

1. Install glassfish(I used 6.2.5)

```
archlinux$ yay -S glassfish
```

```
# /opt/glassfish
```

2. Install netbeans(I used 13.1)

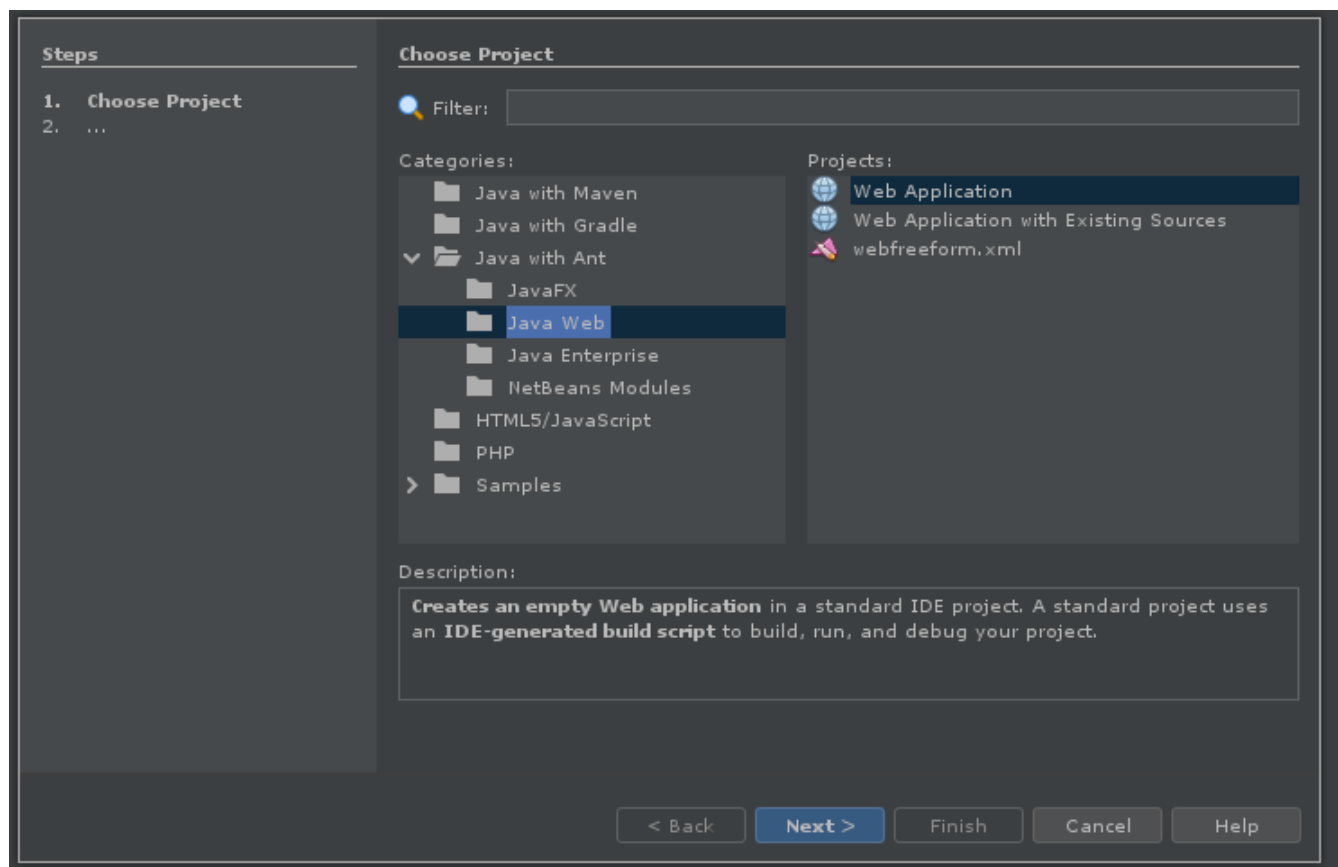
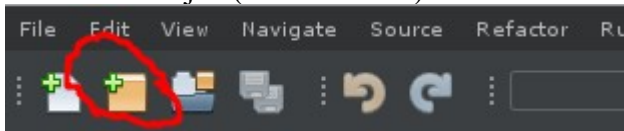
```
archlinux$ sudo pacman -S netbeans
```

