UNIT-V: Web Services for the APIs
Setting up Environment, Creating a New Project, Creating Models, Creating Data Access Object, Creating Controller, Develop Models with Hibernate, Installing MySQL, Create Database and Tables, Making DAO to Perform CRUD.

Web Services for the APIs:

The Internet is the worldwide connectivity of hundreds of thousands of computers of various types that belong to multiple networks. On the World Wide Web, a web service is a standardized method for propagating messages between client and server applications. A web service is a software module that is intended to carry out a specific set of functions. Web services in cloud computing can be found and invoked over the network.
The web service would be able to deliver functionality to the client that invoked the web service.

A web service is a set of open protocols and standards that allow data to be exchanged between different applications or systems. Web services can be used by software programs written in a variety of programming languages and running on a variety of platforms to exchange data via computer networks such as the Internet in a similar way to inter-process communication on a single computer.

Any software, application, or cloud technology that uses standardized web protocols (HTTP or HTTPS) to connect, interoperate, and exchange data messages – commonly XML (Extensible Markup Language) – across the internet is considered a web service.
Web services have the advantage of allowing programs developed in different languages to connect with one another by exchanging data over a web service between clients and servers. A client invokes a web service by submitting an XML request, which the service responds with an XML response.

Functions of Web Services

- It’s possible to access it via the internet or intranet networks.
- XML messaging protocol that is standardized.
- Operating system or programming language independent.
- Using the XML standard, it is self-describing.
- A simple location approach can be used to locate it.

Components of Web Service

XML and HTTP is the most fundamental web services platform. The following components are used by all typical web services:

SOAP (Simple Object Access Protocol)

SOAP stands for “Simple Object Access Protocol.” It is a transport-independent messaging protocol. SOAP is built on sending XML data in the form of SOAP Messages. A document known as an XML document is attached to each message. Only the structure of the XML document, not the content, follows a pattern. The best thing about Web services and SOAP is that everything is sent through HTTP, the standard web protocol.

A root element known as the element is required in every SOAP document. In an XML document, the root element is the first element. The “envelope” is separated into two halves. The header comes first, followed by the body. The routing data, or information that directs the XML document to which client it should be sent to, is contained in the header. The real message will be in the body.

UDDI (Universal Description, Discovery, and Integration)

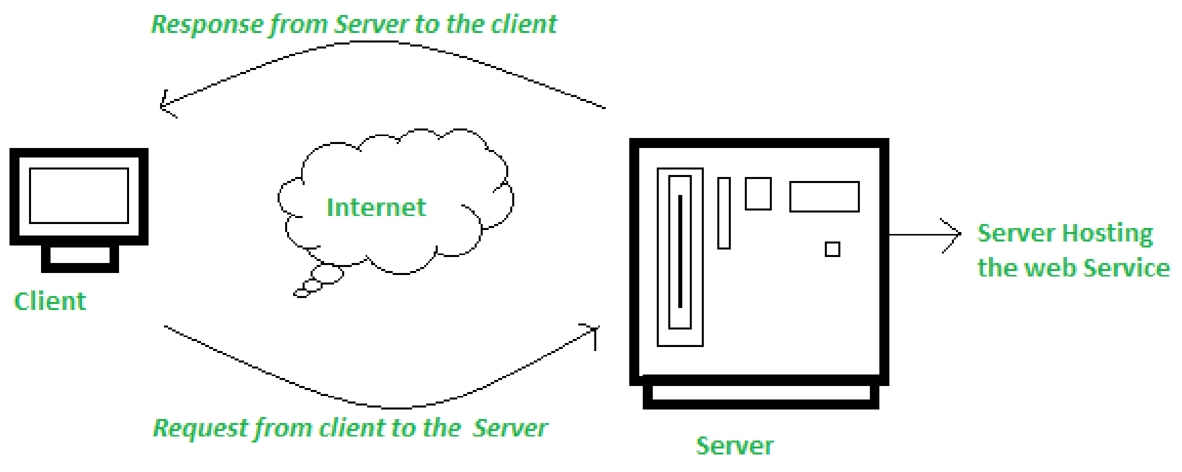
UDDI is a standard for specifying, publishing and discovering a service provider’s online services. It provides a specification that aids in the hosting of data via web services. UDDI provides a repository where WSDL files can be hosted so that a client application can discover a WSDL file to learn about the various actions that a web service offers. As a result, the client application will have full access to the UDDI, which serves as a database for all WSDL files.
The UDDI registry will hold the required information for the online service, just like a telephone directory has the name, address, and phone number of a certain individual. So that a client application may figure out where it is.

WSDL (Web Services Description Language)

If a web service can’t be found, it can’t be used. The client invoking the web service should be aware of the location of the web service. Second, the client application must understand what the web service does in order to invoke the correct web service. The WSDL, or Web services description language, is used to accomplish this. The WSDL file is another XML-based file that explains what the web service does to the client application. The client application will be able to understand where the web service is located and how to use it by using the WSDL document.

How Does Web Service Work?

The diagram depicts a very simplified version of how a web service would function. The client would use requests to send a sequence of web service calls to a server that would host the actual web service.


Response from Server to the client
Internet
Client
Server Hosting the web Service
Request from client to the Server
Server

Remote procedure calls are what are used to make these requests. Calls to methods hosted by the relevant web service are known as Remote Procedure Calls (RPC). Example: Flipkart offers a web service that displays prices for items offered on Flipkart.com. The front end or presentation layer can be written in .Net or Java, but the web service can be communicated using either programming language.

The data that is exchanged between the client and the server, which is XML, is the most important part of a web service design. XML (Extensible markup language) is a simple intermediate language that is understood by various programming languages. It is a counterpart to HTML. As a result, when programs communicate with one another, they do so using XML. This creates a common platform for applications written in different programming languages to communicate with one another.

For transmitting XML data between applications, web services employ SOAP (Simple Object Access Protocol). The data is sent using standard HTTP. A SOAP message is data that is sent from the web service to the application. An XML document is all that is contained in a SOAP message. The client application that calls the web service can be created in any programming language because the content is written in XML.

Features/Characteristics Of Web Service

Web services have the following features:

(a) XML Based: The information representation and record transportation layers of a web service employ XML. There is no need for networking, operating system, or platform binding when using XML. At the middle level, web offering-based applications are highly interoperable.

(b) Loosely Coupled: A customer of an internet service provider isn't necessarily directly linked to that service provider. The user interface for a web service provider can change over time

without impacting the user's ability to interact with the service provider. A strongly coupled system means that the patron's and server's decisions are inextricably linked, indicating that if one interface changes, the other should be updated as well.
A loosely connected architecture makes software systems more manageable and allows for easier integration between different structures.

(c) Capability to be Synchronous or Asynchronous: Synchronicity refers to the client's connection to the function's execution. The client is blocked and the client has to wait for the service to complete its operation, before continuing in synchronous invocations. Asynchronous operations allow a client to invoke a task and then continue with other tasks.
Asynchronous clients get their results later, but synchronous clients get their effect immediately when the service is completed. The ability to enable loosely linked systems requires asynchronous capabilities.

(d) Coarse-Grained: Object-oriented systems, such as Java, make their services available through individual methods. At the corporate level, a character technique is far too fine an operation to be useful. Building a Java application from the ground, necessitates the development of several fine-grained strategies, which are then combined into a rough-grained provider that is consumed by either a buyer or a service.
Corporations should be coarse-grained, as should the interfaces they expose. Web services generation is an easy approach to define coarse-grained services that have access to enough commercial enterprise logic.

(e) Supports Remote Procedure Call: Consumers can use an XML-based protocol to call procedures, functions, and methods on remote objects utilizing web services. A web service must support the input and output framework exposed by remote systems.
Enterprise-wide component development Over the last few years, JavaBeans (EJBs) and.NET Components have become more prevalent in architectural and enterprise deployments. A number of RPC techniques are used to allocate and access both technologies.
A web function can support RPC by offering its own services, similar to those of a traditional role, or by translating incoming invocations into an EJB or.NET component invocation.

(f) Supports Document Exchanges: One of XML's most appealing features is its simple approach to communicating with data and complex entities. These records can be as simple as talking to a current address or as complex as talking to an entire book or a Request for Quotation. Web administrations facilitate the simple exchange of archives, which aids incorporate reconciliation.
The web benefit design can be seen in two ways: **(i)** The first step is to examine each web benefit on-screen character in detail. **(ii)** The second is to take a look at the rapidly growing web benefit convention stack.

Advantages Of Web Service

Using web services has the following advantages:

(a) Business Functions can be exposed over the Internet: A web service is a controlled code component that delivers functionality to client applications or end-users. This capability can be accessed over the HTTP protocol, which means it can be accessed from anywhere on the internet. Because all apps are now accessible via the internet, Web services have become increasingly valuable. Because all apps are now accessible via the internet, Web services have become increasingly valuable. That is to say, the web service can be located anywhere on the internet and provide the required functionality.

(b) Interoperability: Web administrations allow diverse apps to communicate with one another and exchange information and services. Different apps can also make use of web services. A .NET application, for example, can communicate with Java web administrations and vice versa. To make the application stage and innovation self-contained, web administrations are used.

(c) Communication with Low Cost: Because web services employ the SOAP over HTTP protocol, you can use your existing low-cost internet connection to implement them. Web services can be developed using additional dependable transport protocols, such as FTP, in addition to SOAP over HTTP.

(d) A Standard Protocol that Everyone Understands: Web services communicate via a defined industry protocol. In the web services protocol stack, all four layers (Service Transport, XML Messaging, Service Description, and Service Discovery) use well-defined protocols.

(e) Reusability: A single web service can be used simultaneously by several client applications.

<u>Difference between Web Services and APIs:</u>

Web Services	**Web API**
Web services are a type of API, which must be accessed through a network connection.	APIs are application interfaces, implying that one application can communicate with another application in a standardized manner.
Web service is used for REST, SOAP and XML-RPC for communication.	API is used for any style of communication.
All Web services are APIs.	APIs are not web services.
It doesn’t have a lightweight design, and needs a SOAP convention to send or receive data over the system.	It has a light-weight architecture furthermore, useful for gadgets which have constrained transmission capacity like smart phones.

It provides support only for the HTTP protocol.	It provides support for the HTTP/s protocol: URL Request/Response Headers, and so on.
It is not open source, however, can be devoured by any customer that comprehends xml.	It is open source and also ships with .NET framework.
Web service supports only XML.	API supports XML and JSON.
Web Services can be hosted on IIS.	Web API can be hosted only on IIS and self.

Hibernate – Web Application

A web application with hibernate is easier. A JSP page is the best way to get user inputs. Those inputs are passed to the servlet and finally, it is inserted into the database by using hibernate. Here JSP page is used for the presentation logic. The Servlet class is meant for the controller layer. DAO class is meant for database access codes.

As Hibernate – Web Application is going to get prepared, the project is going to get prepared as a dynamic web project. Following jars need to be placed in the WEB-INF/lib folder

Front-end JSP page that gets input from user

register.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Registration Form</title>
</head>
<style>
```

```
body {
   font-family: Verdana, Geneva, sans-serif;
   font-size: 14px;
   background: #006400;
}
.clearfix {
   &:after {
      content: "";
      display: block;
      clear: both;
      visibility: hidden;
      height: 0;
   }
}
.form_wrapper {
   background: #fff;
   width: 600px;
   max-width: 100%;
   box-sizing: border-box;
   padding: 25px;
   margin: 8% auto 0;
   position: relative;
   z-index: 1;
   border-top: 5px solid $yellow;
   -webkit-box-shadow: 0 0 3px rgba(0, 0, 0, 0.1);
   -moz-box-shadow: 0 0 3px rgba(0, 0, 0, 0.1);
   box-shadow: 0 0 3px rgba(0, 0, 0, 0.1);
   -webkit-transform-origin: 50% 0%;
   transform-origin: 50% 0%;
   -webkit-transform: scale3d(1, 1, 1);
   transform: scale3d(1, 1, 1);
   -webkit-transition: none;
   transition: none;
   -webkit-animation: expand 0.8s 0.6s ease-out forwards;
   animation: expand 0.8s 0.6s ease-out forwards;
   opacity: 0;
   h2 {
      font-size: 1.5em;
      line-height: 1.5em;
      margin: 0;
   }
   .title_container {
      text-align: center;
```

```
    padding-bottom: 15px;
}
h3 {
    font-size: 1.1em;
    font-weight: normal;
    line-height: 1.5em;
    margin: 0;
}
label {
    font-size: 12px;
}
.row {
    margin: 10px -15px;
    >div {
        padding: 0 15px;
        box-sizing: border-box;
    }
}
.col_half {
    width: 50%;
    float: left;
}
.input_field {
    position: relative;
    margin-bottom: 20px;
    -webkit-animation: bounce 0.6s ease-out;
      animation: bounce 0.6s ease-out;
    >span {
        position: absolute;
        left: 0;
        top: 0;
        color: #333;
        height: 100%;
        border-right: 1px solid $grey;
        text-align: center;
        width: 30px;
        >i {
            padding-top: 10px;
        }
    }
}
.textarea_field {
    >span {
        >i {
            padding-top: 10px;
        }
    }
}
input {
&[type="text"], &[type="email"], &[type="password"] {
  width: 100%;
  padding: 8px 10px 9px 35px;
  height: 35px;
  border: 1px solid $grey;
  box-sizing: border-box;
  outline: none;
  -webkit-transition: all 0.30s ease-in-out;
  -moz-transition: all 0.30s ease-in-out;
  -ms-transition: all 0.30s ease-in-out;
  transition: all 0.30s ease-in-out;
}
&[type="text"]:hover, &[type="email"]:hover, &[type="password"]:hover {
  background: #fafafa;
}
&[type="text"]:focus, &[type="email"]:focus, &[type="password"]:focus {
  -webkit-box-shadow: 0 0 2px 1px rgba(255, 169, 0, 0.5);
  -moz-box-shadow: 0 0 2px 1px rgba(255, 169, 0, 0.5);
  box-shadow: 0 0 2px 1px rgba(255, 169, 0, 0.5);
  border: 1px solid $yellow;
  background: #fafafa;
}
&[type="submit"] {
    background: $yellow;
    height: 35px;
    line-height: 35px;
    width: 100%;
    border: none;
    outline: none;
    cursor: pointer;
    color: #fff;
    font-size: 1.1em;
    margin-bottom: 10px;
    -webkit-transition: all 0.30s ease-in-out;
    -moz-transition: all 0.30s ease-in-out;
    -ms-transition: all 0.30s ease-in-out;
    transition: all 0.30s ease-in-out;
```

```
        &:hover {
            background: darken($yellow,7%);
        }
        &:focus {
            background: darken($yellow,7%);
        }
    }
    &[type="checkbox"], &[type="radio"] {
      border: 0;
      clip: rect(0 0 0 0);
      height: 1px;
      margin: -1px;
      overflow: hidden;
      padding: 0;
      position: absolute;
      width: 1px;
    }
  }
}
.form_container {
    .row {
        .col_half.last {
            border-left: 1px solid $grey;
        }
    }
}

@-webkit-keyframes check {
  0% { height: 0; width: 0; }
  25% { height: 0; width: 7px; }
  50% { height: 20px; width: 7px; }
}

@keyframes check {
  0% { height: 0; width: 0; }
  25% { height: 0; width: 7px; }
  50% { height: 20px; width: 7px; }
}

@-webkit-keyframes expand {
    0% { -webkit-transform: scale3d(1,0,1); opacity:0; }
    25% { -webkit-transform: scale3d(1,1.2,1); }
    50% { -webkit-transform: scale3d(1,0.85,1); }
```

```
   75% { -webkit-transform: scale3d(1,1.05,1); }
   100% { -webkit-transform: scale3d(1,1,1);  opacity:1; }
}

