FREE eBook

LEARNING mybatis

Free unaffiliated eBook created from **Stack Overflow contributors.**

#mybatis

Table of Contents

About 1

Chapter 1: Getting started with mybatis 2

Remarks 2

Examples 2

Hello world using Spring Boot and Maven 2

Spring Boot- Jersey-Mybatis-MySql REST Maven Project step by step 4

Credits 13

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: mybatis

It is an unofficial and free mybatis ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official mybatis.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with mybatis

Remarks

This section provides an overview of what mybatis is, and why a developer might want to use it.

It should also mention any large subjects within mybatis, and link out to the related topics. Since the Documentation for mybatis is new, you may need to create initial versions of those related topics.

Examples

Hello world using Spring Boot and Maven

1. Add the necessary dependencies to the project POM (`mybatis`, and `mybatis-spring`):

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>x.x.x</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.1-SNAPSHOT</version>
</dependency>
```

A `SqlSessionFactory` is needed to create mybatis sessions. Spring allows one to be quickly configured using `SqlSessionFactoryBean`. Simply define a `SqlSessionFactoryBean` bean, and give the reference to the `DataSource` bean:

```
@Bean
public SqlSessionFactoryBean sqlSessionFactoryBean(DataSource dataSource) {
    SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();
    sqlSessionFactoryBean.setDataSource(dataSource);
    return sqlSessionFactoryBean;
}
```

2. Define a mybatis *mapper*. The mapper is a java interface that will hold SQL queries and translate method calls into `JDBC` queries. If a default Spring Boot `HSQLDB` database is being used, this following query can be created. (There are tables involved; it simply returns a string built using a user-provided parameter).

```
public interface HelloWorldMapper {

    @Select("VALUES ('Hello ' || #{origin})")
    String getString(@Param("origin") String origin);

}
```

3. Register existing mappers so Mybatis and Spring can know about them. Add `@MapperScan` to the spring java configuration with the name of the "root" package containing the mapper interfaces. This annotation will auto-register the interfaces as spring beans, which can be easily injected anywhere in the application.

```
    @Autowired
    private HelloWorldMapper helloWorldMapper;

    // invoking the mapper method
    helloWorldMapper.getString("World");
```

Here's the complete Spring configuration class, with a test invocation of the query:

```
package com.example;

import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.SqlSessionTemplate;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.SpringBootConfiguration;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.TypeExcludeFilter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.FilterType;

import javax.annotation.PostConstruct;
import javax.sql.DataSource;
import java.lang.annotation.*;

@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan
@MapperScan("com.example")
public class DemoApplication {

    @Autowired
    private HelloWorldMapper helloWorldMapper;

    @Bean
    public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {
        SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();
        sqlSessionFactoryBean.setDataSource(dataSource);
        return (SqlSessionFactory) sqlSessionFactoryBean.getObject();
    }

    @PostConstruct
    public void init() {
        System.out.println(helloWorldMapper.getString("World"));
    }

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

Spring Boot- Jersey-Mybatis-MySql REST Maven Project step by step

```
Requirement for Local development
1) MySQL server(I am Using XAMPP for MySQL server)
2) For Test API You can use Postman(optional)
3) Before run application,make sure MySQL server is running, properly prepare your
application.properties file(DB_Name, Username,Password).
```

