

Track Management

A Web Application using JAVA J2EE Framework

Submitted by
Manish Shaw
Roll No.- 15/C.A./601
Registration No.- 20158378

Under Supervision of:
Dr. Anirban Sarkar



DEPARTMENT OF COMPUTER APPLICATIONS

NATIONAL INSTITUTE OF TECHNOLOGY DURGAPUR

December 2017

Acknowledgements

I am pleased to acknowledge Dr. Anirban Sarkar for his invaluable guidance during the course of this project work.

I would also like to thank 'Mkyong' (<http://www.mkyong.com>) for writing the very useful RESTful Java client with Jersey client example under the Web Service banner which greatly helped me in writing the visualization part.

December 2017

Manish Shaw
Roll No.- 15/C.A./601
Registration No.- 20158378

Contents

ACKNOWLEDGEMENTS1

CONTENTS2

INTRODUCTION3

 OVERVIEW3

 BACKGROUND AND MOTIVATION3

 METHODOLOGY4

TOOL DESCRIPTION5

 USER INTERFACE5

 FEATURES5

 VISUALIZATION 6

PROJECT DETAILS 7

 REQUISITES 7

 IMPLEMENTATION 9

 SCREENSHOTS12

FUTURE WORK14

BIBLIOGRAPHY14

Introduction

Overview

This report discusses the result of the work done in development of "Track Management - A Web Application using JAVA J2EE Framework" on Java Platform. It is a part of the academic project going in Department of Computer Application, NIT Durgapur and aims at the development of an application based on J2EE framework for providing a common platform for facilitating the use of methodological approach during the execution of the project.

Background and Motivation

The volume of data transacted by devices grows exponentially. In several vertical markets there are more and more devices that communicate with each other to accomplish tasks. The amount of data generated by these devices is huge and there is a need to develop new M2M communications platforms flexible enough to accommodate a wide range of markets, each with its own specificities. The use of open technologies and interoperable distributed platforms is then a necessity. The present work aims to contribute to the development of a platform for data communications.

Objective

The final goal of the project was two-fold.

1. A Web Service was required for interaction with the various services (like request, respond, manipulate, etc.) with the platform specification being done in the application itself.
2. Based on the final platform configuration and bindings, an Analysis and Visualization framework was required for getting performance metrics of the system and for visualization of the analysis results and the target platform.

Along with above main goals, capability to design the target platform manually was also desired.

Methodology

To implement the above goals, the following methodology needs to be followed:

1. Specifying the Application and various components of the Architecture.
2. Specifying the bindings between the tasks and the resources either manually or by the design tools.
3. Specifying the port interconnections between the resources.
4. Analysis: Extracting the data required for analysis and the doing the analysis.

Tool Description

User Interface

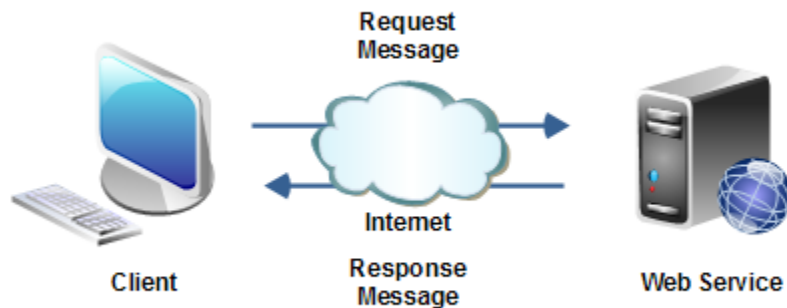
The tool is very user friendly and intuitive and uses a GUI interface implemented in JAVA and Web Tools to communicate with the user. Various features are self – explanatory.

Forms are easy to fill in and components can be added, removed and updated very easily through a single dialog box.

Features

1. Intuitive interface.
2. Clean separation of various components to facilitate easy modification and revision.
3. All the configuration data is maintained in a separate file to facilitate easy modification.

Visualization



Web browsers and web servers function together as a client-server system.

In computer networking, client-server is a standard method for designing applications where data is kept in central locations (server computers) and efficiently shared with any number of other computers (the clients) on request. All web browsers function as clients that request information from websites (servers).

Numerous web browser clients can request data from the same website. Requests can happen at all different times or simultaneously. Client-server systems conceptually call for all requests to the same site to be handled by one server. In practice, however, because the volume of requests to web servers can sometimes grow very large, web servers are often built as a distributed pool of multiple server computers.

For very large websites popular in different countries around the world, this web server pool is geographically distributed to help improve the response time to browsers. If the server is closer to the requesting device, it would follow that the time it takes to deliver the content is faster than if the server were further away.

Project Details

Requisites

Tomcat v7.0 Server

Apache Tomcat, is an open source web server and servlet container developed by the Apache Software Foundation. Basically, it implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run in. In the simplest configuration Tomcat runs in a single operating system process. The process runs a Java virtual machine (JVM) and every single HTTP request from a browser to Tomcat is processed in the Tomcat process in a separate thread.

Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

XAMPP for MySQL

XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P).

It is a free and open source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing and deployment purposes. Everything needed to set up a web server – server application (Apache), database (MariaDB), and scripting language (PHP) – is included in an extractable file.

Google Chrome

The Google Chrome Web browser is based on the open source Chromium project. Google released Chrome in 2008 and issues several updates a year. It is available for Windows, Mac OS X, Linux, Android and iOS operating systems. The Google Chrome browser takes a sandboxing-based approach to Web security.

Eclipse IDE

It is an integrated development environment and It is written mostly in Java so as expected it is mostly used for JAVA SE and JAVA EE applications but it may also be used to develop applications in other programming languages via plug-ins, including: Ada, C, C++, Python, Ruby (including Ruby on Rails framework), etc. The Eclipse software development kit (SDK), which includes the Java development tools initially.

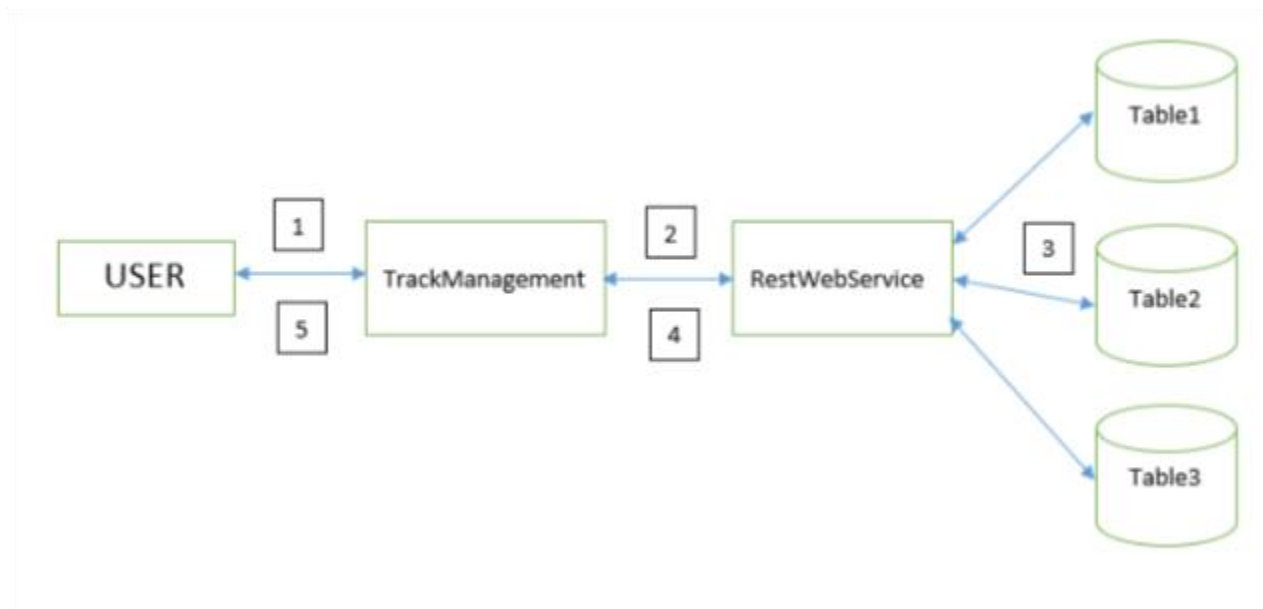
Implementation

1) Inserting Track – This is generally for storing the data into the tables. We are using three tables Table1, Table2 and Table3. The user has the freedom to store Title and Singer in any of the three tables. Using HTML web page, we have taken Title, Singer and the name of the table in which the user want to store his/her data. On getting post response, our TrackManagement Web Service is called. It then requests the RestWebService Web Service to store Title and Singer to the respective table as per said by the user. After the values are stored, a successful message is passed to TrackManagement which then shows the successful message in the browser.



- 1 – User gives Singer and Title.
- 2 – TrackManagement sends Singer and Title values to RestWebService.
- 3 – RestWebService stores the values in respective database.
- 4 – Sends the success message to TrackManagement.
- 5 – TrackManagement shows success message to user.

2) Searching Tracks – We can also implement searching operations. Here, if we give a query, it will be searched in all the tables. If the data is present, it will be displayed in a table. Or else, 'No result to display' will be showed. In FindTracks.jsp, we first select from the drop-down list that what we will provide, Singer or Title. After specifying, we provide the Singer/Title with the values to be searched in the database. This will then be carried forward by TrackManagement Web Service. It then requests the RestWebService Web Service to search for that query in all the tables. If the value is matched with the existing entries, the values are then passed to TrackManagement Web Service. Then TrackManagement displays the Title and Singer just below that page. If the searched key is not present, then 'No result to display' will be showed.



1 – User gives Singer/Title.

