

News Knowledge Base Deals About

g+ f t in

Search...



Java Code Geeks

JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

ANDROID CORE JAVA DESKTOP JAVA ENTERPRISE JAVA JAVA BASICS JVM LANGUAGES SOFTWARE DEVELOPMENT DEVOPS

Home » Enterprise Java » spring » Boot » Spring Boot SOAP Service with Hibernate Example

ABOUT MARY ZHENG



Mary has graduated from Mechanical Engineering department at ShangHai JiaoTong University. She also holds a Master degree in Computer Science from Webster University. During her studies she has been involved with a large number of projects ranging from programming and software engineering. She works as a senior Software Engineer in the telecommunications sector where she acts as a leader and works with others to design, implement, and monitor the software solution.

8+

Spring Boot SOAP Service with Hibernate Example

Posted by: Mary Zheng in Boot September 27th, 2018 1 Comment 803 Views

1. Introduction

Java Persistence API (JPA) is Java's standard API specification for object-relational mapping. Hibernate is a JPA provider and provides a framework for mapping an object-oriented domain model to a relational database. Spring Boot defines a list of starter projects, in which each project includes a set of default component dependencies and an automatic configuration of components. The Spring Web Service starter project enables developers to write the contract-first SOAP service easily.

In this example, I will create a SOAP service with Hibernate in a Spring Boot application.

Want to master Spring Framework ?

Subscribe to our newsletter and download the Spring Framework Cookbook [right now!](#)

In order to help you master the leading and innovative Java framework, we have compiled a kick-ass guide with all its major features and use cases! Besides studying them online you may download the eBook in PDF format!

Email address:

Sign up

2. Technologies Used

The example code in this article was built and run using:

- Java 1.8.101
- Maven 3.3.9
- Eclipse Oxygen
- Spring Boot 1.5.16
- Hibernate 5.0.12.Final

Why
new j

✕

2. Select

with

and Spring Boot version

. Add

, and

in the "search for dependencies" section.

3. Enter the group name:

and artifact:

4. Click the

button.

A maven project will be generated and downloaded to your workstation. Import it into your Eclipse workspace.

3.1 Dependencies

The generated

includes

, and

. I will include

and

[pom.xml](#)

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <project xmlns="http://maven.apache.org/POM/4.0.0"
03     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

✕

x

```

16 <groupId>org.springframework.boot</groupId>
17 <artifactId>spring-boot-starter-parent</artifactId>
18 <version>1.5.15.RELEASE</version>
19 <relativePath /> <!-- lookup parent from repository -->
20 </parent>
21
22 <properties>
23 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
24 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
25 <java.version>1.8</java.version>
26 </properties>
27
28 <dependencies>
29 <dependency>
30 <groupId>org.springframework.boot</groupId>
31 <artifactId>spring-boot-starter-data-jpa</artifactId>
32 </dependency>
33 <dependency>
34 <groupId>org.springframework.boot</groupId>
35 <artifactId>spring-boot-starter-web-services</artifactId>
36 </dependency>
37 <dependency>
38 <groupId>wsdl4j</groupId>
39 <artifactId>wsdl4j</artifactId>
40 </dependency>
41
42 <dependency>
43 <groupId>com.h2database</groupId>
44 <artifactId>h2</artifactId>
45 <scope>runtime</scope>
46 </dependency>
47
48 <dependency>
49 <groupId>org.springframework.boot</groupId>
50 <artifactId>spring-boot-starter-test</artifactId>
51 <scope>test</scope>
52 </dependency>
53 </dependencies>
54
55 <build>
56 <plugins>
57 <plugin>
58 <groupId>org.springframework.boot</groupId>
59 <artifactId>spring-boot-maven-plugin</artifactId>
60 </plugin>
61 <plugin>
62 <groupId>org.jvnet.jaxb2.maven2</groupId>
63 <artifactId>maven-jaxb2-plugin</artifactId>
64 <version>0.13.1</version>
65 <executions>
66 <execution>
67 <goals>
68 <goal>generate</goal>
69 </goals>
70 </execution>
71 </executions>
72 <configuration>
73 <schemaDirectory>src/main/resources/wsdl</schemaDirectory>
74 <schemaIncludes>
75 <include>*.wsdl</include>
76 </schemaIncludes>
77 </configuration>
78 <dependencies>
79 <dependency>
80 <groupId>com.sun.xml.bind</groupId>
81 <artifactId>jaxb-osgi</artifactId>
82 <version>2.2.11</version>
83 </dependency>
84 </dependencies>
85 </plugin>
86 </plugins>
87 </build>
88
89 </project>
90
91

```

3.2 Spring Boot Application

In this step, I will modify the generated Spring Boot application to include a step to save test data.