1.Add following dependency,parent into your POM.xml file

```
    <parent>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-parent</artifactId>
         <version>1.3.0.RELEASE</version>
     </parent>
     <build>
         <plugins>
             <plugin>
                 <groupId>org.springframework.boot</groupId>
                 <artifactId>spring-boot-maven-plugin</artifactId>
             </plugin>
             <plugin>
                 <groupId>org.apache.maven.plugins</groupId>
                 <artifactId>maven-compiler-plugin</artifactId>
                 <configuration>
                     <source>1.8</source>
                     <target>1.8</target>
                 </configuration>
             </plugin>
         </plugins>
     </build>
     <dependencies>
         <dependency>
             <groupId>org.springframework.boot</groupId>
             <artifactId>spring-boot-starter-web</artifactId>
         </dependency>
         <dependency>
             <groupId>org.springframework.boot</groupId>
             <artifactId>spring-boot-starter-data-jpa</artifactId>
         </dependency>
         <dependency>
             <groupId>org.springframework.boot</groupId>
             <artifactId>spring-boot-starter-jersey</artifactId>
         </dependency>
         <dependency>
             <groupId>org.glassfish.jersey.media</groupId>
             <artifactId>jersey-media-multipart</artifactId>
             <version>2.18</version>
         </dependency>
         <dependency>
             <groupId>org.springframework.boot</groupId>
             <artifactId>spring-boot-starter-tomcat</artifactId>
         </dependency>

         <dependency>
             <groupId>commons-validator</groupId>
```

```
            <artifactId>commons-validator</artifactId>
            <version>1.5.0</version>
        </dependency>

        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-sample-annotation</artifactId>
            <version>1.1.1</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.4.1</version>
        </dependency>



        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
    </dependencies>
```

My Project Structure Should be This

```
src-->
     main-->
          java
          |--->com-->controller
          |              |--UserController.java
          |              |--CORSResponseFilter.java
          |           entity
          |              |--User.java
          |           mapper
          |              |--UserMapper.java
          |           JerseyConfig.java
          |           SampleMapperApplication.java
          |
          resources
                 |-->application.properties
```

2.Create an Entity Class named User in src\main\java\com\entity\User.java

```
package com.entity;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.NotEmpty;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

@Entity
public class User {
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    private int id;

    @Size(min = 2, max =50)
    @Pattern(regexp = "[A-Za-z. ]*", message = "First name requires valid character")
    private String firstname;

    @Size(min = 2, max =50)
    @Pattern(regexp = "[A-Za-z. ]*", message = "First name requires valid character")
    private String lastname;


    //@Column(unique = true)
    @NotNull(message =  "Email requires valid value")
    @NotEmpty(message = "Email requires non empty value")
    @Email(message =    "Email requires valid format")
    private String email;

    @Size(min = 2, max =50)
    @Pattern(regexp = "[A-Za-z.0-9 ]*", message = "First name requires valid character")
    private String password;

    @Pattern(regexp = "[0-9.\\-+ ]*", message = "Phone requires valid alphanumaric
characters")
    private String phone;

    private int age;
    /*setter getter method here*/



}
```

3.Create A interface named UserMapper.java into src\main\java\com\mapper\UserMapper.java

```
package com.mapper;


import com.entity.User;
import org.apache.ibatis.annotations.*;

import java.util.List;

@Mapper
public interface UserMapper {
    @Insert("INSERT INTO USER ( firstname,lastname, email,password,age ,phone ) VALUES (
#{user.firstname}, #{user.lastname},#{user.email},#{user.password},
#{user.age},#{user.phone})")
    Integer insertUser(@Param("user") User user) throws Exception;

    @Select("select * from user where email = #{email}")
    User findByEmail(@Param("email") String email);

    @Select("select * from user where email = #{email} AND id!=#{id}")
    User findByEmailNotUser(@Param("email") String email,@Param("id") Integer id);

    @Select("select * from user where id = #{id}")
    User findById(@Param("id") Integer id);

    @Select("select * from user where email = #{email} AND password = #{password}")
    User login(@Param("email") String email,@Param("password") String password);


    @Select("select * from user")
    List<User> getUsers();

    @Update("UPDATE USER SET firstname = #{user.firstname},lastname = #{user.lastname},email =
#{user.email},phone = #{user.phone},age = #{user.age} WHERE id = #{user.id}")
    Integer updateUser(@Param("user") User user) throws Exception;

    @Delete("DELETE from user where id = #{id}")
    Integer deleteById(@Param("id") Integer id);



}
```

