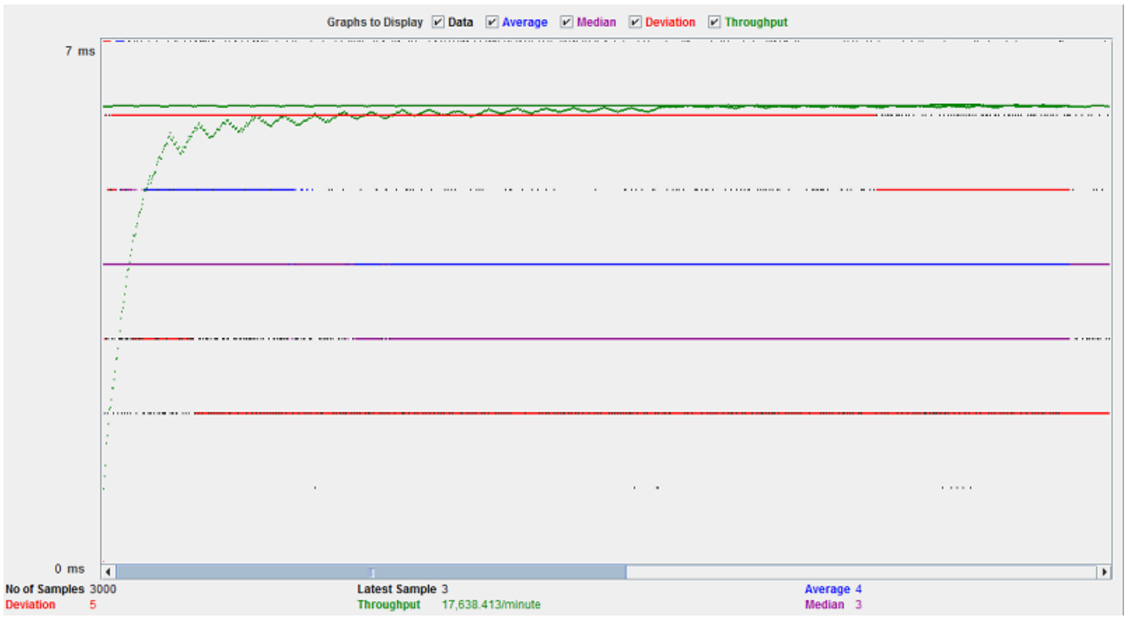
- 100 unique users
- 10 seconds ramp up period
- 10 loops/user
- request branches
- request courses
- request random course

Test results

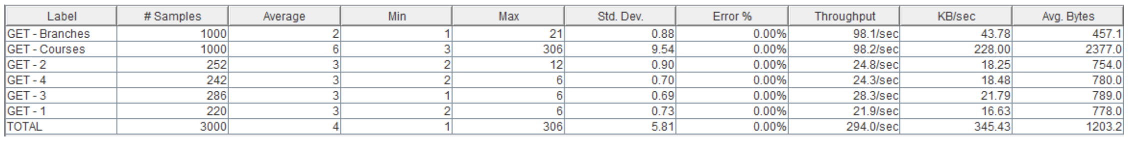
Load tests

OlpStudentHandler load test:

The test shows that the servers easily supports the demand of 100 simultanious users.

