# BAB III EJB *SESSION* BEAN

## 3.1 Tujuan

1. Praktikan mengetahui apa itu *session* bean
2. Praktikan mengenal glashfish server pada netbeans
3. Praktikan mengetahui perbedaan *Stateless* dan *Stateful*l
4. Praktikan dapat mengetahui fungsi dari *class* Servlet

## 3.3 Alat dan Bahan

1. Laptop



Gambar 2.1 Laptop

Laptop digunakan sebagai media instalasi keperluan praktikum seperti NetBeans.

1. NetBeans IDE



Gambar 2.2 NetBeans

NetBeans adalah suatu serambi pengembangan perangkat lunak yang ditulis dalam bahasa pemrograman Java. Serambi Pada NetBeans, pengembangan suatu aplikasi dapat dilakukan dimulai dari setelan perangkat lunak modular bernama *modules*.

1. Java EE SDK



Gambar 2.3 Java *Enterprise* Edition

Java EE adalah plugin netbeans untuk membuat sebuah *Project* web yang berisi modul EJB dan modul WAR.

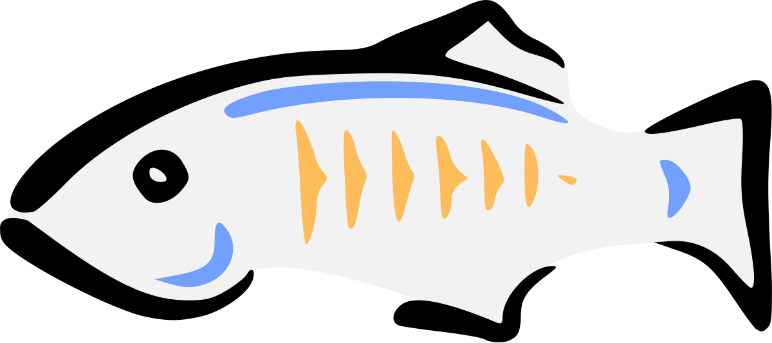
1. Java SDK 7



Gambar 2.4 Jawa SDK

Java Standard Edition Development Kit adalah implementasi salah satu dari paket Edisi standar yang ditujukan untuk pengembangan java di Windows.

1. Glassfish Server



Gambar 2.5 Glassfish Server

Glassfish server adalah salah satu web server yang kerap digunakan dalam pengembangan aplikasi web J2EE.

## 3.3 Dasar Teori

### 3.3.1 Java *Enterprise* Edition (Java EE)

Java *Enterprise* Edition ( Java EE ), sebelumnya Java 2 Platform, *Enterprise* Edition ( J2EE ), saat ini berganti nama menjadi Jakarta EE, adalah seperangkat spesifikasi, memperluas [Java SE](https://en.wikipedia.org/wiki/Java_SE) 8 [[1]](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition#cite_note-1) dengan spesifikasi untuk fitur-fitur perusahaan seperti [komputasi terdistribusi](https://en.wikipedia.org/wiki/Distributed_computing) dan [layanan web](https://en.wikipedia.org/wiki/Web_service) . [[2]](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition#cite_note-2) Aplikasi Java EE dijalankan pada [runtimes referensi](https://en.wikipedia.org/w/index.php?title=Reference_runtime&action=edit&redlink=1) , yang dapat berupa [server layanan](https://en.wikipedia.org/wiki/Application_server) mikro atau [server aplikasi](https://en.wikipedia.org/wiki/Application_server) , yang menangani transaksi, keamanan, skalabilitas, [konkurensi](https://en.wikipedia.org/wiki/Concurrency_(computer_science)) , dan pengelolaan komponen yang dikerahkannya.

Java EE didefinisikan oleh [spesifikasinya](https://en.wikipedia.org/wiki/Program_specification) .  Spesifikasi mendefinisikan [API](https://en.wikipedia.org/wiki/Application_programming_interface) (antarmuka pemrograman aplikasi) dan interaksinya. Seperti spesifikasi [Proses Komunitas Java](https://en.wikipedia.org/wiki/Java_Community_Process) lainnya , penyedia harus memenuhi persyaratan kesesuaian tertentu untuk menyatakan produk mereka sebagai yang *sesuai* dengan *Java EE* .

Java EE mencakup beberapa spesifikasi yang melayani berbagai tujuan, seperti menghasilkan halaman web, membaca dan menulis dari database dengan cara transaksional, mengelola antrian yang didistribusikan. Java EE APIs mencakup beberapa teknologi yang memperluas fungsionalitas [Java SE](https://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition)[API dasar](https://en.wikipedia.org/wiki/Application_programming_interface) , seperti [*Enterprise* JavaBeans](https://en.wikipedia.org/wiki/Enterprise_JavaBean) , [konektor](https://en.wikipedia.org/wiki/Java_EE_Connector_Architecture) , [servlets](https://en.wikipedia.org/wiki/Servlet) , [JavaServer Pages,](https://en.wikipedia.org/wiki/JavaServer_Pages) dan beberapa teknologi [layanan web](https://en.wikipedia.org/wiki/Web_service) .

*Sumber :* [*https://en.wikipedia.org/wiki/Java\_Platform,\_Enterprise\_Edition*](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition)

### 3.3.2 Component *Stateful*l

Stateful adalah disaat informasi yang diberikan sebelumnya disimpan dan mempengaruhi konten/informasi/data yang akan diberikan setelahnya. Berikut ini adalah karakterisktik Statefull komponen:

1. Server menyimpan informasi tentang *file* yang terbuka, dan posisi *file* sekarang (current position),
2. Open (dibuka) sebelum access dan kemudian ditutup, dan
3. Menyediakan *file* locks.

*(Sumber :* [*https://medium.com/@wafaakamilahmaulanihermawan/Statefull-vs-Stateless-application-6d600eddefe3*](https://medium.com/@wafaakamilahmaulanihermawan/statefull-vs-stateless-application-6d600eddefe3)*)*

### 3.3.3 Component *Stateless*

Stateless adalah disaat informasi tidak disimpan sehingga tampilan web akan sama saja kalau dilihat oleh anda berulang kali atau oleh orang lain.

1. Server tidak menyimpan state informasi,
2. Operasi *file* harus mengandung semua yang diperlukan (memuat pesan yang lengkap),
3. Dapat dengan mudah di-recovery apabila terjadi client ataupun server crach, dan
4. Membutuhkan extra lock server untuk mempertahankan state.

*(Sumber :* [*https://medium.com/@wafaakamilahmaulanihermawan/Statefull-vs-Stateless-application-6d600eddefe3*](https://medium.com/@wafaakamilahmaulanihermawan/statefull-vs-stateless-application-6d600eddefe3)*)*

### 3.3.4 Java Server Page (JSP)

JSP adalah sebuah bahasa pemrograman berbasis Java yang diperuntukkan untuk membuat sebuah website. JSP sangat sesuai dan tangguh untuk menangani kebutuhan akan sebuah website. Sama seperti PHP, bahasa yang satu ini dikhususkan untuk berkutat di bagian backend. Artinya, bagian static yang berupa tag-tag HTML akan terpisah dari kode JSP. Kita dapat membuat halaman web static dengan html dan css seperti biasanya, kemudian akan disisipi dengan JSP untuk membuat halaman web menjadi dinamis.

Ada beberapa alasan penting yang membuat JSP cukup terkenal dalam pengembangan aplikasi web:

Yang pertama, JSP menggunakan bahasa Java. Bagi sebagian orang yang sudah terbiasa dengan Java pasti tidak akan kesulitan saat mempelajari ataupun menggunakan bahasa yang satu ini.

Yang kedua, JSP mendukung multiplatform sama seperti Java. Dalam hal ini, JSP memungkinkan kode dapat dipindah-pindahkan ke berbagai platform tanpa perlu melakukan perubahan apapun.