@keyframes expand {
   0% { -webkit-transform: scale3d(1,0,1); transform: scale3d(1,0,1);  opacity:0; }
   25% { -webkit-transform: scale3d(1,1.2,1); transform: scale3d(1,1.2,1); }
   50% { -webkit-transform: scale3d(1,0.85,1); transform: scale3d(1,0.85,1); }
   75% { -webkit-transform: scale3d(1,1.05,1); transform: scale3d(1,1.05,1); }
   100% { -webkit-transform: scale3d(1,1,1); transform: scale3d(1,1,1);  opacity:1; }
}


@-webkit-keyframes bounce {
   0% { -webkit-transform: translate3d(0,-25px,0); opacity:0; }
   25% { -webkit-transform: translate3d(0,10px,0); }
   50% { -webkit-transform: translate3d(0,-6px,0); }
   75% { -webkit-transform: translate3d(0,2px,0); }
   100% { -webkit-transform: translate3d(0,0,0); opacity: 1; }
}

@keyframes bounce {
    0% { -webkit-transform: translate3d(0,-25px,0); transform: translate3d(0,-25px,0); opacity:0;
}
   25% { -webkit-transform: translate3d(0,10px,0); transform: translate3d(0,10px,0); }
   50% { -webkit-transform: translate3d(0,-6px,0); transform: translate3d(0,-6px,0); }
   75% { -webkit-transform: translate3d(0,2px,0); transform: translate3d(0,2px,0); }
   100% { -webkit-transform: translate3d(0,0,0); transform: translate3d(0,0,0); opacity: 1; }
}
@media (max-width: 600px) {
   .form_wrapper {
      .col_half {
         width: 100%;
         float: none;
      }
   }
   .bottom_row {
      .col_half {
         width: 50%;
         float: left;
      }
   }
```

```
}
</style>
<body>
<div class="form_wrapper">
  <div class="form_container">
    <div class="title_container">
    <h1>GeekUser Registration Form</h1>
    <form action="register" method="post">
        <table cellpadding="3pt">
            <tr>
                <td>User Name :</td>
                <td><input type="text" name="userName" size="50" /></td>
            </tr>
            <tr>
                <td>Password :</td>
                <td><input type="password" name="password1" size="50" /></td>
            </tr>

            <tr>
                <td>Confirm Password :</td>
                <td><input type="password" name="password2" size="50" /></td>
            </tr>
            <tr>
                <td>email :</td>
                <td><input type="text" name="email" size="50" /></td>
            </tr>
            <tr>
                <td>Phone :</td>
                <td><input type="text" name="phone" size="50" /></td>
            </tr>
            <tr>
                <td>City :</td>
                <td><input type="text" name="city" size="50" /></td>
            </tr>
        </table>
        <p />
        <input type="submit" value="Register" />
    </form>
    </div>
    </div>
    </div>
</body>
</html>
```

Creating Java class namely a POJO class, a controller class, DAO class

Controller class > GeekUserControllerServlet.java

```
import com.gfg.hibernate.dao.GeekUserDAO;

public class GeekUserControllerServlet extends HttpServlet {

  private static final long serialVersionUID = 1L;

  protected void doPost(HttpServletRequest request,
      HttpServletResponse response) throws ServletException, IOException {

    // From register.jsp, the controls
    // are passed to this controller servlet
    String userName = request.getParameter("userName");
    String password = request.getParameter("password1");
    String email = request.getParameter("email");
    String phone = request.getParameter("phone");
    String city = request.getParameter("city");
    HttpSession session = request.getSession(true);
    try {
      GeekUserDAO userDAO = new GeekUserDAO();
      // Via DAO class addGeekUser method,
      // the data is inserted into the table
      userDAO.addGeekUser(userName, password, email, phone, city);
      // After that it is redirected to the success page
      response.sendRedirect("Success");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Let us see DAO class > GeekUserDAO.java

```
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import com.gfg.hibernate.bean.GeekUser;
public class GeekUserDAO {
    // Method to insert GeekUser in the database
    public Integer addGeekUser(String userName,
     String password,
     String email,
     String phone,
     String city) {
        SessionFactory sessionFactory;
          try {
             sessionFactory = new Configuration().configure().buildSessionFactory();
          } catch (Throwable ex) {
             System.err.println("Failed to create sessionFactory object." + ex);
             throw new ExceptionInInitializerError(ex);
          }
       Session session = sessionFactory.openSession();
       Transaction tx = null;
       Integer employeeID = null;
       try {
          tx = session.beginTransaction();
          GeekUser geekUser = new GeekUser(userName, password, email,phone,city);
          employeeID = (Integer) session.save(geekUser);
          tx.commit();
       } catch (HibernateException e) {
          if (tx != null)
             tx.rollback();
          e.printStackTrace();
       } finally {
          session.close();
       }
       return employeeID;
    }
}
```

Via DAO method, the details are getting inserted into the database. We see that via GeekUser bean class, the data is sent. Let us see the POJO class

POJO class > GeekUser.java

```
public class GeekUser {
    // each and every attribute should have
    // the respective column in the database
    private int id;
    private String geekUserName;
    private String password;
    private String geekEmail;
    private String phone;
    private String city;
    public GeekUser() {

    }
    // parameterized constructors are
    // necessary to carry the details
    public GeekUser(String userName,
     String password,
     String email,
     String phone,
     String city) {
        this.geekUserName = userName;
        this.password = password;
        this.geekEmail = email;
        this.phone = phone;
        this.city = city;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getGeekUserName() {
        return geekUserName;
    }

    public void setGeekUserName(String userName) {
        this.geekUserName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password1) {
        this.password = password1;
    }
```

```
    public String getGeekEmail() {
        return geekEmail;
    }

    public void setGeekEmail(String email) {
        this.geekEmail = email;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

}
```

Once the details are added successfully, it is redirected to **SuccessPage.java**

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SuccessPage extends HttpServlet {

    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        PrintWriter writer = response.getWriter();

                writer.println("<html>" + "<body bgcolor='#006400'>" + "<center><h3><font
color='white'>"
                        + "GeekUser details are added successfully via Hibernate. " +
"</font></h3></center>" + "</body>"
            + "</html>");
    }

}
```

Let us see the output of the flow:

Let us check the table data also parallelly

Hibernate makes the process very easier.

A bean class(Here it is GeekUser.java) is required. Each and every attribute of a bean class has to be mapped in a hbm(hibernate mapping file) file. It is an XML file where each and every attribute are mapped perfectly to the table in the database.

geekuser.hbm.xml (Mapping file which maps bean (POJO) class attributes to corresponding table)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <!-- GeekUser bean class is mapped to GEEKUSER table -->
   <class name="com.gfg.hibernate.bean.GeekUser" table="GEEKUSER">
      <!-- This makes id attribute to be of int datatype and Primary Key -->
      <id column="id" name="id" type="int">
         <generator class="native"/>
        </id>
      <!-- geekUserName,password,geekEmail,phone and city
          are the other attributes that need to be mapped -->
      <property column="geekUserName" name="geekUserName" type="java.lang.String" />
      <property column="password" name="password" type="string" />
      <property column="geekEmail" name="geekEmail" type="java.lang.String" />
      <property column="phone" name="phone" type="java.lang.String" />
      <property column="city" name="city" type="java.lang.String" />
   </class>
```

```
</hibernate-mapping>
```

Main hibernate configuration file > hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
      <!-- As we are going to connect with mysql, this is required -->
      <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
                                                                                    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/geeksforgeeks</property>
      <property name="hibernate.connection.username">root</property>
      <!-- Provide your correct password here -->
      <property name="hibernate.connection.password">XXXX</property>
      <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <!-- In order to show the SQL query in console, this is required -->
      <property name="show_sql">true</property>
      <property name="format_sql">true</property>
      <property name="hbm2ddl.auto">create </property>
      <!-- Specification of exact hibernate mapping file need to be specified -->
      <mapping resource="com/gfg/hibernate/bean/geekuser.hbm.xml" />
   </session-factory>
</hibernate-configuration>
```

Domain Model with Hibernate:

Rich Domain Model in the context of software development and Hibernate is the model where the business logic is encapsulated directly within the domain objects, rather than being separated into the services or other layers. It can allow the domain model to enforce the business rules and ensure data consistency.

In traditional applications, business logic can be spread across multiple layers, leading to a lack of cohesion and harder maintenance. The Rich Domain Model centralizes this logic within the domain entities. Making the code more intuitive and aligned with real-world business operations. This encapsulation aids in maintaining the business rule consistency and logic encapsulation.

For example, consider the simple banking application where the Account entities can handle operations like deposits and withdrawals internally rather than relying on external services to enforce the rules of the application.

Prerequisites:

- Basic knowledge of Java and Spring Boot.
- Maven tool for dependency management.
- The Integrated Development Environment like IntelliJ IDEA.

Implementation of Rich Domain Model with Hibernate

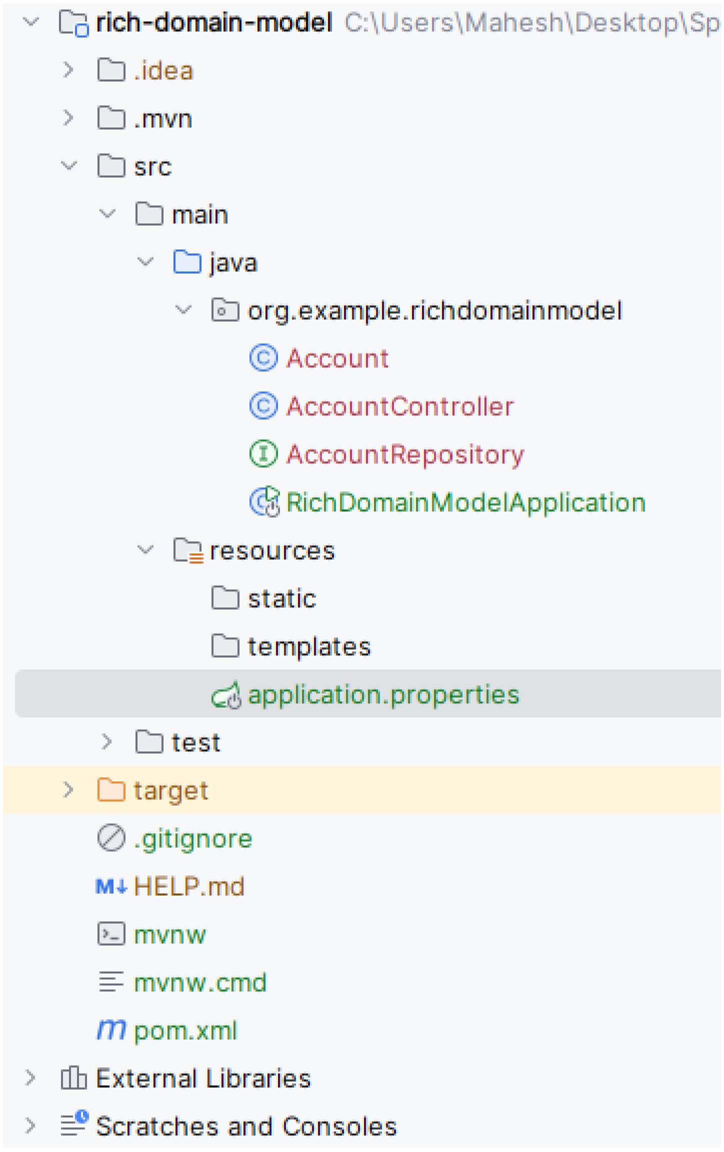
Step 1: Create Spring Boot Project

Create a new Spring project using Spring Initializr and add the below dependencies into the project.

Dependencies:

- Spring Web
- Spring Data JPA
- MySQL Driver
- Lombok
- Spring DevTools

After creating the project, then the file structure will look like the below image.


```
rich-domain-model C:\Users\Mahesh\Desktop\Sp
  .idea
  .mvn
  src
    main
      java
        org.example.richdomainmodel
          Account
          AccountController
          AccountRepository
          RichDomainModelApplication
      resources
        static
        templates
        application.properties
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

Step 2: Application Properties

Open the application.properties file and add the configuration of the MySQL database and Hibernate.

```
spring.application.name=rich-domain-model
# MySQL Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/example
spring.datasource.username=root
spring.datasource.password=

# JPA/Hibernate properties
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
```

Step 3: Create the Account Entity

We will create the Account class with the methods to handle the deposits and withdrawals of the application.

Go to src > main > java > org.example.richdomainmodel > Account and put the below code.

Java

```java
package org.example.richdomainmodel;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Data;

import java.math.BigDecimal;

@Entity
@Data
public class Account {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Long id;
  private BigDecimal balance;

  public Account() {
    this.balance = BigDecimal.ZERO;
  }

  public void deposit(BigDecimal amount) {
    if (amount.compareTo(BigDecimal.ZERO) <= 0) {
      throw new IllegalArgumentException("Deposit amount must be positive");
    }
    this.balance = this.balance.add(amount);
  }

  public void withdraw(BigDecimal amount) {
    if (amount.compareTo(this.balance) > 0) {
      throw new IllegalStateException("Insufficient funds");
    }
    this.balance = this.balance.subtract(amount);
  }
}
```

Step 4: Create the AccountRepository

Go to src > main > java > org.example.richdomainmodel > AccountRepository and put the below code.

Java

```java
package org.example.richdomainmodel;

import org.springframework.data.jpa.repository.JpaRepository;

public interface AccountRepository extends JpaRepository<Account, Long> {
}
```

Step 5: Create AccountController Class

Now, we will create the Simple RESTful endpoints to handle the HTTP requests of the application.

Go to src > main > java > org.example.richdomainmodel > AccountController and put the below code.

Java

```java
package org.example.richdomainmodel;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.math.BigDecimal;
import java.util.Optional;

@RestController
@RequestMapping("/accounts")
public class AccountController {

  @Autowired
  private AccountRepository accountRepository;

  // Create a new account
  @PostMapping
  public ResponseEntity<Account> createAccount(@RequestBody Account account) {

        Account savedAccount = accountRepository.save(account);
        return ResponseEntity.ok(savedAccount);
    }

    // Deposit money into an account
    @PostMapping("/{id}/deposit")
    public ResponseEntity<?> deposit(@PathVariable Long id, @RequestParam BigDecimal amount) {
        Optional<Account> accountOptional = accountRepository.findById(id);
        if (!accountOptional.isPresent()) {
            return ResponseEntity.notFound().build();
        }
        Account account = accountOptional.get();
        account.deposit(amount);
        accountRepository.save(account);
        return ResponseEntity.ok(account);
    }

    // Withdraw money from an account
    @PostMapping("/{id}/withdraw")
    public ResponseEntity<?> withdraw(@PathVariable Long id, @RequestParam BigDecimal amount) {
        Optional<Account> accountOptional = accountRepository.findById(id);
        if (!accountOptional.isPresent()) {
            return ResponseEntity.notFound().build();
        }
        Account account = accountOptional.get();
        try {
            account.withdraw(amount);
            accountRepository.save(account);
            return ResponseEntity.ok(account);
        } catch (IllegalStateException e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }

    // Retrieve the balance of an account
    @GetMapping("/{id}")
    public ResponseEntity<?> getAccount(@PathVariable Long id) {
        Optional<Account> account = accountRepository.findById(id);
        return account
                .map(a -> ResponseEntity.ok("Account ID: " + a.getId() + ", Balance: " + a.getBalance()))
                .orElseGet(() -> ResponseEntity.notFound().build());
    }
}
```

Step 6: Main Class

No changes are required in the main class.

