# Vaadin

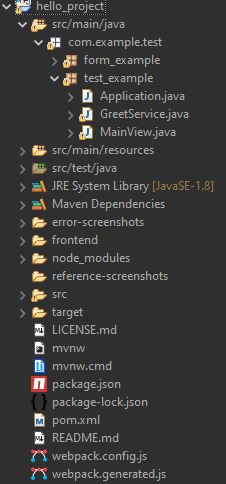
## Was ist Vaadin?

„Vaadin ist ein […] Open-Source-Framework, das die einfache Erstellung auch komplexer browserbasierter Applikationen ermöglicht. Vaadin abstrahiert hierbei für die normale Programmierung von HTML, JavaScript, AJAX und browserspezifschen Details und bietet ein javabasiertes Programmiermodell mit Widgets und Events an, das sehr stark an klassiche UI-Programmierung in Java angelehnt ist.“

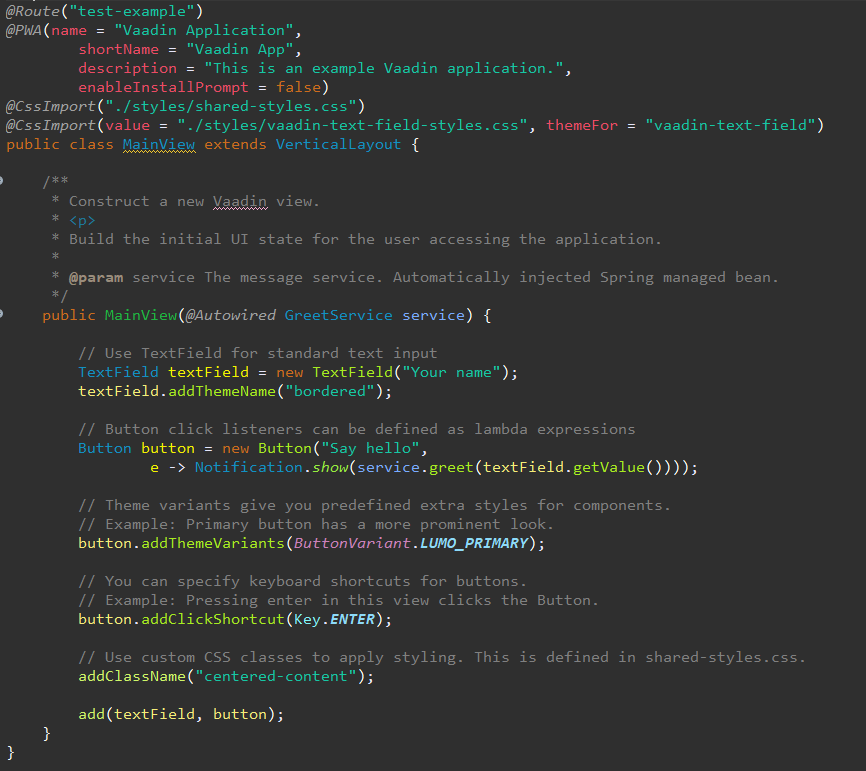
*Baumann, J., Arndt, D., Engelen, F., Hardy, F. & Mjartan, C. (2014). Vaadin: Der kompakte Einstieg für Java-Entwickler (1. Aufl.). dpunkt.verlag GmbH.*

## Mit Vaadin arbeiten

Um in Eclipse ein neues Vaadin-Projekt anzulegen geht man im Workspace auf New/Other und wählt dann im Wizard ein neues Vaadin-Projekt aus. Vaadin hat mehrere unterschiedliche Templates von denen man für seine Anwendung auswählen kann. Hat man sich für eines entschieden erstellt Eclipse automatisch ein neues Projekt mit allen nötigen Ordnern und Dateien.



Am wichtigsten für die Arbeit an dem Projekt ist der src/main/java-Ordner der alle Java-Klassen enthält. In diesen findet man bei allen erstellten Projekten immer eine *Application*.java und *MainView*.java. Die *Application*.*java* ist die Datei die die *main*-Methode enthält und immer ausgeführt werden soll, da sie die komplette Anwendung startet. Die *MainView.java* enthält den Quellcode für die Seite die beim ersten Starten der Anwendung am Anfang angezeigt wird. Über Annotations kann man der Seite eine bestimmte Route, also die URL unter der sie angezeigt wird, angeben, sowie ebenfalls entsprechende CSS-Dateien einbinden. Über die verschiedenen Komponenten die in den entsprechenden Bibliotheken von Java zu finden sind, kann man sich Seiten mit verschiedensten Komponenten und Design erstellen und so seine Anwendung zubauen.