3. Install java(I used java17)

```
archlinux$ sudo pacman -S jdk17-openjdk
```

4. Create java web project with ant in Netbeans

File>New Project(Ctrl+Shift+N)>Java with Ant>Java Web>Web application



add glassfish server:

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

Server: GlassFish Server Add...

Java EE Version: Jakarta EE 9 Web

Note: Source Level 8 will be set for Java EE 8 project.

Context Path: /lr2

< Back Next > Finish Cancel Help

Steps

1. **Choose Server**
2. ...

Choose Server

Server: Amazon Beanstalk
Apache Tomcat or TomEE
GlassFish Server
Payara Server
WildFly Application Server

Name: GlassFish Server 6.2.5

< Back Next > Finish Cancel Help

SERVER and Settings

Steps

1. Choose Server

2. **Server Location**

3. Domain Name/Location

Installation Location:

/opt/glassfish

Browse...

☒ Local Domain

☐ Remote Domain


Choose server to download:

GlassFish Server 6.2.1

▼

Download Now...

☐ [I have read and accept the license agreement... \(click\)](#)

 No usable default domain. Use Next to create a personal domain.

< Back

Next >

Finish

Cancel

Help

< Back

Next >

Finish

Cancel

Help

Steps

1. Choose Server
2. Server Location
3. **Domain Name/Location**

Domain Location

Domain:

Host: ☒ Loopback

DAS Port: HTTP Port: ☐ Default

Target:

User Name:

Password:

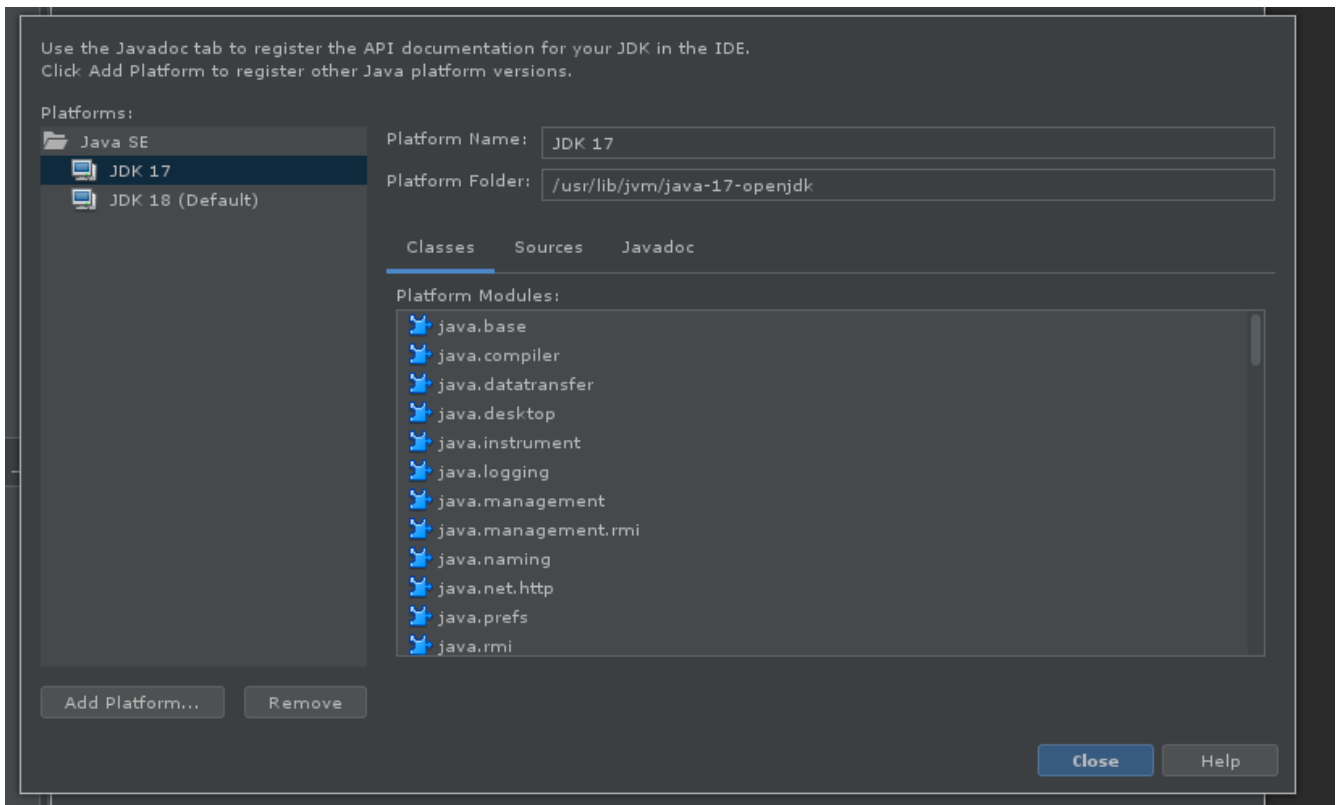
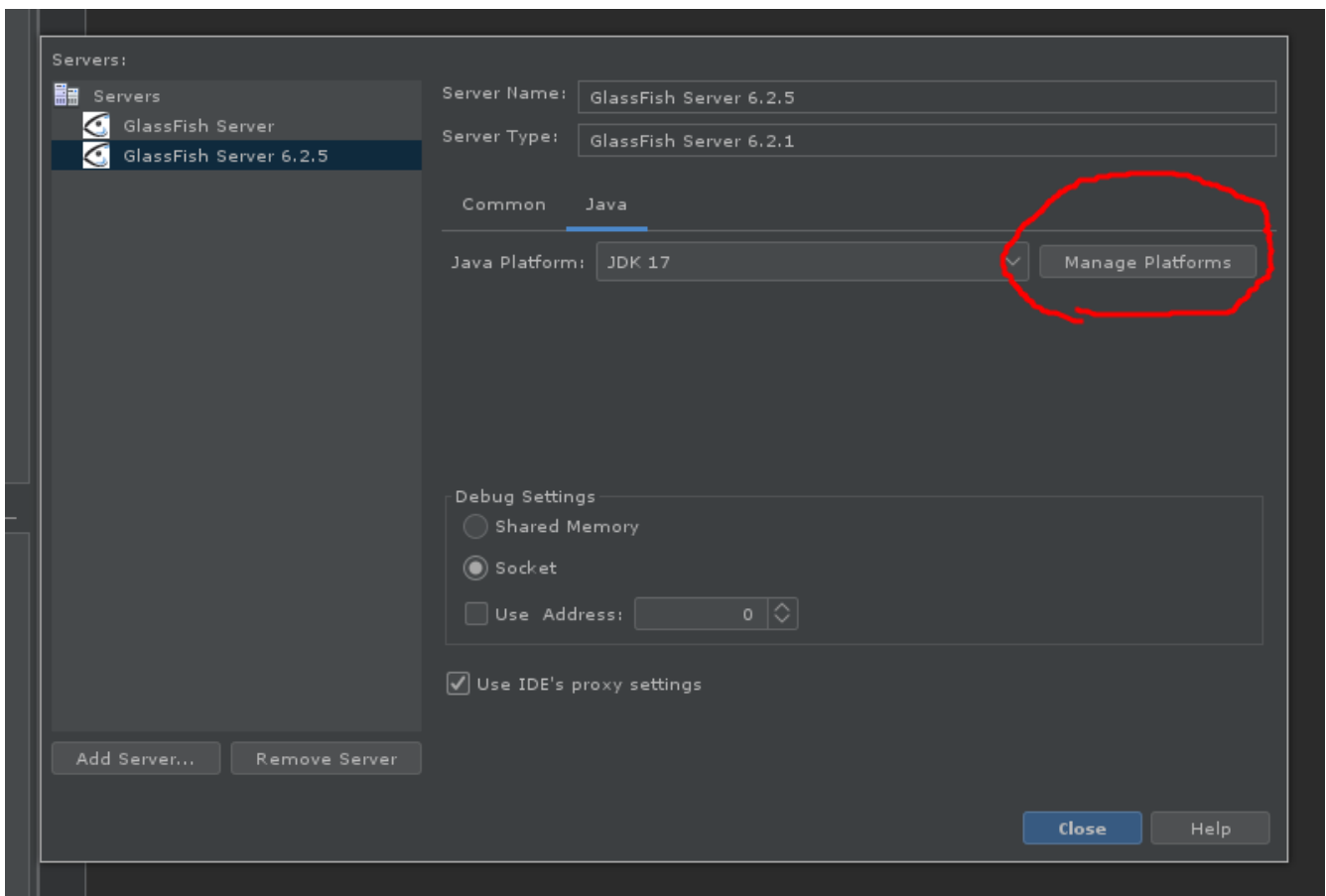
Create domain at /home/dazzlemon/personal_domain1