Java

```
package org.example.richdomainmodel;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

```

```
@SpringBootApplication
public class RichDomainModelApplication {

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

}
```

pom.xml:

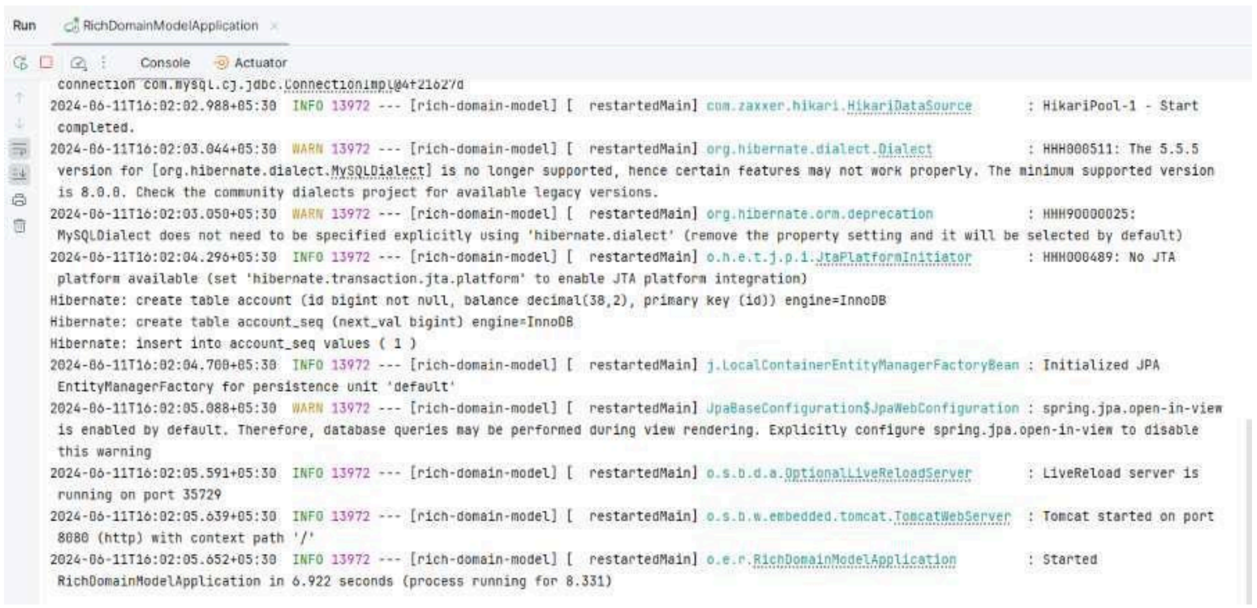
XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project
 xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                                    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>3.3.0</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>org.example</groupId>
   <artifactId>rich-domain-model</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>rich-domain-model</name>
   <description>rich-domain-model</description>
   <properties>
      <java.version>17</java.version>
   </properties>
   <dependencies>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-data-jpa</artifactId>
      </dependency>
      <dependency>
```

```xml
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

Step 7: Run the application

Now, run the application and it will start at port 8080.



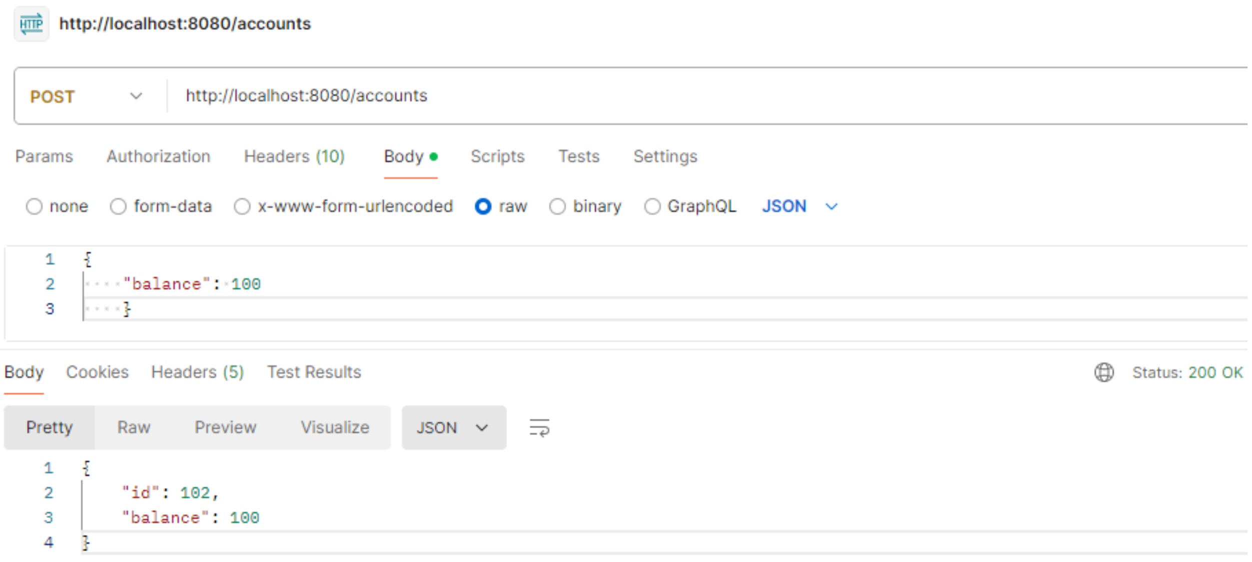
```
 connection com.mysql.cj.jdbc.ConnectionImpl@4f21627d
2024-06-11T16:02:02.988+05:30  INFO 13972 --- [rich-domain-model] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start
 completed.
2024-06-11T16:02:03.044+05:30  WARN 13972 --- [rich-domain-model] [  restartedMain] org.hibernate.dialect.Dialect            : HHH000511: The 5.5.5
 version for [org.hibernate.dialect.MySQLDialect] is no longer supported, hence certain features may not work properly. The minimum supported version
 is 8.0.0. Check the community dialects project for available legacy versions.
2024-06-11T16:02:03.050+05:30  WARN 13972 --- [rich-domain-model] [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025:
 MySQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)
2024-06-11T16:02:04.296+05:30  INFO 13972 --- [rich-domain-model] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA
 platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
Hibernate: create table account (id bigint not null, balance decimal(38,2), primary key (id)) engine=InnoDB
Hibernate: create table account_seq (next_val bigint) engine=InnoDB
Hibernate: insert into account_seq values ( 1 )
2024-06-11T16:02:04.700+05:30  INFO 13972 --- [rich-domain-model] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA
 EntityManagerFactory for persistence unit 'default'
2024-06-11T16:02:05.088+05:30  WARN 13972 --- [rich-domain-model] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view
 is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable
 this warning
2024-06-11T16:02:05.591+05:30  INFO 13972 --- [rich-domain-model] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is
 running on port 35729
2024-06-11T16:02:05.639+05:30  INFO 13972 --- [rich-domain-model] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port
 8080 (http) with context path '/'
2024-06-11T16:02:05.652+05:30  INFO 13972 --- [rich-domain-model] [  restartedMain] o.e.r.RichDomainModelApplication         : Started
 RichDomainModelApplication in 6.922 seconds (process running for 8.331)
```

Step 8: Testing the Endpoints

1. Create Account Endpoint

POST http://localhost:8080/accounts

Output:



http://localhost:8080/accounts

POST http://localhost:8080/accounts

Body: ( ) none ( ) form-data ( ) x-www-form-urlencoded [✓ raw] ( ) binary ( ) GraphQL JSON

```
{
····"balance":·100
····}
```

Status: 200 OK

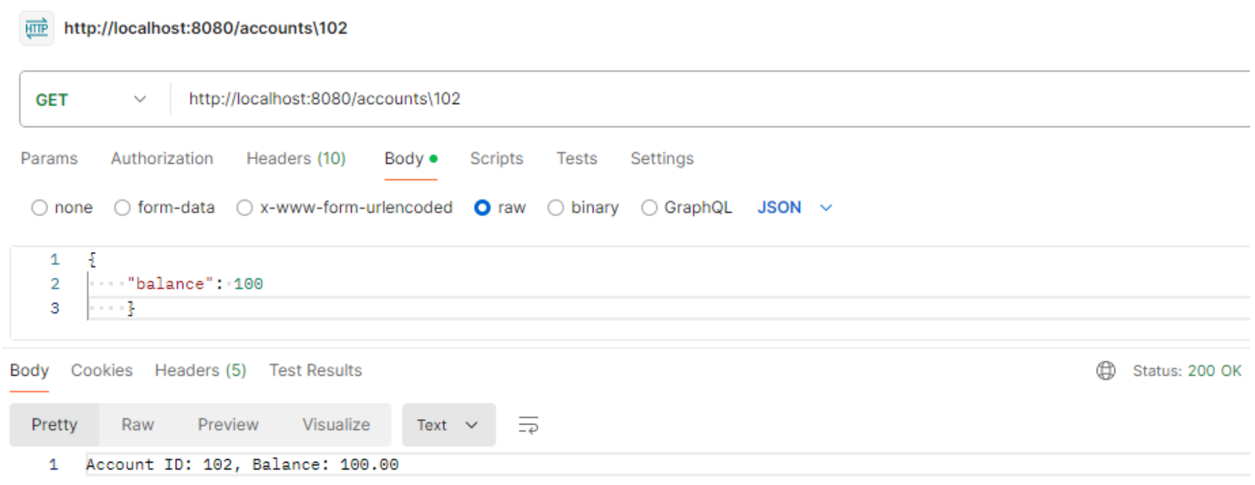
JSON

```
{
    "id": 102,
    "balance": 100
}
```

2. Retrieve Account balance through Id

GET http://localhost:8080/accounts/102

Output:



http://localhost:8080/accounts\102

GET http://localhost:8080/accounts\102

Params Authorization Headers (10) Body Scripts Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "balance": 100
    }
```

Body Cookies Headers (5) Test Results Status: 200 OK

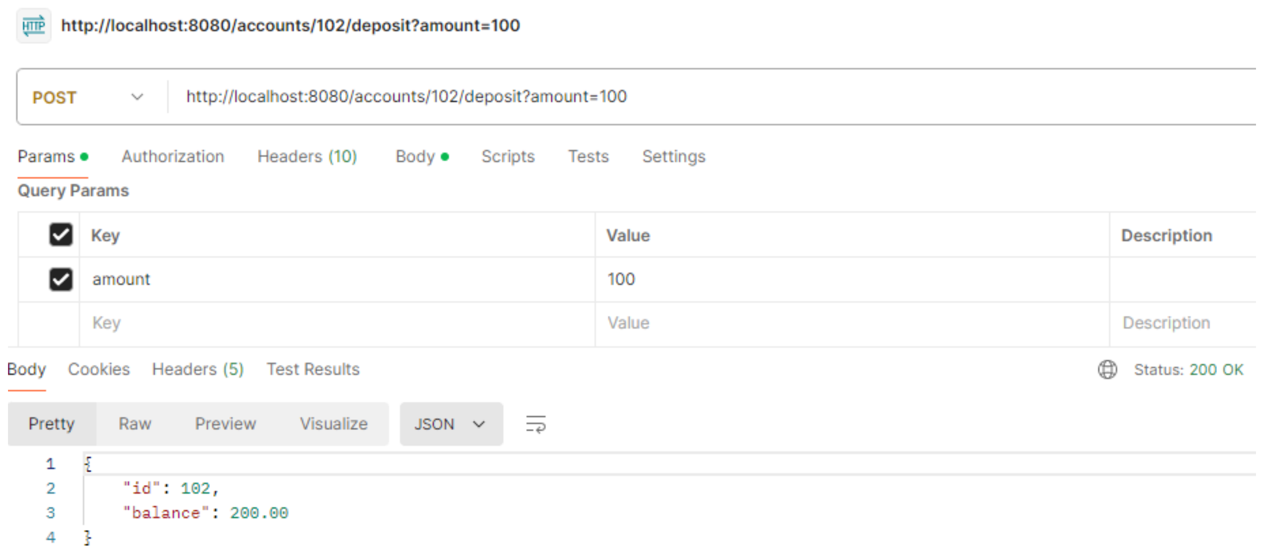
Pretty Raw Preview Visualize Text

```
Account ID: 102, Balance: 100.00
```

4. Deposit Endpoint

POST http://localhost:8080/accounts/102/deposit?amount=100

Output:



http://localhost:8080/accounts/102/deposit?amount=100

POST http://localhost:8080/accounts/102/deposit?amount=100

Params Authorization Headers (10) Body Scripts Tests Settings

Query Params

	Key	Value	Description
✓	amount	100	
	Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK

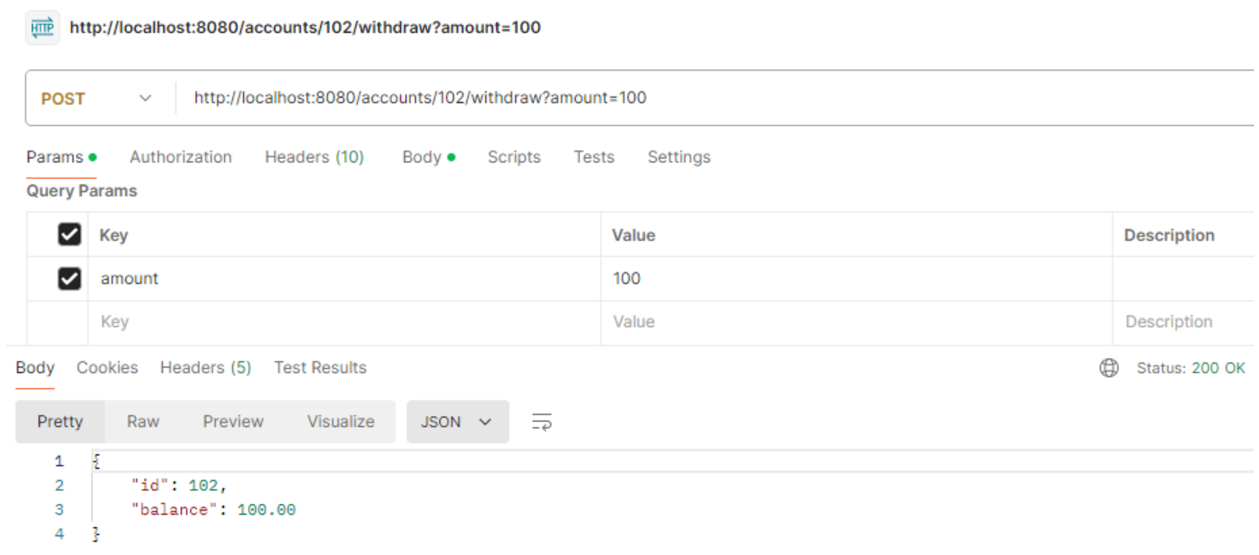
Pretty Raw Preview Visualize JSON

```
{
    "id": 102,
    "balance": 200.00
}
```

5. Withdraw Endpoint

POST http://localhost:8080/accounts/102/withdraw?amount=100

Output:


http://localhost:8080/accounts/102/withdraw?amount=100

POST http://localhost:8080/accounts/102/withdraw?amount=100

✓	Key	Value	Description
✓	amount	100	
	Key	Value	Description