Dan yang ketiga, JSP cenderung memiliki perfomansi yang lebih baik, karena dalam JSP, dilakukan proses compile terlebih dahulu menjadi servlet. Servlet adalah *class* Java yang siap dijalankan oleh web server.

*(Sumber :* [*https://www.mahirkoding.com/pengenalan-jsp/*](https://www.mahirkoding.com/pengenalan-jsp/)*)*

### 3.3.5 Java Servlet

Servlet adalah bahasa pemrograman Java kelas digunakan untuk memperluas kemampuan dari server yang tuan aplikasi mengakses melalui model pemrograman request-respon. Meskipun servlet dapat menanggapi setiap jenis permintaan, mereka biasanya digunakan untuk memperpanjang aplikasi host oleh server Web. Dengan demikian, dapat dianggap sebagai Java Applet yang berjalan pada server bukan browser.

Servlet adalah berbasis Java server-side teknologi web. Sesuai namanya, melayani permintaan klien dan menerima respon dari server. Secara teknis, Servlet adalah kelas Java dalam Java EE yang sesuai dengan Java Servlet API, sebuah protokol di mana kelas Java mungkin merespon permintaan. Mereka tidak terikat dengan sebuah protokol client-server khusus, tetapi yang paling sering digunakan dengan protokol HTTP. Oleh karena itu, kata “Servlet” sering digunakan dalam arti “Servlet HTTP”. [2] Dengan demikian, seorang pengembang perangkat lunak dapat menggunakan servlet untuk menambahkan konten dinamis ke server Web menggunakan platform Java. Isi dihasilkan umumnya HTML, tetapi mungkin data lain seperti XML. Servlets adalah mitra Jawa non-Jawa teknologi konten web yang dinamis seperti PHP dan ASP.NET. Servlets dapat mempertahankan negara dalam variabel sesi transaksi di server banyak dengan menggunakan cookie HTTP, atau menulis ulang URL.  
Untuk menyebarkan dan menjalankan Servlet, wadah Web harus digunakan. Sebuah wadah Web (juga dikenal sebagai wadah Servlet) pada dasarnya adalah komponen Web server yang berinteraksi dengan servlet. Wadah Web bertanggung jawab untuk mengelola siklus hidup servlets, pemetaan URL ke servlet tertentu dan memastikan bahwa URL pemohon memiliki hak akses yang benar.  
API servlet, yang terkandung dalam hirarki javax.servlet paket Java, mendefinisikan interaksi yang diharapkan dari wadah Web dan servlet [2].  
Servlet adalah sebuah objek yang menerima permintaan dan menghasilkan respon berdasarkan permintaan itu. Paket servlet mendefinisikan dasar Jawa objek untuk mewakili servlet permintaan dan tanggapan, serta sebagai objek untuk mencerminkan konfigurasi servlet parameter dan lingkungan eksekusi. Paket javax.servlet.http mendefinisikan HTTP spesifik sub*class* dari elemen servlet generik, termasuk objek manajemen sesi yang melacak beberapa permintaan dan tanggapan antara Web server dan klien. Servlets dapat dikemas dalam *file* WAR sebagai aplikasi Web.  
Servlets dapat dihasilkan secara otomatis dari JavaServer Pages (JSP) oleh kompilator JavaServer Pages. Perbedaan antara Servlets dan JSP adalah bahwa Servlets biasanya menanamkan HTML di dalam kode Java, sedangkan JSP embed kode Java dalam HTML. Sedangkan penggunaan langsung dari Servlets untuk menghasilkan HTML (seperti yang ditunjukkan pada contoh di bawah) telah menjadi langka, tingkat yang lebih tinggi MVC kerangka web di Java EE (JSF) masih secara eksplisit menggunakan teknologi Servlet untuk penanganan permintaan / tanggapan tingkat rendah melalui FacesServlet yang . Sebuah penggunaan agak lebih tua adalah dengan menggunakan servlet bersama dengan JSP dalam pola yang disebut “Model 2”, yang merupakan rasa pola model-view-controller.

*(Sumber :* [*https://aeroyid.wordpress.com/2012/07/16/javamemperkenalkan-java-servlet-dan-membuat-helloworld/*](https://aeroyid.wordpress.com/2012/07/16/javamemperkenalkan-java-servlet-dan-membuat-helloworld/)*)*

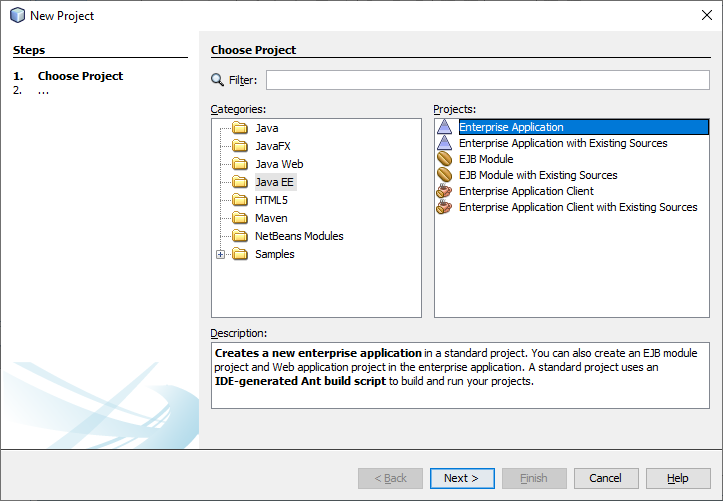
## 3.4 Langkah Kerja

1. Buka aplikasi **NETBEANS IDE**.

Di sini menggunakan Netbeans 8.0.2

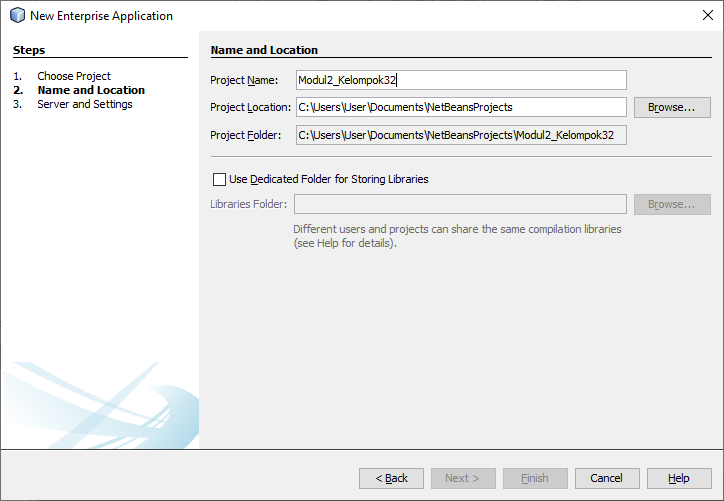
1. Buat *Project* baru pilih Java EE > *Enterprise* Application lalu klik next

Java EE adalah *Project* yang digunakan untuk membuat web service



Gambar 2.6 New *Project* Java EE

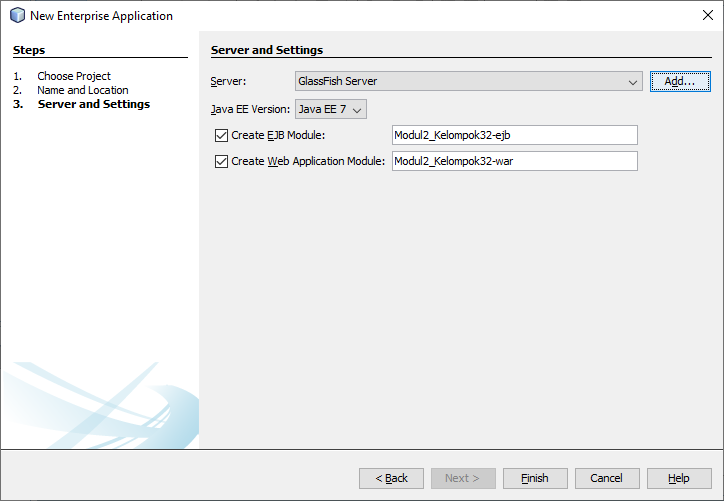
1. Pilih location *Project* lalu klik next



Gambar 2.7 Memberi nama *Project*

1. Pilih server glashfish server yang, ikuti seperti gambar.

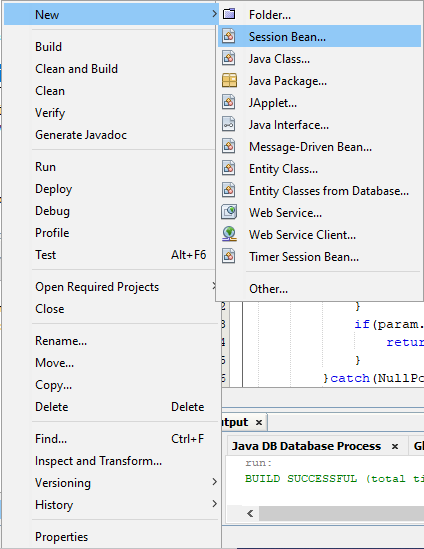
GlassFish Server adalah web server yang digunakan untuk *Project* ini.



