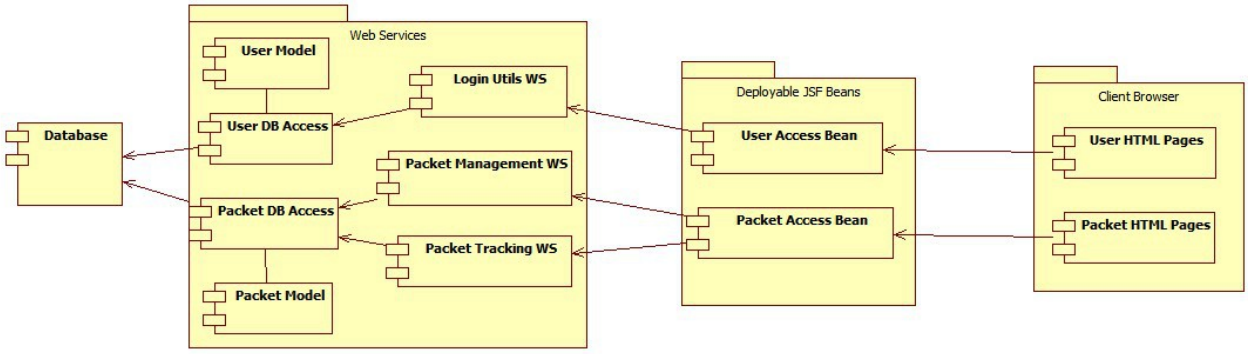
Distributed Systems – assign 5

-Stefan Prisca, group 30441-

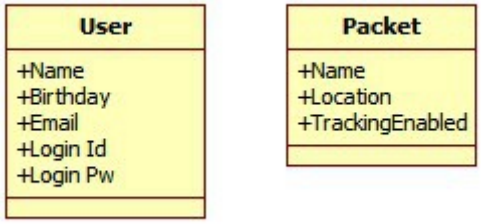
Conceptual Diagram



In the above diagram, you can clearly see all the important parts of this application and how they interect with each other:

1. The Applicaiton Database: is the database used to store information about books, which also includes book publishers, and about the application users.
2. The Web Sevices component: This includes all components that run as Web Services :
    1. The User DB Access Logic: this is the persistence API and extra bussines logic that is used to access the DB to gain information about users.
    For this application, Eclipse Link was chosen as the persistence API. It offers a wide range of possible database connections (including SQL Dbs and NonSql dbs). It is also easy to configure through xml files, and has a central maven repository that facilitates the maven build.
    2. The Packet DB Access logic will handle data access for the packets.
    For this, .NET Entity framework is used.
    3. Application Models: These are in-memory representations of the data models from the database. There are two different models, the User model, and the Packet Model. These have been separated because they are used by different parts of the application.
    4. The LoginUtils Web Service is a SOAP web service that handles user management requests like login validation, registering a new user, etc.
    5. The Packet Management component will deal with CRUD operations on packets, and will register packets for tracking.
    6. The Packet Tracking WS will provide methods to update the packet's location and to check where the packet is.
3. The Deployable JSF Beans are the backing beans of the User HTML browser pages.
There is one for user management and one for packet management.
4. Client-side HTML pages: contains the web pages that will be displayed in the browser.
These were build using the JSF2 technologies.

Data Design



Packet

The representation of packets:

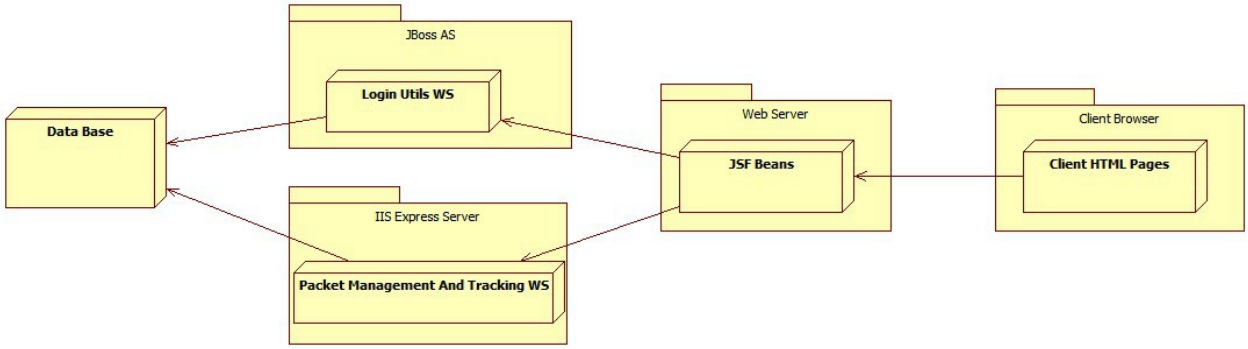
- Name: The name of this packet.
- Location: The location of this packet.
  It can be null, but if tracking is enabled, it should not be.
- Tracking Enabled: a boolean value that determines if this packet is beeing tracked or not.