```
{
    "id": 102,
    "balance": 100.00
}
```

This example demonstrates the embedded critical business logic within the domain model itself and adhering to the principles of the Rich Domain Model with Hibernate in the Spring Boot application.

Feeling lost in the vast world of Backend Development? It's time for a change! Join our Java Backend Development - Live Course and embark on an exciting journey to master backend development efficiently and on schedule.

How to install MySQL:

MySQL is one of the most popular relational database management software that is widely used in today's industry. It provides multi-user access support with various storage engines. It is backed by Oracle Company. In this section, we are going to learn how we can download and install MySQL for beginners.

Prerequisites

The following requirements should be available in your system to work with MySQL:

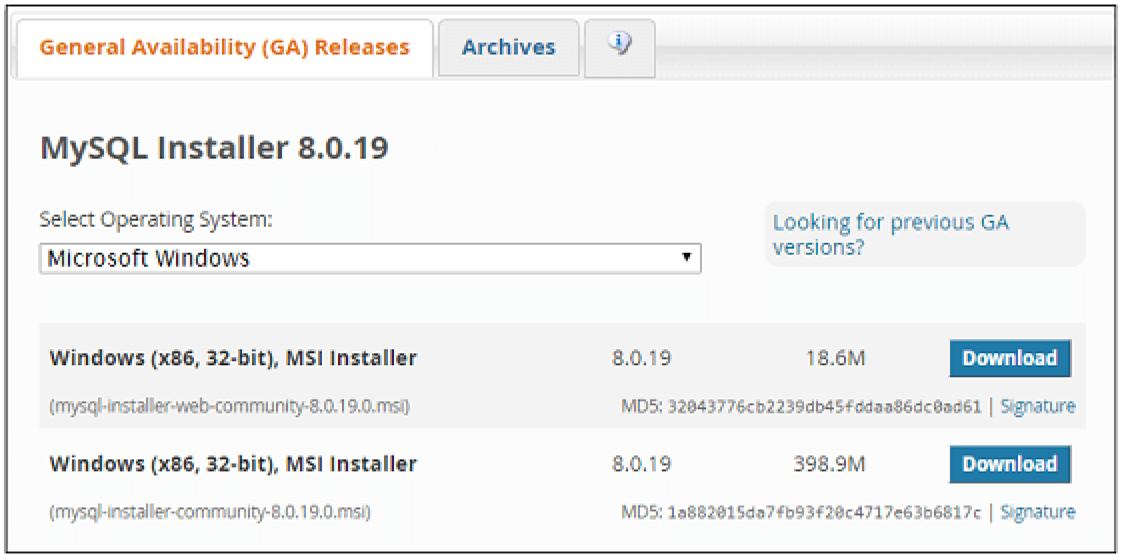
- MySQL Setup Software
- Microsoft .NET Framework 4.5.2
- Microsoft Visual C++ Redistributable for Visual Studio 2019
- RAM 4 GB (6 GB recommended)

Download MySQL:

Follow these steps:

Step 1: Go to the official website of MySQL and download the community server edition software. Here, you will see the option to choose the Operating System, such as Windows.

Step 2: Next, there are two options available to download the setup. Choose the version number for the MySQL community server, which you want. If you have good internet connectivity, then choose the mysql-installer-web-community. Otherwise, choose the other one.



MySQL Installer 8.0.19

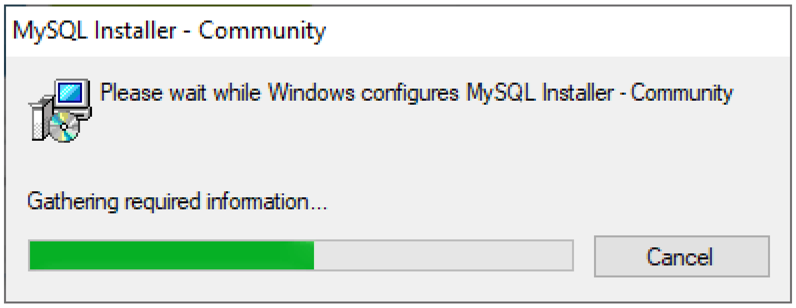
Select Operating System:
Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer	8.0.19	18.6M	Download
(mysql-installer-web-community-8.0.19.0.msi)	MD5: 32043776cb2239db45fddaa86dc0ad61 | Signature		
Windows (x86, 32-bit), MSI Installer	8.0.19	398.9M	Download
(mysql-installer-community-8.0.19.0.msi)	MD5: 1a882015da7fb93f20c4717e63b6817c | Signature		

Installing MySQL on Windows

Step 1: After downloading the setup, unzip it anywhere and double click the MSI installer .exe file. It will give the following screen:

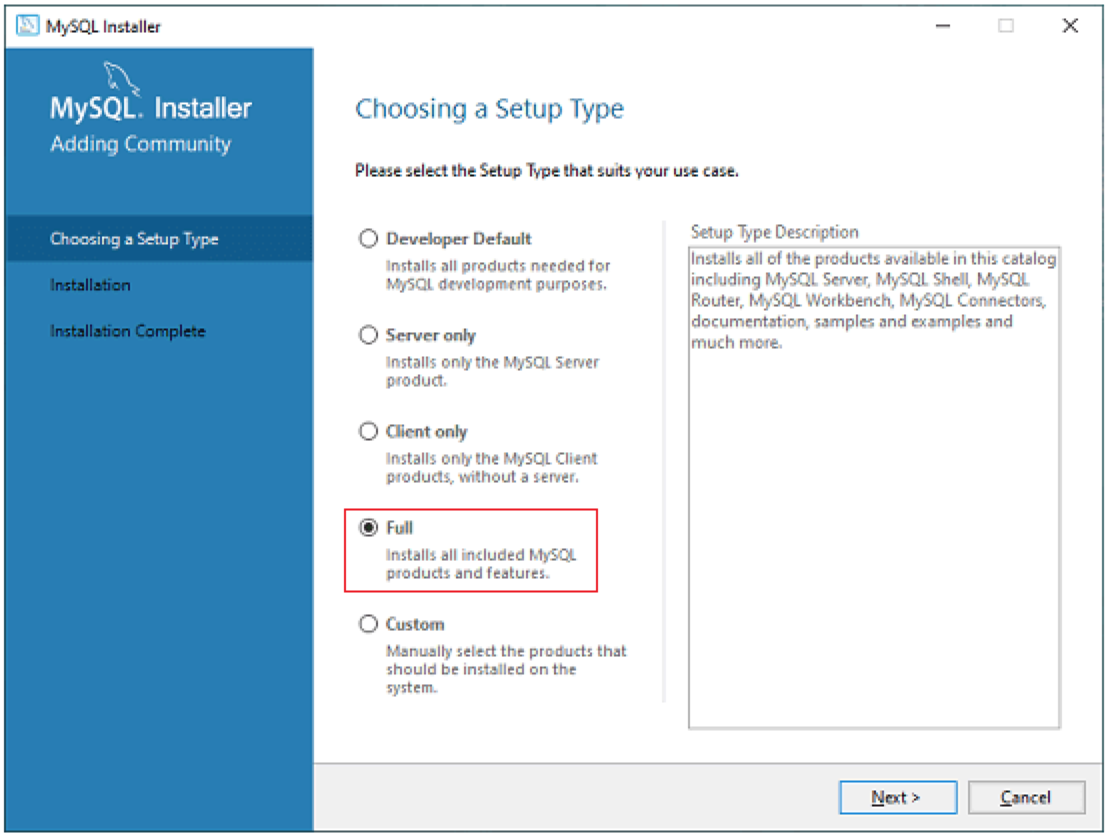


MySQL Installer - Community

Please wait while Windows configures MySQL Installer - Community

Gathering required information...

Step 2: In the next wizard, choose the Setup Type. There are several types available, and you need to choose the appropriate option to install MySQL product and features. Here, we are going to select the Full option and click on the Next button.


MySQL Installer

MySQL. Installer
Adding Community

Choosing a Setup Type
Installation
Installation Complete

Choosing a Setup Type

Please select the Setup Type that suits your use case.

○ Developer Default
Installs all products needed for MySQL development purposes.

○ Server only
Installs only the MySQL Server product.

○ Client only
Installs only the MySQL Client products, without a server.

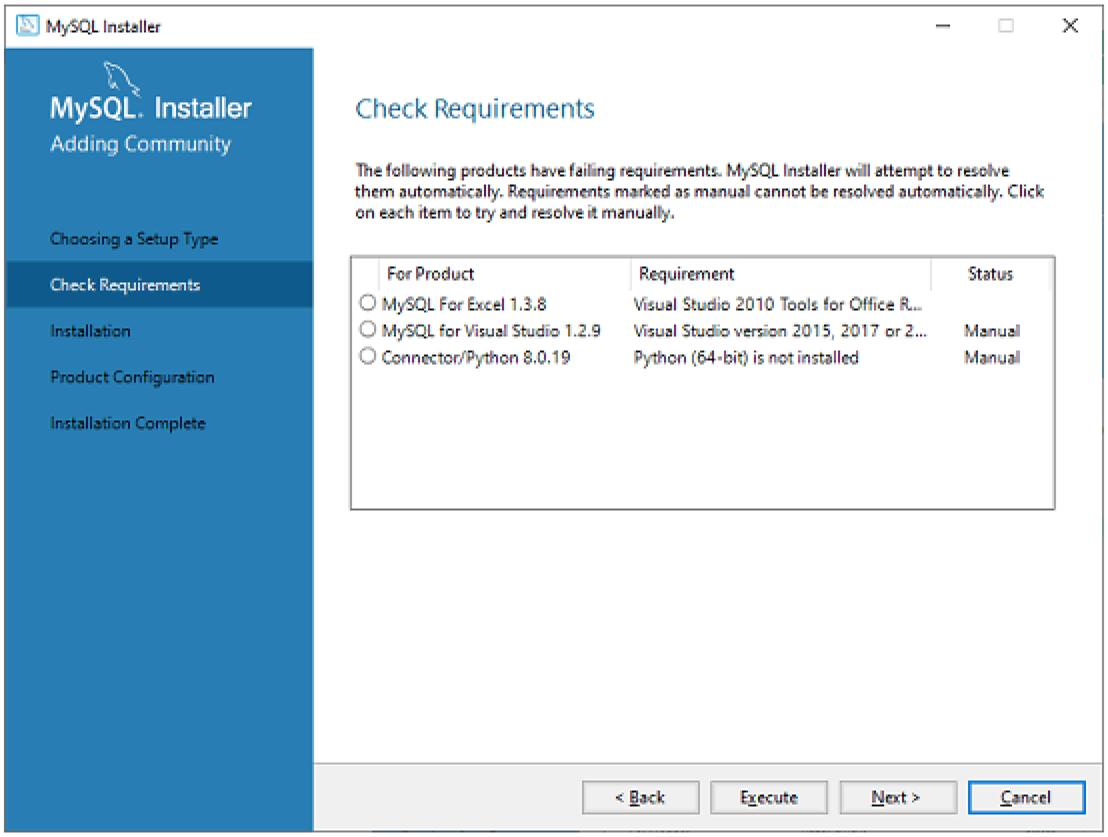
● Full
Installs all included MySQL products and features.

○ Custom
Manually select the products that should be installed on the system.

Setup Type Description
Installs all of the products available in this catalog including MySQL Server, MySQL Shell, MySQL Router, MySQL Workbench, MySQL Connectors, documentation, samples and examples and much more.

This option will install the following things: MySQL Server, MySQL Shell, MySQL Router, MySQL Workbench, MySQL Connectors, documentation, samples and examples, and many more.

Step 3: Once we click on the Next button, it may give information about some features that may fail to install on your system due to a lack of requirements. We can resolve them by clicking on the Execute button that will install all requirements automatically or can skip them. Now, click on the Next button.


MySQL Installer

MySQL. Installer
Adding Community

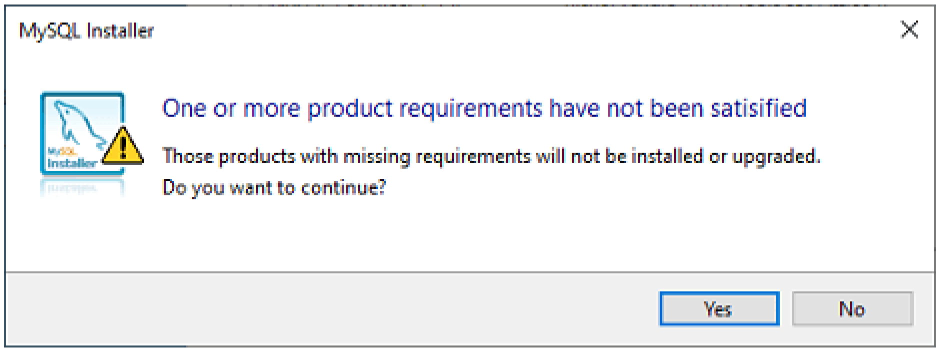
- Choosing a Setup Type
- Check Requirements
- Installation
- Product Configuration
- Installation Complete

Check Requirements

The following products have failing requirements. MySQL Installer will attempt to resolve them automatically. Requirements marked as manual cannot be resolved automatically. Click on each item to try and resolve it manually.

For Product	Requirement	Status
MySQL For Excel 1.3.8	Visual Studio 2010 Tools for Office R...	
MySQL for Visual Studio 1.2.9	Visual Studio version 2015, 2017 or 2...	Manual
Connector/Python 8.0.19	Python (64-bit) is not installed	Manual

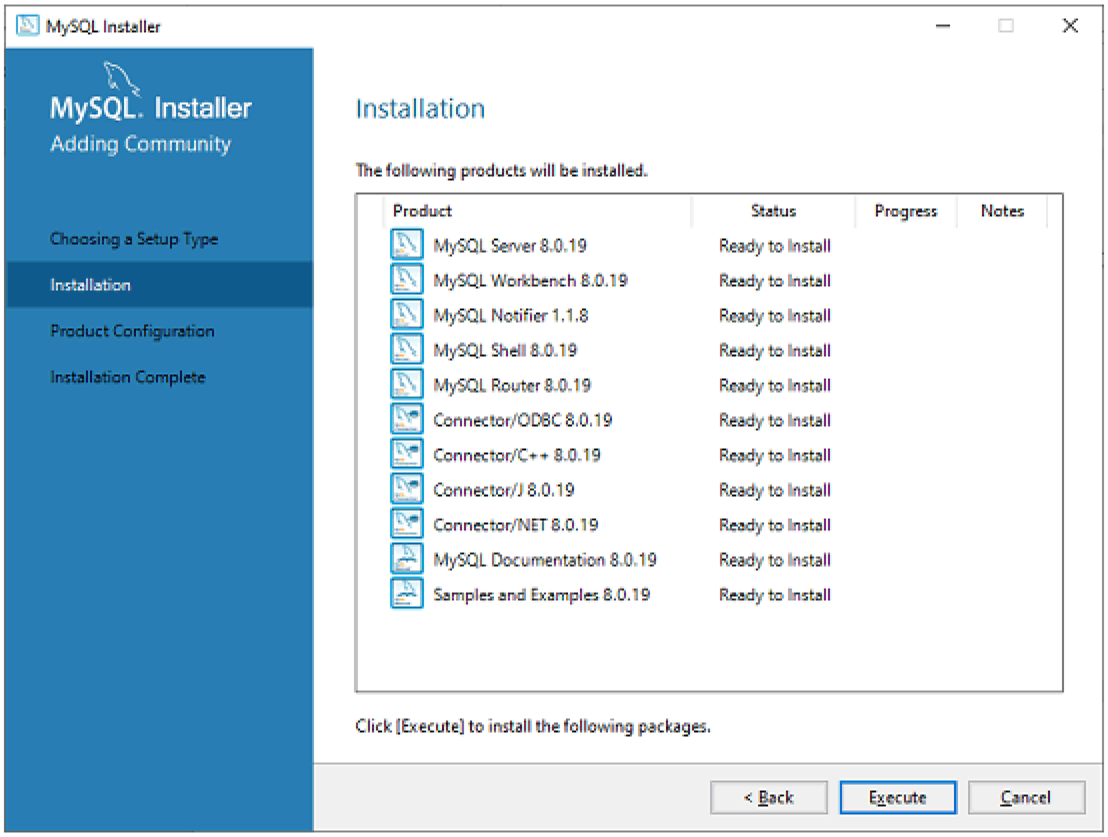
Step 4: In the next wizard, we will see a dialog box that asks for our confirmation of a few products not getting installed. Here, we have to click on the Yes button.


MySQL Installer

One or more product requirements have not been satisified

Those products with missing requirements will not be installed or upgraded.
Do you want to continue?

After clicking on the Yes button, we will see the list of the products which are going to be installed. So, if we need all products, click on the Execute button.



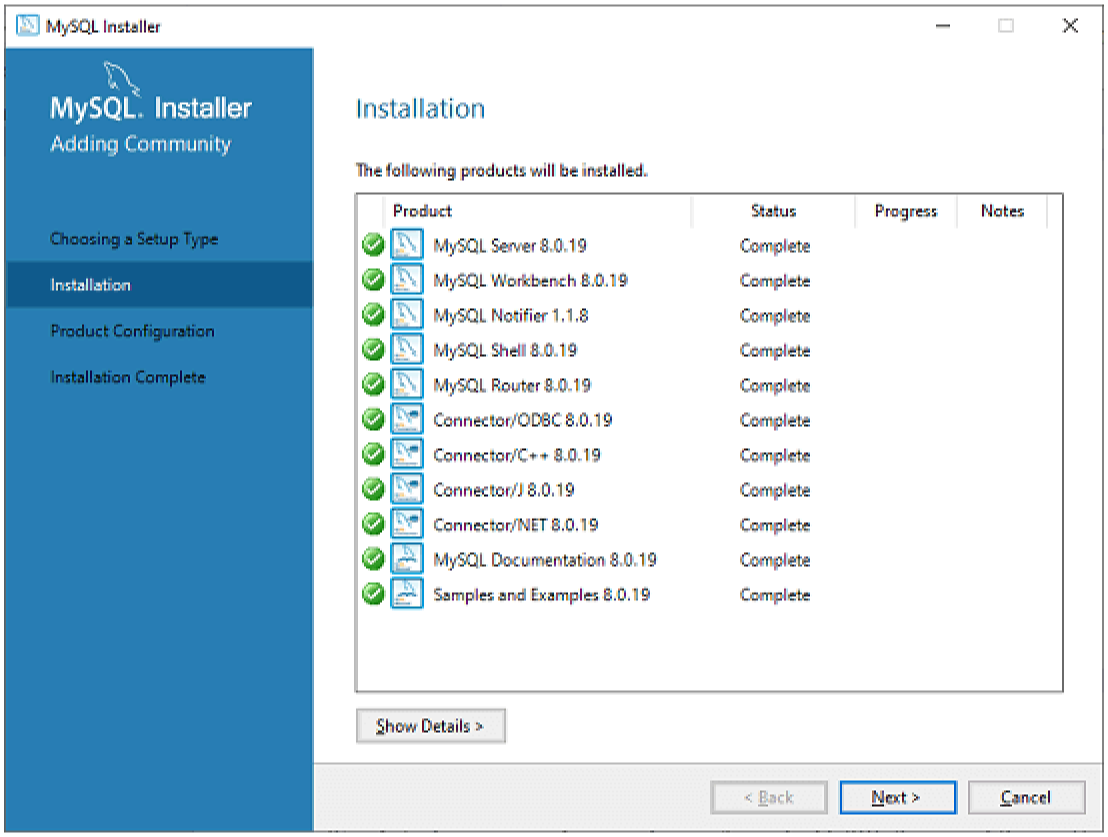
Installation

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.19	Ready to Install		
MySQL Workbench 8.0.19	Ready to Install		
MySQL Notifier 1.1.8	Ready to Install		
MySQL Shell 8.0.19	Ready to Install		
MySQL Router 8.0.19	Ready to Install		
Connector/ODBC 8.0.19	Ready to Install		
Connector/C++ 8.0.19	Ready to Install		
Connector/J 8.0.19	Ready to Install		
Connector/NET 8.0.19	Ready to Install		
MySQL Documentation 8.0.19	Ready to Install		
Samples and Examples 8.0.19	Ready to Install		

Click [Execute] to install the following packages.

Step 5: Once we click on the Execute button, it will download and install all the products. After completing the installation, click on the Next button.

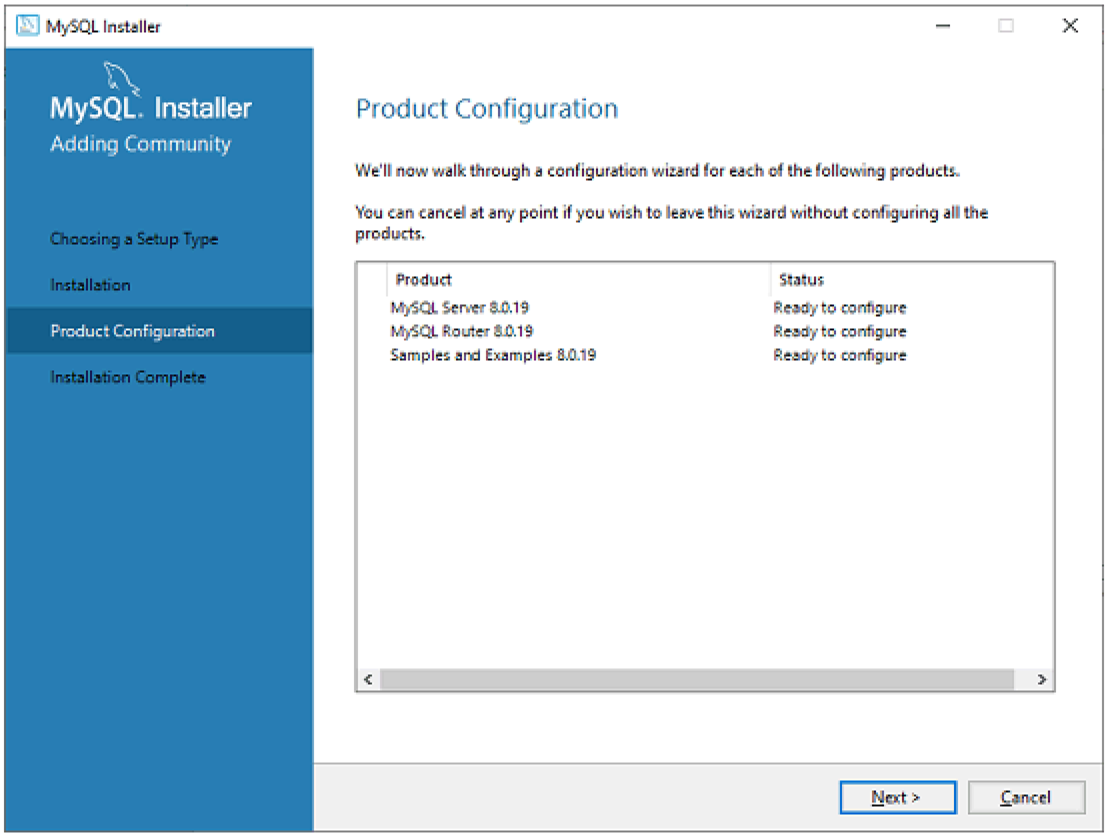


Installation

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.19	Complete		
MySQL Workbench 8.0.19	Complete		
MySQL Notifier 1.1.8	Complete		
MySQL Shell 8.0.19	Complete		
MySQL Router 8.0.19	Complete		
Connector/ODBC 8.0.19	Complete		
Connector/C++ 8.0.19	Complete		
Connector/J 8.0.19	Complete		
Connector/NET 8.0.19	Complete		
MySQL Documentation 8.0.19	Complete		
Samples and Examples 8.0.19	Complete		

Step 6: In the next wizard, we need to configure the MySQL Server and Router. Here, I am not going to configure the Router because there is no need to use it with MySQL. We are going to show you how to configure the server only. Now, click on the Next button.


MySQL Installer

MySQL. Installer
Adding Community

Choosing a Setup Type
Installation
Product Configuration
Installation Complete

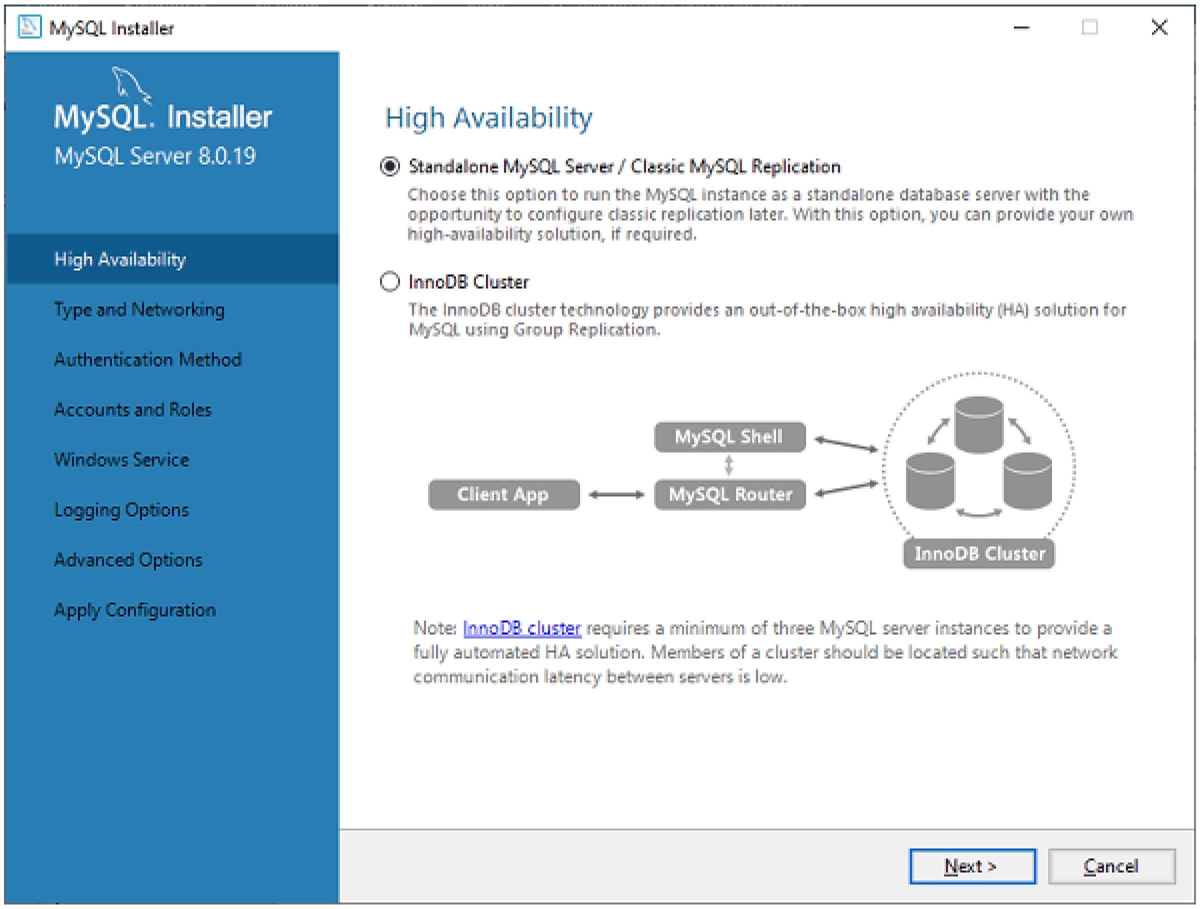
Product Configuration

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.19	Ready to configure
MySQL Router 8.0.19	Ready to configure
Samples and Examples 8.0.19	Ready to configure

Step 7: As soon as you will click on the Next button, you can see the screen below. Here, we have to configure the MySQL Server. Now, choose the Standalone MySQL Server/Classic MySQL Replication option and click on Next. Here, you can also choose the InnoDB Cluster based on your needs.


MySQL Installer

MySQL. Installer
MySQL Server 8.0.19

- High Availability
- Type and Networking
- Authentication Method
- Accounts and Roles
- Windows Service
- Logging Options
- Advanced Options
- Apply Configuration

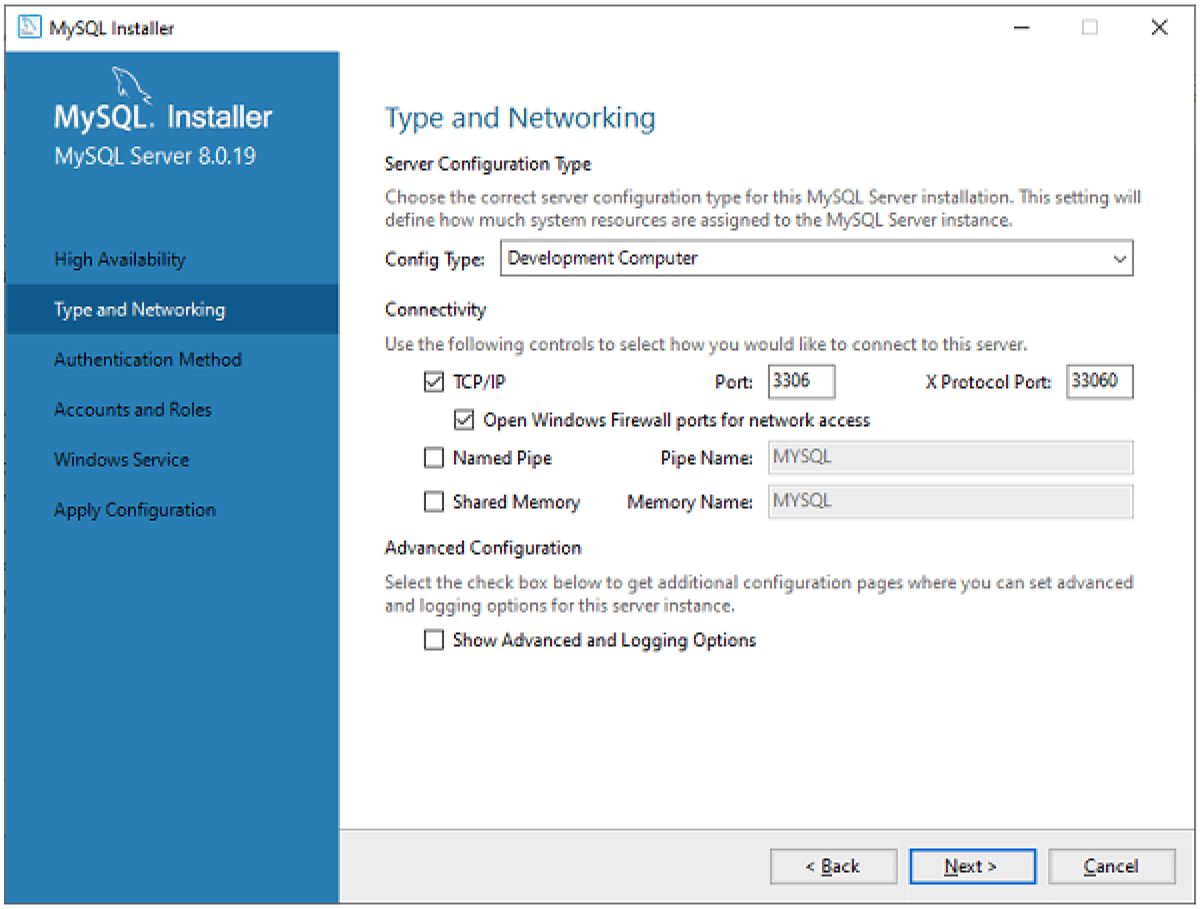
High Availability

[● Standalone MySQL Server / Classic MySQL Replication]
Choose this option to run the MySQL instance as a standalone database server with the opportunity to configure classic replication later. With this option, you can provide your own high-availability solution, if required.

○ InnoDB Cluster
The InnoDB cluster technology provides an out-of-the-box high availability (HA) solution for MySQL using Group Replication.

Note: InnoDB cluster requires a minimum of three MySQL server instances to provide a fully automated HA solution. Members of a cluster should be located such that network communication latency between servers is low.

Step 8: In the next screen, the system will ask you to choose the Config Type and other connectivity options. Here, we are going to select the Config Type as 'Development Machine' and Connectivity as TCP/IP, and Port Number is 3306, then click on Next.


