



About Siva Reddy

Katanreddy Siva Prasad is a Senior Software Engineer working in E-Commerce domain. His areas of interest include Object Oriented Design, SOLID Design principles, RESTful WebServices and OpenSource softwares including Spring, MyBatis and Jenkins.



MyBatis Tutorial – CRUD Operations and Mapping Relationships – Part 1

by Siva Reddy on November 16th, 2012 | Filed In: [Enterprise Java](#) Tags: [MyBatis](#)



CRUD Operations

MyBatis is an SQL Mapper tool which greatly simplifies the database programming when compared to using JDBC directly.

Build Responsive ASP.NET MVC Apps Faster

70+ Extensions for Every Need



Get Your Free Trial

Telerik

Online Japanese Study


 study.u-biq.org/english.htm



Step1: Create a Maven project and configure MyBatis dependencies.

```
01 <project xmlns="http://maven.apache.org/POM/4.0.0"
02 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
04
05 http://maven.apache.org/xsd/maven-4.0.0.xsd">
06
07   <modelVersion>4.0.0</modelVersion>
08
09   <groupId>com.sivalabs</groupId>
10   <artifactId>mybatis-demo</artifactId>
11   <version>0.0.1-SNAPSHOT</version>
12   <packaging>jar</packaging>
13
14   <name>mybatis-demo</name>
15   <url>http://maven.apache.org</url>
16
17   <properties>
18     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
19   </properties>
20
21   <build>
22     <plugins>
23       <plugin>
24         <groupId>org.apache.maven.plugins</groupId>
25         <artifactId>maven-compiler-plugin</artifactId>
26         <version>2.3.2</version>
27         <configuration>
28           <source>1.6</source>
29           <target>1.6</target>
30           <encoding>${project.build.sourceEncoding}</encoding>
31         </configuration>
32       </plugin>
33     </plugins>
34   </build>
35
36   <dependencies>
37     <dependency>
38       <groupId>junit</groupId>
39       <artifactId>junit</artifactId>
40       <version>4.10</version>
41       <scope>test</scope>
42     </dependency>
43
44     <dependency>
45       <groupId>org.mybatis</groupId>
46       <artifactId>mybatis</artifactId>
47       <version>3.1.1</version>
48     </dependency>
49   </dependencies>
50
51   <dependency>
52     <groupId>mysql</groupId>
53     <artifactId>mysql-connector-java</artifactId>
54     <version>5.1.21</version>
55     <scope>runtime</scope>
```


New sletter



20,709 insiders are already enjoying weekly updates and complimentary whitepapers! **Join them now** to gain exclusive access to the latest news in the Java world, as well as insights about Android, Scala, Groovy and other related technologies.

As an **extra bonus**, by joining you will get our **brand new e-books**, published by Java Code Geeks and their JCG partners for your reading pleasure!

Join Us



With **748,895** Jun unique visitors and over **500** authors we are placed among the top Java related sites around. Constantly being on the lookout for partners; we encourage you to join us. So if you have a blog with unique and interesting content then you should check out our **JCG** partners program. You can also be a **guest writer** for Java Code Geeks and hone your writing skills in addition to utilizing our **revenue sharing model** to **monetize** your technical writing!

Carrer Opportunities

Technical Solutions Consultant, Social and Knowledge (FULL-TIME) July 20th, 2014

Technical Lead/Manager/Director Engineer- Core Systems (FULL-TIME) July 19th, 2014

```

54         </dependency>
55     </dependencies>
56 </project>

```

Java Developer for Curam Training (FULL-TIME)
July 19th, 2014

Java Developer/Analyst (FULL-TIME) July 19th,
2014

Software Developer, Principal (FULL-TIME) July
18th, 2014

Tags

[Akka](#) [Apache Camel](#) [Apache Hadoop](#)
[Apache Maven](#) [Apache Tomcat](#) [Big Data](#)
[Cloud](#) [Concurrency](#) [Databases](#)
[Design Patterns](#) [Eclipse](#) [Git](#) [Google Guava](#)
[Gradle](#) [Grails](#) [IDE](#) [Java 7](#) [Java 8](#) [Java EE6](#)
[JavaFX](#) [JAXB](#) [JBoss](#) [Hibernate](#) [JMS](#) [JPA](#)
[JSF](#) [JSON](#) [JUnit](#) [JVM](#) [Lambdas](#) [Logging](#)
[MongoDB](#) [News](#) [NoSQL](#) [Oracle GlassFish](#)
[Performance](#) [Play Framework](#) [Project Management](#)
[RESTful](#) [Web Services](#) [Security](#)
[Spring](#) [Spring Data](#) [Spring MVC](#) [SQL](#)
[Testing](#) [XML](#)