Users

First data element consists of the table that contains all the user information that is required for the application to run:

- id: database identifier for the current user
- name: name of the user
- birth date: the birth date of the user
- email address: the email address of the user.
- LoginID: the id the user is using to log in the application
- LoginPW: the password used to log in to the application
- type: the type of the user: administrator or regular user.

Deployment Diagram



The above diagram shows how the components will be deployment into the running application:

1. The Application Database will be separate from the application, and can be hosted on any Micrisoft SQL server.
2. The Application Server will contain the login utilities WS, including the data base access and models.
   Note that the web services and servlets can be deployed on any other AS or Container.
3. The IIS Express Server will contain the logic required for managing packets, with all db access and models.
4. The JSF Client Beans (Including the UserManagement beans) will also be deployed on an Web Server. As these are developed as a web application, the logic will be available on the entire network.
5. The User HTML pages will be displayed in the user's browser.

Installing the application

It is recommended to use maven tool for building and deploying the project.
All the pom files are configured, so in order to run the build, do the following:

1. Clone the project git repository
2. Go to the App4 folder.
   There should be two projects here:
   1. ro.stefanprisca.distsystems.app5 -> The jsf client-side implementation.
      This is the JSF client project
   2. PacketTracer_AddRm
      This contains the .NET Web services for packet tracing and management.

*(A) LoginUtils and the jsf Client (*ro.stefanprisca.distsystems.app5*):*

1. Before installing the application, you'll have to install the mssql jdbc driver and the web service for user management:
   1. MSSQL JDBC driver:
      1. go to *git_root/utils/mssqljdbc* folder
      2. you should have the *sqljdbc4.jar* driver there.
      3. For windows systems, there is a *mvn_install.bat* file that will install this driver using maven.
      4. For other systems, use the command:
         mvn install:install-file -Dfile=sqljdbc4.jar -DgroupId=com.microsoft.sqlserver -DartifactId=sqljdbc4 -Dversion=4.0 -Dpackaging=jar
   2. LoginUtils Web Service:
      You need this installed in the local maven repository because the JSF client project has a dependency on it, as it is required to use the objects provided by the service.
      1. go to *git_root/utils/* folder
      2. you should have the *ro.stefanprisca.distsystems.utils.login* project there.
      3. Also, there is a maven aggregator for installing all the utilities (if there will be more in the future). So, the only thing you need to do is run a *mvn clean install* on that aggregator.
2. After maven has installed the jdbc driver and the usermanagement utils in your local repository, go back to the application folder (App5) and run *mvn clean install*
3. This will install the application, and generate the *ro.stefanprisca.distsystems.app5* file in ..\App5\ro.stefanprisca.distsystems.app5\target

An alternative to steps 3-5 is using the common maven aggregator from the repository root folder. This will install the App4 project and the login utilities project. Also, you can add any other project to the module list in this aggregator, and it will install it.
Note: you will still need to manually install the jdbc driver as explined in step 3.1 from above.

Deploying the JSF application using Jboss AS 8.1(Wildfly).

In order to deploy the application, you'll have to install Jboss AS 8.1 as a local service first.

After you did this, copy the above mentioned *ro.stefanprisca.distsystems.app5.war* and the *utils/usermanager/ro.stefanprisca.distsystems.utils.login.war* files in the standalone deployment folder of JBoss.

Now you just have to start Jboss and access the page:
http://localhost:xxxx/PacketTracer/home.xhtml
Further instructions on how to deploy applications with jboss can be found here.

Opening the application in Eclipse:

The App5 project:
If you have the Eclipse Maven plug-in installed, all you need to do is import the projects as Maven projects.
Note that this is the recommended way to import the projects in Eclipse.

If not, you'll have to select *Create New Project,* deselect the *Use default location* option, and set the project location to the *ro.stefanprisca.distsystems.app5b* folder.