< Back Next > **Finish** Cancel Help

5. Start glassfish

Change java version(if you have multiple installed it might auto detect the wrong one):

In your project: Services>Servers>Glassfish Server(Right click)>Properties>Java>Manage Platforms



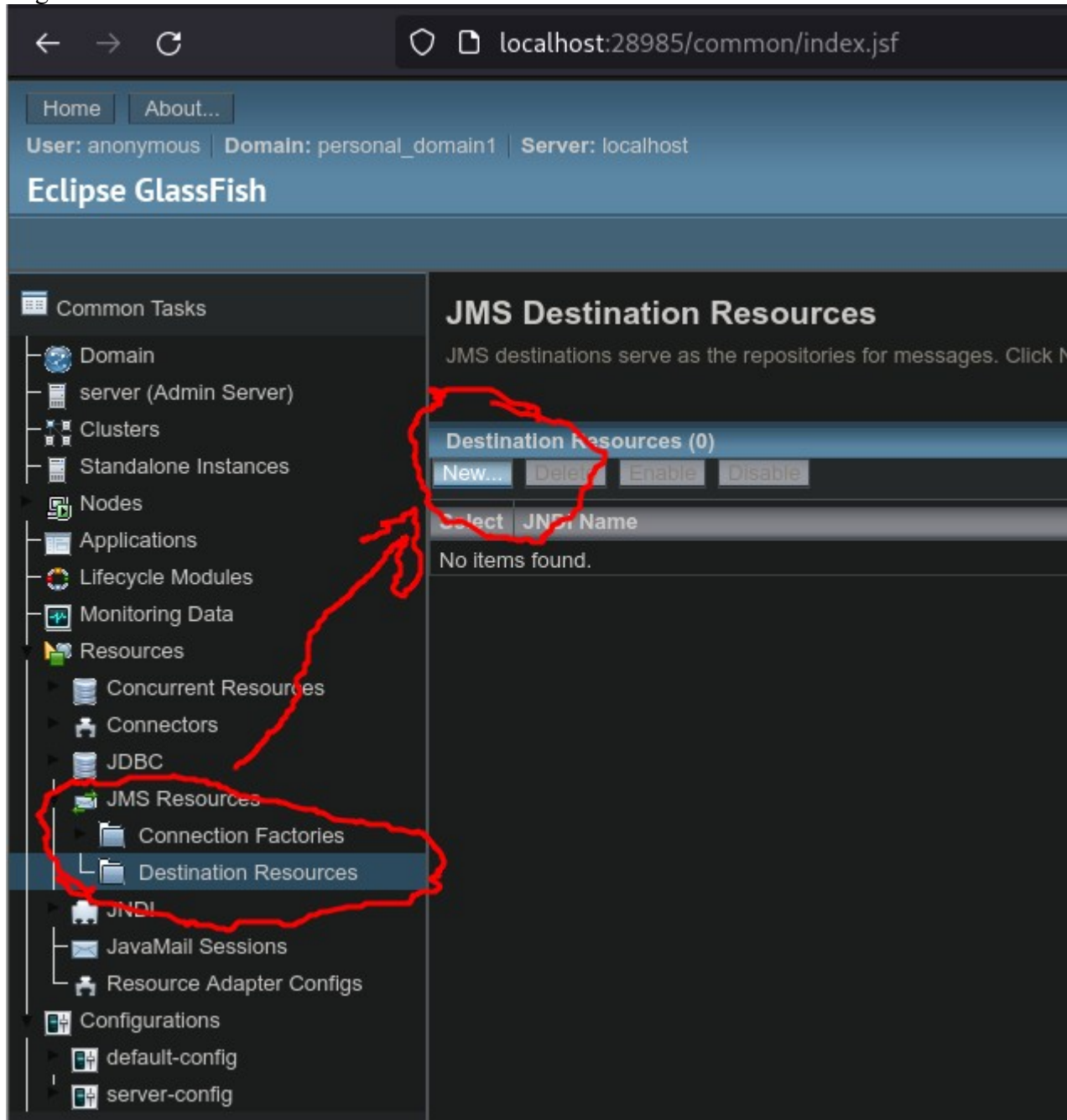
In your project: Services>Servers>Glassfish Server(Right click)>Start