Step#2: Create the table USER and a Java domain Object User as follow s:

```

1 CREATE TABLE user (
2     user_id int(10) unsigned NOT NULL auto_increment,
3     email_id varchar(45) NOT NULL,
4     password varchar(45) NOT NULL,
5     first_name varchar(45) NOT NULL,
6     last_name varchar(45) default NULL,
7     PRIMARY KEY (user_id),
8     UNIQUE KEY Index_2_email_uniq (email_id)
9 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;

01 package com.sivalabs.mybatisdemo.domain;
02 public class User
03 {
04     private Integer userId;
05     private String emailId;
06     private String password;
07     private String firstName;
08     private String lastName;
09
10     @Override
11     public String toString() {
12         return 'User [userId=' + userId + ', emailId=' + emailId
13             + ', password=' + password + ', firstName=' + firstName
14             + ', lastName=' + lastName + ']';
15     }
16     //setters and getters
17 }

```

Step#3: Create MyBatis configuration files.

a) Create jdbc.properties file in src/main/resources folder

```

1 jdbc.driverClassName=com.mysql.jdbc.Driver
2 jdbc.url=jdbc:mysql://localhost:3306/mybatis-demo
3 jdbc.username=root
4 jdbc.password=admin

```

b) Create mybatis-config.xml file in src/main/resources folder

```

01 <?xml version='1.0' encoding='UTF-8' ?>
02 <!DOCTYPE configuration
03     PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
04     'http://mybatis.org/dtd/mybatis-3-config.dtd'>
05 <configuration>
06     <properties resource='jdbc.properties' />
07     <typeAliases>
08         <typeAlias type='com.sivalabs.mybatisdemo.domain.User' alias='User'></typeAlias>
09     </typeAliases>
10     <environments default='development'>
11         <environment id='development'>
12             <transactionManager type='JDBC' />
13             <dataSource type='POOLED'>
14                 <property name='driver' value='${jdbc.driverClassName}' />
15                 <property name='url' value='${jdbc.url}' />
16                 <property name='username' value='${jdbc.username}' />
17                 <property name='password' value='${jdbc.password}' />
18             </dataSource>
19         </environment>
20     </environments>
21     <mappers>
22         <mapper resource='com/sivalabs/mybatisdemo/mappers/UserMapper.xml' />
23     </mappers>
24 </configuration>

```

Step#4: Create an interface UserMapper.java in src/main/java folder in com.sivalabs.mybatisdemo.mappers package.

```

01 package com.sivalabs.mybatisdemo.mappers;
02
03 import java.util.List;
04 import com.sivalabs.mybatisdemo.domain.User;
05
06 public interface UserMapper
07 {
08
09     public void insertUser(User user);
10
11     public User getUserById(Integer userId);
12
13     public List<User> getAllUsers();
14
15     public void updateUser(User user);
16
17     public void deleteUser(Integer userId);
18
19 }

