# Werkzeuge zur Entwicklung von Client/Server Applikationen

## Client

Auf dem Client muss eine passende Clientapplikation laufen. Im Normalfall ist das einfach ein Webbrowser welcher HTML-Seiten anzeigt. Auch denkbar wäre ein Rich-Client welcher sich die passenden Daten beim Server abholt, sie anzeigt und die Userinteraktionen als Befehle an den Server absetzt. Die Art des Clients bestimmt somit die Funktionalität welche der Controller der Serverapplikation anzubieten hat.

## Server

Der Server muss in der Lage sein, die von uns programmierte Applikation laufen zu lassen. Je nach der Art oder den Anforderungen der Applikation müssen spezielle Server verwendet werden. Ggf. muss der Server um Funktionalität (die benötigten Bibliotheken) ergänzt werden.

Der Server muss so konfiguriert sein, dass die Applikation ihre Daten (die DB) findet.

Gegebenenfalls müssen im Server Sicherheitseinstellungen für unsere Applikation konfiguriert werden.

## Datenbank

Mindestens ein laufender Datenbankserver muss vorhanden sein. In diesem muss zumindest eine Datenbank definiert sein. Allfällige Zugangsdaten müssen im Server konfiguriert werden.

## Entwicklungsumgebung

Die Entwicklungsumgebung muss in der Lage sein, ein Programm so zu erstellen, dass es innerhalb eines Servers ausführbar ist.

Weiter muss sie wissen, welchen Server gestartet werden muss. Die ausführbare Datei muss im Server installiert werden (deployement). Anschliessend muss die Entwicklungsumgebung den Server instruieren, unseren Code laufen zu lassen. Selbstverständlich muss zu diesem Zeitpunkt der Datenbankserver gestartet sein.

Idealerweise wird die DB bei jedem Neustart der Applikation in seinen Ursprungszustand versetzt, so dass wir einfacher Testen können.

Weiter wäre es ideal, wenn wir auf dem laufenden Server debuggen könnten.

Obwohl die Erstellung des Clientoutputs nicht Teil der Serverprogrammierung ist, sollte uns die Entwicklungsumgebung bei diesem Vorgang unterstützen, und sei es nur, damit wir sehen können, ob wir die richtigen Daten ausliefern.

# DIY oder ein Framework

Wir können bei der Entwicklung unserer Applikation auf ein Framework verzichten und unsere Applikation von Hand mittels der vorhandenen Java-Bibliotheken bauen. Alternativ können wir ein Framework verwenden.

## Do it yourself

Das Selberschreiben einer Serverapplikation in Java ist nicht schwierig. Die vorhandenen Bibliotheken sind in ihrer Funktionalität ausreichend und gut verständlich. Servlets bieten alles, was wir im Zusammenhang mit einem Server brauchen. Unangenehm ist, dass wir sehr viel Code schreiben müssen, welcher nicht direkt mit unserem Problem zu tun hat, wie Beispielsweise das Parsen von Parametern bei einer Anfrage eines Clients und das Bestimmen der Funktion, welche anschliessen laufen muss. Ein weiteres grosses Problem ist die Fehlerbehandlung.

Der Vorteil dieser Lösung ist eher pädagogischer Natur. Wir verstehen wie eine Serverapplikation grundsätzlich aufgebaut ist.

Der Nachteil ist, dass wir viel fehleranfälligen Code schreiben welcher wenig zur eigentlichen Funktionalität beiträgt.

Schon aus Zeitgründen fällt diese Lösung weg.

## Ein Framework benutzen

Ein Framework unterscheiden sich von einer Bibliothek dadurch, dass wir nicht selber sagen müssen wie etwas funktioniert, wir ergänzen das vorhandene Programm (das Framework) nur noch durch diejenigen Funktionalitäten, welche für unser Programm typisch sind.

Der Vorteil eines Frameworks ist es, dass wir bereits beim Start der Entwicklung ein laufendes Programm haben, welches allerdings noch nichts macht.

