Table of Contents

[CurrencyFair TradeMessage API Project Documentation 1](#_Toc509787382)

[Reference Specification: 1](#_Toc509787383)

[Project Information: 1](#_Toc509787384)

[Reasons for Implementing with given technology 1](#_Toc509787385)

[Jersey 1](#_Toc509787386)

[Hibernate 1](#_Toc509787387)

[JSP-JavaScript-JQuery-Ajax 1](#_Toc509787388)

[Needed to Use: 2](#_Toc509787389)

[Running operations: 3](#_Toc509787390)

[API: 5](#_Toc509787391)

# CurrencyFair TradeMessage API Project Documentation

## Reference Specification:

[https://docs.google.com/document/d/1st5i5OpravXYhXBZry8qZwP-ish-ZfQQcSOBduKf73k/edit#](https://docs.google.com/document/d/1st5i5OpravXYhXBZry8qZwP-ish-ZfQQcSOBduKf73k/edit)

<http://hibernate.org/>

<https://jersey.github.io/>

<https://www.mysql.com/>

## Project Information:

This project is a Greenfields project using the following technology:

JavaEE, Jax-rs(Jersey), Tomcat 8, Hibernate, MySQL, JSP, JavaScript, JQuery, AJax.

## Reasons for Implementing with given technology

### Jersey

Jersey is almost and industry standard and has a wide range of features and support. The implementation allows for your application to scale easily and takes out lots of the overhead.

### Hibernate

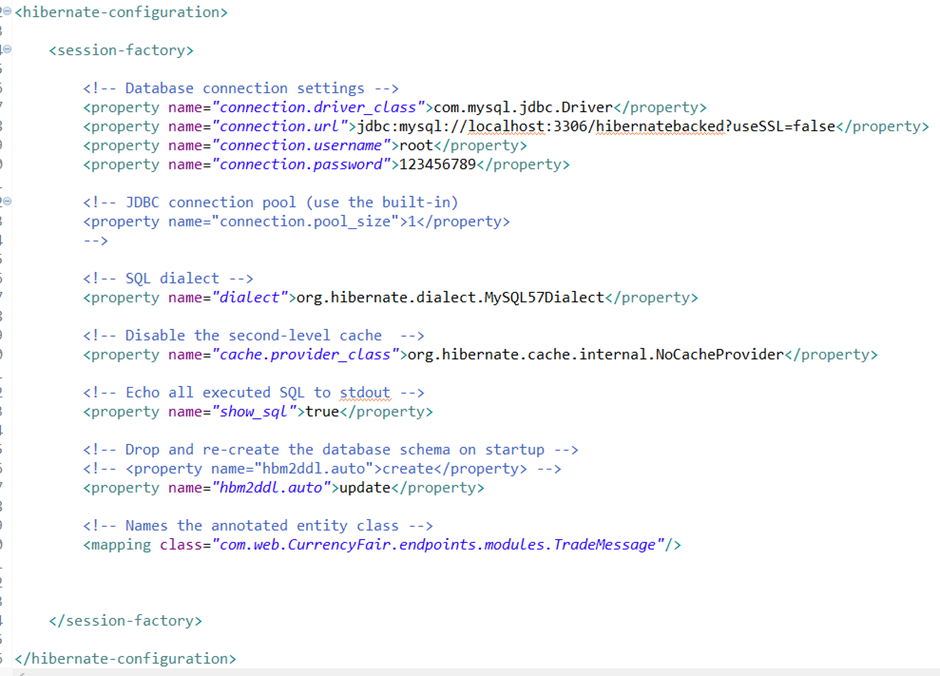
Hibernate is one of the mostly widely used interfaces for DB’s and has the ability to be configured to work with nearly any DB seamlessly. There is also a large focus of Caching and optimised requests to the database. Once a request is made, the data is stored in memory, vastly increasing the performance of the queries. This is particularly useful when instances are called repeatedly

### JSP-JavaScript-JQuery-Ajax

This makes the application implementation portable to a wide audience. All you need is a Browser to access, and can implement HTML charts, Reports easily. <https://canvasjs.com/>