```

Step#5: Create UserMapper.xml file in src/main/resources folder in com.sivalabs.mybatisdemo.mappers package.

```

01 <?xml version='1.0' encoding='UTF-8' ?>
02 <!DOCTYPE mapper PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN"
03 'http://mybatis.org/dtd/mybatis-3-mapper.dtd'>
04
05 <mapper namespace='com.sivalabs.mybatisdemo.mappers.UserMapper'>
06
07   <select id='getUserById' parameterType='int' resultType='com.sivalabs.mybatisdemo.domain.User'>
08     SELECT
09     user_id as userId,
10     email_id as emailId ,
11     password,
12     first_name as firstName,
13     last_name as lastName
14   FROM USER
15   WHERE USER_ID = #{userId}
16   </select>
17   <!-- Instead of referencing Fully Qualified Class Names we can register Aliases in mybatis-
18   config.xml and use Alias names. -->
19   <resultMap type='User' id='UserResult'>
20     <id property='userId' column='user_id' />
21     <result property='emailId' column='email_id' />
22     <result property='password' column='password' />
23     <result property='firstName' column='first_name' />
24     <result property='lastName' column='last_name' />
25   </resultMap>
26
27   <select id='getAllUsers' resultMap='UserResult'>
28     SELECT * FROM USER
29   </select>
30
31   <insert id='insertUser' parameterType='User' useGeneratedKeys='true' keyProperty='userId'>
32     INSERT INTO USER(email_id, password, first_name, last_name)
33     VALUES(#{emailId}, #{password}, #{firstName}, #{lastName})
34   </insert>
35
36   <update id='updateUser' parameterType='User'>
37     UPDATE USER
38     SET
39     PASSWORD= #{password},
40     FIRST_NAME = #{firstName},
41     LAST_NAME = #{lastName}
42     WHERE USER_ID = #{userId}
43   </update>
44
45   <delete id='deleteUser' parameterType='int'>
46     DELETE FROM USER WHERE USER_ID = #{userId}
47   </delete>
48 </mapper>

```

Step#6: Create MyBatisUtil.java to instantiate SqlSessionFactory.

```

01 package com.sivalabs.mybatisdemo.service;
02
03 import java.io.IOException;
04 import java.io.Reader;
05 import org.apache.ibatis.io.Resources;
06 import org.apache.ibatis.session.SqlSessionFactory;
07 import org.apache.ibatis.session.SqlSessionFactoryBuilder;
08
09 public class MyBatisUtil
10 {
11   private static SqlSessionFactory factory;
12
13   private MyBatisUtil() {
14   }
15
16   static
17   {
18     Reader reader = null;
19     try {
20       reader = Resources.getResourceAsReader('mybatis-config.xml');
21     } catch (IOException e) {
22       throw new RuntimeException(e.getMessage());
23     }
24     factory = new SqlSessionFactoryBuilder().build(reader);
25   }
26
27   public static SqlSessionFactory getSqlSessionFactory()
28   {
29     return factory;
30   }
31 }

```

Step#7: Create UserService.java in src/main/java folder.

```

01 package com.sivalabs.mybatisdemo.service;
02
03 import java.util.List;
04 import org.apache.ibatis.session.SqlSession;
05 import com.sivalabs.mybatisdemo.domain.User;
06 import com.sivalabs.mybatisdemo.mappers.UserMapper;
07
08 public class UserService
09 {
10
11   public void insertUser(User user) {
12     SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
13     try{
14       UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
15       userMapper.insertUser(user);

```

```

16     sqlSession.commit();
17 }finally{
18     sqlSession.close();
19 }
20 }
21
22 public User getUserById(Integer userId) {
23     SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
24     try{
25         UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
26         return userMapper.getUserById(userId);
27     }finally{
28         sqlSession.close();
29     }
30 }
31
32 public List<User> getAllUsers() {
33     SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
34     try{
35         UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
36         return userMapper.getAllUsers();
37     }finally{
38         sqlSession.close();
39     }
40 }
41
42 public void updateUser(User user) {
43     SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
44     try{
45         UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
46         userMapper.updateUser(user);
47         sqlSession.commit();
48     }finally{
49         sqlSession.close();
50     }
51 }
52
53
54 public void deleteUser(Integer userId) {
55     SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
56     try{
57         UserMapper userMapper = sqlSession.getMapper(UserMapper.class);
58         userMapper.deleteUser(userId);
59         sqlSession.commit();
60     }finally{
61         sqlSession.close();
62     }
63 }
64
65
66 }

```

Step#8: Create a JUnit Test class to test UserService methods.

```

01 package com.sivalabs.mybatisdemo;
02
03 import java.util.List;
04
05 import org.junit.AfterClass;
06 import org.junit.Assert;
07 import org.junit.BeforeClass;
08 import org.junit.Test;
09
10 import com.sivalabs.mybatisdemo.domain.User;
11 import com.sivalabs.mybatisdemo.service.UserService;
12
13 public class UserServiceTest
14 {
15     private static UserService userService;
16
17     @BeforeClass
18     public static void setup()
19     {
20         userService = new UserService();
21     }
22
23     @AfterClass
24     public static void teardown()
25     {
26         userService = null;
27     }
28
29     @Test
30     public void testGetUserById()
31     {
32         User user = userService.getUserById(1);
33         Assert.assertNotNull(user);
34         System.out.println(user);
35     }
36
37     @Test
38     public void testGetAllUsers()
39     {
40         List<User> users = userService.getAllUsers();
41         Assert.assertNotNull(users);
42         for (User user : users)
43         {
44             System.out.println(user);
45         }
46     }
47
48
49     @Test
50     public void testInsertUser()
51     {