6. Open glassfish console

In your project: Services>Servers>Glassfish Server(Right click)>View domain admin console

7. Create Destination Resource

In glassfish console: Resources>JMS Resources>Destination Resources>New



New JMS Destination Resource

The creation of a new Java Message Service (JMS) destination resource also creates an admin object resource.

JNDI Name: *

Physical Destination Name: *
Destination name in the Message Queue broker. If the destination does not exist, it will be created automatically when needed.

Resource Type: *

Description:

Status: ☒

8. Create Connection Factory

In glassfish console: Resources>JMS Resources>Connection Factories>New

Home About...

User: anonymous Domain: personal_domain1 Server: localhost

Eclipse GlassFish

Common Tasks

- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - Connection Factories
 - Destination Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
 - Configurations
 - default-config
 - server-config

JMS Connection Factories

Java Message Service (JMS) connection factories are objects that allow an application

Connection Factories (1)

☒ ☐ |

Select	JNDI Name	Logical Name
<input type="checkbox"/>	java:/jms/_defaultConnectionFactory	java:comp/...

New JMS Connection Factory

The creation of a new Java Message Service (JMS) connection factory also creates a connector connection pool for the factory and a connector resource.

General Settings

JNDI Name:
Resource Type:
Description:
Status: ☒

Pool Settings

Initial and Minimum Pool Size: Connections
Minimum and initial number of connections maintained in the pool

Maximum Pool Size: Connections
Maximum number of connections that can be created to satisfy client requests

Pool Resize Quantity: Connections
Number of connections to be removed when pool idle timeout expires

Idle Timeout: Seconds
Maximum time that connection can remain idle in the pool

Max Wait Time: Milliseconds
Amount of time caller waits before connection timeout is sent

On Any Failure: ☐ Close All Connections
Close all connections and reconnect on failure, otherwise reconnect only when used

Transaction Support:
Level of transaction support. Overwrite the transaction support attribute in the Resource Adapter in a downward compatible way.