MySQL Installer

MySQL. Installer
MySQL Server 8.0.19

- High Availability
- Type and Networking
- Authentication Method
- Accounts and Roles
- Windows Service
- Apply Configuration

Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

Connectivity

Use the following controls to select how you would like to connect to this server.

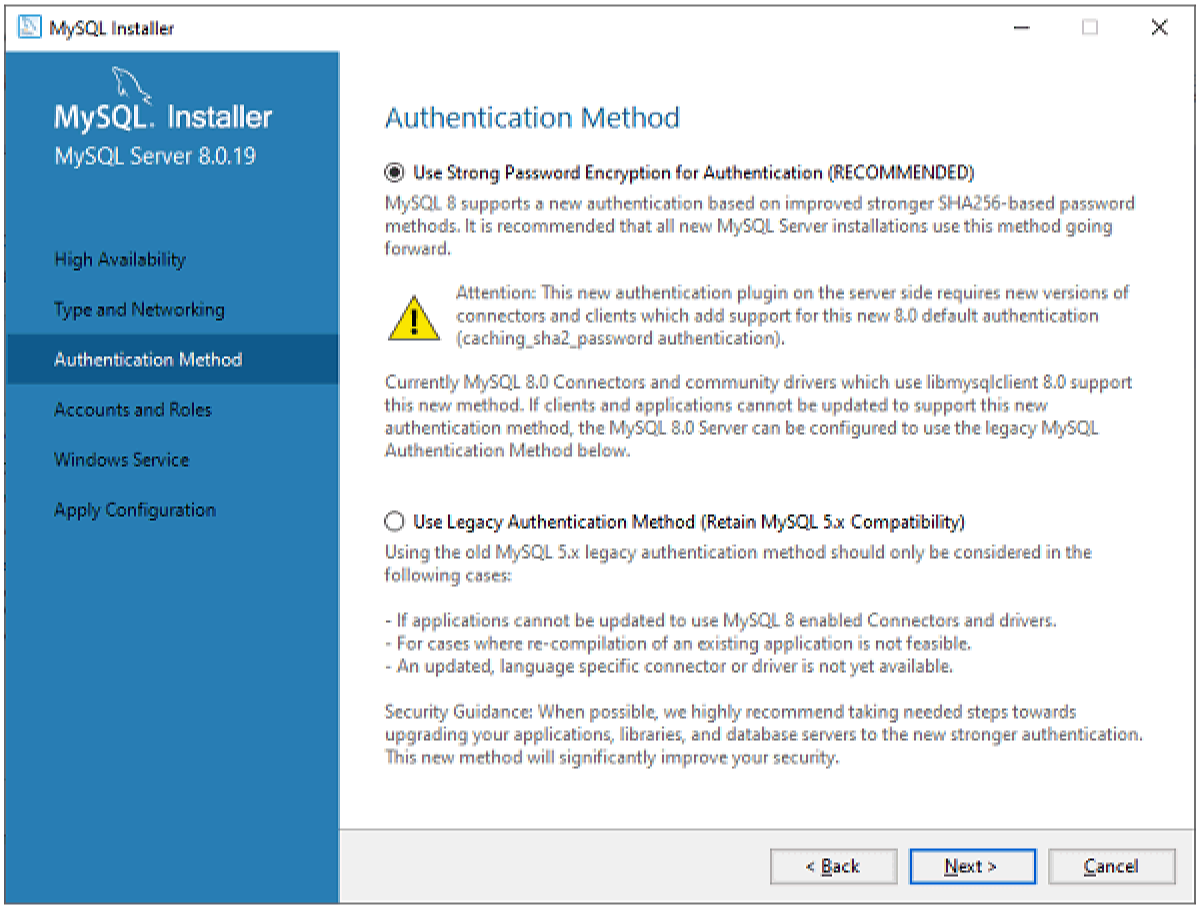
- [✓] TCP/IP — Port: 3306 — X Protocol Port: 33060
  - [✓] Open Windows Firewall ports for network access
- [ ] Named Pipe — Pipe Name: MYSQL
- [ ] Shared Memory — Memory Name: MYSQL

Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

- [ ] Show Advanced and Logging Options

Step 9: Now, select the Authentication Method and click on Next. Here, I am going to select the first option.



Authentication Method

[✓ Use Strong Password Encryption for Authentication (RECOMMENDED)]

MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.

Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

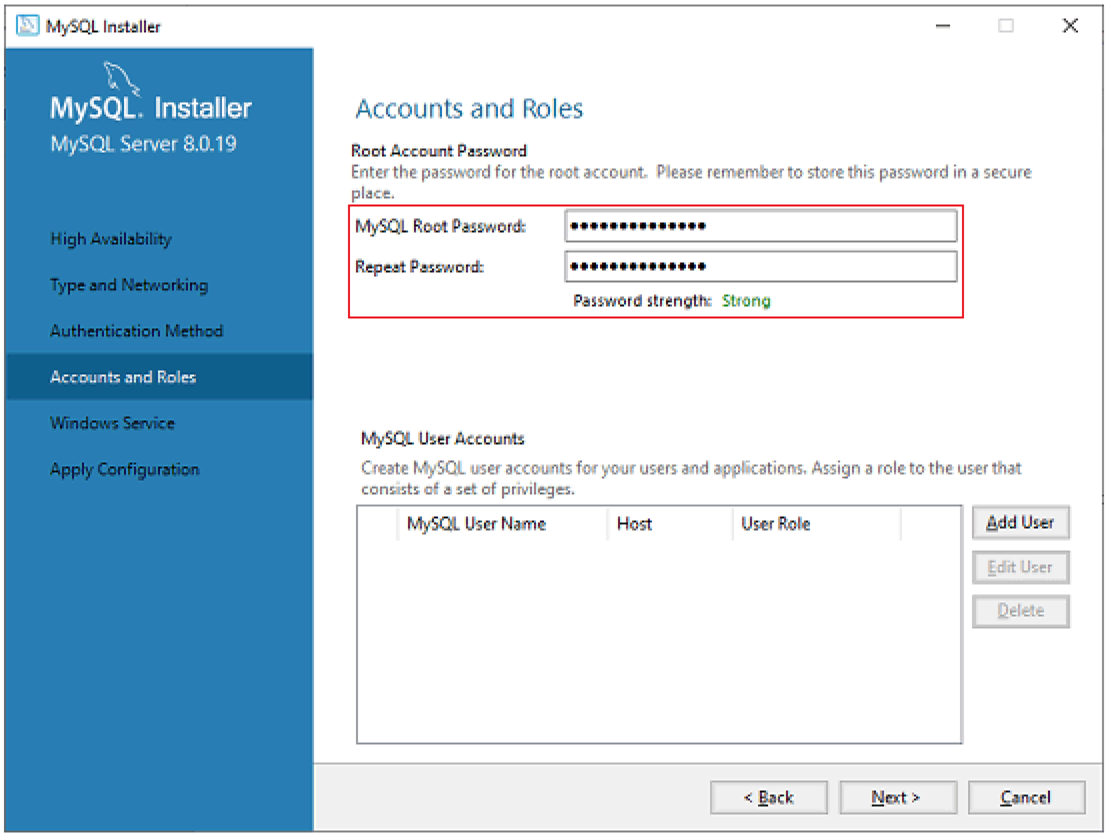
Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)

Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.

Step 10: The next screen will ask you to mention the MySQL Root Password. After filling the password details, click on the Next button.


MySQL Installer

MySQL. Installer
MySQL Server 8.0.19

- High Availability
- Type and Networking
- Authentication Method
- Accounts and Roles
- Windows Service
- Apply Configuration

Accounts and Roles

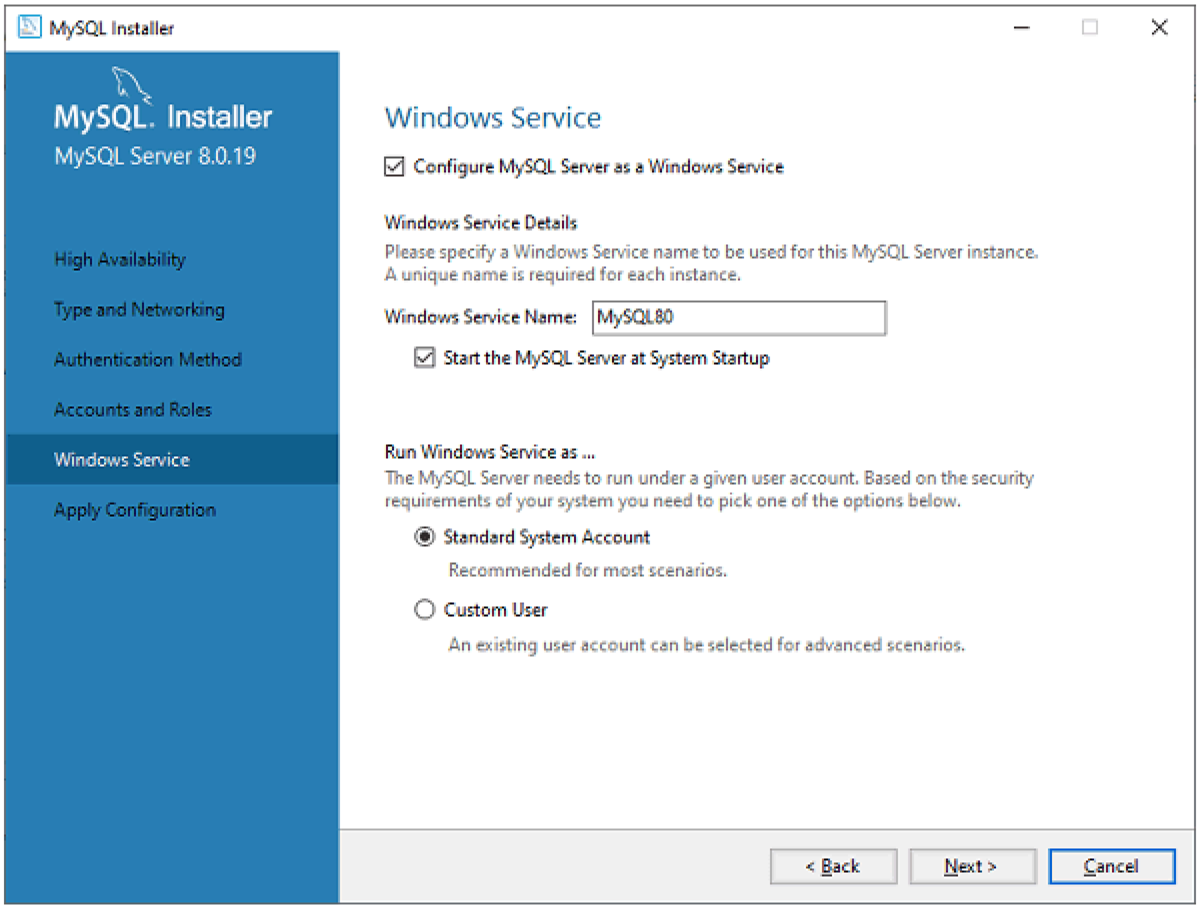
Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password: ••••••••••••••
Repeat Password: ••••••••••••••
Password strength: Strong

MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role

Step 11: The next screen will ask you to configure the Windows Service to start the server. Keep the default setup and click on the Next button.



Windows Service

☑ Configure MySQL Server as a Windows Service

Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.

Windows Service Name: MySQL80

☑ Start the MySQL Server at System Startup

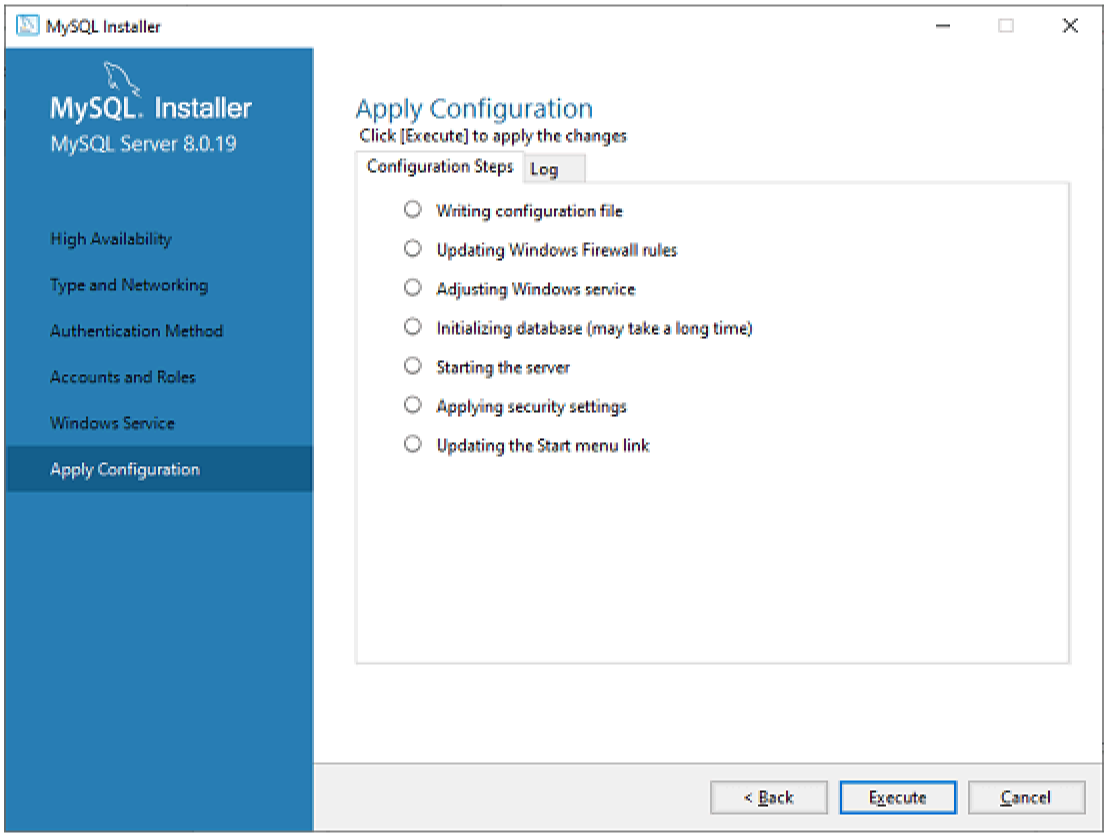
Run Windows Service as ...

The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

● Standard System Account
Recommended for most scenarios.

○ Custom User
An existing user account can be selected for advanced scenarios.

Step 12: In the next wizard, the system will ask you to apply the Server Configuration. If you agree with this configuration, click on the Execute button.


MySQL Installer

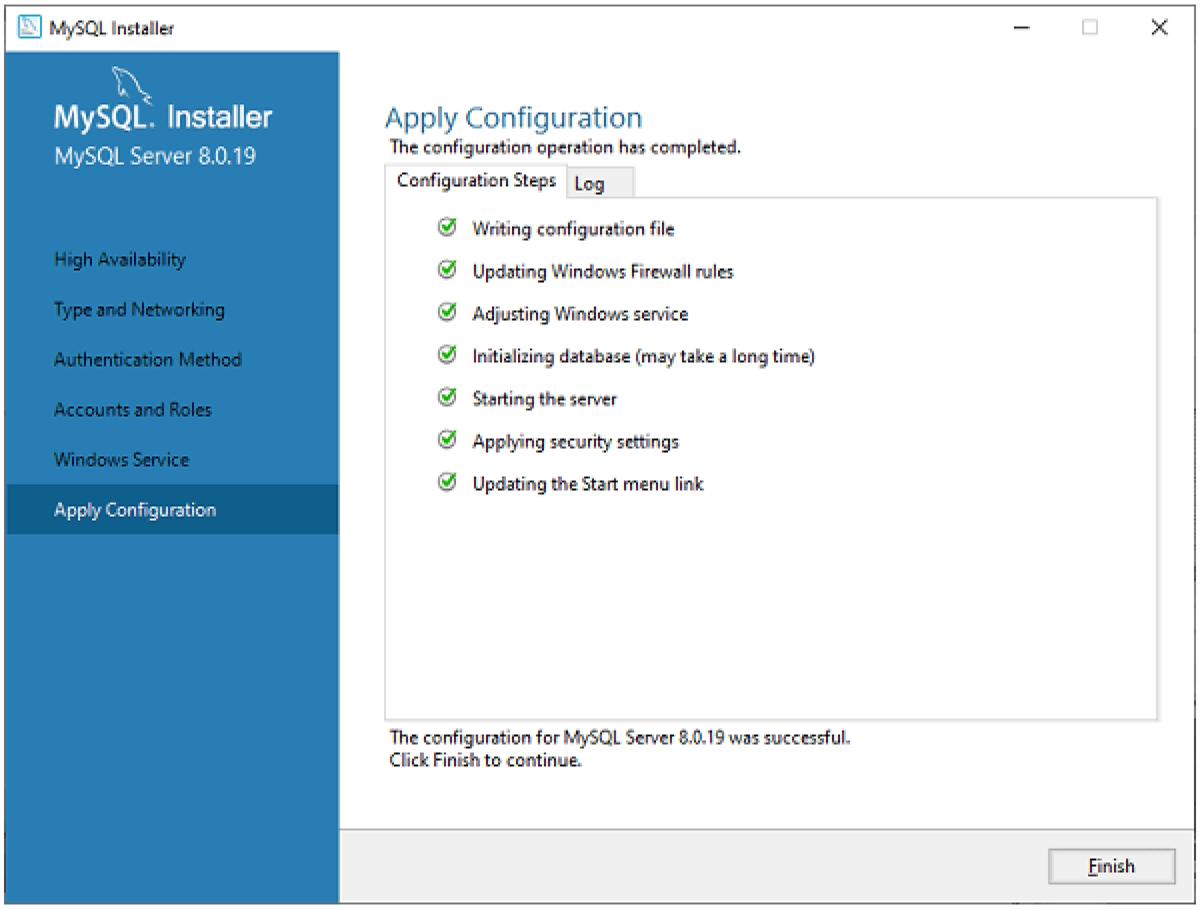
MySQL. Installer
MySQL Server 8.0.19

Apply Configuration

Click [Execute] to apply the changes

- Writing configuration file
- Updating Windows Firewall rules
- Adjusting Windows service
- Initializing database (may take a long time)
- Starting the server
- Applying security settings
- Updating the Start menu link

Step 13: Once the configuration has completed, you will get the screen below. Now, click on the Finish button to continue.


MySQL Installer

MySQL. Installer
MySQL Server 8.0.19

- High Availability
- Type and Networking
- Authentication Method
- Accounts and Roles
- Windows Service
- Apply Configuration

Apply Configuration

The configuration operation has completed.

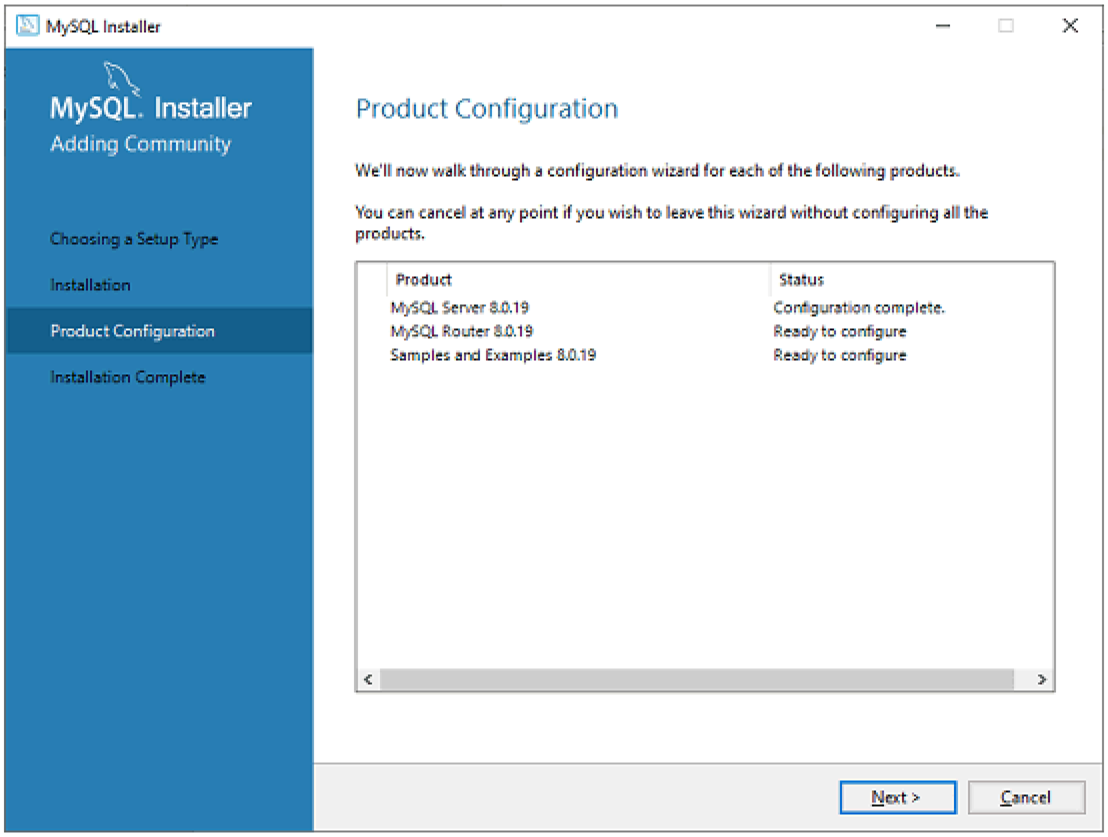
Configuration Steps | Log

- ✓ Writing configuration file
- ✓ Updating Windows Firewall rules
- ✓ Adjusting Windows service
- ✓ Initializing database (may take a long time)
- ✓ Starting the server
- ✓ Applying security settings
- ✓ Updating the Start menu link

The configuration for MySQL Server 8.0.19 was successful.
Click Finish to continue.

Finish

Step 14: In the next screen, you can see that the Product Configuration is completed. Keep the default setting and click on the Next-> Finish button to complete the MySQL package installation.



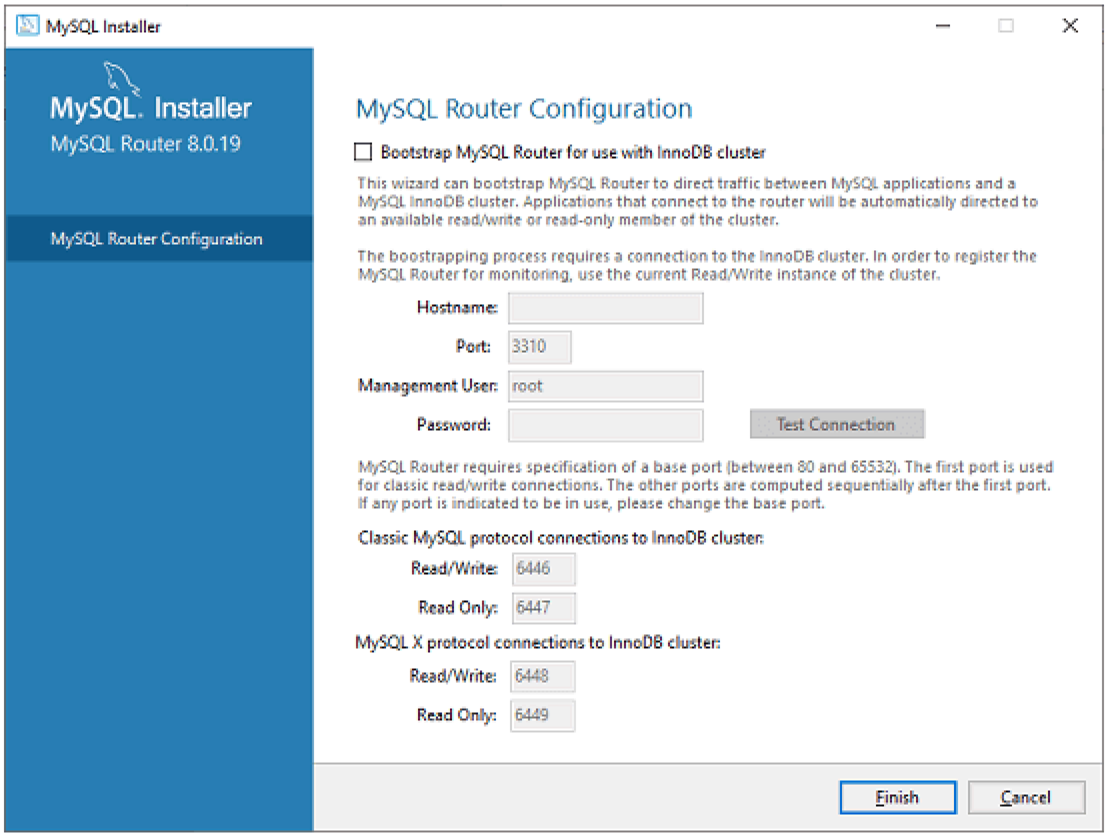
Product Configuration

We'll now walk through a configuration wizard for each of the following products.

You can cancel at any point if you wish to leave this wizard without configuring all the products.

Product	Status
MySQL Server 8.0.19	Configuration complete.
MySQL Router 8.0.19	Ready to configure
Samples and Examples 8.0.19	Ready to configure

Step 15: In the next wizard, we can choose to configure the Router. So click on Next->Finish and then click the Next button.



MySQL Installer

MySQL. Installer
MySQL Router 8.0.19

MySQL Router Configuration

MySQL Router Configuration

☐ Bootstrap MySQL Router for use with InnoDB cluster

This wizard can bootstrap MySQL Router to direct traffic between MySQL applications and a MySQL InnoDB cluster. Applications that connect to the router will be automatically directed to an available read/write or read-only member of the cluster.

The boostrapping process requires a connection to the InnoDB cluster. In order to register the MySQL Router for monitoring, use the current Read/Write instance of the cluster.

Hostname:
Port: 3310
Management User: root
Password:

Test Connection

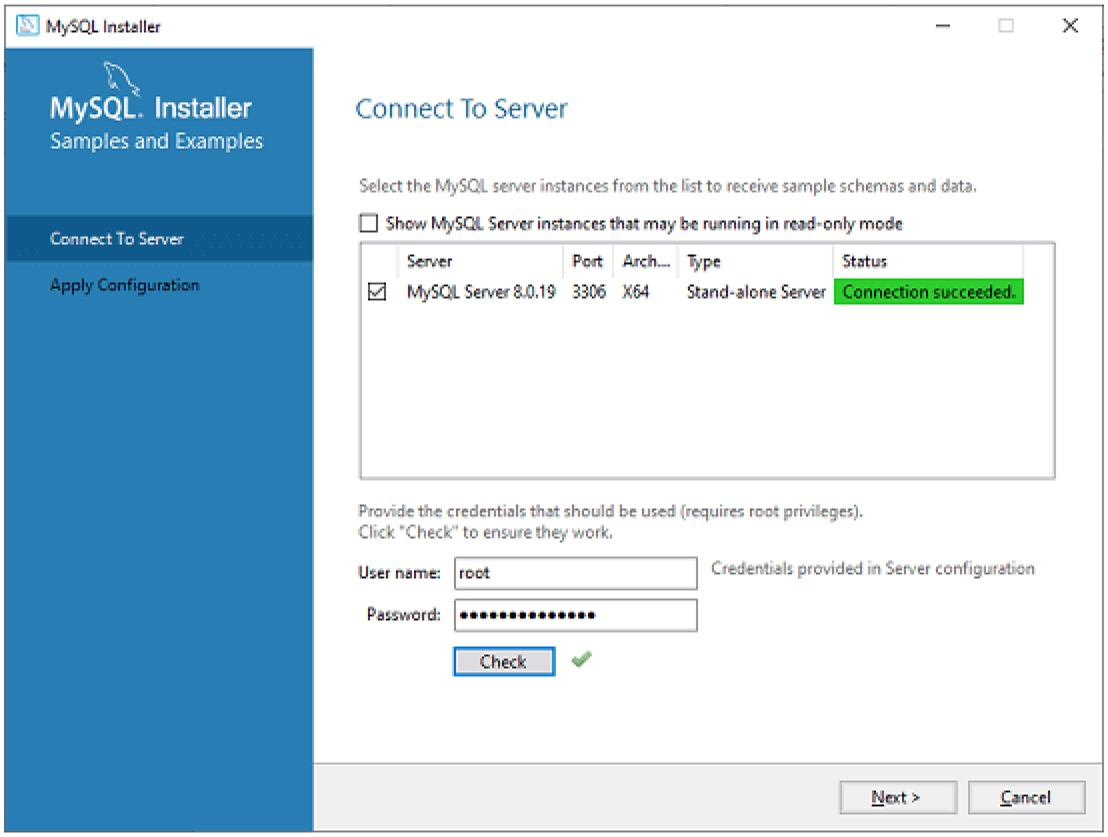
MySQL Router requires specification of a base port (between 80 and 65532). The first port is used for classic read/write connections. The other ports are computed sequentially after the first port. If any port is indicated to be in use, please change the base port.

Classic MySQL protocol connections to InnoDB cluster:
Read/Write: 6446
Read Only: 6447

MySQL X protocol connections to InnoDB cluster:
Read/Write: 6448
Read Only: 6449

Finish Cancel

Step 16: In the next wizard, we will see the Connect to Server option. Here, we have to mention the root password, which we had set in the previous steps.


MySQL Installer

MySQL. Installer
Samples and Examples

Connect To Server
Apply Configuration

Connect To Server

Select the MySQL server instances from the list to receive sample schemas and data.

☐ Show MySQL Server instances that may be running in read-only mode

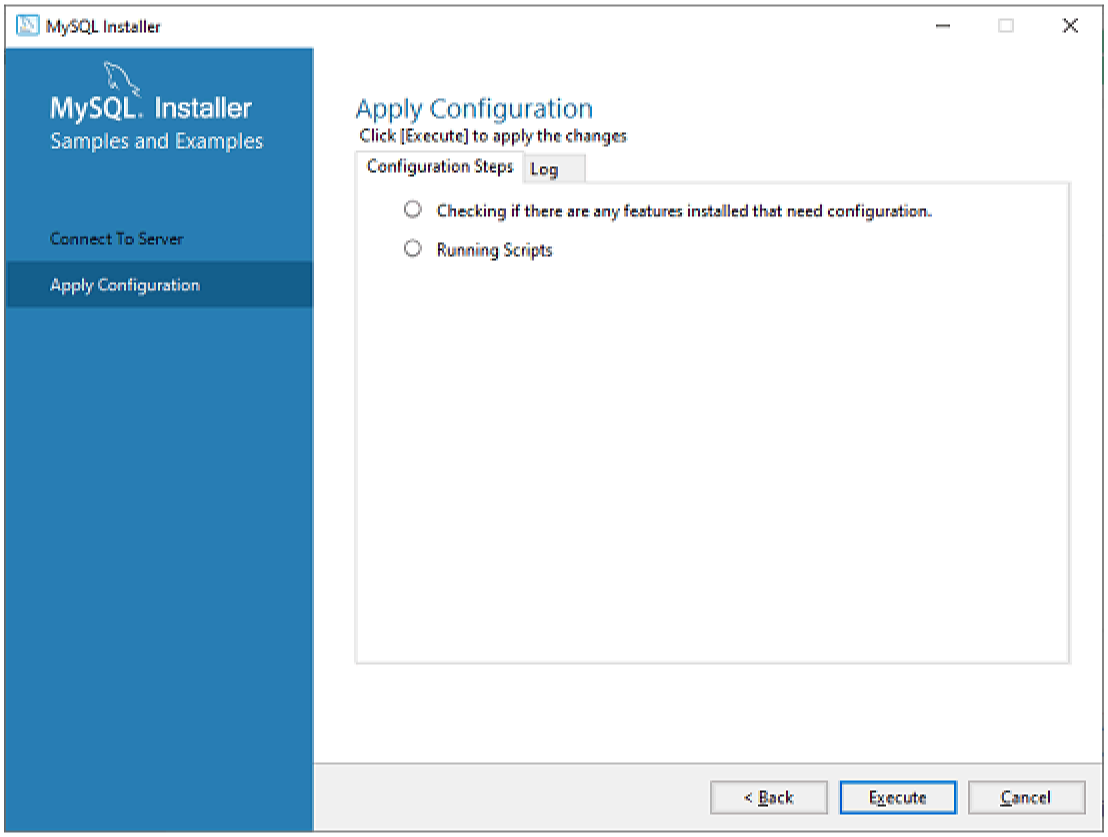
	Server	Port	Arch...	Type	Status
☑	MySQL Server 8.0.19	3306	X64	Stand-alone Server	Connection succeeded.

Provide the credentials that should be used (requires root privileges).
Click "Check" to ensure they work.

User name: root
Password: ••••••••••••••
Credentials provided in Server configuration

In this screen, it is also required to check about the connection is successful or not by clicking on the Check button. If the connection is successful, click on the Execute button. Now, the configuration is complete, click on Next.

Step 17: In the next wizard, select the applied configurations and click on the Execute button.



Apply Configuration

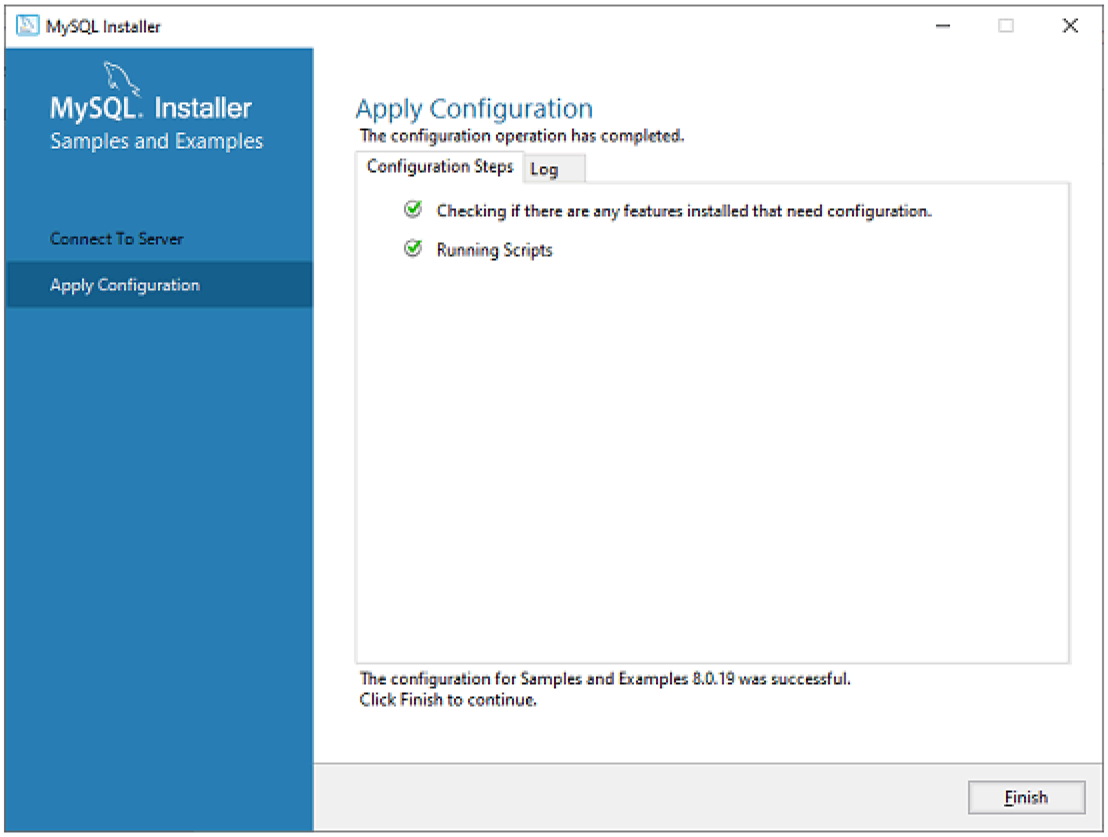
Click [Execute] to apply the changes

Configuration Steps | Log

○ Checking if there are any features installed that need configuration.

○ Running Scripts

Step 18: After completing the above step, we will get the following screen. Here, click on the Finish button.



Apply Configuration

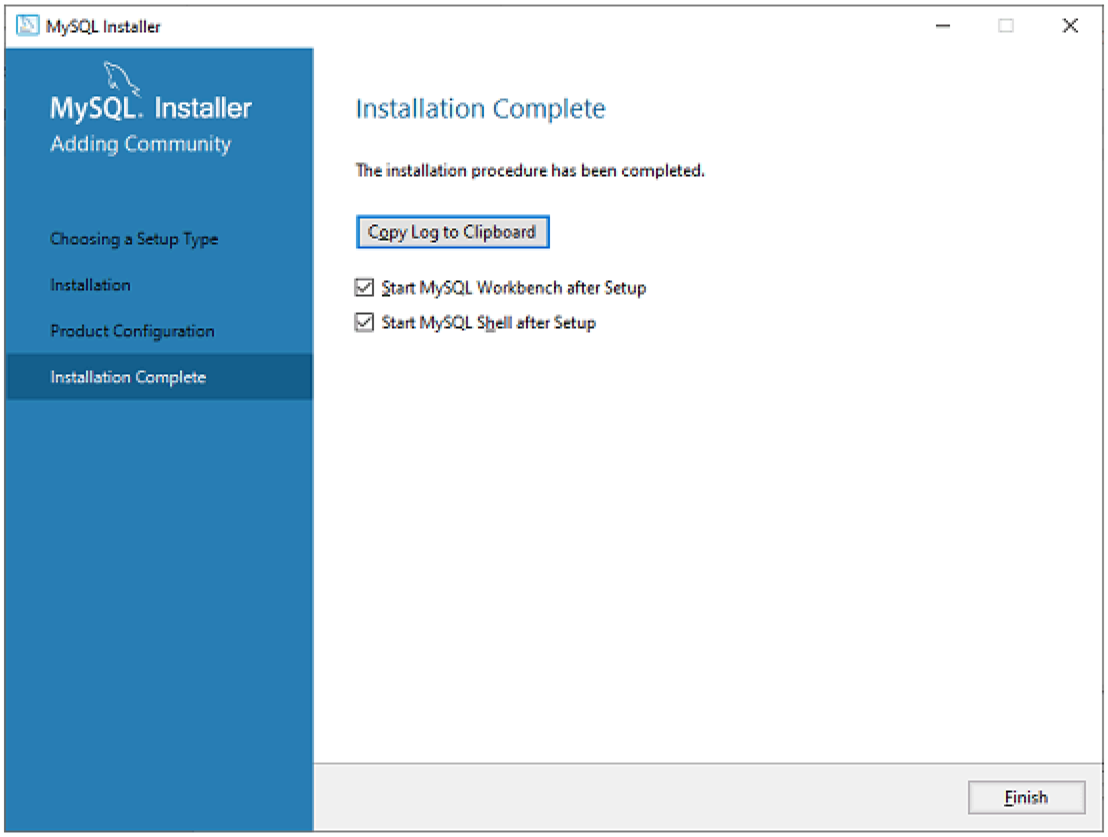
The configuration operation has completed.

✓ Checking if there are any features installed that need configuration.

✓ Running Scripts

The configuration for Samples and Examples 8.0.19 was successful.
Click Finish to continue.

Step 19: Now, the MySQL installation is complete. Click on the Finish button.



Installation Complete

The installation procedure has been completed.

☑ Start MySQL Workbench after Setup

☑ Start MySQL Shell after Setup

<u>SQL CREATE TABLE:</u>

The SQL CREATE TABLE statement is a **foundational command** used to define and structure a new table in a database. By specifying the columns, data types, and constraints such as **PRIMARY KEY**, **NOT NULL**, and CHECK, helps you design the database schema.

SQL CREATE TABLE Statement

To create a new table in the database, use the SQL CREATE TABLE statement. A table's structure, including column names, data types, and constraints like NOT NULL, PRIMARY KEY, and CHECK, are defined when it is created in SQL.

The CREATE TABLE command is a crucial tool for database administration because of these limitations, which aid in ensuring data integrity.

Syntax:

To create a table in SQL, use this **CREATE TABLE syntax**:

```
CREATE table table_name
(
Column1 datatype (size),
column2 datatype (size),
.
.
columnN datatype(size)
);
```

- **table_name**: The name you assign to the new table.
- **column1, column2, …** : The names of the columns in the table.
- **datatype(size):** Defines the data type and size of each column.

Here **table_name** is name of the table, **column** is the name of column

SQL CREATE TABLE Example

Let's look at examples of CREATE TABLE command in SQL and see **how to create table in SQL.**

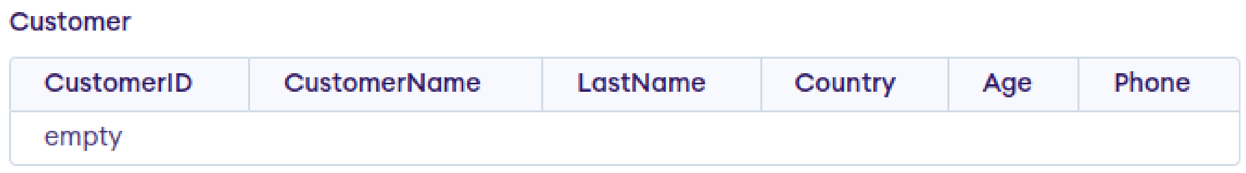
CREATE TABLE in SQL and Insert Data

In this example, we will create a new table and insert data into it.

Let us create a table to store data of Customers, so the table name is Customer, Columns are Name, Country, age, phone, and so on.