```

```

52     User user = new User();
53     user.setEmailId('test_email_'+System.currentTimeMillis()+'@gmail.com');
54     user.setPassword('secret');
55     user.setFirstName('TestFirstName');
56     user.setLastName('TestLastName');
57
58     userService.insertUser(user);
59     Assert.assertTrue(user.getUserId() != 0);
60     User createdUser = userService.getUserById(user.getUserId());
61     Assert.assertNotNull(createdUser);
62     Assert.assertEquals(user.getEmailId(), createdUser.getEmailId());
63     Assert.assertEquals(user.getPassword(), createdUser.getPassword());
64     Assert.assertEquals(user.getFirstName(), createdUser.getFirstName());
65     Assert.assertEquals(user.getLastName(), createdUser.getLastName());
66
67 }
68
69 @Test
70 public void testUpdateUser()
71 {
72     long timestamp = System.currentTimeMillis();
73     User user = userService.getUserById(2);
74     user.setFirstName('TestFirstName'+timestamp);
75     user.setLastName('TestLastName'+timestamp);
76     userService.updateUser(user);
77     User updatedUser = userService.getUserById(2);
78     Assert.assertEquals(user.getFirstName(), updatedUser.getFirstName());
79     Assert.assertEquals(user.getLastName(), updatedUser.getLastName());
80 }
81
82 @Test
83 public void testDeleteUser()
84 {
85     User user = userService.getUserById(4);
86     userService.deleteUser(user.getUserId());
87     User deletedUser = userService.getUserById(4);
88     Assert.assertNull(deletedUser);
89 }
90
91 }

```

Now, I will explain how to perform CRUD operations using MyBatis Annotation support without need of Queries configuration in XML mapper files.

Step#1: Create a table BLOG and a java domain Object Blog.

```

1 CREATE TABLE blog (
2     blog_id int(10) unsigned NOT NULL auto_increment,
3     blog_name varchar(45) NOT NULL,
4     created_on datetime NOT NULL,
5     PRIMARY KEY (blog_id)
6 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

01 package com.sivalabs.mybatisdemo.domain;
02
03 import java.util.Date;
04
05 public class Blog {
06
07     private Integer blogId;
08     private String blogName;
09     private Date createdOn;
10
11     @Override
12     public String toString() {
13         return 'Blog [blogId=' + blogId + ', blogName=' + blogName
14             + ', createdOn=' + createdOn + ']';
15     }
16     //Setters and getters
17 }

```

Step#2: Create UserMapper.java interface with SQL queries in Annotations.

```

01 package com.sivalabs.mybatisdemo.mappers;
02
03 import java.util.List;
04
05 import org.apache.ibatis.annotations.Delete;
06 import org.apache.ibatis.annotations.Insert;
07 import org.apache.ibatis.annotations.Options;
08 import org.apache.ibatis.annotations.Result;
09 import org.apache.ibatis.annotations.Results;
10 import org.apache.ibatis.annotations.Select;
11 import org.apache.ibatis.annotations.Update;
12
13 import com.sivalabs.mybatisdemo.domain.Blog;
14
15 public interface BlogMapper
16 {
17     @Insert('INSERT INTO BLOG(BLOG_NAME, CREATED_ON) VALUES(#{blogName}, #{createdOn})')
18     @Options(useGeneratedKeys=true, keyProperty='blogId')
19     public void insertBlog(Blog blog);
20
21     @Select('SELECT BLOG_ID AS blogId, BLOG_NAME as blogName, CREATED_ON as createdOn FROM BLOG WHERE BLOG_ID=#{blogId}')
22     public Blog getBlogById(Integer blogId);
23
24     @Select('SELECT * FROM BLOG ')
25     @Results({
26         @Result(id=true, property='blogId', column='BLOG_ID'),