The Usermanagement web service:

This project can also be imported using Eclipse Maven.
You can also import this from eclipse, using the method presented above (by creating a new project in that file location).
Note that after you create the project, you'll have to make sure to import all the required libraries.

Further note: you don't really need to open this project. If you want to run the application, just copy the war file in Tomcat, and the service will perform as expected (tests have been made :-).

(B) Opening the PacketTracer_AddRm

Prerequirements:

The application uses the following NuGet packages:

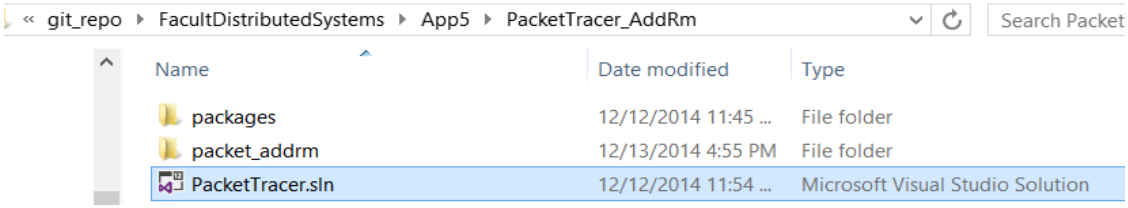
- EntityFramework v6.0.0

You can get this packages using the NuGet Package manager from VS.

It is recommended to use Visual Studio 2013 for this application, with the above-mentioned versions of the packages.

Opening the application

1. Clone the repository on your local machine.
2. From VS 2013:
    1. Go to File -> Open Project
    2. Go to the repository folder (named *FacultDistributedSystems)*, and select the *App5/PacketTracer_AddRM/PacketTracer.sln* solution file:



    3. Click on Open.
3. The solution should now be opened in your Visual Studio IDE.

After the application is opened, you'll have to configure the connection string for the db connection. I do not provide my database in the repository.
The best way to do this is to allow Entity to create a db and configure the connection string for you from the Model/Packet.cs class.
See this tutorial on how to configure code-first databases with Entity framework.

Running the whole application.

Before starting the application, you'll need to also configure the local database, for the users and the packets. You will find the specifications for this data bases in the *persistence.xml* file:

- *\App5\ro.stefanprisca.distsystems.app5\src\main\resources\META-INF\persistence.xml*

Also, you might want to add some accounts in the Users database in order to log in. I usually use the following two:

1. administrator account: id = admin; pw = admin
   Use this account to test the administrator functions
2. regular user account: id = user; pw = user
   Use this acount to test the regular user functions.

Alternativly, you can use the Register service to register a new user.

After you deploy the application using jboss, and you access the application through the browser, you'll have to log in, or to register.

1. **Regular user page**:
   Here you will find all packets.

Packet Name	Tracking	Tracking Actions		Delete
hello	true	Update	Check	Delete
Anoter Packet	true	Update	Check	Delete

Add New Packet

LogOut

Furthermore, you can:

1. Add a new packet.
   At this point you will be redirected to the Add Packet page:

Note that by adding a location the packet will be registered for tracking automatically.

Packet Name : New Packet

Packet Location : Some Packet Location

Add Packet

2. Register Packets for tracking:

Packet Name	Tracking	Tracking Actions		Delete
hello	true	Update	Check	Delete
Anoter Packet	true	Update	Check	Delete
New Packet	true	Update	Check	Delete
Untracked Packet	false	Enable Tracking		Delete

Add New Packet

LogOut

3. Check or update Tracking Information:

Packet Name	Tracking	Tracking Actions		Delete
hello	true	Update	Check	Delete
Anoter Packet	true	Update	Check	Delete
New Packet	true	Update	Check	Delete
Untracked Packet	false	Enable Tracking		Delete

- The requested location is: "The second Location "

When updating, you will be redirected to the update page.

Enter new location: New Test Location Confirm

References

- JSF2: <http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>
- EclipseLink: <https://www.eclipse.org/eclipselink/>
- Jboss: <http://www.jboss.org/>
- Maven: <http://maven.apache.org/>
- Entity Framework: <http://msdn.microsoft.com/en-us/data/ef.aspx>
- Visual Studio 2013 Express: <http://www.visualstudio.com/downloads/download-visual-studio-vs>
- Project github repo: <https://github.com/stefanprisca/FacultDistributedSystems>