[SpringBootSoapHibernateApplication.java](#)

```

01 package jcg.zheng.demo.springbootsoaphibernate;
02
03 import java.util.Random;
04
05 import org.springframework.boot.SpringApplication;
06 import org.springframework.boot.autoconfigure.SpringBootApplication;
07 import org.springframework.context.ConfigurableApplicationContext;
08 import org.springframework.scheduling.annotation.EnableAsync;
09
10 import jcg.zheng.demo.springbootsoaphibernate.entity.Employee;
11 import jcg.zheng.demo.springbootsoaphibernate.repository.EmployeeRepository;
12 import payroll.bestpay.employee.EmployeeType;
13
14 @SpringBootApplication
15 @EnableAsync

```

x



```
27 private static Employee buildDummyEmployee(String firstName, String lastName) {
28     Employee emp = new Employee();
29     emp.setType(EmployeeType.HOURLY);
30     Random rand = new Random();
31
32     emp.setFirstName(firstName);
33     emp.setLastName(lastName);
34     emp.setDepartment("dummy dept");
35     emp.setManagerId("1");
36
37     emp.setHourlyRate(rand.nextInt(100));
38
39     return emp;
40 }
41
42 }
```

3.3 Application Properties

In this step, I will include two properties to set the Hibernate SQL logger level to

```
DEBUG
```

. Click here to see the default properties.

[applicaiton.properties](#)

```
1 logging.level.root=INFO
2 logging.level.org.hibernate.SQL=DEBUG
3
4 spring.jpa.generate-ddl=true
```

4. Employee WSDL

The Web Services Description Language (WSDL) is an XML-based interface definition language that is used for describing the functionality offered by a web service. In this step,

```
employee.wsdl
```

is an XML document which describes the

```
EmployeeLookup
```

web service and specifies how to access and use its methods.

[employee.wsdl](#)

```
001 <wsdl:definitions
002     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
003     xmlns:tns="http://bestpay.payroll/employee"
004     xmlns:xs="http://www.w3.org/2001/XMLSchema"
005     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
006     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
007     targetNamespace="http://bestpay.payroll/employee">
008
009     <wsdl:documentation>
010         Service: EmployeeService
011         Version: 1.0
012         Owner: Mary
013         Zheng
014     </wsdl:documentation>
015     <wsdl:types>
016         <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
017             targetNamespace="http://bestpay.payroll/employee"
018             xmlns:tns="http://bestpay.payroll/employee"
019             elementFormDefault="qualified">
020
021             <xs:simpleType name="EmployeeId">
022                 <xs:restriction base="xs:string">
023                     <xs:length value="7" />
024                     <xs:pattern value="[E0-9]{7}" />
025                 </xs:restriction>
026             </xs:simpleType>
027
028             <xs:simpleType name="EmployeeType">
029                 <xs:restriction base="xs:string">
030                     <xs:enumeration value="Hourly" />
031                     <xs:enumeration value="Salary" />
032                 </xs:restriction>
033             </xs:simpleType>
034
035             <xs:complexType name="EmployeeInfo">
036                 <xs:sequence>
037                     <xs:element name="eid" type="tns:EmployeeId"
038                         minOccurs="0" nillable="false" />
039                     <xs:element name="firstName" type="xs:string"
040                         minOccurs="0" nillable="false" />
041                     <xs:element name="lastName" type="xs:string"
042                         minOccurs="0" nillable="false" />
043                     <xs:element name="hourlyRate" type="xs:decimal"
044                         minOccurs="0" nillable="false" />
045                     <xs:element name="type" type="tns:EmployeeType"
```