```

```

27     @Result(property='blogName', column='BLOG_NAME'),
28     @Result(property='createdOn', column='CREATED_ON')
29     })
30     public List<Blog> getAllBlogs();
31
32     @Update('UPDATE BLOG SET BLOG_NAME=#{blogName}, CREATED_ON=#{createdOn} WHERE BLOG_ID=#{blogId}')
33     public void updateBlog(Blog blog);
34
35     @Delete('DELETE FROM BLOG WHERE BLOG_ID=#{blogId}')
36     public void deleteBlog(Integer blogId);
37
38 }

```

Step#3: Configure BlogMapper in mybatis-config.xml

```

01 <?xml version='1.0' encoding='UTF-8' ?>
02 <!DOCTYPE configuration
03 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
04 "http://mybatis.org/dtd/mybatis-3-config.dtd">
05 <configuration>
06 <properties resource='jdbc.properties' />
07 <environments default='development'>
08 <environment id='development'>
09 <transactionManager type='JDBC' />
10 <dataSource type='POOLED'>
11 <!-- <property name='driver' value='com.mysql.jdbc.Driver' />
12 <property name='url' value='jdbc:mysql://localhost:3306/mybatis-demo' />
13 <property name='username' value='root' />
14 <property name='password' value='admin' /> -->
15 <property name='driver' value='${jdbc.driverClassName}' />
16 <property name='url' value='${jdbc.url}' />
17 <property name='username' value='${jdbc.username}' />
18 <property name='password' value='${jdbc.password}' />
19 </dataSource>
20 </environment>
21 </environments>
22 <mappers>
23 <mapper class='com.sivalabs.mybatisdemo.mappers.BlogMapper' />
24 </mappers>
25 </configuration>

```

Step#4: Create BlogService.java

```

01 package com.sivalabs.mybatisdemo.service;
02
03 import java.util.List;
04
05 import org.apache.ibatis.session.SqlSession;
06
07 import com.sivalabs.mybatisdemo.domain.Blog;
08 import com.sivalabs.mybatisdemo.mappers.BlogMapper;
09
10 public class BlogService
11 {
12
13     public void insertBlog(Blog blog) {
14         SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
15         try{
16             BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
17             blogMapper.insertBlog(blog);
18             sqlSession.commit();
19         }finally{
20             sqlSession.close();
21         }
22     }
23
24     public Blog getBlogById(Integer blogId) {
25         SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
26         try{
27             BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
28             return blogMapper.getBlogById(blogId);
29         }finally{
30             sqlSession.close();
31         }
32     }
33
34     public List<Blog> getAllBlogs() {
35         SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
36         try{
37             BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
38             return blogMapper.getAllBlogs();
39         }finally{
40             sqlSession.close();
41         }
42     }
43
44     public void updateBlog(Blog blog) {
45         SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
46         try{
47             BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);
48             blogMapper.updateBlog(blog);
49             sqlSession.commit();
50         }finally{
51             sqlSession.close();
52         }
53     }
54
55     public void deleteBlog(Integer blogId) {
56         SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory().openSession();
57         try{
58             BlogMapper blogMapper = sqlSession.getMapper(BlogMapper.class);

```

```

59     blogger.deleteBlog(blogId);
60     sqlSession.commit();
61 }finally{
62     sqlSession.close();
63 }
64 }
65 }
66 }
67 }