Connection Validation: ☐ Required
Validate connections, allow server to reconnect in case of failure

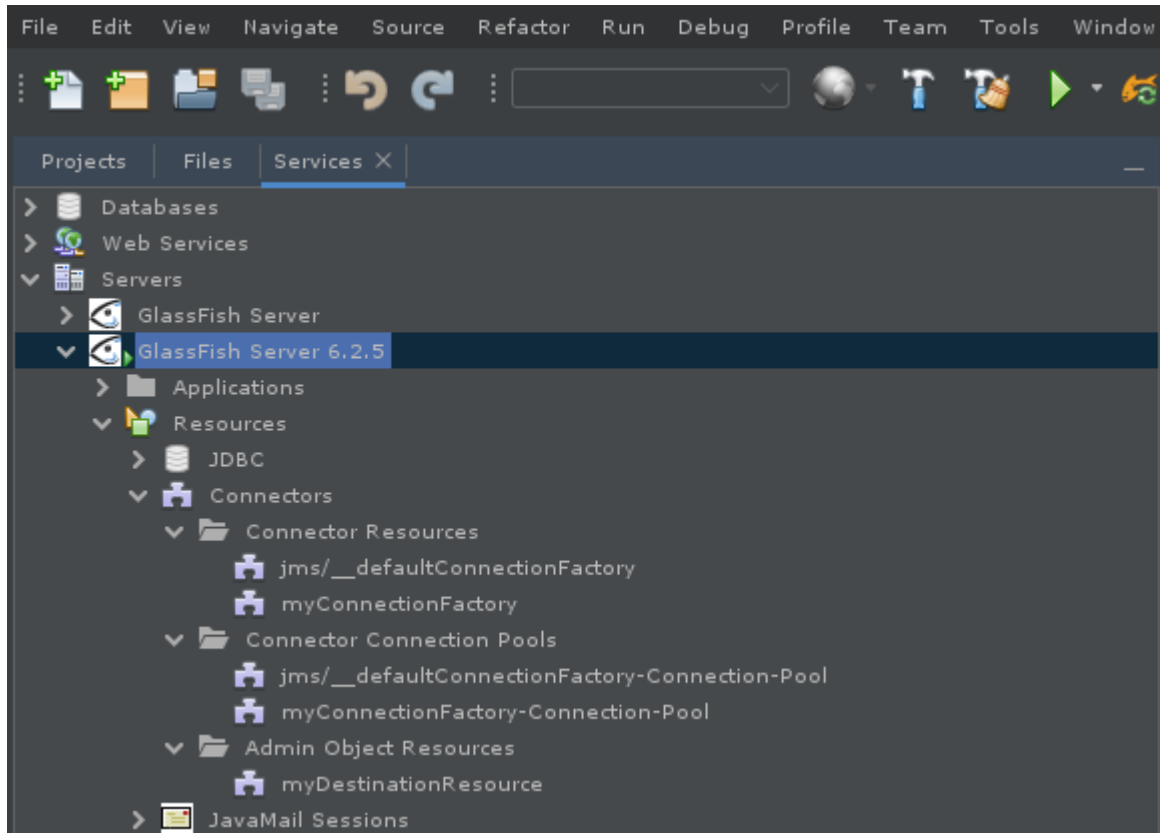
9. Restart Glassfish

in Glassfish console: `server>restart`

The screenshot shows the Glassfish console interface. On the left sidebar, the 'server (Admin Server)' is selected under the 'Domain' section. In the top navigation bar, the 'Restart' button is highlighted with a red circle. The main panel displays 'General Information' for the server, including the following details:

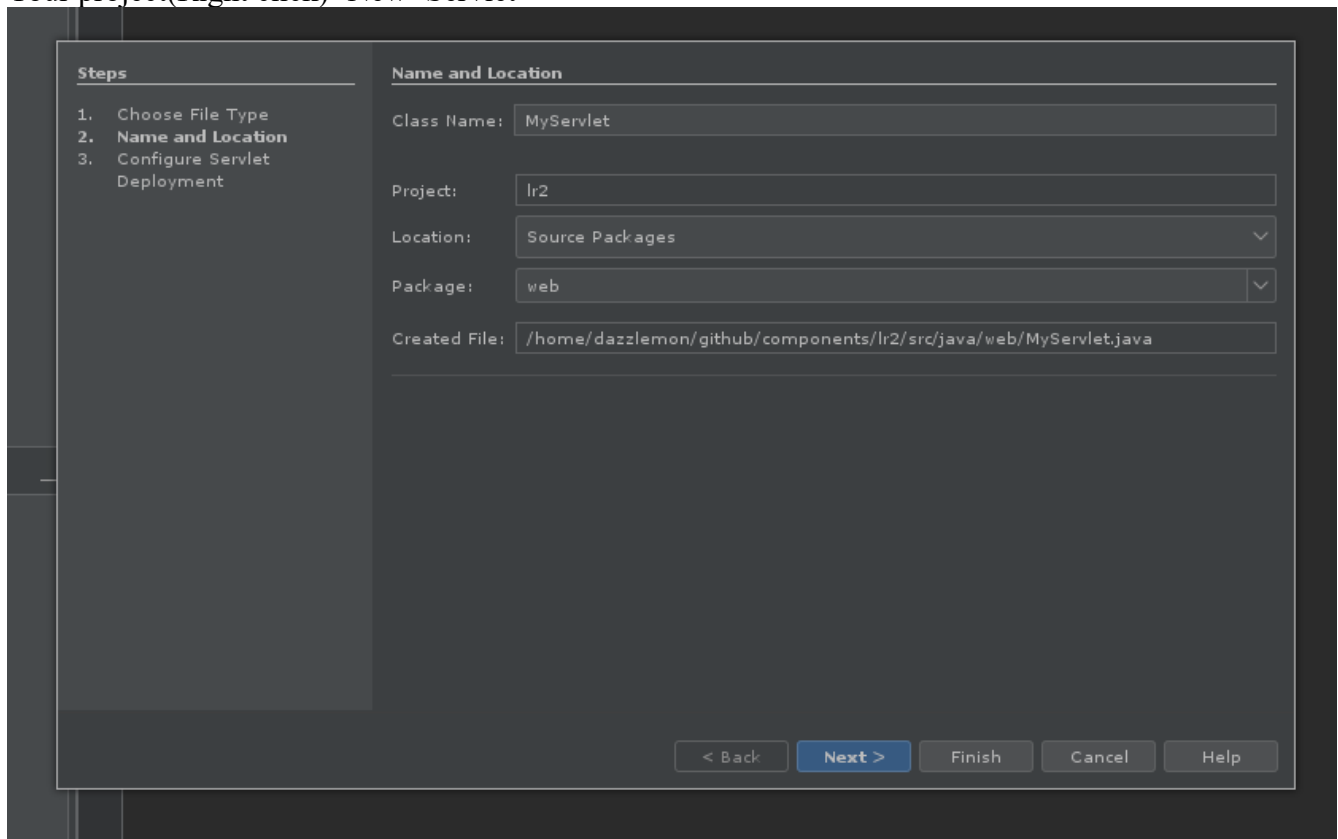
Property	Value
Name	server
Status	Running
JVM	JVM Report
Configuration	server-config
Installation Directory	/opt/glassfish/glassfish
Installed Version	Eclipse GlassFish 6.2.5 (build 6.x-b66-g0159b68 2022-02-12T17:39:59+0000)
Secure Administration	Not Enabled
Debug	Not Enabled
Up Time	8 Minutes 38 Seconds
HTTP Port(s)	28985,29017,29018
IIOP Port(s)	28975,28976,28974

Here are our resources in netbeans



10. Create servlet:

Your project(Right click)>New>Servlet



11. Edit index.html

```
<!DOCTYPE html>
```

```
<!--
```

Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Html.html to edit this template

```
-->
```

```
<html>
```

```
<head>
```

```
<title>TODO supply a title</title>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
</head>
```

```
<body>
```

```
<div>Enter a name</div>
```

```
<<form action="MyServlet" method="get">
```

```
<input type="text" name="name"/>
```

```
<button type="submit">Submit</button>
```

```
</form>
```

```
</body>
```

```
</html>
```