Wie im obigen Screenshot zu sehen ist, arbeitet Vaadin in seinen Klassen mit Annotations die verwendet werden um beispielsweise die Route einer Klasse oder CSS-Importe zu deklarieren. Im Konstruktor der *MainView*-Klasse werden dann ein Text-Feld und ein Knopf initialisiert. Der Button wird dann noch erweitert in dem man ihm ein bestimmtes Design hinzufügt und ihm die Funktionalität gibt, per Enter-Taste gedrückt zu werden. Fügt man diese Komponenten am Ende der *MainView-*Klasse hinzu, werden diese in der Browserseite angezeigt.

Beim Starten der Anwendung wird automatisch ein Tomcat-Server hochgefahren auf dem die erstellten Seiten laufen. Unter src/main/resources findet man die application.properties-Datei mit der man Einstellungen am Server vornehmen kann. Ist der Server gestartet wird automatisch ein Browser-Fenster geöffnet das auf die angegebene URL zeigt. Beim Erstellen einer Anwendung lohnt es sich sehr in der offiziellen Dokumentation von Vaadin zu suchen (<https://vaadin.com/docs/latest/components>). Hier gibt es beispielsweise für die meisten UI-Komponenten Beispiele mit Code die man direkt kopieren und in die eigenen Klassen integrieren kann. Der folgende Screenshot ist ein Beisepiel für eine mit Vaadin geschriebene Anwendung mit der man eine CSV-Datei einlesen und den Inhalt in einer Tabelle anzeigen lassen kann.

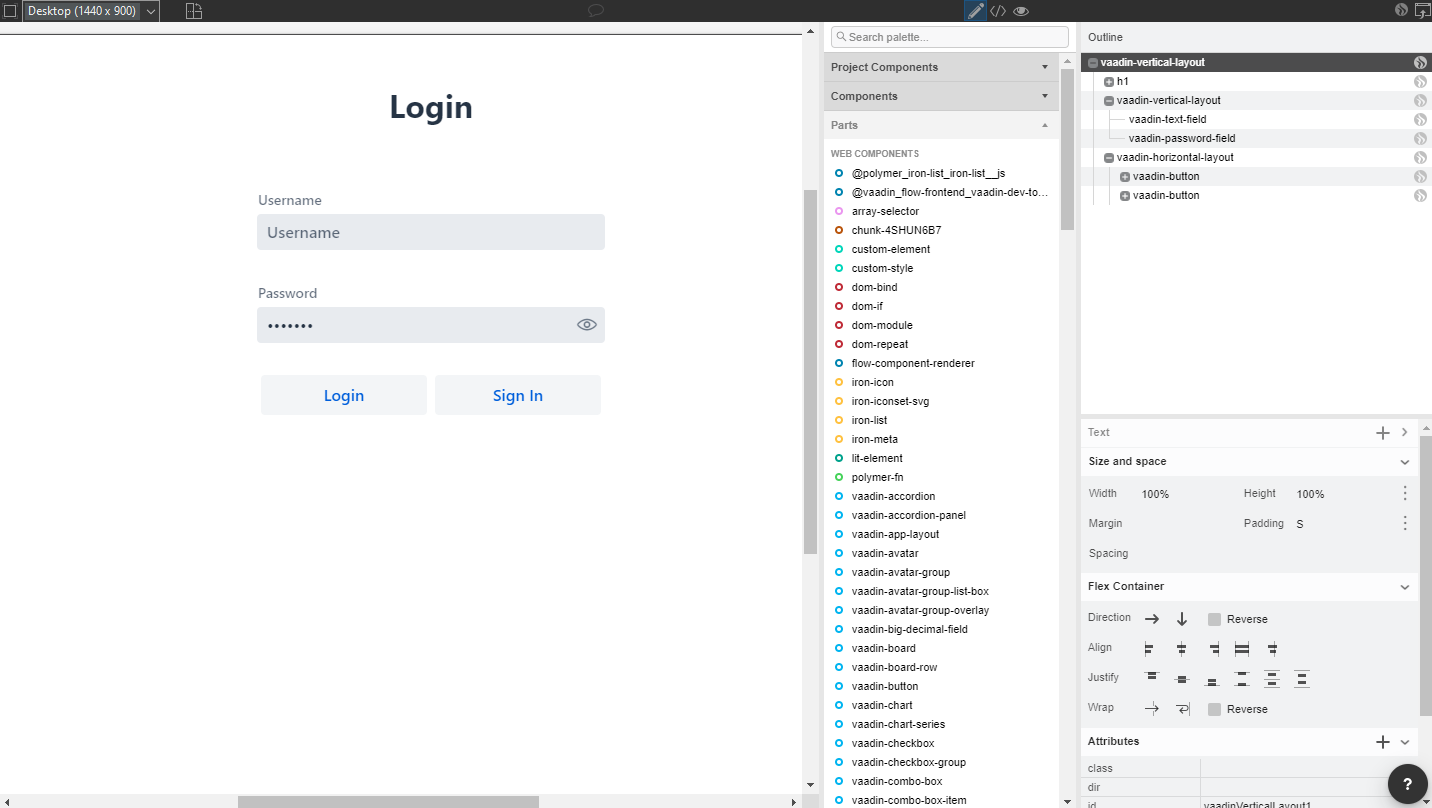


## Vaadin-Plattformen

Vaadin bietet als Framework neben den eigentlichen Grundbausteinen zur Erstellung von Web-Applikationen auch weitere Plattformen an, die den Entwicklungsprozess beschleunigen und angenehmer machen. Im Folgenden daher eine Zusammenfassung der drei wichtigsten Plattformen für Entwickler.