Gambar 2.8 Memilih GlassFish Server untuk Web Server

1. Buat *file* *Session* Been baru dengan cara klik kanan *Project* EJB pilih New>*Session* Bean.

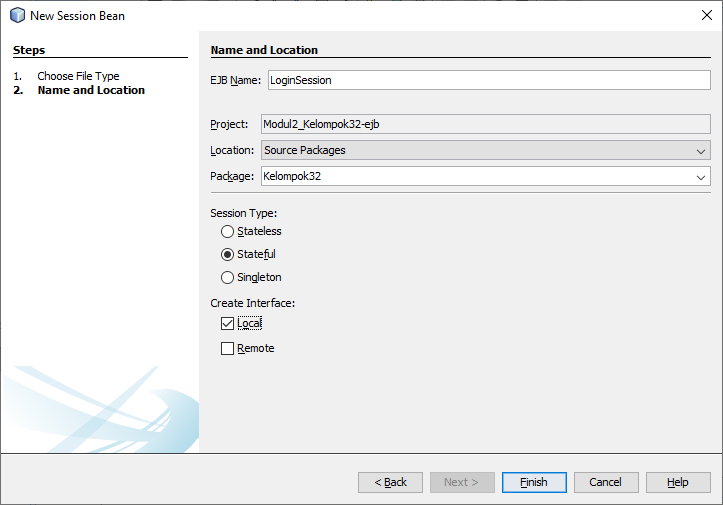
*Session* bean adalah sebuah *file* java yang dapat di panggil ke dalam servlet nantinya.



Gambar 2.9 Membuat *Session* Bean

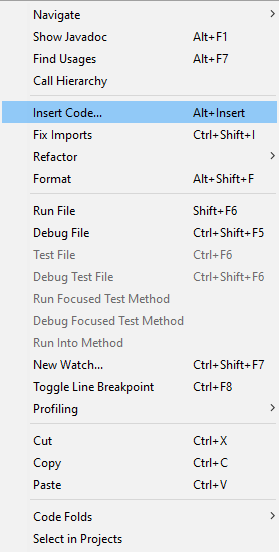
1. Pada Isikan Login*Session* pada **EJB Name**. Pada ***Session* Type** pilih *Stateful* dan centang pilihan Local pada **Create Interface**.

*Stateful* artinya data dapat disimpan secara sementara, sedangkan local interface akan membuat *file* interface yang terhubung dengan *session* bean ini.



Gambar 2.10 Memberi nama *Session* Bean

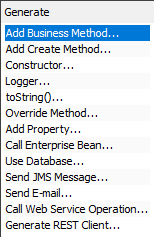
1. Klik kanan pada text editor dan pilih **Insert *Code***



Gambar 2.11 Insert *Code*

1. Pilih **Add Business *Method*e.**

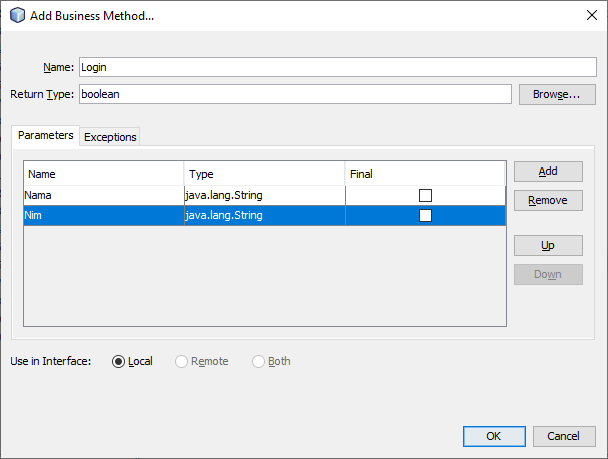
Fitur ini digunakan untuk memasukan fungsi yang terhubung langsung ke interface dengan mudah.



Gambar 2.12 Generate *code*

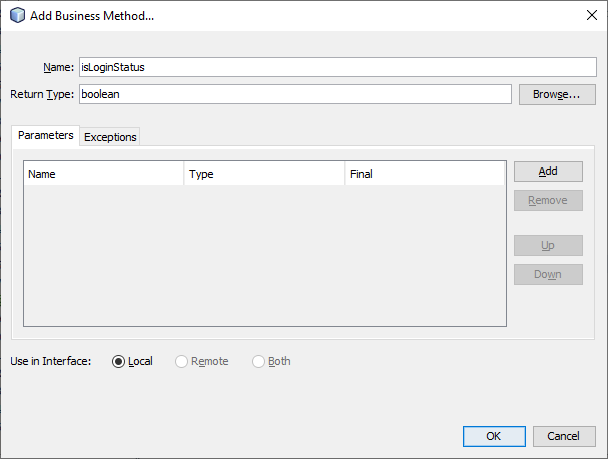
1. Masukan nama Login dengan **Return Type** boolean. Tambahkan 2 parameter dengan nama **Nama** dan **Nim**.

Parameter di masukan beserta nama fungsinya, yang otomatis akan membuat sebuah fungsi return.

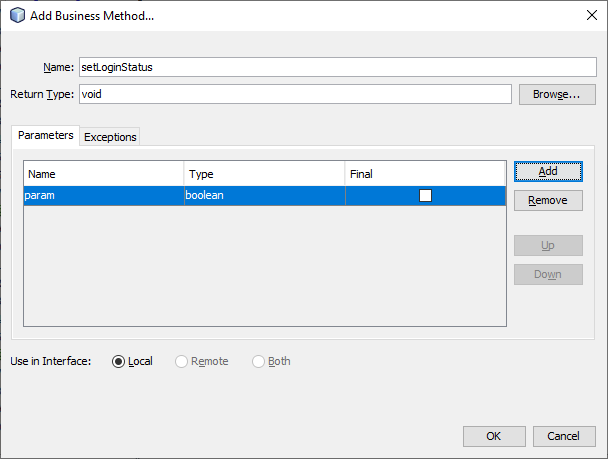


Gambar 2.13 Menambah fungsi login

1. Lakukan hal yang sama untuk *Method* **isLoginStatus** dan **setLoginStatus**



Gambar 2.14 Menambah fungsi isLoginStatus



Gambar 2.15 Menambah fungsi setLoginStatus

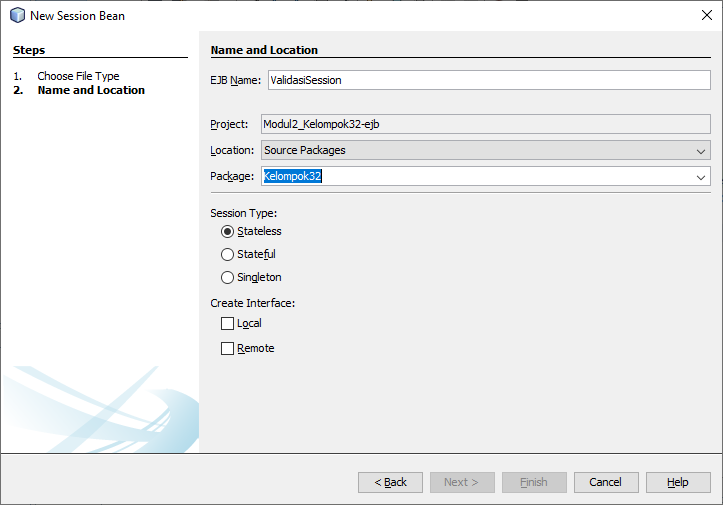
1. Tambahkan *source* *code* berikut didalam *class* Login*Session*