x

```
050         <xs:sequence>
051             <xs:element name="eid" type="tns:EmployeeId"
052                 minOccurs="0" maxOccurs="unbounded" nillable="false" />
053         </xs:sequence>
054     </xs:complexType>
055
056     <xs:element name="employeeLookupRequest"
057         type="tns:EmployeeIdWrapper" />
058     <xs:element name="employeeServiceResponse"
059         type="tns:EmployeeInfoWrapper" />
060
061 </xs:schema>
062 </wsdl:types>
063
064 <wsdl:message name="employeeLookupRequest">
065     <wsdl:part element="tns:employeeLookupRequest"
066         name="employeeLookupRequest" />
067 </wsdl:message>
068
069 <wsdl:message name="employeeLookupResponse">
070     <wsdl:part element="tns:employeeServiceResponse"
071         name="employeeServiceResponse" />
072 </wsdl:message>
073
074 <wsdl:portType name="employeeLookupService">
075     <wsdl:documentation>Employee Lookup interface</wsdl:documentation>
076     <wsdl:operation name="employeeLookup">
077         <wsdl:input message="tns:employeeLookupRequest" />
078         <wsdl:output message="tns:employeeLookupResponse" />
079     </wsdl:operation>
080 </wsdl:portType>
081
082 <wsdl:binding name="employeeLookupBinding"
083     type="tns:employeeLookupService">
084     <soap:binding style="document"
085         transport="http://schemas.xmlsoap.org/soap/http" />
086     <wsdl:operation name="employeeLookup">
087         <soap:operation
088             soapAction="http://bestpay.payroll/employee/employeeLookup" />
089         <wsdl:input>
090             <soap:body parts="employeeLookupRequest" use="literal" />
091         </wsdl:input>
092         <wsdl:output>
093             <soap:body parts="employeeServiceResponse" use="literal" />
094         </wsdl:output>
095     </wsdl:operation>
096 </wsdl:binding>
097
098 <wsdl:service name="employeeLookupService">
099     <wsdl:port binding="tns:employeeLookupBinding"
100         name="employeeLookupPort">
101         <soap:address
102             location="http://localhost:8080/soap/ws/employee" />
103     </wsdl:port>
104 </wsdl:service>
105 </wsdl:definitions>
```

4.1 WSDL Validation

Predic8 provides a free online tool to validate the WSDL file. I used it to validate

employee.wsdl

. The validation results included reports for

TargetNamespace

Message

PortType

Operation

Binding

, and

Service

4.2 Generated Java Files

x

x

```
mvn clean install
```

to generate the Java stubs under

```
C:\MZheng_Java_workspace\Java Code Geek Examples\spring-boot-soap-hibernate\target\generated-sources\xjc
```

No modification needed to the generated files.

Generated Java Stubs

```
01 C:\MZheng_Java_workspace\Java Code Geek Examples\spring-boot-soap-hibernate\target\generated-
02 sources\xjc\payroll\bestpay\employee>dir
03 Volume in drive C is OSDisk
04 Volume Serial Number is 3A10-C6D4
05 Directory of C:\MZheng_Java_workspace\Java Code Geek Examples\spring-boot-soap-hibernate
\target\generated-sources\xjc\payroll\bestpay\employee
06
07 09/18/2018  08:13 PM    <DIR>          .
08 09/18/2018  08:13 PM    <DIR>          ..
09 09/18/2018  08:13 PM                2,174 EmployeeIdWrapper.java
10 09/18/2018  08:13 PM                4,480 EmployeeInfo.java
11 09/18/2018  08:13 PM                2,298 EmployeeInfoWrapper.java
12 09/18/2018  08:13 PM                1,549 EmployeeType.java
13 09/18/2018  08:13 PM                3,148 ObjectFactory.java
14 09/18/2018  08:13 PM                 530 package-info.java
15                6 File(s)              14,179 bytes
16                2 Dir(s)            16,617,570,304 bytes free
```

5. Employee Service

In this step, I will create a Spring service to look up the employee based on its identifier and transform it to

```
EmployeeInfo
```

5.1 Employee Entity

```
@javax.persistence.Entity
```

annotation defines that a class can be mapped to a table. An entity class must have a field annotated with

```
@Id
```

In this step, I will create an

```
Employee
```

entity to map to a

```
T_Employee
```

table.

Employee.java

```
001 package jcg.zheng.demo.springbootsoaphibernate.entity;
002
003 import javax.persistence.Column;
004 import javax.persistence.Entity;
005 import javax.persistence.EnumType;
006 import javax.persistence.Enumerated;
007 import javax.persistence.GeneratedValue;
008 import javax.persistence.Id;
009 import javax.persistence.Table;
010
011 import payroll.bestpay.employee.EmployeeType;
012
013 @Entity
014 @Table(name = "T_Employee")
015 public class Employee {
016     @Id
017     @GeneratedValue
018     @Column(name = "Id")
019     private Long id;
020
021     @Column(name = "First_Name")
022     private String firstName;
023
024     @Column(name = "Last_Name")
025     private String lastName;
026 }
```