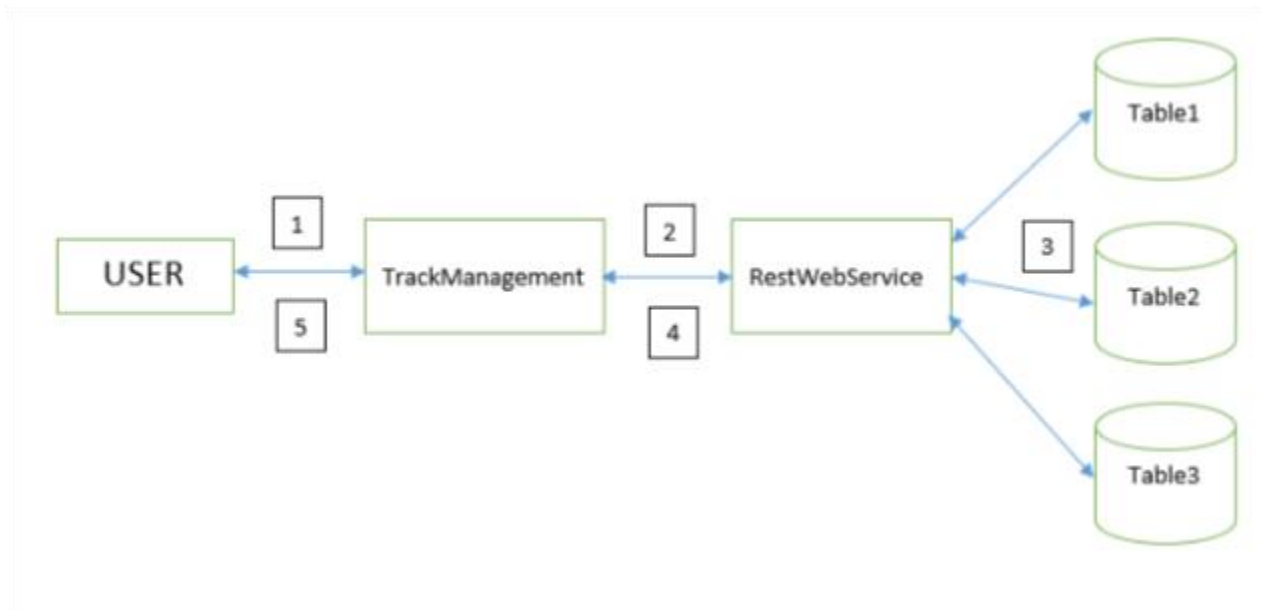
2 – TrackManagement sends Singer/Title value to RestWebService to search in the tables.

3 – RestWebService searches the value in all the tables.

4 – On getting the result, it is stored in a list and sent to TrackManagement.

5 – TrackManagement displays the result to user.

3) View All Tracks – This generally shows the union of all values in the database. In this, the TrackManagement Web Service requests for all the values to RestWebService Web Service. It then, makes a list of values and returns to the TrackManagement. Which in turn displays the result in ViewTracks.jsp.



- 1 – User request to show all the entries.
 - 2 – TrackManagement sends the request to RestWebService to give entries of all the tables.
 - 3 – RestWebService collects the values from all the tables.
 - 4 – On getting the result, it is stored in a list and sent to TrackManagement.
 - 5 – TrackManagement displays the result to user.
-

Screenshots

Home Page – index.jsp



Select an operation

[Insert track](#)

[Search Track](#)

[View all Tracks](#)

Insert Track – CreateTrack.jsp



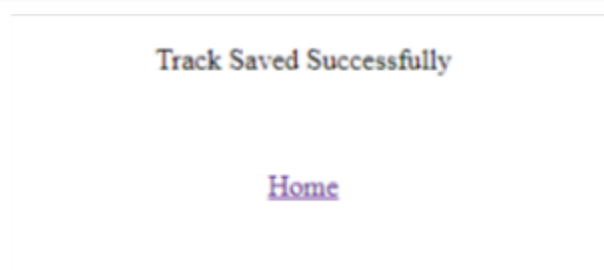
Track Details

Title :

Singer :

Database :

Successful Page – Success.jsp



Track Saved Successfully

[Home](#)

Search an Entry

Choose :

Title ▼

Title/Singer :

Baarish

Search

Back

Title	Singer
Baarish	Arijit

View Track Details	
Title	Singer
hello	hi
hello	hi
asd	123
manish	shaw
Nitish	Nitish
manish	Manish
sampletile	samplesinger
insert	insert
Baarish	Arijit
Hello	say
Home	

Future Work

- Introducing Security measures.
- Making the project based on real time.
- Working on increased overhead.
- Introducing Distributed Database.

Bibliography

- <http://www.mkyong.com/webservices/jax-rs/restful-java-client-with-jersey-client/>
- <http://www.tutorialspoint.com/listtutorials/java/j2ee/1>
- <https://stackoverflow.com/>