Ini adalah *code* utama dari fungsi java.

|  |
| --- |
| private final String [] Nama = {"Praktikum RSBK"};  private final String [] Nim = {"21120116000001"};  private boolean loginStatus = false;  @Override  public boolean Login(String Nama, String Nim) {  return Nama.equals(this.Nama[0]) && Nim.equals(this.Nim[0]);  }  @Override  public boolean isLoginStatus() {  return loginStatus;  }  @Override  public void setLoginStatus(boolean param) {  this.loginStatus = param;  } |

1. Ikuti langkah 5 untuk membuat *Session* Bean baru
2. Pada Isikan Validasi*Session* pada **EJB Name**. Pada ***Session* Type** pilih *Stateless*.

*Stateless* artinya tidak menyimpan data.



Gambar 2.16 Membuat *Session* Bean Validasi *Session*

1. Tambakan *source* *code* berikut didalam *class* **Validasi*Session***.

Ini adalah *source* *code* yang digunakan untuk memvalidasi *session* login.

|  |
| --- |
| public boolean nama(String param) {  try{  if(param.isEmpty()){  return false;  }  if(param.length()<=5){  return false;  }  }catch(NullPointerException e){  return false;  }  return true;  }  public boolean nim(String param) {  try{  Long.parseLong(param);  }catch(NumberFormatException e){  return false;  }  if(param.isEmpty()){  return false;  }  if(param.length()<14){  return false;  }  return true;  } |

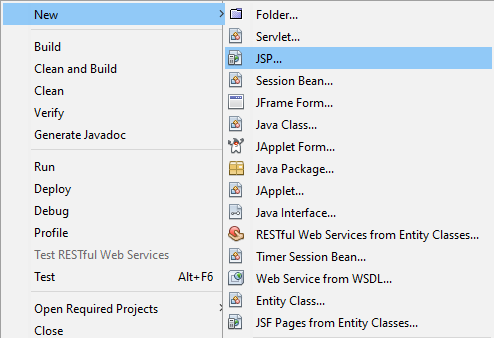
1. Buka *file* index.html di directory **Modul2\_Kelompok32-war>Web Page** dan timpa *source* *code* yang sudah ada dengan *source* *code* berikut.

*File* index.html berisikan baris kode untuk tampilan halaman awal sebuah web service.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en" dir="ltr">  <head>  <meta charset="utf-8">  <title>2020 Gelar ST</title>  </head>  <body>  <h1 style="text-align:center"> 2020 GELAR ST</h1>  <h3 style="text-align:center">Tahun depan kita wisuda</h3>  <h3 style="text-align:center">Pelajari program berikut biar bisa mewujudkannya</h3>  <h3 style="text-align:center;"><a href="Login" style="text-decoration:none"><< Klik Disini >></a></h3>  <h4 style="text-align:center">&copy; Copyright Praktikum Rekayasa Perangkat Lunak 2019</h4>  </body>  </html> |

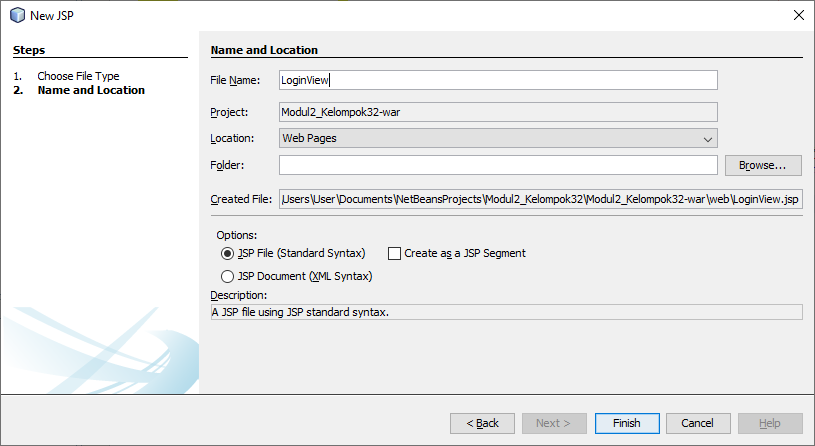
1. Buat *file* JSP dengan cara klik kanan pada *Project* war pilih **New>JSP**

*File* Javascript berguna sebagai tampilan untuk halaman baru.



Gambar 2.17 Membuat JSP Baru

1. Beri nama LoginView lalu klik finish



Gambar 2.18 Membuat LoginView.jsp

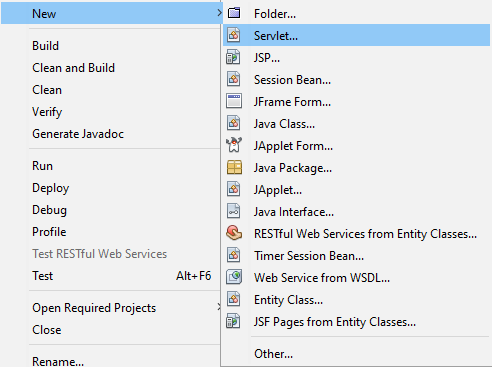
1. Paste *source* *code* berikut ke dalam *file* JSP yang dibuat.

*Source* *code* ini berisi sebuah tampilan untuk login.

|  |
| --- |
| <%@page contentType="text/html" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html lang="en" dir="ltr">  <head>  <meta charset="utf-8">  <title>Tahun Depan Wisuda</title>  <meta name="viewport" content="width=device-width, initial-scale=1.0">  </head>  <body>  <header>  <h1 style="text-align:center">SEMANGAT, TAHUN DEPAN WISUDA</h1>  </header>  <main style="display:flex;justify-content:center;flex-direction: column;">  <% String stat = (String)request.getAttribute("status");  if(stat != "true"){%>  <form action="Login" *Method*="post" style="display:flex;justify-content:center;flex-direction: column;align-items: center;">  <label for="nama" style="margin-bottom:5px">Nama Mahasiswa</label>  <input id="nama" type="text" name="nama" placeholder="Nama Mahasiswa">  <span style="color:red">${namaEr}</span>  <br>  <label for="nim" style="margin-bottom:5px">NIM Mahasiswa</label>  <input id="nim" type="text" name="nim" placeholder="Nim Mahasiswa">  <span style="color:red">${nimEr}</span>  <span style="color:red">${userEr}</span>  <br>  <input type="submit" name="submit" value="Cari" style="width:150px">  </form>  <%} else {%>  <h2 style="text-align: center">Hallo ${namaMahasiswa}, Semoga cepat lulus ya</h2>  <form action="Login" *Method*="get" style="display:flex;justify-content:center;flex-direction: column;align-items: center;">  <input type="submit" name="keluar" value="Kembali" style="width:150px">  </form>  <%}%>  </main><br>  <footer style="text-align:center">&copy; Copyright Praktikum Rekayasa Perangkat Lunak 2019</footer>  </body>  </html> |

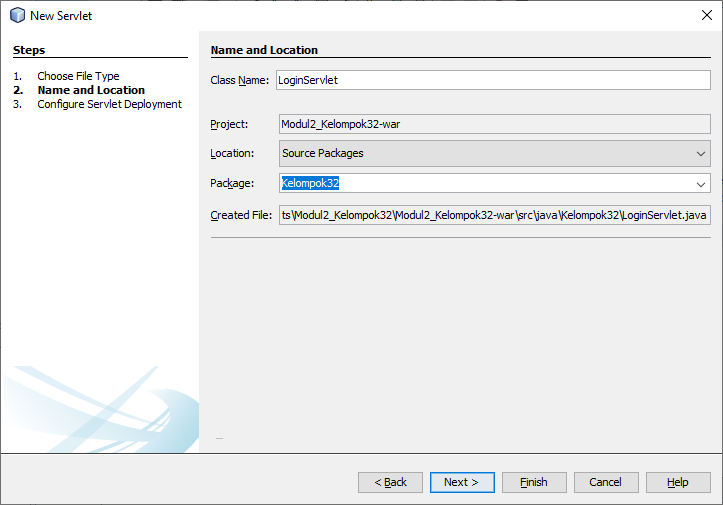
1. Buat *file* Sevlet dengan cara klik kanan pada *Project* web lalu pilih **New>Sevlet.**

*File* Servlet adalah sebuah penghubung antara *code* pada java ke web service.