x

x

```

039     return id;
040 }
041
042 public void setId(Long id) {
043     this.id = id;
044 }
045
046 public String getFirstName() {
047     return firstName;
048 }
049
050 public void setFirstName(String firstName) {
051     this.firstName = firstName;
052 }
053
054 public String getLastName() {
055     return lastName;
056 }
057
058 public void setLastName(String lastName) {
059     this.lastName = lastName;
060 }
061
062 public float getHourlyRate() {
063     return hourlyRate;
064 }
065
066 public void setHourlyRate(float hourlyRate) {
067     this.hourlyRate = hourlyRate;
068 }
069
070 public EmployeeType getType() {
071     return type;
072 }
073
074 public void setType(EmployeeType type) {
075     this.type = type;
076 }
077
078 public String getDepartment() {
079     return department;
080 }
081
082 public void setDepartment(String department) {
083     this.department = department;
084 }
085
086 public String getManagerId() {
087     return managerId;
088 }
089
090 public void setManagerId(String managerId) {
091     this.managerId = managerId;
092 }
093
094 @Override
095 public String toString() {
096     return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName +
097     ", hourlyRate=" + hourlyRate + ", type=" + type + ", department=" + department + ", managerId="
098     + managerId + "]\n";
099 }
100 }

```

5.2 EmployeeTransformer

In this step, I will create an

EmployeeTransformer

which transforms the

Employee

entity to the generated

EmployeeInfo

class.

EmployeeTransformer.java

```

01 package jcg.zheng.demo.springbootsoaphibernate.component;
02
03 import java.math.BigDecimal;
04
05 import org.springframework.beans.BeanUtils;
06 import org.springframework.stereotype.Component;
07
08 import jcg.zheng.demo.springbootsoaphibernate.aop.LoggableService;
09 import jcg.zheng.demo.springbootsoaphibernate.entity.Employee;
10 import payroll.bestpay.employee.EmployeeInfo;
11
12 @Component
13 public class EmployeeTransformer {

```

x

x

5.3 EmployeeRepository

Spring data scans the base package and all its sub-packages for any interfaces extending

@Repository

or one of its sub-interfaces. For each interface found, Spring creates the appropriate bean to handle invocation of the query methods. In this step, I will create an

EmployeeRepository

interface.

EmployeeRepository.java

```
01 package jcg.zheng.demo.springbootsoaphibernate.repository;
02
03 import org.springframework.data.jpa.repository.JpaRepository;
04 import org.springframework.stereotype.Repository;
05 import org.springframework.transaction.annotation.Transactional;
06
07 import jcg.zheng.demo.springbootsoaphibernate.entity.Employee;
08
09 @Repository
10 @Transactional
11 public interface EmployeeRepository extends JpaRepository<Employee, Long> {
12
13 }
```

5.4 EmployeeService

In this step, I will create an

EmployeeService

which injects

EmployeeRepository

and

EmployeeTransformer

to look up an employee and convert it to

EmployeeInfo

EmployeeService.java

```
01 package jcg.zheng.demo.springbootsoaphibernate.component;
02
03 import org.springframework.beans.factory.annotation.Autowired;
04 import org.springframework.stereotype.Service;
05
06 import jcg.zheng.demo.springbootsoaphibernate.aop.LoggableService;
07 import jcg.zheng.demo.springbootsoaphibernate.entity.Employee;
08 import jcg.zheng.demo.springbootsoaphibernate.repository.EmployeeRepository;
09 import payroll.bestpay.employee.EmployeeInfo;
10
11 @Service
12 public class EmployeeService {
13
14     @Autowired
15     private EmployeeRepository empRepo;
16
17     @Autowired
18     private EmployeeTransformer convertor;
19
20     @LoggableService
21     public EmployeeInfo employeeLookup(String employeeId) {
22
23         Employee employee = empRepo.findOne(Long.parseLong(employeeId));
24         if (employee != null) {
25             return convertor.convert(employee);
26         }
27
28         return null;
29     }
30
31 }
```

6. Employee SOAP Service

I will create an

EmployeeServiceEndPoint

x

x

In this step, I will create an

EmployeeServiceEndPoint

annotate it with

@Endpoint

. I will annotate the web method with

@PayloadRoot

and

@ResponsePayload

. It uses the generated Java stubs in step 4.2 and the

EmployeeService

created in step 5.4.