12. Edit MyServlet.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
 */
package web;

import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
 *
 * @author dazzlemon
 */
@WebServlet(name = "MyServlet", urlPatterns = {"/MyServlet"})
public class MyServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP GET and POST
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // Reading inputs from form
        String name = request.getParameter("name");

        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet MyServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Your name is " + name + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```

```

    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to
    edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    } // </editor-fold>

}

```

13. Build project
14. run project(it will open in your browser)
15. update code to add post method

```
/*  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template  
 */
```

```
package web;
```

```
import jakarta.annotation.Resource;  
import jakarta.jms.Connection;  
import jakarta.jms.ConnectionFactory;  
import jakarta.jms.JMSException;  
import jakarta.jms.MessageProducer;  
import jakarta.jms.Queue;  
import jakarta.jms.Session;  
import jakarta.jms.TextMessage;  
import java.io.IOException;  
import java.io.PrintWriter;  
import jakarta.servlet.ServletException;  
import jakarta.servlet.annotation.WebServlet;  
import jakarta.servlet.http.HttpServlet;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;
```

```
/**
```

```
 *
```

```
 * @author dazzlemon
```

```
 */
```

```
@WebServlet(name = "MyServlet", urlPatterns = {"/MyServlet"})
```

```
public class MyServlet extends HttpServlet {
```

```
    @Resource(mappedName="myConnectionFactory")
```

```
    private ConnectionFactory connectionFactory;
```

```
    @Resource(mappedName="myDestinationResource")
```

```
    private Queue destinationQueue;
```

```
/**
```

```
 * Processes requests for both HTTP GET and POST
```

```
 * methods.
```

```
 *
```

```
 * @param request servlet request
```

```
 * @param response servlet response
```

```
 * @throws ServletException if a servlet-specific error occurs
```

```
 * @throws IOException if an I/O error occurs
```

```
 */
```

```
protected void processRequest(  
    HttpServletRequest request, HttpServletResponse response
```

```
) throws ServletException, IOException {
```

```
String name = request.getParameter("name");
```

```
try (  
    Connection connection = connectionFactory.createConnection();  
    Session session = connection.createSession(  
        false, Session.AUTO_ACKNOWLEDGE);  
    MessageProducer messageProducer = session.createProducer(  
        destinationQueue)  
    ) {  
        TextMessage textMessage = session.createTextMessage();  
        textMessage.setText(name);  
        messageProducer.send(textMessage);  
    } catch (JMSEException e) {  
        // TODO  
    }  
}
```

```
response.setContentType("text/html;charset=UTF-8");  
try ( PrintWriter out = response.getWriter()) {  
    /* TODO output your page here. You may use following sample code. */  
    out.println("<!DOCTYPE html>");  
    out.println("<html>");  
    out.println("<head>");  
    out.println("<title>Servlet MyServlet</title>");  
    out.println("</head>");  
    out.println("<body>");  
    out.println("<h1>Your name is " + name + "</h1>");  
    out.println("</body>");  
    out.println("</html>");  
}  
}
```

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

```
/**  
 * Handles the HTTP <code>GET</code> method.  
 *  
 * @param request servlet request  
 * @param response servlet response  
 * @throws ServletException if a servlet-specific error occurs  
 * @throws IOException if an I/O error occurs  
 */  
@Override  
protected void doGet(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException {  
    processRequest(request, response);  
}  
  
/**  
 * Handles the HTTP <code>POST</code> method.
```

```

*
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
} // </editor-fold>

}

```

16. Build

17. Deploy

18. You can see number of POSTs here:

Server>JMS Physical Destinations>myDestinationPhysical>View>Number of Messages