Gambar 2.19 Membuat Servlet Baru

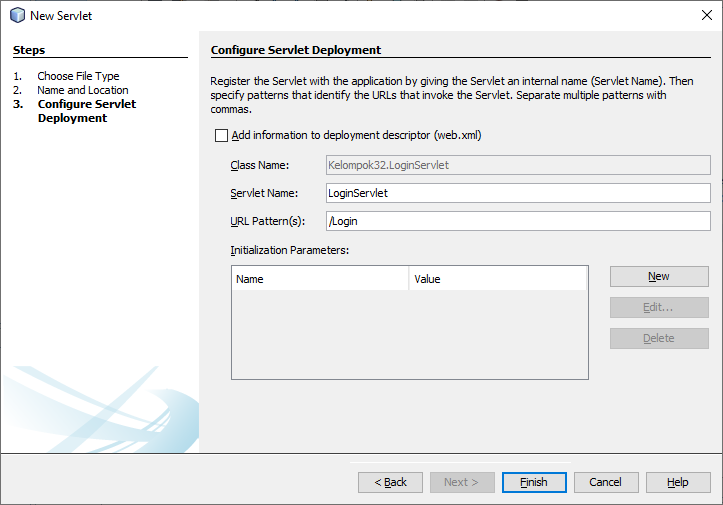
1. Beri nama LoginSevlet pada kolom ***Class* Name** dan Kelompok32padapackage.



Gambar 2.20 Membuat LoginServlet

1. Pastikan **Add information** di uncheck dan ganti URL Pattern(s) menjadi /Login

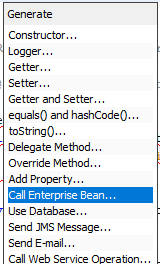
Dengan mengatur URL Pattern, kita dapat menentukan keluaran URL dari *file* tersebut.



Gambar 2.21 Memberi URL Pattern

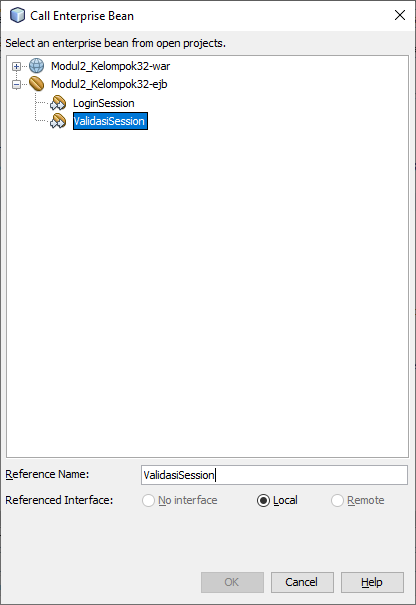
1. Import Login*Session* dan Validasi*Session* dengan cara klik kanan pada text editor dan pilih *Call* *Enterprise* Bean.

*Call* *Enterprise* Bean berfungsi untuk memanggil fungsi *Session* Bean yang telah di buat.



Gambar 2.22 *Call* *Enterprise* Bean

1. Pilih Login*Session* lalu klik OK dan ulang dari langkah 22 untuk impor Validasi*Session*.



Gambar 2.23 Memanggil EJB ke servlet

1. Hapus *source* *code* pada *Method* **processRequest,** lalutambahkan *source* *code* berikut.

*Source* *code* ini berfungsi untuk memuat *file* LoginView.jsp

|  |
| --- |
| validasi*Session* = new Validasi*Session*();    request.setAttribute("status", "false");  RequestDispatcher rd = getServletContext().getRequestDispatcher("/LoginView.jsp");  rd.forward(request, response); |

1. Tambahkan *source* *code* berikut didalam *Method* **doGet**.

*Source* *code* ini adalah *source* *code* logout atau kembali dengan menghapus *session*

|  |
| --- |
| if("Keluar".equals(request.getParameter("keluar"))){  login*Session*.setLoginStatus(false);  request.setAttribute("nimEr", "");  request.setAttribute("namaEr", "");  } |

1. Tambahkan *source* *code* berikut didalam *Method* **doPost**.

*Source* *code* ini adalah *source* *code* yang dieksekusi ketika menekan button Login.

|  |
| --- |
| String nama = request.getParameter("nama");  String nim = request.getParameter("nim");  boolean stNama, stNim = false;  stNama = validasi*Session*.nama(nama);  stNim = validasi*Session*.nim(nim);  if (stNim && stNama){  if (login*Session*.Login(nama, nim)) {  login*Session*.setLoginStatus(true);  } else {  request.setAttribute("userEr", "Mahasiswa tidak terdaftar");  }  }  else{  if (!stNim) request.setAttribute("nimEr", "Inputan Salah");  if (!stNama) request.setAttribute("namaEr", "Inputan Salah");  }    if (login*Session*.isLoginStatus()) {  request.setAttribute("status", "true");  RequestDispatcher rd = getServletContext().getRequestDispatcher("/LoginView.jsp");  rd.forward(request, response);  } else {  request.setAttribute("status", "false");  RequestDispatcher rd = getServletContext().getRequestDispatcher("/LoginView.jsp");  rd.forward(request, response);  } |

Jalankan *Project* *Enterprise* dengan cara klik kanan *Project* *Enterprise* lalu klik pilih run.

## 3.5 Hasil Percobaan

*Source* *code* Login*Session*.java

|  |
| --- |
| private final String [] Nama = {"Rio Kisna Eka Putra"};  private final String [] Nim = {"21120116130060"};  private boolean loginStatus = false;  @Override  public boolean Login(String Nama, String Nim) {  return Nama.equals(this.Nama[0]) && Nim.equals(this.Nim[0]);  }  @Override  public boolean isLoginStatus() {  return loginStatus;  }  @Override  public void setLoginStatus(boolean param) {  this.loginStatus = param;  } |

*Source* *code* diatas adalah kelas utama dari program ini yang berisi sebuah model untuk variabel nama dan nim yang sudah terisi. Kemudian juga ada pernyataan loginStatus untuk *session* dan beberapa fungsi seperti fungsi *login* untuk pembanding Nama dan nim, fungsi isLoginStatus() untuk mengambil nilai loginStatus, dan setLoginStatus() untuk mengatur nilai loginStatus.

Validasi*Session*.java

|  |
| --- |
| public boolean nama(String param) {  try{  if(param.isEmpty()){  return false;  }  if(param.length()<=5){  return false;  }  }catch(NullPointerException e){  return false;  }  return true;  }  public boolean nim(String param) {  try{  Long.parseLong(param);  }catch(NumberFormatException e){  return false;  }  if(param.isEmpty()){  return false;  }  if(param.length()<14){  return false;  }  return true;  } |

Ini adalah *Source* *code* yang digunakan untuk validasi saat login dimana mengambil nilai dari kotak teks untuk nama dan nim. Dilakukan pengecekan dari variabel yang dimasukan tersebut pada fungsi nama dan nim di kelas ini.