EmployeeServiceEndPoint.java

```
01 package jcg.zheng.demo.springbootsoaphibernate.soap;
02
03 import javax.xml.bind.JAXBElement;
04
05 import org.springframework.beans.factory.annotation.Autowired;
06 import org.springframework.ws.server.endpoint.annotation.Endpoint;
07 import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
08 import org.springframework.ws.server.endpoint.annotation.RequestPayload;
09 import org.springframework.ws.server.endpoint.annotation.ResponsePayload;
10
11 import jcg.zheng.demo.springbootsoaphibernate.aop.LoggableService;
12 import jcg.zheng.demo.springbootsoaphibernate.component.EmployeeService;
13 import payroll.bestpay.employee.EmployeeIdWrapper;
14 import payroll.bestpay.employee.EmployeeInfo;
15 import payroll.bestpay.employee.EmployeeInfoWrapper;
16 import payroll.bestpay.employee.ObjectFactory;
17
18 @Endpoint
19 public class EmployeeServiceEndPoint {
20
21     private static final String NAMESPACE_URI = "http://bestpay.payroll/employee";
22
23     @Autowired
24     private EmployeeService empService;
25
26     @PayloadRoot(namespace = NAMESPACE_URI, localPart = "employeeLookupRequest")
27     @ResponsePayload
28     @LoggableService
29     public JAXBElement<EmployeeInfoWrapper> employeeLookup(@RequestPayload JAXBElement<Employee
30         ObjectFactory factory = new ObjectFactory();
31         EmployeeInfoWrapper response = factory.createEmployeeInfoWrapper();
32         for (String empId : request.getValue().getEid()) {
33             EmployeeInfo found = empService.employeeLookup(empId);
34             if (found != null) {
35                 response.getEmployeeInfo().add(found);
36             }
37         }
38         return factory.createEmployeeServiceResponse(response);
39     }
40 }
41 }
```

6.2 SoapServiceConfiguration

In this step, I will create a

SoapServiceConfiguration

with

@EnableWs

and configure a SOAP service with

employee.wsdl

SoapServiceConfiguration.java

```
01 package jcg.zheng.demo.springbootsoaphibernate.soap;
02
03 import org.springframework.boot.web.servlet.ServletRegistrationBean;
04 import org.springframework.context.ApplicationContext;
05 import org.springframework.context.annotation.Bean;
06 import org.springframework.context.annotation.Configuration;
```

x

x

```
19 applicationContext {  
20     MessageDispatcherServlet servlet = new MessageDispatcherServlet();  
21     servlet.setApplicationContext(applicationContext);  
22  
23     return new ServletRegistrationBean(servlet, "/soap/ws/*");  
24 }  
25  
26 @Bean(name = "employee")  
27 public Wsdl11Definition employeeWsdl11Definition() {  
28     SimpleWsdl11Definition wsdl11Definition = new SimpleWsdl11Definition();  
29     wsdl11Definition.setWsdl(new ClassPathResource("/wsdl/employee.wsdl"));  
30  
31     return wsdl11Definition;  
32 }  
33 }
```

7. Demo

SoapUI is a great tool for testing web services. Click here to download it.

Start the Spring Boot application created in step 3.2. Verify that the service is started with four employees in the server log.