### Vaadin Designer

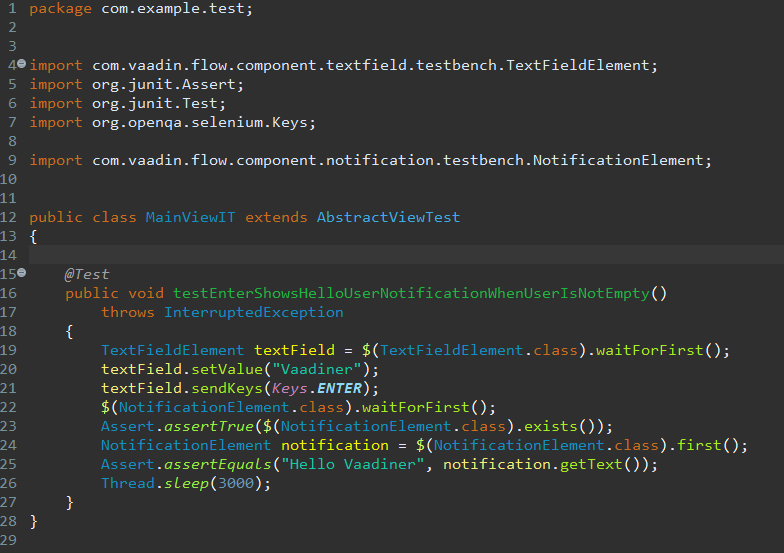
Dieser ermöglicht es per Drag & Drop die einzelnen UI Komponenten von Vaadin in ein Fenster zu ziehen und so das Erstellen des passenden Designs zu erleichtern.



Der Designer ist in drei Abschnitte unterteilt. Im linken Fenster wird das Design der HTML-Seite angezeigt. Hier kann man verschiedene Auflösungen für Smartphones, Tablets oder Desktops auswählen. Im mittleren Abschnitt kann man nach den gewünschten Komponenten suchen und diese in das linke Design-Fenster schieben. Im rechten Fenster wird die hierarchische Struktur der Komponenten angezeigt und es können einzelne Komponenten ausgewählt und angepasst werden. Hierbei kann man jede einzelne Komponente mit klassischen CSS-Attributen wie *Width*, *Height*, *Margin* oder *Padding* anpassen. Bei den Komponenten die in der fertigen Seite vorkommen sollen, ist es wichtig das ID-Attribut anzugeben, damit Vaadin weiß das diese Komponente ein Teil der Java-Klasse sein soll. Möchte man für sein Design nun eine entsprechende Java-Klasse erstellen, die angezeigten Seite entspricht, nutzt man den kleinen Button oben rechts im Screenshot der automatisch eine Java-Klasse erzeugt. Diese kann man dann in seiner Anwendung verwenden.

### TestBench

TestBench ist ein Werkzeug welches auf das Test-Framework Selenium aufbaut und genutzt werden kann um Integration Tests zu erstellen. Der Vorteil gegenüber Selenium ist, dass die Selektion und Manipulation von UI Komponenten erheblich erleichtert wird und so Testklassen deutlich leichter zu schreiben sind.



Um UI-Tests mithilfe von TestBench zuschreiben wird die Hilfsklasse AbstractViewTest erstellt, von der die Testklasse erben muss. Diese sorgt dafür das automatisch die Testumgebung erzeugt wird. Bezüglich der Testfälle verhält sich TestBench ähnlich wie Selenium, da man ebenfalls mit Annotations wie Before, After und Test arbeitet. In den einzelnen Methoden können UI-Komponenten mithilfe des *$-*Operator anhand ihrer Klasse selektiert wird. Dabei verhält sich TestBench ähnlich wie JQuery da man verschiedene Selektoren aneinanderreihen kann, umso einzelne Komponenten gezielt auszuwählen. TestBench bietet für diese Elemente dann Methoden wie *setValue* oder *sendKeys* um Nutzereingaben simulieren zu können. Über die *Assert*-Methoden kann man dann die entsprechenden Ausgaben wieder überprüfen.

### Design System Publisher

Diese Plattform kann genutzt werden um individuelle UI-Komponenten und entsprechende Dokumentation an einem Ort zu speichern und abrufen zu können.