Index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en" dir="ltr">  <head>  <meta charset="utf-8">  <title>2020 Gelar ST</title>  </head>  <body>  <h1 style="text-align:center"> 2020 GELAR ST</h1>  <h3 style="text-align:center">Tahun depan kita wisuda</h3>  <h3 style="text-align:center">Pelajari program berikut biar bisa mewujudkannya</h3>  <h3 style="text-align:center;"><a href="Login" style="text-decoration:none"><< Klik Disini >></a></h3>  <h4 style="text-align:center">&copy; Copyright Praktikum Rekayasa Perangkat Lunak 2019</h4>  </body>  </html> |

Ini adalah *source* *code* yang berfungsi untuk menampilkan halaman utama web yang dibuat. Hanya berisi teks-teks yang diantaranya ada sebuah teks yang ketika di klik akan mengarahkan ke LoginView.jsp.

LoginView.jsp

|  |
| --- |
| <%@page contentType="text/html" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html lang="en" dir="ltr">  <head>  <meta charset="utf-8">  <title>Tahun Depan Wisuda</title>  <meta name="viewport" content="width=device-width, initial-scale=1.0">  </head>  <body>  <header>  <h1 style="text-align:center">SEMANGAT, TAHUN DEPAN WISUDA</h1>  </header>  <main style="display:flex;justify-content:center;flex-direction: column;">  <% String stat = (String)request.getAttribute("status");  if(stat != "true"){%>  <form action="Login" *Method*="post" style="display:flex;justify-content:center;flex-direction: column;align-items: center;">  <label for="nama" style="margin-bottom:5px">Nama Mahasiswa</label>  <input id="nama" type="text" name="nama" placeholder="Nama Mahasiswa">  <span style="color:red">${namaEr}</span>  <br>  <label for="nim" style="margin-bottom:5px">NIM Mahasiswa</label>  <input id="nim" type="text" name="nim" placeholder="Nim Mahasiswa">  <span style="color:red">${nimEr}</span>  <span style="color:red">${userEr}</span>  <br>  <input type="submit" name="submit" value="Cari" style="width:150px">  </form>  <%} else {%>  <h2 style="text-align: center">Hallo ${namaMahasiswa}, Semoga cepat lulus ya</h2>  <form action="Login" *Method*="get" style="display:flex;justify-content:center;flex-direction: column;align-items: center;">  <input type="submit" name="keluar" value="Kembali" style="width:150px">  </form>  <%}%>  </main><br>  <footer style="text-align:center">&copy; Copyright Praktikum Rekayasa Perangkat Lunak 2019</footer>  </body>  </html> |

LoginView.jsp berisikan sebuah tampilan yang digunakan untuk login dan menampilkan sebuah teks ketika berhasil login. Didalamnya terdapat dua kotak masukan, dengan sebuah button. kotak masukan tersebut akan di kirimkan ke servlet untuk di olah ketika button di tekan, dan kemudian akan mengeksekusi sesuai pengkondisian yang ada di servlet.

processRequest.LoginServlet.java

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");  validasi*Session* = new Validasi*Session*();    request.setAttribute("status", "false");  RequestDispatcher rd = getServletContext().getRequestDispatcher("/LoginView.jsp");  rd.forward(request, response); |

Fungsi processRequest bersisi RequestDispatcher yang berfungsi untuk memuat halaman LoginView.jsp.

doGet.LoginServlet.java

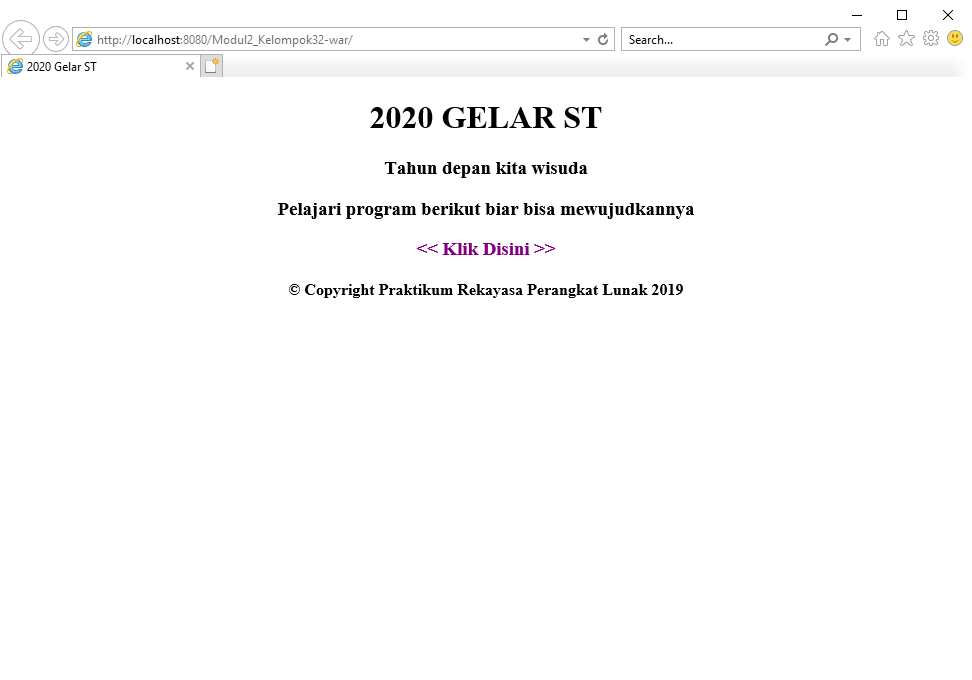
|  |
| --- |
| if("Kembali".equals(request.getParameter("keluar"))){  login*Session*.setLoginStatus(false);  request.setAttribute("nimEr", "");  request.setAttribute("namaEr", "");    }  processRequest(request, response); |

Fungsi doGet di eksekusi ketika tombol kembali di tekan. Fungsi ini akan mengubah nilai loginStatus pada login*Session* menjadi false dan menghapus nilai nimEr dan namaEr.

doPost.LoginServlet.java

|  |
| --- |
| String nama = request.getParameter("nama");  String nim = request.getParameter("nim");  boolean stNama, stNim = false;  stNama = validasi*Session*.nama(nama);  stNim = validasi*Session*.nim(nim);  if (stNim && stNama){  if (login*Session*.Login(nama, nim)) {  login*Session*.setLoginStatus(true);  } else {  request.setAttribute("userEr", "Mahasiswa tidak terdaftar");  }  }  else{  if (!stNim) request.setAttribute("nimEr", "Inputan Salah");  if (!stNama) request.setAttribute("namaEr", "Inputan Salah");  }  if (login*Session*.isLoginStatus()) {  request.setAttribute("status", "true");  request.setAttribute("namaMahasiswa", nama);  RequestDispatcher rd = getServletContext().getRequestDispatcher("/LoginView.jsp");  rd.forward(request, response);  } else {  request.setAttribute("status", "false");  RequestDispatcher rd = getServletContext().getRequestDispatcher("/LoginView.jsp");  rd.forward(request, response);  }  processRequest(request, response); |

Fungsi doPost dijalankan ketika menekan tombol login, terdapat pengkondisian berdasarkan nilai yang di masukan pada kotak masukan nama dan nim. Kemudian nilai tersebut akan di bandingkan dengan nilai yang sudah di inisialisasi pada Login*Session*.java. jika sama maka akan mengubah statusLogin menjadi true, sehingga fungsi yang di jalankan pada LoginView.jsp akan berubah ke halaman yang telah *login*. Lalu nilai namaMahasiswa akan di set sesuai dengan variabel nama pada *file* Login*Session*.java