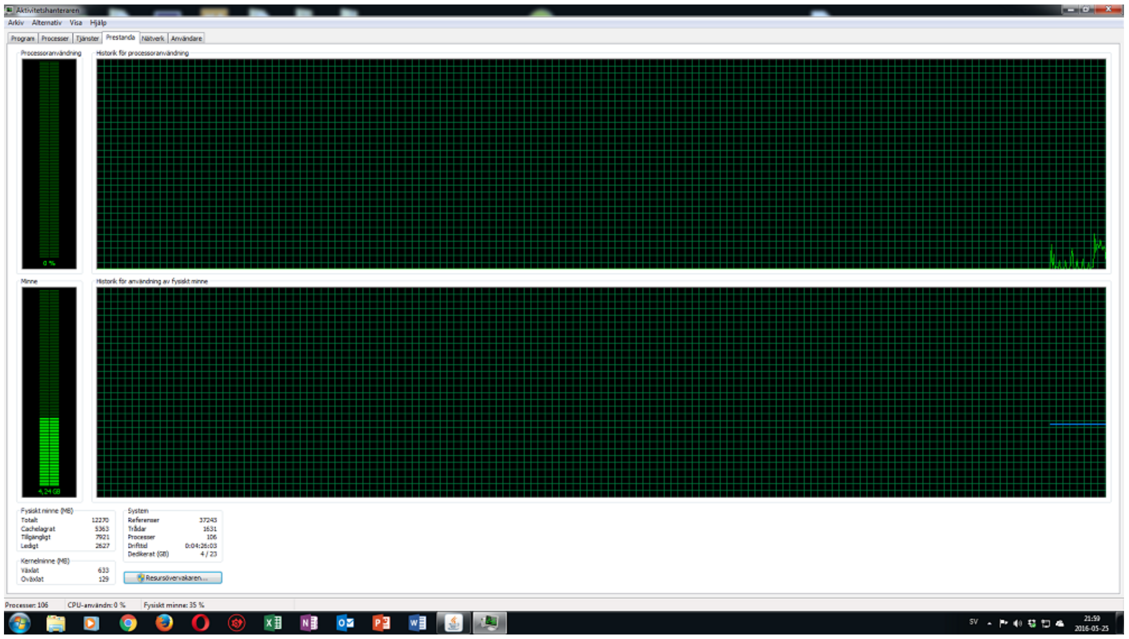


No of Samples 3000
Deviation 5
Latest Sample 3
Throughput 17,638.413/minute
Average 4
Median 3



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
GET - Branches	1000	2	1	21	0.88	0.00%	98.1/sec	43.78	457.1
GET - Courses	1000	6	3	306	9.54	0.00%	98.2/sec	228.00	2377.0
GET - 2	252	3	2	12	0.90	0.00%	24.8/sec	18.25	754.0
GET - 4	242	3	2	6	0.70	0.00%	24.3/sec	18.48	780.0
GET - 3	286	3	1	6	0.69	0.00%	28.3/sec	21.79	789.0
GET - 1	220	3	2	6	0.73	0.00%	21.9/sec	16.63	778.0
TOTAL	3000	4	1	306	5.81	0.00%	294.0/sec	345.43	1203.2

The request for courses is significant higher, but within boundarys, than the other requests. It suggests there are room for investigatin and tweeking to increase performance.



Aktivitetshanteraren
Arkiv Alternativ Visa Hjälp
Program Processer Tjänster Prestanda Nätverk Användare
Processoranvändning
Historik för processoranvändning
Minne
Historik för användning av fysiskt minne
4,24 GB
Resursövervakaren...

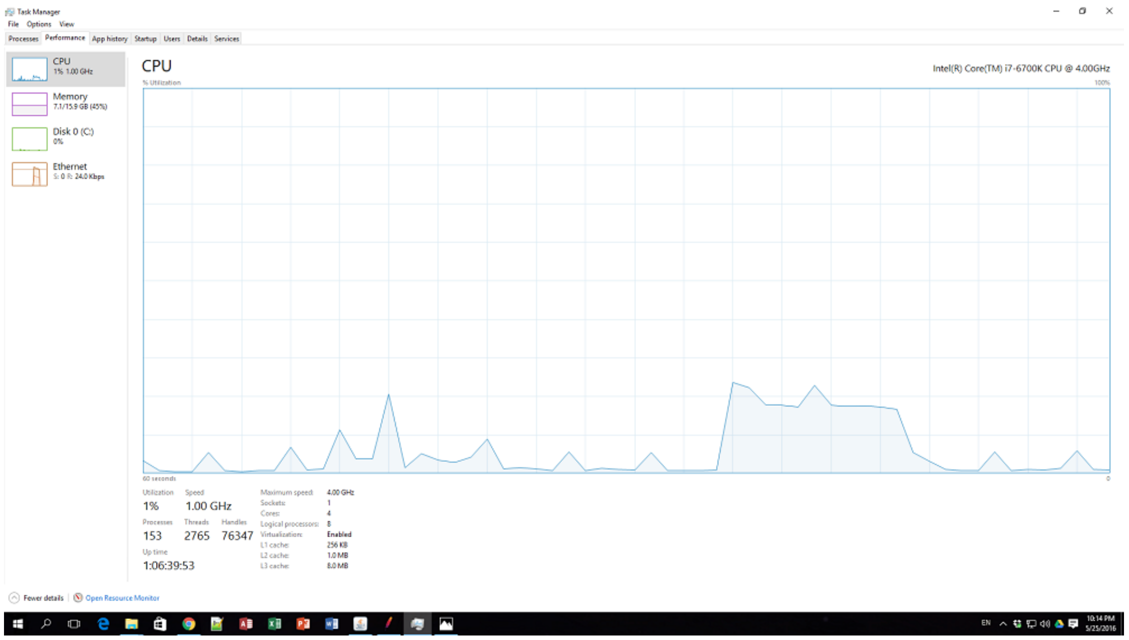
Fysiskt minne (MB)
Totalt 12270
Cachelagrat 5363
Tillgängligt 7921
Ledigt 2627

Kernelminne (MB)
Växlat 633
Oväxlat 129

System
Referenser 37243
Trådar 1631
Processer 106
Drifttid 0:04:26:03
Dedikerat (GB) 4 / 23

Processer: 106 CPU-användn: 0 % Fysiskt minne: 35 %

The main server have no problems with the workload of 100 users.



