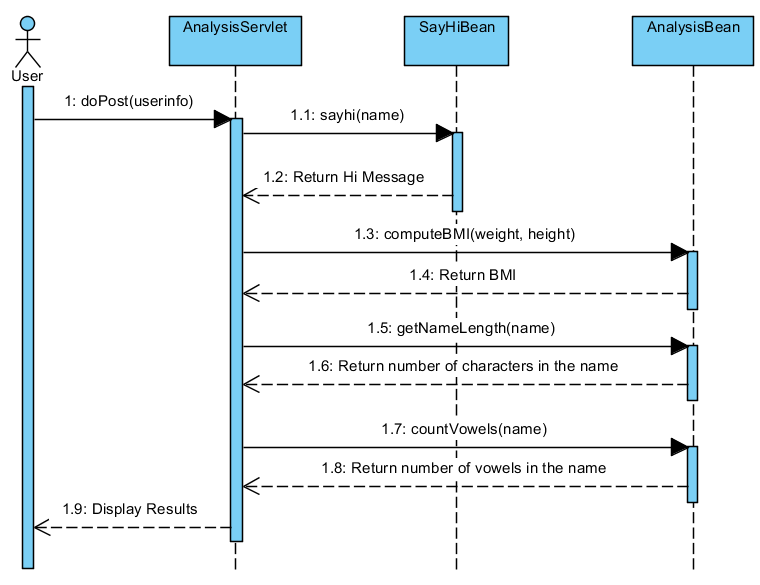
Course : Diploma in Multimedia & Infocomm Technology (EGDF15)

Module : Java Enterprise Development (EG3752)

|  |  |
| --- | --- |
| Laboratory : | Lab 6 – Enterprise JavaBeans (EJB) |
| Objectives : | At the end of this session, you should know how to:   * Construct a no-interface stateless session bean * Develop Java EE Web Application that uses EJB to provide database transactional services |
| Software Used : | Java Standard Development Toolkit (JDK™) 8.0  NetBeans IDE 8.0 with GlassFish Server 4.0 bundle  MySQL Community Edition 5.7 or WAMP Server + MySQL Workbench 6.3 |

Analysis Application

In this task, we will be creating a very simple programme that will say Hi to a person as shown in the sequence diagram below. A basic no-interface stateless session EJB will be used.



Setting up the Skeleton of the Application

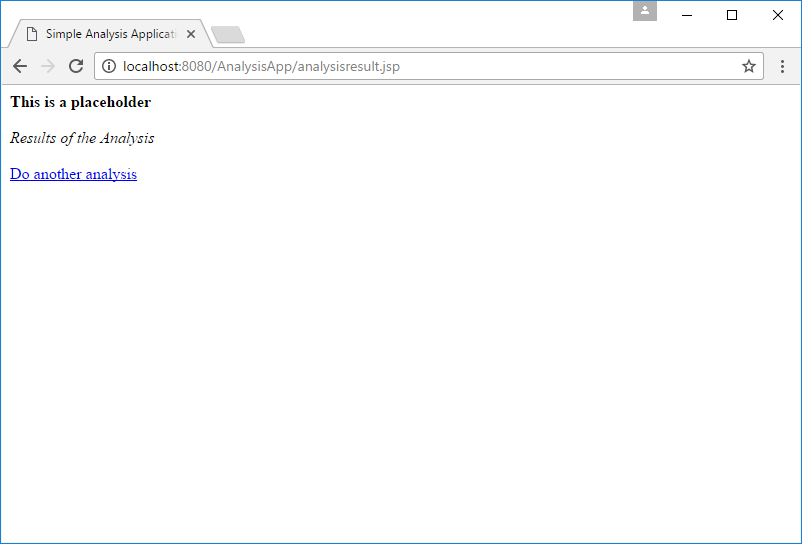
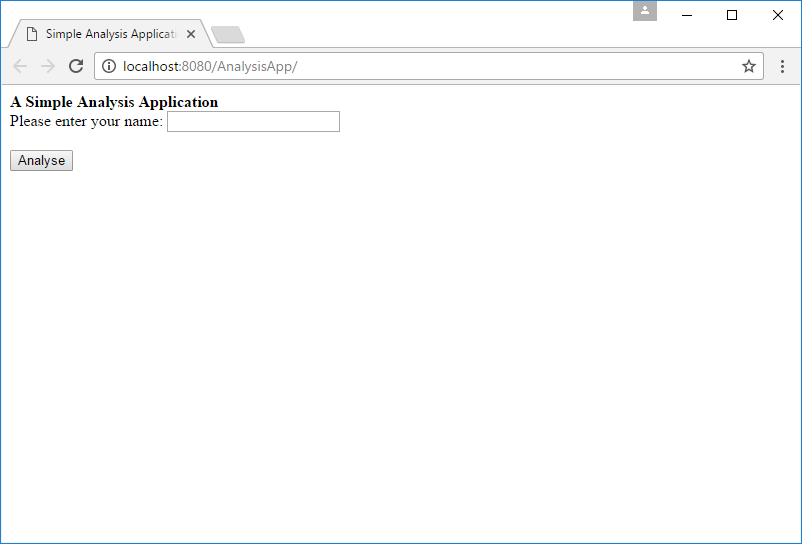
We will create a skeleton for the web application first to lay the foundation for us to code the EJB later.