Server Log

```
01  
02  
03  
04  
05  
06  
07 :: Spring Boot :: (v1.5.15.RELEASE)  
08  
09 2018-09-21 16:17:25.114 INFO 7176 --- [main] z.d.s.SpringBootSoapHibernateApplicati  
10 2018-09-21 16:17:25.119 INFO 7176 --- [main] z.d.s.SpringBootSoapHibernateApplicati  
11 2018-09-21 16:17:25.196 INFO 7176 --- [main] ationConfigEmbeddedWebApplicationConte  
12 2018-09-21 16:17:26.670 INFO 7176 --- [main] trationDelegatesBeanPostProcessorCheck  
13 2018-09-21 16:17:26.720 INFO 7176 --- [main] .w.s.a.s.AnnotationActionEndpointMappi  
14 2018-09-21 16:17:26.925 INFO 7176 --- [main] trationDelegatesBeanPostProcessorCheck  
15 2018-09-21 16:17:28.133 INFO 7176 --- [main] s.b.c.e.t.TomcatEmbeddedServletContain  
16 2018-09-21 16:17:28.175 INFO 7176 --- [main] o.apache.catalina.core.StandardService  
17 2018-09-21 16:17:28.175 INFO 7176 --- [main] org.apache.catalina.core.StandardEngin  
18 2018-09-21 16:17:28.359 INFO 7176 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]  
19 2018-09-21 16:17:28.359 INFO 7176 --- [ost-startStop-1] o.s.web.context.ContextLoader  
20 2018-09-21 16:17:28.614 INFO 7176 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBea  
21 2018-09-21 16:17:28.615 INFO 7176 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBea  
22 2018-09-21 16:17:28.619 INFO 7176 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean  
23 2018-09-21 16:17:28.620 INFO 7176 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean  
24 2018-09-21 16:17:28.620 INFO 7176 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean  
25 2018-09-21 16:17:28.620 INFO 7176 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean  
26 2018-09-21 16:17:29.357 INFO 7176 --- [main] j.LocalContainerEntityManagerFactoryBe  
27 2018-09-21 16:17:29.380 INFO 7176 --- [main] o.hibernate.jpa.internal.util.LogHelpe  
28 name: default  
29 ...]  
30 2018-09-21 16:17:29.492 INFO 7176 --- [main] org.hibernate.Version  
31 2018-09-21 16:17:29.495 INFO 7176 --- [main] org.hibernate.cfg.Environment  
32 2018-09-21 16:17:29.497 INFO 7176 --- [main] org.hibernate.cfg.Environment  
33 2018-09-21 16:17:29.560 INFO 7176 --- [main] o.hibernate.annotations.common.Version  
34 2018-09-21 16:17:29.759 INFO 7176 --- [main] org.hibernate.dialect.Dialect  
35 2018-09-21 16:17:30.343 INFO 7176 --- [main] org.hibernate.tool.hbm2ddl.SchemaExpor  
36 2018-09-21 16:17:30.350 DEBUG 7176 --- [main] org.hibernate.SQL  
37 2018-09-21 16:17:30.351 DEBUG 7176 --- [main] org.hibernate.SQL  
38 2018-09-21 16:17:30.361 INFO 7176 --- [main] org.hibernate.tool.hbm2ddl.SchemaExpor  
39 2018-09-21 16:17:30.412 INFO 7176 --- [main] j.LocalContainerEntityManagerFactoryBe  
40 2018-09-21 16:17:31.259 INFO 7176 --- [main] s.w.s.m.m.a.RequestMappingHandlerAdapt  
41 2018-09-21 16:17:31.374 INFO 7176 --- [main] s.w.s.m.m.a.RequestMappingHandlerMappi  
42 2018-09-21 16:17:31.376 INFO 7176 --- [main] s.w.s.m.m.a.RequestMappingHandlerMappi  
43 2018-09-21 16:17:31.428 INFO 7176 --- [main] o.s.w.s.handler.SimpleUrlHandlerMappin  
44 2018-09-21 16:17:31.428 INFO 7176 --- [main] o.s.w.s.handler.SimpleUrlHandlerMappin  
45 2018-09-21 16:17:31.489 INFO 7176 --- [main] o.s.w.s.handler.SimpleUrlHandlerMappin  
46 2018-09-21 16:17:31.825 INFO 7176 --- [main] o.s.j.e.a.AnnotationMBeanExporter  
47 2018-09-21 16:17:31.880 INFO 7176 --- [main] s.b.c.e.t.TomcatEmbeddedServletContain  
48 2018-09-21 16:17:31.886 INFO 7176 --- [main] z.d.s.SpringBootSoapHibernateApplicati  
49 2018-09-21 16:17:31.929 DEBUG 7176 --- [main] org.hibernate.SQL  
50 2018-09-21 16:17:31.959 DEBUG 7176 --- [main] org.hibernate.SQL  
51 2018-09-21 16:17:31.961 DEBUG 7176 --- [main] org.hibernate.SQL  
52 2018-09-21 16:17:31.962 DEBUG 7176 --- [main] org.hibernate.SQL
```

In this step, I will create a new SOAP project:

1. Click

File

-->

New SOAP Project

2. Enter the Initial WSDL:

http://localhost:8080/soap/ws/employee.wsdl

and Click

OK

x

x

Request 1

4. A SOAP message is populated, replace

?

with the test data

5. Submit the request
6. Wait for a response

SoapUI Input

```
01 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
02   xmlns:emp="http://bestpay.payroll/employee">
03   <soapenv:Header/>
04   <soapenv:Body>
05     <emp:employeeLookupRequest>
06       <!-- Zero or more repetitions:-->
07       <emp:eid>1</emp:eid>
08       <emp:eid>2</emp:eid>
09       <emp:eid>3</emp:eid>
10       <emp:eid>4</emp:eid>
11     </emp:employeeLookupRequest>
12   </soapenv:Body>
13 </soapenv:Envelope>
```