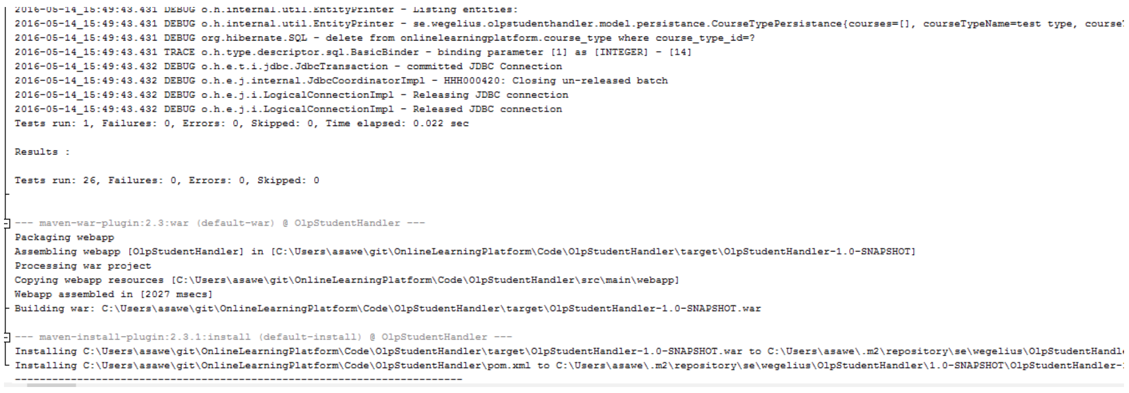
Utilization 1%
Speed 1.00 GHz
Processes 153
Threads 2765
Handles 76347
Up time 1:06:39:53

Maximum speed: 4.00 GHz
Sockets: 1
Cores: 4
Logical processors: 8
Virtualization: Enabled
L1 cache: 256 KB
L2 cache: 1.0 MB
L3 cache: 8.0 MB

The seccondary server have no problem with the workload of 100 users

5. Unit tests

We used unit tests to verify some crusial classes and methods like ensuring our generic dao setup works. Looking at the OLPStudentHandler we have over all 26 Junit tests:

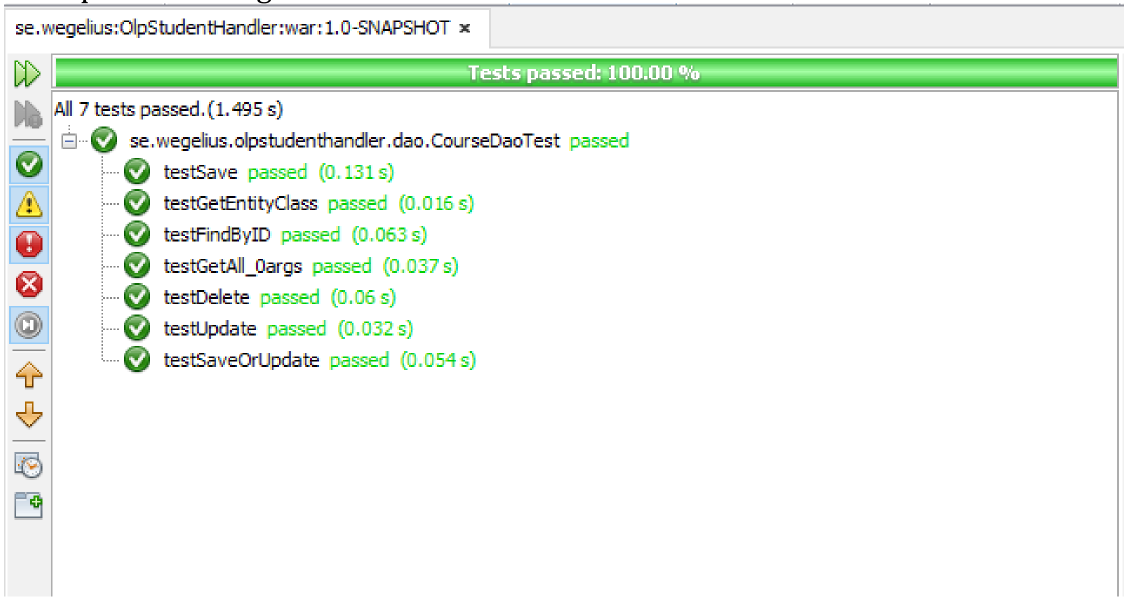


```
2016-05-14_15:49:43.431 DEBUG o.h.internal.util.EntityPrinter - Listing entities:
2016-05-14_15:49:43.431 DEBUG o.h.internal.util.EntityPrinter - se.wegelius.olpstudenthandler.model.persistance.CourseTypePersistance{courses=[], courseTypeName=test type, course
2016-05-14_15:49:43.431 DEBUG org.hibernate.SQL - delete from onlinelearningplatform.course_type where course_type_id=?
2016-05-14_15:49:43.431 TRACE o.h.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [14]
2016-05-14_15:49:43.432 DEBUG o.h.e.t.i.jdbc.JdbcTransaction - committed JDBC Connection
2016-05-14_15:49:43.432 DEBUG o.h.e.j.internal.JdbcCoordinatorImpl - HHH000420: Closing un-released batch
2016-05-14_15:49:43.432 DEBUG o.h.e.j.i.LogicalConnectionImpl - Releasing JDBC connection
2016-05-14_15:49:43.432 DEBUG o.h.e.j.i.LogicalConnectionImpl - Released JDBC connection
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 sec

Results :

Tests run: 26, Failures: 0, Errors: 0, Skipped: 0



--- maven-war-plugin:2.3:war (default-war) @ OlpStudentHandler ---
Packaging webapp
Assembling webapp [OlpStudentHandler] in [C:\Users\asawe\git\OnlineLearningPlatform\Code\OlpStudentHandler\target\OlpStudentHandler-1.0-SNAPSHOT]
Processing war project
Copying webapp resources [C:\Users\asawe\git\OnlineLearningPlatform\Code\OlpStudentHandler\src\main\webapp]
Webapp assembled in [2027 msecs]
Building war: C:\Users\asawe\git\OnlineLearningPlatform\Code\OlpStudentHandler\target\OlpStudentHandler-1.0-SNAPSHOT.war

--- maven-install-plugin:2.3.1:install (default-install) @ OlpStudentHandler ---
Installing C:\Users\asawe\git\OnlineLearningPlatform\Code\OlpStudentHandler\target\OlpStudentHandler-1.0-SNAPSHOT.war to C:\Users\asawe\.m2\repository\se\wegelius\OlpStudentHandl
Installing C:\Users\asawe\git\OnlineLearningPlatform\Code\OlpStudentHandler\pom.xml to C:\Users\asawe\.m2\repository\se\wegelius\OlpStudentHandler\1.0-SNAPSHOT\OlpStudentHandler-
------------------------------------------------------------------------
```