Der Nachteil eines Frameworks ist es, dass wir verstehen müssen wie es benutzt werden muss, und wo und wie wir es mit unserer Funktionalität ergänzen müssen. Weiter kann das Finden von Fehlern innerhalb eines Frameworks richtig schwierig sein. Das Nichteinhalten von Framework Konventionen oder falsche Konfigurationen im Zusammenhang mit dem Server oder der DB führen oft zu Fehlern, welche mit unserem Code absolut nichts zu tun haben aber doch irgendwie mit unserem Code zu tun haben.

# Frameworks

## Java Plattform, Enterprise Edition (JEE)

Auszug aus Wikipedia (https://de.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition)

> *Java Plattform, Enterprise Edition, abgekürzt Java EE oder früher J2EE, ist die Spezifikation einer Softwarearchitektur für die transaktionsbasierte Ausführung von in Java programmierten Anwendungen und insbesondere Web-Anwendungen.*
>
> *…*
>
> *Bestandteile der Spezifikation werden innerhalb des Java Community Process von diversen Unternehmen erarbeitet und schließlich der Öffentlichkeit in Form eines Dokuments und einer Referenzimplementierung zur Verfügung gestellt.*

Dieser Auszug beschreibt JEE relativ gut. JEE ist ein ‚Quasistandart' welcher von einem Industrie-Konsortium erstellt wurde. Wie bei allen «Standards» ist seine Weiterentwicklung ein langwieriger Prozess. Das ist einerseits gut, da sich nicht ständig etwas Ändert, andererseits schlecht, da neue Erkenntnisse und Arbeitsverfahren erst nach und nach in die Definition aufgenommen werden.

Beachten Sie auch die Aussage «von diversen Unternehmen erarbeitet». Unternehmen arbeiten nicht gratis. JEE braucht einen speziellen Server und dieser soll dann das investierte Geld zurückspielen.

Nichts desto trotz ist JEE ein gutes Framework, zumal «Referenzimplementierungen» und ein «public domain» Server zur Verfügung stehen.

### Herausragende Eigenschaften von JEE

JEE ist eigentlich nur eine Ansammlung von Vorschriften (Interfaces). Implementierungen (und Support) von beliebig guter Qualität können kommerziell erstanden werden. Dies macht JEE vor allem im kommerziellen Umfeld zum bevorzugten Framework.

## Spring Framework

Auszug aus Wikipedia ( https://de.wikipedia.org/wiki/Spring_(Framework) )

> *Das Spring Framework (kurz Spring) ist ein quelloffenes Framework für die Java-Plattform. Ziel des Spring Frameworks ist es, die Entwicklung mit Java/Java EE zu vereinfachen und gute Programmierpraktiken zu fördern. Spring bietet mit einem breiten Spektrum an Funktionalität eine ganzheitliche Lösung zur Entwicklung von Anwendungen und deren Geschäftslogiken; dabei steht die Entkopplung der Applikationskomponenten im Vordergrund.*

Nicht viel Neues hier. Im Unterschied zu JEE ist Spring aus den gemachten Erfahrungen bei der Implementierung von Serveranwendungen entstanden. Spring wird nicht von einem Konsortium, sondern von den Mitgliedern der Open-Source-Community weiterentwickelt.

Das ist gut, weil sich neue Ideen schneller durchsetzen, schlecht, weil man nie weiss ob das Neue noch kompatibel ist zum Alten.

Spring existiert schon lange und ist mittlerweile hinreichend stabil. Die Weiterentwicklung von Spring findet heute i.A. durch das Hinzufügen von neuen Komponenten statt. Spring stellt geringe Anforderungen an den Webserver und läuft auf nahezu jedem Server.

### Herausragende Eigenschaften von Spring

Spring basierte auf der Idee von «Dependency Injection» und ermöglichte schon immer die Programmierung mittels POJOs (Plan old Java objects). JEE kann das mittlerweile auch. Beide Frameworks unterstützen die Aspekt orientierte Programmierung.