1.1 Create a new **Java Web Application** named **AnalysisApp**.

1.2 Replace **index.html** with **Code Snippet Lab 6 P 1.2**.

1.3 Create a Java class named **AnalysisServlet** under a package named **com.analysis**. Replace all the codes in there with **Code Snippet Lab 6 P 1.3** for quick creation.

1.4 Create a results page named **analysisresult.jsp**. Use **Code Snippet Lab 6 P 1.4** to help you create the page.

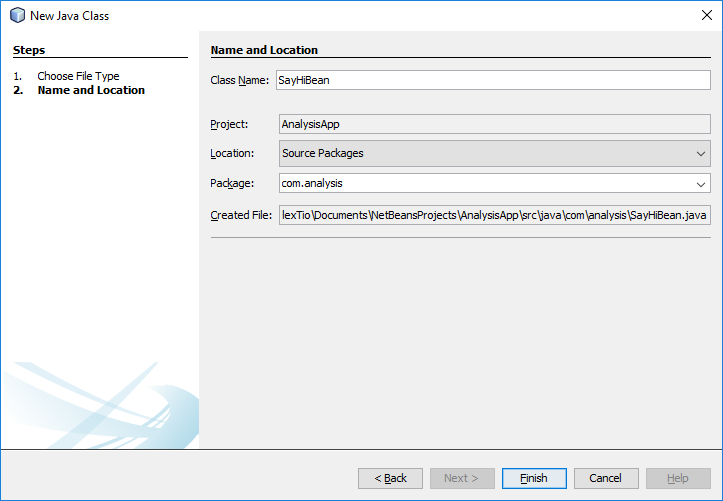
1.5 Test your application to ensure that it runs smoothly. It should show the following screens:

Coding your first EJB

Next, we will create a stateless session EJB named **SayHiBean**.

2.1 Add a new **Java Class** named **SayHiBean** to the **com.analysis** package of your project.

**IMPORTANT NOTE: Your EJB must have a package in order to run!**



2.2 Import **javax.ejb.Stateless** into the class.

2.3 Convert the class into an EJB by adding **@Stateless** before the line **public class SayHiBean**.

2.4 Write a method call **sayHi** that takes in the name that the user provided as an argument and returns a message *“Hi “ + <name which the user provided> + “!”*.

Injecting the EJB and Invoking the Function

3.1 Go back to AnalysisServlet.java and import **javax.ejb.EJB** into the class.

3.2 Inject the **SayHiBean** into the Servlet by declaring **SayHiBean** as a private member variable of the **AnalysisServlet** and annotating it with the **@EJB** annotation just below the comment //Inject your EJB here found near line 19 of your codes.