```

Step#5: Create JUnit Test for BlogService methods

```

01 package com.sivalabs.mybatisdemo;
02
03 import java.util.Date;
04 import java.util.List;
05
06 import org.junit.AfterClass;
07 import org.junit.Assert;
08 import org.junit.BeforeClass;
09 import org.junit.Test;
10
11 import com.sivalabs.mybatisdemo.domain.Blog;
12 import com.sivalabs.mybatisdemo.service.BlogService;
13
14 public class BlogServiceTest
15 {
16     private static BlogService blogService;
17
18     @BeforeClass
19     public static void setup()
20     {
21         blogService = new BlogService();
22     }
23
24     @AfterClass
25     public static void teardown()
26     {
27         blogService = null;
28     }
29
30     @Test
31     public void testGetBlogById()
32     {
33         Blog blog = blogService.getBlogById(1);
34         Assert.assertNotNull(blog);
35         System.out.println(blog);
36     }
37
38     @Test
39     public void testGetAllBlogs()
40     {
41         List<Blog> blogs = blogService.getAllBlogs();
42         Assert.assertNotNull(blogs);
43         for (Blog blog : blogs)
44         {
45             System.out.println(blog);
46         }
47     }
48 }
49
50 @Test
51 public void testInsertBlog()
52 {
53     Blog blog = new Blog();
54     blog.setBlogName('test_blog_'+System.currentTimeMillis());
55     blog.setCreatedOn(new Date());
56
57     blogService.insertBlog(blog);
58     Assert.assertTrue(blog.getBlogId() != 0);
59     Blog createdBlog = blogService.getBlogById(blog.getBlogId());
60     Assert.assertNotNull(createdBlog);
61     Assert.assertEquals(blog.getBlogName(), createdBlog.getBlogName());
62 }
63
64 @Test
65 public void testUpdateBlog()
66 {
67     long timestamp = System.currentTimeMillis();
68     Blog blog = blogService.getBlogById(2);
69     blog.setBlogName('TestBlogName'+timestamp);
70     blogService.updateBlog(blog);
71     Blog updatedBlog = blogService.getBlogById(2);
72     Assert.assertEquals(blog.getBlogName(), updatedBlog.getBlogName());
73 }
74
75 @Test
76 public void testDeleteBlog()
77 {
78     Blog blog = blogService.getBlogById(4);
79     blogService.deleteBlog(blog.getBlogId());
80     Blog deletedBlog = blogService.getBlogById(4);
81     Assert.assertNull(deletedBlog);
82 }
83
84 }

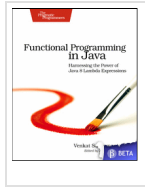
```

Reference: MyBatis Tutorial: Part1 – CRUD Operations from our JCG partner, MyBatis Tutorial: Part-2: CRUD operations Using Annotations from our JCG partner Siva Reddy at the [My Experiments on Technology](#) blog.

You might also like:

- [MyBatis Tutorial – CRUD Operations and Mapping Relationships – Part 2](#)
- [Spring MVC 3 Controller for MyBatis CRUD operation](#)
- [MyBatis 3 – Spring integration tutorial](#)

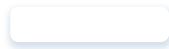
Related Whitepaper:



Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions

Get ready to program in a whole new way!

Functional Programming in Java will help you quickly get on top of the new, essential Java 8 language features and the functional style that will change and improve your code. This short, targeted book will help you make the paradigm shift from the old imperative way to a less error-prone, more elegant, and concise coding style that's also a breeze to parallelize. You'll explore the syntax and semantics of lambda expressions, method and constructor references, and functional interfaces. You'll design and write applications better using the new standards in Java 8 and the JDK.



App. Server
Integration

 myeclipse.com



11 Responses to "MyBatis Tutorial – CRUD Operations and Mapping Relationships – Part 1"



Yannick Majoros

November 18th, 2012 at 3:05 pm

Come on, I hope nobody uses that instead of JPA?

[Reply](#)



Siva Prasad Reddy

November 19th, 2012 at 8:58 am

Try building a RESTful services application which contain very complex object graph structure using JPA. If you try to marshal JPA loaded proxy after it got disconnected from Session it will throw LazyLoadingException, if you try to marshal JPA entity by attaching it to JPA session it will load the entire database, if you are planning to use DTO pattern and populate the necessary properties only All the BEST :-). And then take a look at MyBatis :-)

[Reply](#)



Yannick Majoros

December 1st, 2012 at 10:41 pm

How would MyBatis solve that, besides the fact that it's non-standard and as such, the only implementation? If you need to load a complex object graph, you'll have to maintain it somehow. Perhaps you should not make your object graph that complex in the first place (seems it's what you do here anyway, small object graph for which JPA is a much better choice). Perhaps you should better find the bounds of your business operations, and know when to merge and when not to end your transaction or to detach. This thing is a relic, should disappear asap instead of letting people write crap that will eventually have to be refactored to JPA or be forever lost in spaghetti-code land.