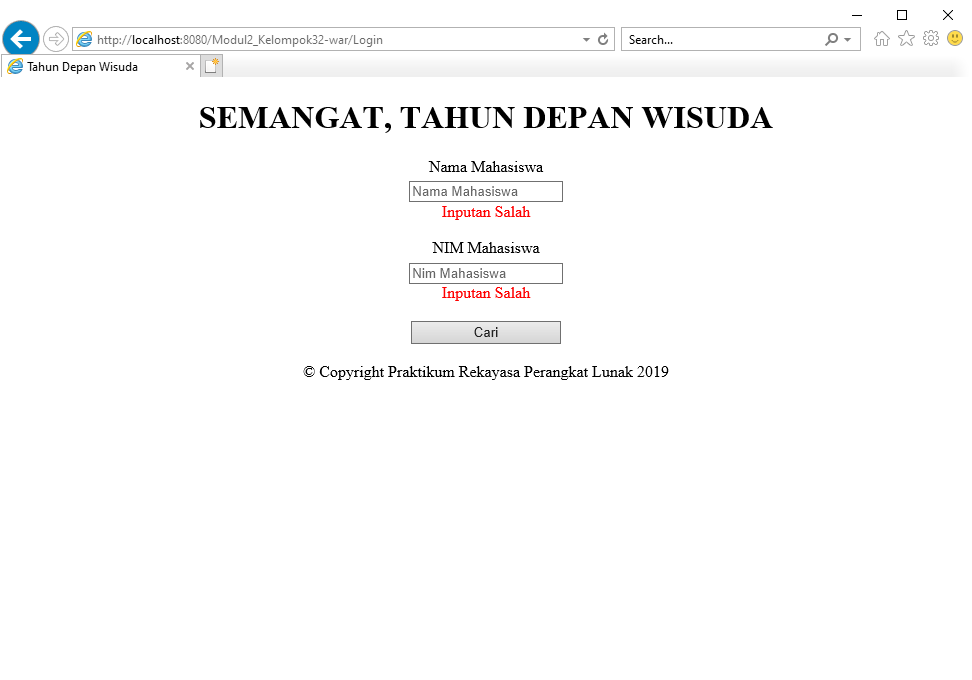


Gambar 2.24 index.html

Ketika program di jalankan akan di alihkan ke halaman ini dan dapat menekan sebuah teks “Klik Disini”. Selanjutnya akan dialihkan ke halaman baru.

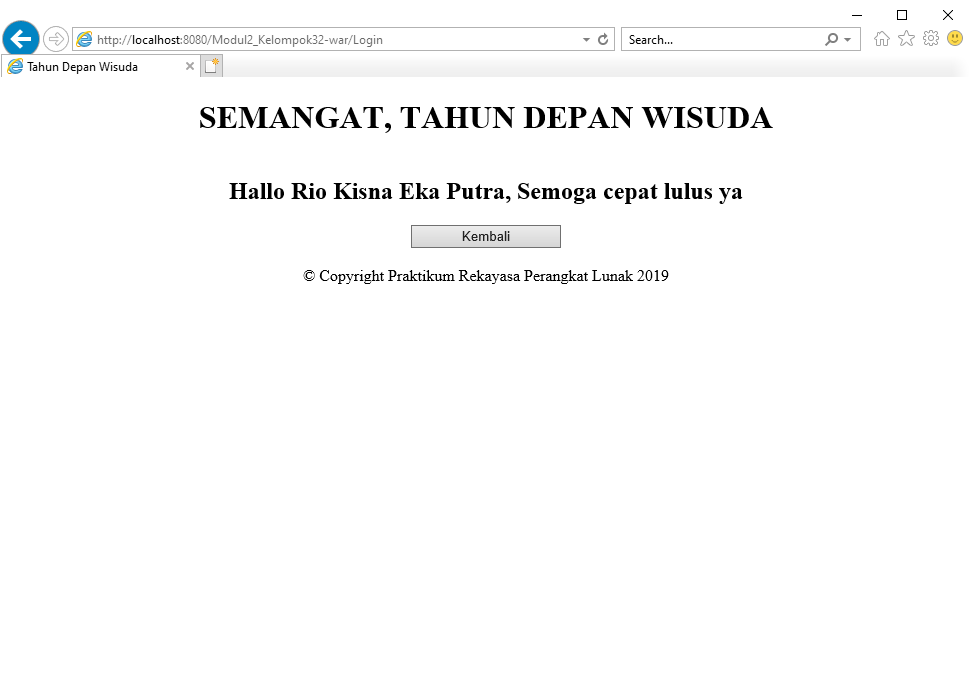


Gambar 2.25 Halaman Login



Gambar 2.26 Memasukan inputan salah

Jika masukan salah, akan di tampilkan petunjuk seperti diatas.



Gambar 2.27 Memasukan Inputan Benar

Ketika memasukan nama dan nim yang benar, akan keluar tampilan seperti diatas, dimana variabel nama akan ditampilkan

## 3.6 Tugas dan Pembahasan

Mahasiswa.java

|  |
| --- |
| String nama, nim;  public Mahasiswa(String nama, String nim) {  this.nama = nama;  this.nim = nim;  }  public String getNama() {  return nama;  }  public void setNama(String nama) {  this.nama = nama;  }  public String getNim() {  return nim;  }  public void setNim(String nim) {  this.nim = nim;  } |

*Source* *code* ini adalah sebuah model Mahasiswa yang di gunakan sebagai object yang akan di masukan kedalam array list. Terdapat 2 attribute, yaitu nama dan nim dan beberapa fungsi setter getter.

cariMahasiswa.java

|  |
| --- |
| ArrayList<Mahasiswa> datar = new ArrayList<Mahasiswa>();  public void cekpicek(String nama,String nim){  datar.add(new Mahasiswa(nama,nim));  }    public cariMahasiswa() {  cekpicek("Rio Kisna Eka Putra","21120116130060");  cekpicek("Alfian Aulia Firdaus","21120116130035");  cekpicek("Favo Perdana HS","21120116120015");  cekpicek("Farrell Denando","21120116120026");  cekpicek("Faisal Rizki R","21120116120014");  }  private String checkNama(String param){  for (int i = 0; i < datar.size(); i++){  if(param.equals(datar.get(i).nama)){  return "Nama Praktikan : " +datar.get(i).nama+"("+datar.get(i).nim+")";  }  }  return null;  }  private String checkNIM(String param){  for (int i = 0; i < datar.size(); i++) {  if(param.equals(datar.get(i).nim)){  return "Nama Praktikan : " +datar.get(i).nama+"("+datar.get(i).nim+")";  }  }  return null;  }  public String search(String param){  if (checkNama(param) != null) {  return checkNama(param);  }  else if (checkNIM(param) !=null) {  return checkNIM(param);  }  else {  return "Nope";  }  }; |

*Source* *code* diatas adalah kelas yang berisi fungsi-fungsi yang akan di panggil di servlet serta berbagai variabel yang sudah ditentukan. Pertama-tama adalah pembuatan array list, kemudian membuat sebuah fungsi yang digunakan untuk menambahkan mahasiswa ke dalam array list. Selanjutnya membuat constractor yang di dalamnya sudah membuat 5 objek mahasiswa. Kemudian ada fungsi checkNama untuk mengecek nama yang nanti dicari, sedangkan fungsi checkNim untuk mengecek nim yang nanti dicari. Lalu fungsi terakhir adalah mengambil informasi yang di gabungkan dalam sebuah string dari nama dan nim, yang nantinya akan digunakan sebagai string yang di tampilkan dalam label. Hanya satu *Method* public yang di buat, yaitu *Method* search, hal ini di karenakan untuk melindungi penggunaan *Method* lain yang bersifat tidak lengkap. Selain itu akan lebih efektif jika hanya satu *Method* public yang di panggil di servlet nantinya.