Your codes will look something like this:

|  |
| --- |
| @EJB  private SayHiBean sayHiBean**;** |

Take note of the coding convention. The first SayHiBean which represents the name of the class begins with an uppercase “S” while the second sayHiBean which represents the name of the variable begins with a lowercase “s”.

3.3 To invoke the **sayHi** method within the EJB, treat the EJB just as any other Java class and call its method in the same manner as how you could call it using other Java class. Remember to assign the return data into the variable named **hiMessage**.

You can fill in this line of codes near line 40, under the comments //Invoke the EJB function to create the customised message.

3.4 Test your web application once again to check if your EJB is working.

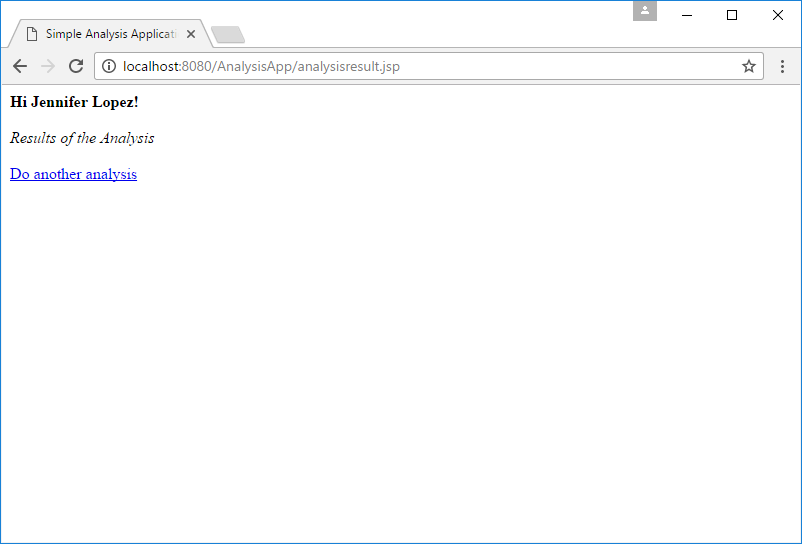


Figure 3.4. A screenshot of what the working application should display

More EJB Practice

4 Using the same project you have created, implement the following *additional* functions into a second EJB called **AnalysisBean** and display the results in analysisresult.jsp:

* Request for the weight and height of the user
* Compute and display the BMI of the user
* Compute the length of the name of the user in number of characters
* Count the number of vowels that appears in the name of the user

Managing Database Transactions in EJB

One of the main usage of EJBs is to interface database related transaction so that we will not overload the servlets with transaction management related codes.

|  |
| --- |
| When using the EJB, please take note of the following:   * Keep servlet related codes in the servlets. Do not pass the session, request or response object into the EJB for processing. * Put database related codes in the EJBs. This is to take advantage of the EJB’s database transactional services. * Do not call for commit and rollback. EJB handles commit and rollback automatically and calling commit and rollback will corrupt your application’s database processes. * As such, there is no need to set auto commit parameters for the connections. |

5. Create a new **BookManagement** application such that all database related codes are being handled by the EJB instead. You may use a single EJB named **CatalogueBean** for all database related operations.

**WARNING: Do not overwrite your original BookManagement application. This project is to be used again in a self-directed practical session on the Java Persistence API and Login Authentication.**

Below are the requirement specifications for your reference once again.

|  |  |
| --- | --- |
| **Requirement Reference** | **Requirement Specifications** |
| 1.1 Search | The system shall provide the functionality for users to search books available in the library using part of the title. |
| 1.2 Search Result Display | The system shall display all the details of the book. |
| 2.1 Add Book | The system shall provide the functionality for users add a new book into the system.  The system shall validate all data input by the user. In particular, the system will have to verify that a valid ISBN number is provided. |