SoapUI Output

```
01 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
02   <SOAP-ENV:Header/>
03   <SOAP-ENV:Body>
04     <ns2:employeeServiceResponse xmlns:ns2="http://bestpay.payroll/employee">
05       <ns2:employeeInfo>
06         <ns2:eid>1</ns2:eid>
07         <ns2:firstName>John</ns2:firstName>
08         <ns2:lastName>Zhang</ns2:lastName>
09         <ns2:hourlyRate>14</ns2:hourlyRate>
10         <ns2:type>Hourly</ns2:type>
11       </ns2:employeeInfo>
12       <ns2:employeeInfo>
13         <ns2:eid>2</ns2:eid>
14         <ns2:firstName>Dan</ns2:firstName>
15         <ns2:lastName>Zhao</ns2:lastName>
16         <ns2:hourlyRate>32</ns2:hourlyRate>
17         <ns2:type>Hourly</ns2:type>
18       </ns2:employeeInfo>
19       <ns2:employeeInfo>
20         <ns2:eid>3</ns2:eid>
21         <ns2:firstName>Tom</ns2:firstName>
22         <ns2:lastName>Zheng</ns2:lastName>
23         <ns2:hourlyRate>25</ns2:hourlyRate>
24         <ns2:type>Hourly</ns2:type>
25       </ns2:employeeInfo>
26       <ns2:employeeInfo>
27         <ns2:eid>4</ns2:eid>
28         <ns2:firstName>Mary</ns2:firstName>
29         <ns2:lastName>Zheng</ns2:lastName>
30         <ns2:hourlyRate>87</ns2:hourlyRate>
31         <ns2:type>Hourly</ns2:type>
32       </ns2:employeeInfo>
33     </ns2:employeeServiceResponse>
34   </SOAP-ENV:Body>
35 </SOAP-ENV:Envelope>
```

Capture the server.log and compare the time executed for each request. You will see that the first query took 60ms and all subsequent queries take less than 10ms. Hibernate improves the performance.

Server.log

```
01 2018-09-21 16:18:55.489 INFO 7176 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
02 2018-09-21 16:18:55.490 INFO 7176 --- [nio-8080-exec-1] o.s.w.t.http.MessageDispatcherServlet
03 2018-09-21 16:18:55.496 INFO 7176 --- [nio-8080-exec-1] o.s.ws.soap.saaj.SaajSoapMessageFactory
04 2018-09-21 16:18:55.511 INFO 7176 --- [nio-8080-exec-1] o.s.w.t.http.MessageDispatcherServlet
05 2018-09-21 16:18:55.750 DEBUG 7176 --- [nio-8080-exec-1] org.hibernate.SQL
06 2018-09-21 16:18:55.784 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeTransformer
07 2018-09-21 16:18:55.784 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeService
08 2018-09-21 16:18:55.786 DEBUG 7176 --- [nio-8080-exec-1] org.hibernate.SQL
09 2018-09-21 16:18:55.787 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeTransformer
10 2018-09-21 16:18:55.787 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeService
11 2018-09-21 16:18:55.789 DEBUG 7176 --- [nio-8080-exec-1] org.hibernate.SQL
12 2018-09-21 16:18:55.790 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeTransformer
13 2018-09-21 16:18:55.790 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeService
14 2018-09-21 16:18:55.791 DEBUG 7176 --- [nio-8080-exec-1] org.hibernate.SQL
15 2018-09-21 16:18:55.792 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeTransformer
16 2018-09-21 16:18:55.793 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.component.EmployeeService
17 2018-09-21 16:18:55.794 INFO 7176 --- [nio-8080-exec-1] j.z.d.s.soap.EmployeeServiceEndPoint
18 2018-09-21 16:22:33.348 DEBUG 7176 --- [nio-8080-exec-3] org.hibernate.SQL
19 2018-09-21 16:22:33.350 INFO 7176 --- [nio-8080-exec-3] j.z.d.s.component.EmployeeTransformer
20 2018-09-21 16:22:33.353 INFO 7176 --- [nio-8080-exec-3] j.z.d.s.component.EmployeeService
21 2018-09-21 16:22:33.353 DEBUG 7176 --- [nio-8080-exec-3] org.hibernate.SQL
22 2018-09-21 16:22:33.354 INFO 7176 --- [nio-8080-exec-3] j.z.d.s.component.EmployeeTransformer
```