# Entwicklungsumgebung (IDE)

Entwicklungsumgebungen sind Geschmackssache. Je nach verwendetem Framework ist die eine oder andere etwas besser geeignet bezüglich der Integration mit dem Server und dem Debugging. Dank Erweiterungsmodulen lässt sich aber jede Entwicklungsumgebung mit jedem Framework brauchen.

## Entwicklungsanforderungen

- Als Applikationsentwickler wollen wir möglichst nichts mit der Serverkonfiguration und dem Deployment zu tun haben. XML-Konfigurationsdateien sind ein Gräuel. Falls Konfigurationen nötig sind, gehören sie in den Code.
- Datenbankserver sollen einfach laufen, die passenden DBs sollen existieren und die nötigen Tabellen vorhanden sein.
- Debugging auf dem laufenden Server muss möglich sein, auch ohne Konfiguration der IDE oder des Servers.
- «Instant Reload» muss möglich sein. Das bedeutet, dass Änderungen im Code augenblicklich auf den Server gespiegelt werden, ohne weitere Interaktionen unsererseits. Dasselbe sollte (soweit möglich) für Codeänderungen während einer Debugging-Session gelten.
- Last but not least soll das Erstellen eines neuen Projektes schnell und Schmerzlos sein. Falls wir neue Komponenten brauchen, sollte das Auflösen der Abhängigkeiten und das Nachladen der benötigten Bibliotheken automatisch passieren.

Die oben gestellten Anforderungen sind umfangreich und schwer zu erfüllen. Wenn wir das schaffen, fühlt sich das an wie Ostern und Weinachten gleichzeitig ☺

## Eclipse, Spring Boot & Maven

Nach sorgfältiger Evaluation der obenstehenden Kriterien, habe ich folgende Wahl für die Entwicklungsumgebung getroffen:

- Ecilpse als IDE, einfach, weil Sie Eclipse schon kennen und wir alle benötigten Erweiterungen nachinstallieren können.
- Maven als Build- und Dependency-Resolution tool. Gradle wäre genauso gut gewesen, macht aber in Zusammenhang mit Eclipse mehr Ärger.
- Spring Boot als Framework und Rapid-Prototyping tool.

Die Wahl von Spring Boot muss noch etwas genauer erklärt werden. Spring boot ist ein Werkzeug das ursprünglich für die Erstellung von Microservices gedacht war. Es erstellt eine .jar-Datei, welche bereits einen Server, eine Datenbank und die erstellte Applikation enthält. Diese kann dann direkt ausgeführt werden und stellt einen lauffähigen (Web)-Server dar. Die Grundsatzphilosophie von Spring Boot ist es, dass alle nötigen Konfigurationen direkt im Code mittels Annotation vorgenommen werden. Für die Implementierung der eigentlichen Applikation integriert Spring Boot das Spring Framework. Die Wahl des Spring Frameworks als solches anstelle des JEE Frameworks spielt für Ausbildungszwecke keine entscheidende Rolle. Sollten Sie in der Industrie später mit JEE arbeiten, sind alle Gelernten Ideen 1:1 übertragbar.

Sollte es nötig sein, unsere Applikation auf einem vorhandenen Server laufen zu lassen, können wir uns Anstelle einer .jar- auch eine .war-Datei erstellen lassen und diese dann auf einen bestehenden Server «Deployen».

Ein weiterer grosser Vorteil von Spring Boot ist es, dass wir ein Entwicklermodul laden können, welche Codeänderungen direkt in den laufenden Server übernimmt, ohne dass wir ein «Redeploy» starten müssen. Das Debuggen auf dem Server klappt Problemlos.

Dank seiner Rapid-Prototyping-Struktur, sind Spring Boot-Projekte in wenigen Minuten aufgesetzt und lauffähig, so, dass wir unmittelbar mit der Applikationsentwicklung beginnen können.