## Needed to Use:

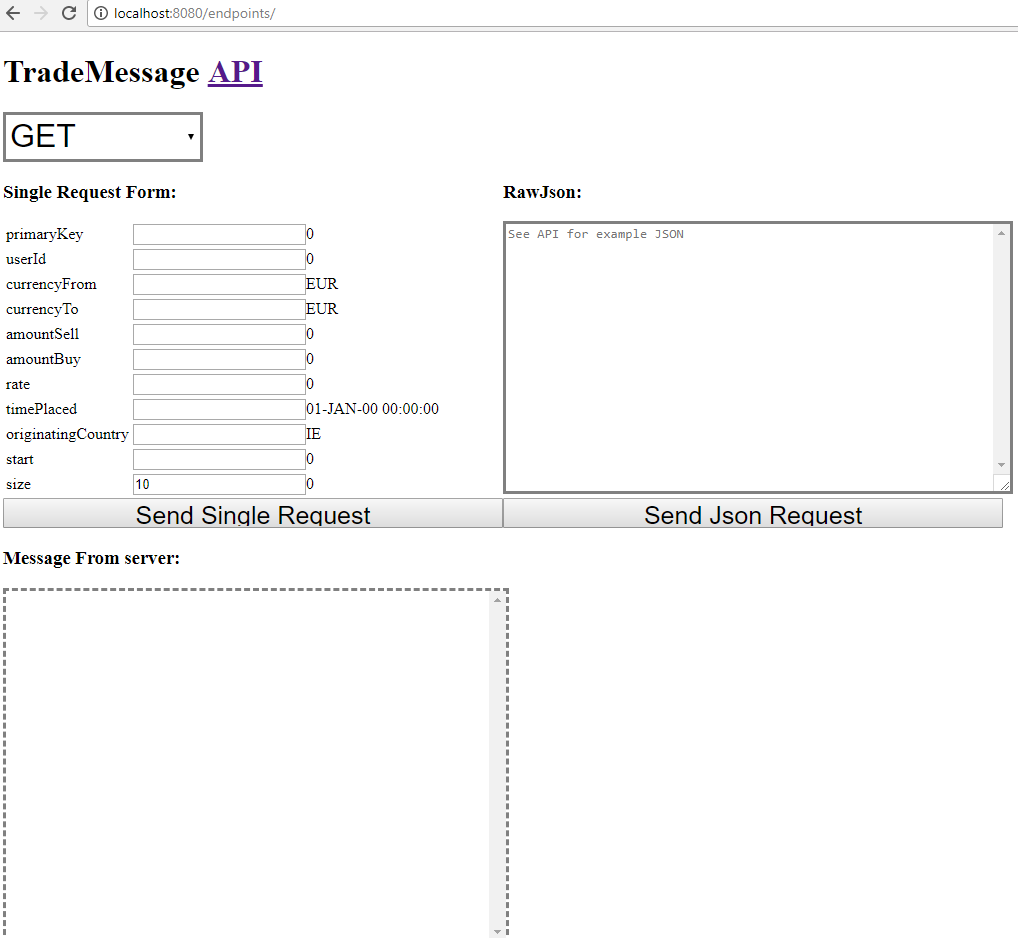
You will need to configure the database connection to work with the DB. For my project I chose MySQL, however Hibernate is compatible with most DB’s mealy need to be configured here with credentials and URL connection. See sample image for details. Note you will need to download the connector jar and add the dependency in Maven or add the Jar to the build path.



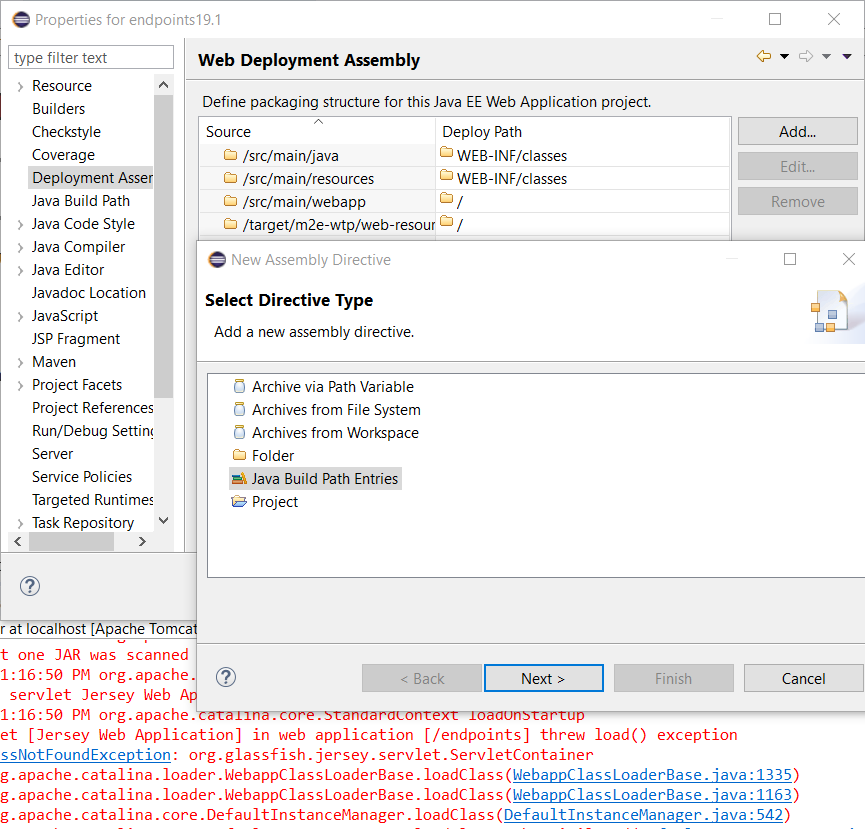
## Running operations:

[http://localhost:8080/endpoints/](http://localhost:8080/endpoints/webapi/jsonTradeEndpoint)

This is the default index.jsp page ran when the application starts. The UI is a barebones implementation for the purpose of interacting with the DB.



When you start the project, you may get this for the fist time. Just go to Deployment assembly and add maven in build path so it’s available when apache runs and can find the servlet.



## API:

This is the API documintaion for http://localhost:8080/endpoints/webapi/jsonTradeEndpoint/

This is a list of the API URI's:

\*Please ensuer all / json body message have surounding [] annotation

/ CRUD opperations consume and produce Content-Type = application/json in format [TradeMessage]

/{primaryKey} RD will do the respective opperation of the instance in the DB

/create/{number} Creates $ records on the DB of random values in EUR market

filterParmiters:

primaryKey=0&start=0&size=0&userId=0&year=0&originatingCountry=0&rate=0&currencyFrom=0&currencyTo=0&amountSell=0&amountBuy=0&timePlaced=0

Sample TradeMessage:

[

{

"amountBuy": 7976,

"amountSell": 5140,

"currencyFrom": "EUR",

"currencyTo": "EUR",

"originatingCountry": "IE",

"primaryKey": 1,

"rate": 52.462395688840104,

"timePlaced": "11-NOV-05 09:16:48",

"userId": 93522

}

]