x



35	2018-09-21	16:22:40.335	INFO	7176	---	[nio-8080-exec-4]	j.z.d.s.component.EmployeeService
36	2018-09-21	16:22:40.335	DEBUG	7176	---	[nio-8080-exec-4]	org.hibernate.SQL
37	2018-09-21	16:22:40.338	INFO	7176	---	[nio-8080-exec-4]	j.z.d.s.component.EmployeeTransformer
38	2018-09-21	16:22:40.338	INFO	7176	---	[nio-8080-exec-4]	j.z.d.s.component.EmployeeService
39	2018-09-21	16:22:40.339	DEBUG	7176	---	[nio-8080-exec-4]	org.hibernate.SQL
40	2018-09-21	16:22:40.341	INFO	7176	---	[nio-8080-exec-4]	j.z.d.s.component.EmployeeTransformer
41	2018-09-21	16:22:40.341	INFO	7176	---	[nio-8080-exec-4]	j.z.d.s.component.EmployeeService
42	2018-09-21	16:22:40.341	INFO	7176	---	[nio-8080-exec-4]	j.z.d.s.soap.EmployeeServiceEndPoint
43	2018-09-21	16:22:41.594	DEBUG	7176	---	[nio-8080-exec-5]	org.hibernate.SQL
44	2018-09-21	16:22:41.594	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.component.EmployeeService
45	2018-09-21	16:22:41.595	DEBUG	7176	---	[nio-8080-exec-5]	org.hibernate.SQL
46	2018-09-21	16:22:41.596	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.component.EmployeeTransformer
47	2018-09-21	16:22:41.596	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.component.EmployeeService
48	2018-09-21	16:22:41.596	DEBUG	7176	---	[nio-8080-exec-5]	org.hibernate.SQL
49	2018-09-21	16:22:41.598	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.component.EmployeeTransformer
50	2018-09-21	16:22:41.598	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.component.EmployeeService
51	2018-09-21	16:22:41.602	DEBUG	7176	---	[nio-8080-exec-5]	org.hibernate.SQL
52	2018-09-21	16:22:41.604	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.component.EmployeeTransformer
53	2018-09-21	16:22:41.604	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.component.EmployeeService
54	2018-09-21	16:22:41.604	INFO	7176	---	[nio-8080-exec-5]	j.z.d.s.soap.EmployeeServiceEndPoint

8. Spring Boot SOAP Service with Hibernate- Summary

In this article, I demonstrated how to create a SOAP web service using Hibernate in a Spring Boot Maven project in four steps:

1. Generate Java stubs from a WSDL file.
2. Create an

Entity

class and

Repository

Interface

3. Create a web service endpoint

4. Configure a

SimpleWsdl11Definition

with WSDL

9. Download the Source Code

This tutorial consists of a Spring Boot Maven project which creates a SOAP service with Hibernate from a WSDL file.

Download

You can download the full source code of this example here: [Spring Boot SOAP Service with Hibernate Example](#)

Tagged with: HIBERNATE

(No Ratings Yet) 1 Comment 803 Views Tweet it!

Do you want to know how to develop your skillset to become a **Java Rockstar?**

Subscribe to our newsletter to start Rocking right now!
To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more

Email address:

Your email address


☒ Receive Java & Developer job alerts in your Area

Sign up






LIKE THIS ARTICLE? READ MORE FROM JAVA CODE GEEKS




Microservices training

Ad Chris Richardson
javacodegeeks.com




Top 20 Libraries and APIs Java Developer should know

javacodegeeks.com



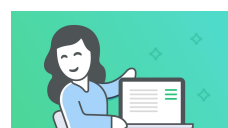
#1 Mind Map Tool for PC or Mac

Ad MindManager
javacodegeeks.com




Writing and Consuming SOAP Webservice with...

javacodegeeks.com




Free Writing Tool

Ad Grammarly
iavacodegeeks.com



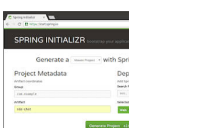
Restful API using Spring Rest & Spring Data JPA & H2 with...

iavacodegeeks.com



Query Databases Using Java Streams

iavacodegeeks.com



Top 5 Spring Features Jav Developers S

iavacodegeeks.com

1 Leave a Reply

Join the discussion...

1 0 0 0 0 0

1

☒ Subscribe

newest oldest most voted

fairymaiden

Nice post !

Guest

+ 0

Reply

3 months ago



NEWSLETTER

165,703 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.

Email address:

☒ Receive Java & Developer job alerts in your Area

Sign up

JOIN US

With **1,240,600** monthly 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!

✕

x

Courses

Minibooks

News

Resources

Tutorials

THE CODE GEEKS NETWORK

.NET Code Geeks

Java Code Geeks

System Code Geeks

Web Code Geeks

Android AsyncTask Example

Android OnClickListener Example

How to convert Character to String and a String to Character Array in Java

Java Inheritance example

Java write to File Example

java.io.FileNotFoundException – How to solve File Not Found Exception

java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception

java.lang.NoClassDefFoundError – How to solve No Class Def Found Error

JSON Example With Jersey + Jackson

Spring JdbcTemplate Example

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 news written by domain experts, articles, tutorials, reviews, announcements, code snippets and open source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples Java Code Geeks is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

Examples Java Code Geeks and all content copyright © 2010-2019, Exelixis Media P.C. | Terms of Use | Privacy Policy | Contact

g+

f

t

in

x