[Reply](#)



Siva Prasad Reddy

February 4th, 2013 at 11:51 am

Hi,

I am not going to enter into never ending debate on whether STANDARDS are important

or not.

Here what I am trying to say is if I load a complex object structure using Mybatis they are really POJOs, not proxies. So I can use any of the marshalling/unmarshalling tools like Jackson/xstream etc to generate XML/JSON. If some property is null it will ignore to generate the tag or generates an empty tag.

Where as if I load a complex object by JPA which has lazy child collection then if I try to marshal that object it will throw LazyLoadingException if the object is detached from EntityManager. If that object is still attached to JPA entityManager it will load the whole database as the marshalling libraries navigate through all the properties triggering loading the data from DB.

Anyway it would be great if you can show an example on how to marshal a JPA loaded object which has a lazy child collection property using Jackson without using manual copy-object (DTO) creation.

[Reply](#)



Phongveth

January 23rd, 2013 at 8:00 am

Hi Siva, I still confuse that can we define method in the interface which have return type boolean
example like `public void insertBlog(Blog blog);` into
`public boolean insertBlog(Blog blog);`

[Reply](#)



Siva Prasad Reddy

February 4th, 2013 at 11:31 am

No, for insert/update/delete queries you can return int representing the no of rows affected by the query. You can't return boolean.

[Reply](#)



Phongveth Luangsisongkham

February 5th, 2013 at 3:03 am

Thank you very much for your reply

[Reply](#)



krish

February 7th, 2013 at 9:35 am

hello friend you have done very good work ... i used it in my app.... thx
but i have a prob. when i open it from differ pc then it not show me onl9 to others ...
and when i open it in my own pc with differ browser then it works good ... please help me

[Reply](#)



Jed

February 27th, 2014 at 12:47 pm

Hi Siva Reddy,
Thanks for your tutorial, I learned a lot.
I do have some points to address, I hope you dont mind sharing it.
On your Service objects (BlogService.java and UserService.java), i think these object are for actual DB execution. My concern is, how would you handle the Exceptions thrown by your JDBC driver?
as far as I can see, try{} – finally{} are the only ones there. can you explain how to determine all the exceptions thrown by the driver? and is it possible to handle those? please provide a sample try{} – catch{} combination.
Please reply thru my mail.
Thank you and Best Regards.

[Reply](#)



Siva

February 27th, 2014 at 2:41 pm

Hi Jed,

I haven't covered Exception handling in Service classes in the article. Yes, if you are using plain MyBatis you need to handle JDBC Exceptions using try-catch blocks. If you are using MyBatis with Spring then Spring will translate JDBC Exceptions into Spring's more appropriate DataAccessExceptions hierarchy and they are all RuntimeExceptions. If you want to handle case-by-case then you can catch those Exceptions and take necessary action.
Hope it helps.

[Reply](#)



deepak

June 27th, 2014 at 5:33 pm

Hi Siva,

Thanks for this wonderful tutorial. I was trying a POC with ibatis 2 version and got stuck with some strange errors. Then I wanted to switch over to ibatis. With your tutorial, I just completed it with half an hour. Thanks :) very useful

[Reply](#)

Leave a Reply

Name (Required)
 Mail (will not be published) (Required)
 Website

6 + = ten

☐ Notify me of follow up comments via e-mail

☒ Sign me up for the new sletter!

Knowledge Base

[Academy](#)
[Examples](#)
[Resources](#)
[Tutorials](#)
[Whitepapers](#)

Partners

[Mkyong](#)

The Code Geeks Network

[Java Code Geeks](#)
[.NET Code Geeks](#)
[Web Code Geeks](#)

Hall Of Fame

["Android Full Application Tutorial" series](#)
[GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial](#)
[Android Game Development Tutorials](#)
[Android Google Maps Tutorial](#)
[Android Location Based Services Application – GPS location](#)
[Funny Source Code Comments](#)
[Java Best Practices – Vector vs ArrayList vs HashSet](#)
[Android JSON Parsing with Gson Tutorial](#)
[Android Quick Preferences Tutorial](#)

About Java Code Geeks

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical architect, technical team lead (senior developer), project manager and junior developers alike. JCGs serve the Java, SOA, Agile and Telecom communities with daily new s written by domain experts, articles, tutorials, review s, announcements, code snippets and open source projects.