```
CREATE TABLE Customer(
   CustomerID INT PRIMARY KEY,
   CustomerName VARCHAR(50),
   LastName VARCHAR(50),
   Country VARCHAR(50),
   Age INT CHECK (Age >= 0 AND Age <= 99),
   Phone int(10)
);
```

Output:



Customer

CustomerID	CustomerName	LastName	Country	Age	Phone
empty					

To add data to the table, we use **<u>INSERT INTO</u>** command, the syntax is as shown below:

Syntax:

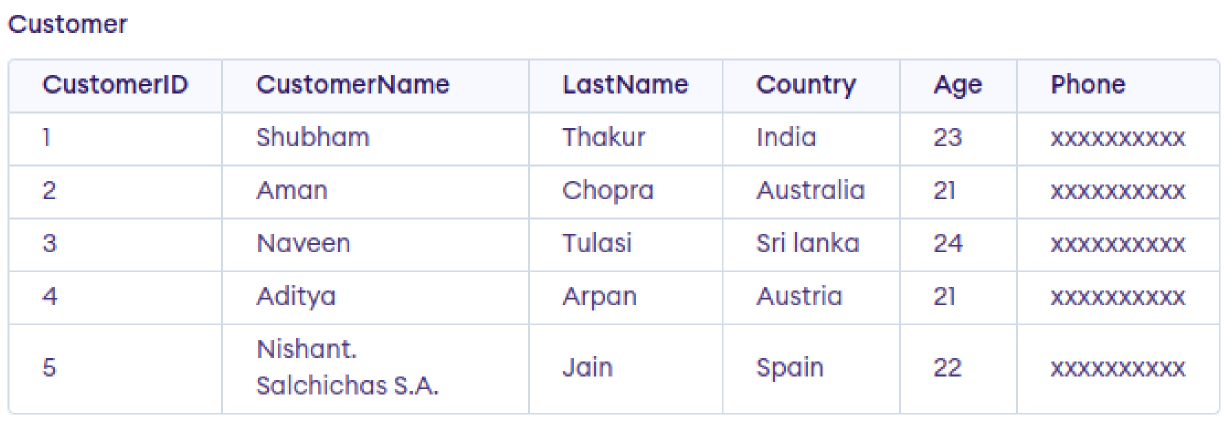
INSERT INTO *table_name (column1, column2, ...) VALUES (value1, value2, ...);*

Example Query

This query will add data in the table named Subject