Example of a Junit test of the dao class CourseDao:

```
public class CourseDaoTest {
    private static CourseBranchPersistance branch;
    private static ContentProviderPersistance provider;
    private static CourseTypePersistance type;

    public CourseDaoTest() {
    }

    @BeforeClass
    public static void setUpClass() {
        branch = new CourseBranchPersistance();
        branch.setCourseBranchName("Test Branch");
        CourseBranchDao branchDao = new CourseBranchDao();
        branchDao.save(branch);
        provider = new ContentProviderPersistance();
        provider.setContentProviderName("testprovider");
        provider.setContentProviderEmail("test@test.dk");
        provider.setContentProviderDescription("expert in tests");
        ContentProviderDao providerDao = new ContentProviderDao();
        providerDao.save(provider);
        type = new CourseTypePersistance();
        type.setCourseTypeName("test type");
        type.setCtCourseBranchFk(branch.getCourseBranchId());
        CourseTypeDao typeDao = new CourseTypeDao();
        typeDao.save(type);
    }

    @AfterClass
    public static void tearDownClass() {
        CourseBranchDao branchDao = new CourseBranchDao();
        ContentProviderDao providerDao = new ContentProviderDao();
        CourseTypeDao typeDao = new CourseTypeDao();
        branchDao.delete(branch);
        providerDao.delete(provider);
        typeDao.delete(type);
    }


    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
        System.out.println("@After - tearDown");

        CourseDao courseDao = new CourseDao();
        for (CoursePersistance type : courseDao.getAll()) {
            switch (type.getCourseName()) {
                case "Test save":
                    courseDao.delete(type);
                    System.out.println("deleting" +
type.getCourseName());
                    break;
                case "to be tested":
                    courseDao.delete(type);
                    System.out.println("deleting " +
type.getCourseName());
                    break;
                case "Test update":
                    courseDao.delete(type);
                    System.out.println("deleting " +
type.getCourseName());
                    break;
                case "test FindByID":
                    courseDao.delete(type);
                    System.out.println("deleting " +
type.getCourseName());
                    break;
                case "Test SaveOrUpdate":
                    courseDao.delete(type);
                    System.out.println("deleting " +
type.getCourseName());
                    break;
                default:
                    System.out.println("default " +
type.getCourseName());
                    break;
            }
        }
    }
    /**
     * Test of save method, of class OlpDao.
     */
    @Test
    public void testSave() {
        System.out.println("save");
        CoursePersistance course = new CoursePersistance();
        course.setCourseBranch(branch);
        course.setContentProvider(provider);
        course.setCourseType(type);
        course.setCourseName("Test save");
        CourseDao courseDao = new CourseDao();
        int sum = courseDao.count();
        courseDao.save(course);
        int newSum = courseDao.count();
        System.out.println("sum = " + sum + " newSum = " + newSum);
        assertTrue(sum < newSum);
    }

    /**
     * Test of update method, of class OlpDao.
     */
    @Test
    public void testUpdate() {
        System.out.println("update");
        CourseDao courseDao = new CourseDao();
        CoursePersistance course = new CoursePersistance();
        course.setCourseBranch(branch);
        course.setContentProvider(provider);
        course.setCourseType(type);
        course.setCourseName("to be tested");
        courseDao.saveOrUpdate(course);
        course.setCourseName("Test update");
        courseDao.update(course);
        CoursePersistance test =
courseDao.findByID(course.getCourseId());
        assertEquals(test.getCourseName(), "Test update");
    }

    /**
     * Test of saveOrUpdate method, of class OlpDao.
     */
    @Test
    public void testSaveOrUpdate() {
        System.out.println("saveOrUpdate");
        CoursePersistance course = new CoursePersistance();
        course.setCourseBranch(branch);
        course.setContentProvider(provider);
        course.setCourseType(type);
        course.setCourseName("to be tested");
        CourseDao courseDao = new CourseDao();
        courseDao.save(course);
        CoursePersistance type2 =
courseDao.findByID(course.getCourseId());
        type2.setCourseName("Test SaveOrUpdate");
        courseDao.saveOrUpdate(type2);

assertEquals(courseDao.findByID(type2.getCourseId()).getCourseName(),
courseDao.findByID(course.getCourseId()).getCourseName());
    }

    /**
     * Test of findByID method, of class OlpDao.
     */
    @Test
    public void testFindByID() {
        System.out.println("findByID");
        CoursePersistance course = new CoursePersistance();
        course.setCourseBranch(branch);
        course.setContentProvider(provider);
        course.setCourseType(type);
        course.setCourseName("test FindByID");
        CourseDao courseDao = new CourseDao();
        courseDao.save(course);
        Integer id = course.getCourseId();
        Object expResult = id;
        Object result = courseDao.findByID(id).getCourseId();
        assertEquals(expResult, result);
    }

    /**
     * Test of getAll method, of class OlpDao.
     */
    @Test
    public void testGetAll_0args() {
        System.out.println("getAll");
        CourseDao instance = new CourseDao();
        Set result = instance.getAll();
        assertEquals(instance.count(), result.size());
    }
    @Test
    public void testDelete() {
        System.out.println("delete");
        CoursePersistance course = new CoursePersistance();
        course.setCourseBranch(branch);
        course.setContentProvider(provider);
        course.setCourseType(type);
        course.setCourseName("Test delete");
        CourseDao courseDao = new CourseDao();
        int sum = courseDao.count();
        courseDao.save(course);
        courseDao.delete(course);
        int newSum = courseDao.count();
        assertEquals(sum, newSum);
    }

    /**
     * Test of getEntityClass method, of class OlpDao.
     */
    @Test
    public void testGetEntityClass() {
        System.out.println("getEntityClass");
        CourseDao courseDao = new CourseDao();
        Class expResult = CoursePersistance.class;
        Class result = courseDao.getEntityClass();
        assertEquals(expResult, result);
    }


}
```

Example of running the CourseDaoTest:



se.wegelius:OlpStudentHandler:war:1.0-SNAPSHOT ×

Tests passed: 100.00 %

```
All 7 tests passed.(1.495 s)
  se.wegelius.olpstudenthandler.dao.CourseDaoTest passed
    testSave passed (0.131 s)
    testGetEntityClass passed (0.016 s)
    testFindByID passed (0.063 s)
    testGetAll_0args passed (0.037 s)
    testDelete passed (0.06 s)
    testUpdate passed (0.032 s)
    testSaveOrUpdate passed (0.054 s)
```