Search.jsp

|  |
| --- |
| <%@page contentType="text/html" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html lang="en" dir="ltr">  <head>  <meta charset="utf-8">  <title>Mesin Pencari Mahasiswa</title>  <meta name="viewport" content="width=device-width, initial-scale=1.0">  </head>  <body style="background-color: #b3ffec">  <header>  <h1 style="text-align:center; background-color: #33ffcc;">MESIN PENCARIAN MAHASISWA YANG SEGERA WISUDA</h1>  </header>  <main style="display:flex;justify-content:center;flex-direction: column;">  <form action="search" *Method*="post" style="display:flex;justify-content:center;flex-direction: column;align-items: center;">  <h2 for="nama" style="margin-bottom:5px">----------------------------Cari Mahasiswa----------------------------</h2>  <input id="nama" type="text" name="param" style="background-color: white;color: #3CBC8D;border: none;width: 90%;padding: 12px 20px; margin: 8px 0; box-sizing: border-box;border-radius: 3px;" placeholder="Masukan Nama / NIM">  <span style="color:red ;font-size: 20px">${show}</span>  <br>  <input type="submit" name="submit" value="Cari" style=" background-color: #4CAF50; border: none; color: white; padding: 16px 32px; text-decoration: none; margin: 4px 2px; cursor: pointer; width: 90%">  </form>  <br>  <form action="search" *Method*="get" style="display:flex;justify-content:center;flex-direction: column;align-items: center;">  <h2 for="nama" style="margin-bottom:5px">----------------------------Tambah Mahasiswa----------------------------</h2>  <input id="nama1" type="text" name="nama1" style="background-color: white;color: #3CBC8D;border: none;width: 90%;padding: 12px 20px; margin: 8px 0; box-sizing: border-box;border-radius: 3px;" placeholder="Masukan Nama">  <br>  <input id="nim1" type="text" name="nim1" style="background-color: white;color: #3CBC8D;border: none;width: 90%;padding: 12px 20px; margin: 8px 0; box-sizing: border-box;border-radius: 3px;" placeholder="Masukan NIM">  <br>  <input type="submit" name="submit" value="Tambah" style=" background-color: #4CAF50; border: none; color: white; padding: 16px 32px; text-decoration: none; margin: 4px 2px; cursor: pointer; width: 90%">  </form>  </main><br>  <footer style="text-align:center;background-color: #33ffcc;">&copy; Copyright Kelompok 32 Praktikum RSBK 2019</footer>  </body>  </html> |

Search.jsp adalah *file* yang berisikan sebuah *code* javascript yang digunakan untuk menampilkan sebuah halaman pencarian dan penambahan data mahasiswa. Pada *source* *code* diatas terdapat 2 buah action, yaitu post search dan get search. Post search digunakan untuk mencari data mahasiswa yang sudah terdaftar. Terdapat sebuah kotak masukan, serta sebuah button untuk mencari. Kemudian pada action get search, berisi 2 buah kotak masukan dan sebuah button.

Penggunaan request get disini dilakukan agar action untuk menambah data dapat di lakukan pada servlet yang sama. Karena servlet tidak dapat menyimpan data ketika beralih ke servlet lainnya, sehingga untuk menambahkan data pada servlet terkait harus dilakukan pada halaman ini juga.

processRequest.cariServlet.java

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");    RequestDispatcher rd = getServletContext().getRequestDispatcher("/search.jsp");  rd.forward(request, response); |

Fungsi processRequest berisi RequestDispatcher yang digunakan untuk memuat *file* search.jsp saat program dijalankan.

doGet.cariServlet.java

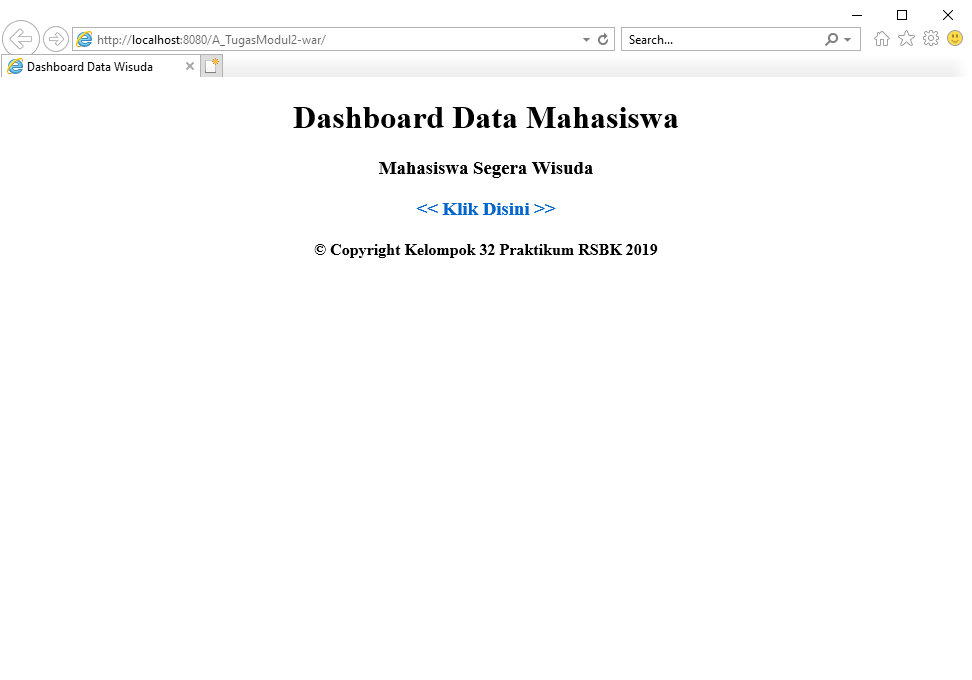
|  |
| --- |
| cariMahasiswa.cekpicek(request.getParameter("nama1"), request.getParameter("nim1"));  processRequest(request, response); |

Fungsi doGet di eksekusi ketika action get search pada search.jsp di jalankan. Isinya berupa pemanggilan fungsi cekpicek dari *class* cariMahasiswa dengan parameter nama1 dan nim1 untuk menambahkan objek baru pada sebuah arraylist.

doPost.cariServlet.java

|  |
| --- |
| String param = request.getParameter("param");  String cek = cariMahasiswa.search(param);  if (cek=="Nope"){  request.setAttribute("show", "Mahasiswa tidak terdaftar");  }  else {  request.setAttribute("show", cek);  }  processRequest(request, response); |

Fungsi doPost digunakan untuk memanggil fungsi search pada *class* cariMahasiswa dengan parameter param, kemudian nilai di simpan pada variabel cek. Variabel cek kemudian di bandingkan, jika bernilai “Nope” (nilai default diubah ke bentuk “Nope), maka data mahasiswa tidak ada, sehingga atribut show di tuliskan demikian, namun jika ternyata nilai cek itu lainnya, maka segera di tuliskan nilai cek tersebut ke atribut “show”.



Gambar 2.28 index.html tugas



Gambar 2.29 Halaman pencarian



Gambar 2.30 Mencari mahasiswa yang terdaftar



Gambar 2.31 Mencoba mencari mahasiswa yang belum terdaftar



Gambar 2.32 Mahasiswa tidak ditemukan



Gambar 2.33 Menambahkan mahasiswa baru



Gambar 2.34 Mahasiswa baru telah di tambahkan

Link Github : <https://github.com/riokisna/Praktikum-RSBK-Kelompok32>

## 3.7 Kesimpulan

1. *Stateful* komponen berguna untuk membuat *session* yang dapat menyimpan informasi secara sementara.
2. *Call* *Enterprise* Bean memudahkan pemrogram untuk memasukan sebuah fungsi *Session* Bean ke dalam Servlet.
3. Servlet tidak dapat menyimpan data ketika ditutup atau berpindah ke servlet lainnya.
4. RequestDispatcher berfungsi untuk memuat sebuah *file* JSP, sehingga bahasa permorgraman untuk tampilan web bisa di lakukan di *file* tersebut.
5. Local Interface digunakan digunakan untuk menghubungkan fungsi dan *Method* yang ada di *session* bean ke sebuah Servlet