```
INSERT INTO Customer (CustomerID, CustomerName, LastName, Country, Age, Phone)
VALUES (1, 'Shubham', 'Thakur', 'India','23','xxxxxxxxxx'),
       (2, 'Aman ', 'Chopra', 'Australia','21','xxxxxxxxxx'),
       (3, 'Naveen', 'Tulasi', 'Sri lanka','24','xxxxxxxxxx'),
       (4, 'Aditya', 'Arpan', 'Austria','21','xxxxxxxxxx'),
       (5, 'Nishant. Salchichas S.A.', 'Jain', 'Spain','22','xxxxxxxxxx');
```

Output:



Customer

CustomerID	CustomerName	LastName	Country	Age	Phone
1	Shubham	Thakur	India	23	xxxxxxxxxx
2	Aman	Chopra	Australia	21	xxxxxxxxxx
3	Naveen	Tulasi	Sri lanka	24	xxxxxxxxxx
4	Aditya	Arpan	Austria	21	xxxxxxxxxx
5	Nishant. Salchichas S.A.	Jain	Spain	22	xxxxxxxxxx

Create Table From Another Table:

We can also use CREATE TABLE to create a copy of an existing table. In the new table, it gets the exact column definition so all columns or specific columns can be selected.

If an existing table was used to create a new table, by default the new table would be populated with the existing values from the old table.

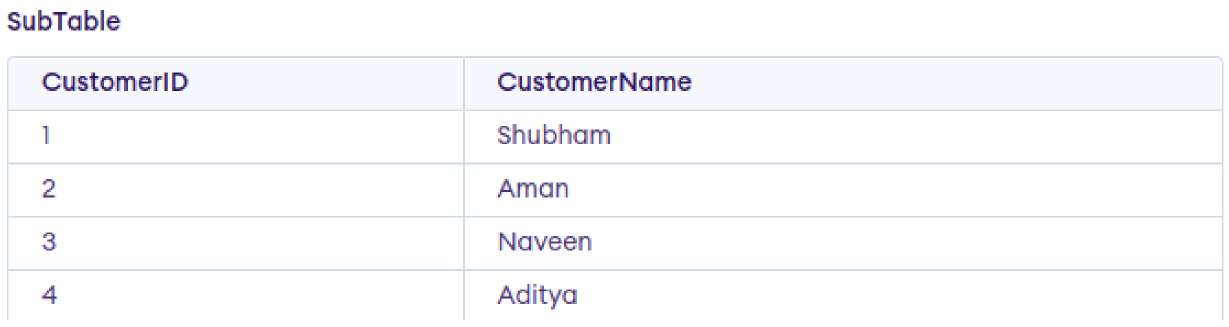
Syntax:

CREATE TABLE** new_table_name **AS
***SELECT** column1, column2,…*
***FROM** existing_table_name*
***WHERE** ….;*

Query:

CREATE TABLE SubTable **AS**
SELECT CustomerID, CustomerName
FROM customer;

Output:



SubTable

CustomerID	CustomerName
1	Shubham
2	Aman
3	Naveen
4	Aditya