Vaadin

Dependencies:

• Java JDK 8
• Latest Eclipse Neon Vaadin plug-in installed
• Vaadin 8
• Tomcat Server 9

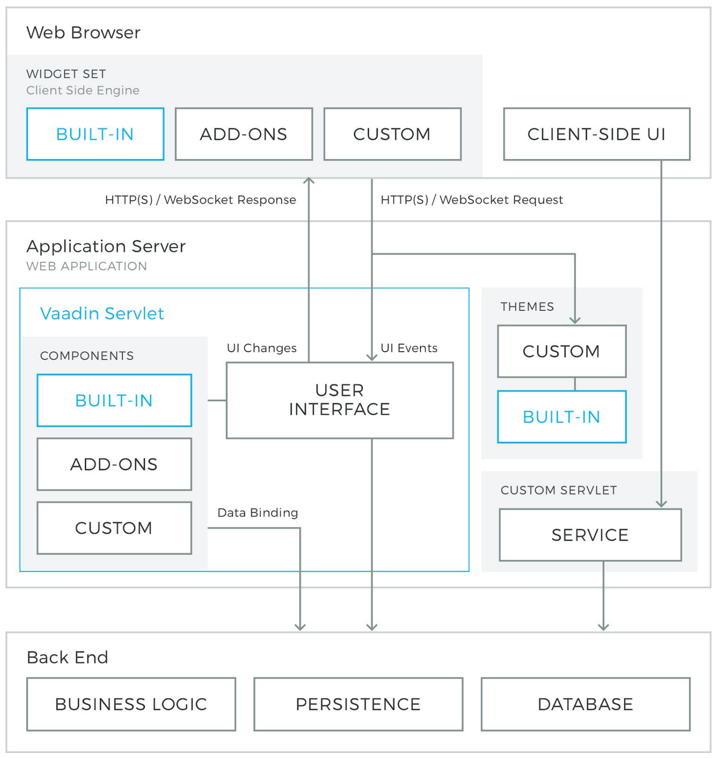
What is it ?

Vaadin Framework 7 is a server side web framework that uses Ajax to keep the UI synchronized with the server. Usually on each user interaction, Vaadin makes one or more Ajax calls against the server to "keep the server informed" whats going on with the UI.

Get started:

From the project selection menu, select Project -> Vaadin 8 Projects
Uses Maven for dependency management.

Architecture:



3 Layered Architecture

• Client widgets
• Web Server layer
• Persistence layer

Vaadin Framework consists of a server-side API, a client-side API, a horde of user interface components/widgets on the both sides, themes for controlling the appearance, and a data model that allows binding the server-side components directly to data. For client-side development, it includes the Vaadin Compiler, which allows compiling Java to JavaScript.

A server-side Vaadin application runs as a servlet in a Java web server, serving HTTP requests. The VaadinServlet is normally used as the servlet class. The servlet receives client requests and interprets them as events for a particular user session. Events are associated with user interface components and delivered to the event listeners defined in the application. If the UI logic makes changes to the server-side user interface components, the servlet renders them in the web browser by generating a response. The client-side engine running in the browser receives the responses and uses them to make any necessary changes to the page in the browser.

Code:

MyUI class is the entry point of your application. All objects handling UI should return here to be displayed.
A UI represents an HTML fragment in which a Vaadin application runs in a web page. It typically fills the entire page, but can also be just a part of a page. You normally develop an application with Vaadin Framework by extending the UI class and adding content to it. A UI is essentially a viewport connected to a user session of an application, and you can have many such views, especially in a multi-window application. Normally, when the user opens a new page with the URL of the UI, a new UI (and the associated Page object) is automatically created for it. All of them share the same user session.

Layouts:

The Layout is based on two main classes Vertical Layout and Horizontal Layout. All components in HL will be put horizontally after each other, whereas components will be put vertically for VL. [https://vaadin.com/docs/v8/framework/layout/layout-orderedlayout.html]

Components:

Label, Buttons, Text Field etc, are called Components. To make a UI you put components in a layout, and set the content. Different types of Component:
https://vaadin.com/docs/v8/framework/components/components-overview.html

Themes:

Themes can be defined separately, enabling to develop and design independently. Ref:
[https://vaadin.com/docs/v8/framework/themes/themes-overview.html]

Event Handling:

To handle Events, it is necessary to bind event listeners to components. Event Listeners take a lambda function which will be call when the event is triggered. Ref: [https://vaadin.com/docs/v8/framework/datamodel/datamodel-overview.html]

Building and Deploying:

Since Vaadin uses Maven as dependency manager, building is as easy as doing a "maven install". This will make the .war file. The war file can be deployed to servers. For local testing use a Apache Tomcat localhost.
Use link [https://help.eclipse.org/neon/topic/org.eclipse.stardust.docs.wst/html/wst-integration/configuration.html] to create a server config in Eclipse.
Then run the application as "Run As" -> "Run on Server" and the application will be deployed o local host.