4.Create Two class UserController.java into src\main\java\com\controller\UserController.java

```
package com.controller;

import com.entity.User;
import com.mapper.UserMapper;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;


import javax.validation.ConstraintViolation;
import javax.validation.Validation;
import javax.validation.Validator;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Set;


@Controller
@Path("/users")
public class UserController {
    static final Logger logger = LoggerFactory.getLogger(UserController.class);
    private static Validator validator =
Validation.buildDefaultValidatorFactory().getValidator();

    @Autowired
    private UserMapper userMapper;

    @POST
    @Consumes("application/json")
    @Produces("application/json")
    public Response create(User user) {
        LinkedHashMap<Object, Object> serviceResponse = new LinkedHashMap<Object, Object>();
        logger.info("Starting to create a user");

        try {
            Set<ConstraintViolation<User>> validateErrors = validator.validate(user);
            if (validateErrors.isEmpty()) {
                if (userMapper.findByEmail(user.getEmail()) != null) {
                    serviceResponse.put("duplicate_Email", "user already exist");
                } else {
                    Integer createPerson = userMapper.insertUser(user);

                    if (createPerson != 1) {
                        serviceResponse.put("created", "unable to create user");
                    } else {
                        logger.info("Successfully created User.");
                        serviceResponse.put("created_msg", "Successfully created User");
                    }
                }
                return
Response.status(Response.Status.CREATED).entity(serviceResponse).build();
            } else {
                logger.info("Failed to create a user due to field validation errors.");
                logger.debug("Unable to create a user due to validation errors using {}",
user);
                //JSONObject jsonObj = new JSONObject(validateErrors.toString());
                serviceResponse.put("error", validateErrors.toString());
                return Response.status(400).entity(serviceResponse).build();


            }
        } catch (Exception e) {
            logger.debug("<< create()");
            return
Response.status(Response.Status.INTERNAL_SERVER_ERROR).entity(serviceResponse).build();
        }
    }


    @GET
    @Produces("application/json")
    public Response getUsers() {

        LinkedHashMap<String, Object> response = new LinkedHashMap<String, Object>();
        try {
            List<User> listUsers = userMapper.getUsers();
            if (listUsers == null) {
                response.put("users", Collections.emptyMap());
            } else {
                response.put("total", listUsers.size());
                response.put("users", listUsers);
            }
            return Response.status(Response.Status.OK).entity(listUsers).build();
        } catch (Exception ex) {

            response.put("user", "Not Found");
            return Response.status(Response.Status.BAD_REQUEST).entity(response).build();
        }
    }

    @PUT
    @Path("/{id}")
    @Produces("application/json")
    public Response updateUser(@PathParam("id") Integer id, User user) {
        //String personJson = gson.toJson(user);
        //logger.debug(">> create({})", personJson);
        //LinkedHashMap<Object, Object> apiResponse = new LinkedHashMap<>();
        LinkedHashMap<Object, Object> serviceResponse = new LinkedHashMap<Object, Object>();
        logger.info("Starting to create a person");

        try {
            Set<ConstraintViolation<User>> validateErrors = validator.validate(user);
            if (validateErrors.isEmpty()) {
                if (userMapper.findById(id) == null) {
                    serviceResponse.put("msg", "User Not Found");
                } else {
                    if (userMapper.findByEmailNotUser(user.getEmail(), user.getId()) != null)
{
                        serviceResponse.put("duplicate_Email", "user already exist");
                    } else {
                        int updateUser = userMapper.updateUser(user);
                        if (updateUser == 0) {
                            serviceResponse.put("created", "unable to update User");
                        } else {
                            logger.info("Successfully update user.");
                            serviceResponse.put("update", user);
                        }
                    }
                }
                return Response.status(Response.Status.OK).entity(user).build();
            } else {
                logger.info("Failed to update a user due to field validation errors.");
                // logger.debug("Unable to update a user due to validation errors using {}",
personJson);
                serviceResponse.put("error", validateErrors.toString());

                return Response.status(400).entity(serviceResponse).build();
            }
        } catch (Exception e) {
            logger.debug("<< create()");
            return
Response.status(Response.Status.INTERNAL_SERVER_ERROR).entity(serviceResponse).build();
        }
    }

    @GET
    @Path("/{id}")
    @Produces("application/json")
    public Response getPerson(@PathParam("id") int userId) {

        LinkedHashMap<String, Object> response = new LinkedHashMap<String, Object>();

        try {
            User user = userMapper.findById(userId);

            if (user == null) {
                response.put("User", Collections.emptyMap());
            } else {
                response.put("user", user);
            }
            return Response.status(Response.Status.OK).entity(user).build();
        } catch (Exception ex) {

            response.put("user", "Not Found");
            return Response.status(Response.Status.BAD_REQUEST).entity(response).build();
        }
    }
    @DELETE
    @Path("/{id}")
    @Consumes("application/json")
    @Produces("application/json")
    public Response deleteUser(@PathParam("id") int userId) {

        //LinkedHashMap<Object, Object> apiResponse = new LinkedHashMap<>();
        LinkedHashMap<Object, Object> serviceResponse = new LinkedHashMap<Object, Object>();
        logger.info("Starting to delete a user");

        try {
            int deletePerson = userMapper.deleteById(userId);
            if (deletePerson == 0) {
                serviceResponse.put("delete", "unable delete user");
            } else {
                logger.info("Successfully delete user.");
                serviceResponse.put("delete", "Successfully delete user.");
            }
            return Response.status(Response.Status.OK).entity(serviceResponse).build();

        } catch (Exception e) {

            logger.debug("<< create()");
            return
Response.status(Response.Status.INTERNAL_SERVER_ERROR).entity(serviceResponse).build();
        }
    }

}
```

5.Create CORSResponseFilter.java class into src\main\java\com\controller\CORSResponseFilter.java

```
package com.controller;

import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.core.MultivaluedMap;
import java.io.IOException;


public class CORSResponseFilter implements ContainerResponseFilter {
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext
responseContext)
            throws IOException {

        MultivaluedMap<String, Object> headers = responseContext.getHeaders();

        headers.add("Access-Control-Allow-Origin", "*");
        //headers.add("Access-Control-Allow-Origin", "http://abcd.org"); //allows CORS
requests only coming from abcd.org
        headers.add("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
        headers.add("Access-Control-Allow-Headers", "X-Requested-With, Content-Type");
    }
}
```

6.Register your Configuration Class into JerseyConfig class

src\main\java\com\JerseyConfig.java

```
package com;

import com.controller.CORSResponseFilter;
import com.controller.UserController;
import org.glassfish.jersey.server.ResourceConfig;
import org.springframework.context.annotation.Configuration;

import javax.ws.rs.ApplicationPath;


@Configuration
@ApplicationPath("/api")
public class JerseyConfig extends ResourceConfig {
    public JerseyConfig() {
        register(CORSResponseFilter.class);
        register(UserController.class);



    }

}
```

7. Create your application class named SampleMapperApplication into src\main\java\com\SampleMapperApplication.java

```
package com;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication
public class SampleMapperApplication implements CommandLineRunner {

    public static void main(String[] args) {
        SpringApplication.run(SampleMapperApplication.class, args);

    }



}
```

8. Prepare your application.properties file src\main\resources\application.properties

```
# localhost:3306 will be replace your DB server url if you want. I am using MySql local DB
server
spring.datasource.url = jdbc:mysql://localhost:3306/YOUR_DB_NAME
db.driver=com.mysql.jdbc.Driver
#if you want update, just replace create with update
spring.jpa.hibernate.ddl-auto=create
#If you want to change default port(8080) number, Comment out below line code
#server.port = 8090
```

```
# Username and password
spring.datasource.username = YOUR_DB_USER_NAME
spring.datasource.password =YOUR_DB_USER_PASSWORD
```

For Testing user REST API

```
  List of User(GET) ---->http://localhost:8080/api/users
  Create User(POST) ---->http://localhost:8080/api/users
  Get User(GET)    ----->http://localhost:8080/api/users/1
  Update User(PUT) ----->http://localhost:8080/api/users/1
  Delete User(Delete)--->http://localhost:8080/api/users/1
```

Read Getting started with mybatis online: https://riptutorial.com/mybatis/topic/6692/getting-started-with-mybatis

Credits

S. No	Chapters	Contributors
1	Getting started with mybatis	4444, Abdullah Al Hafiz, Community, Fabien Benoit-Koch, Roman Konoval, Skyler Tao