Web-Apps mit Play! entwickeln - nichts leichter als das!

Andreas Koop
enpit consulting OHG
33102 Paderborn

Schlüsselworte
Playframework, Play!, Erfahrungsbericht

Einleitung
Mit dem Open Source Webframework Play! erhalten etablierte Java-Webframeworks ernstzunehmende Konkurrenz. Nicht zu letzt die Integration des Play!Frameworks in der Version 2 in den Typesafe-Stack der Scala-Macher bringt frischen Wind in den Dunstkreis von Struts, JSF, Tapestry und Grails.

Das Play!Framework verzichtet auf Servlets und bricht auch sonst mit weit verbreiteten Konventionen im Java-Enterprise-Umfeld, um die Produktivität und Leichtgewichtigkeit wieder in den Vordergrund zu rücken. Lernen Sie in dieser Session die Grundlagen der skalierbaren Architektur von Play! kennen und erleben die Konzepte anhand von Live-Codings. Die Entwicklung von Tests, deren Durchführung, Bereitstellung von Testdaten, sowie Deployment in die Cloud runden den Vortrag ab.

Architektur und Technologien
Play 2 bezeichnet sich selbst als „Web framework for a new era“ und setzt damit die Messlatte sehr hoch an. Die Architektur basiert auf einem asynchronen Request-Verarbeitungsmodell. Es ist für hochperformante und -skalierbare Anwendungen konzipiert. Folgende Grafik veranschaulicht die wesentlichen Komponenten:

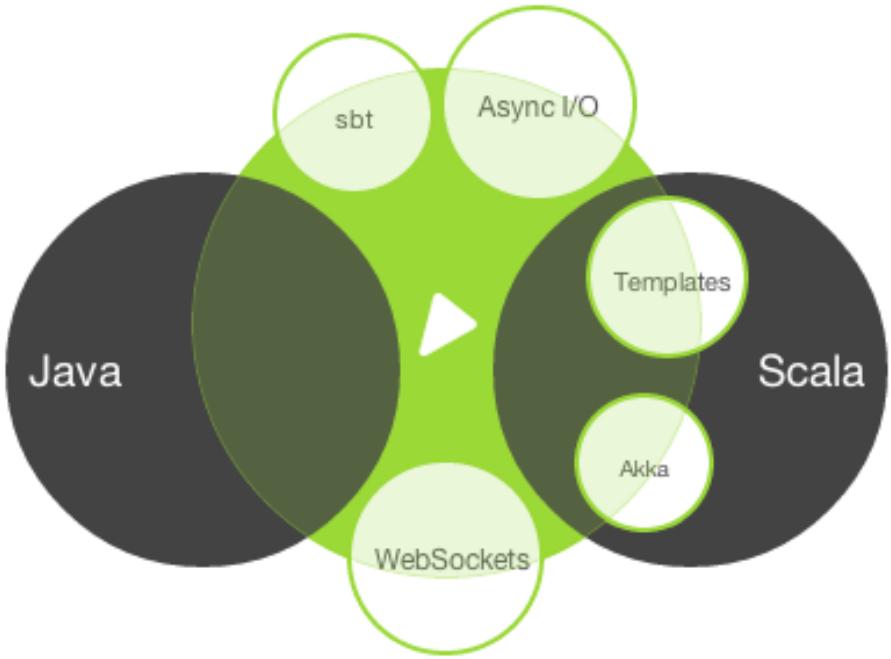

Abb. 1: Architektur- und Technologiestack von Play 2 (Quelle: www.playframework.org)

Eine Besonderheit von Play ist, dass Anwendungen sowohl in Java als auch Scala programmiert werden können. Dies zeigt sich unmittelbar bei Erstellung einer neuen Play-Anwendung, was im Übrigen durch Ausführung eines einzigen Befehls auf der Kommandozeile erledigt wird:

```
$ play new myApplication
```

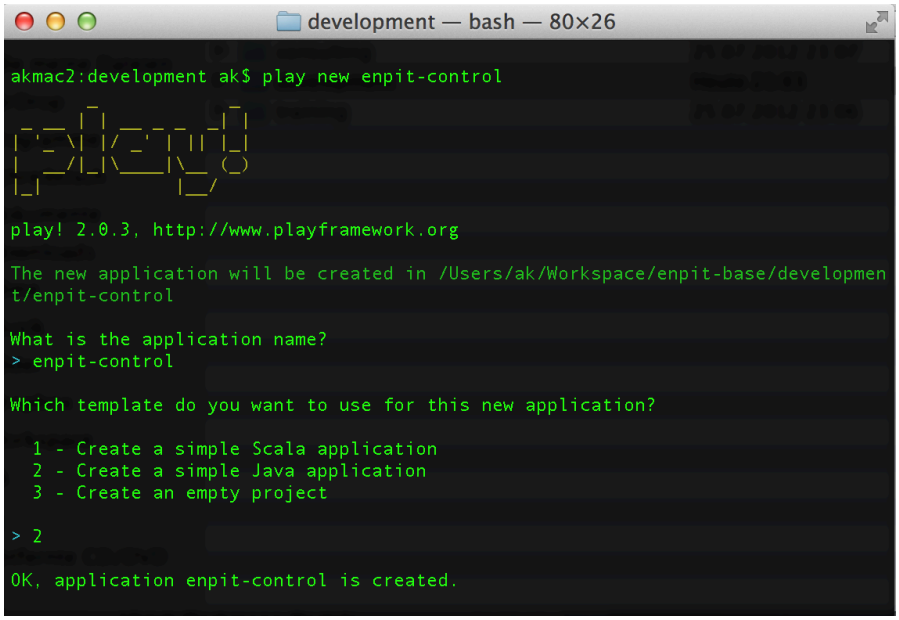

Abb. 2: Erstellung einer neuen Play-Anwendung

Bei der Erstellung einer neuen Anwendung werden per Konvention alle notwendigen Dateien und Verzeichnisse erstellt, so dass im Anschluss die Anwendung per „play run“ unmittelbar über den built-in Server auf dem Defaultport 9000 erreichbar ist: http://localhost:9000 . Über die Struktur einer Play-Anwendung gibt Abb. 2 Aufschluss:

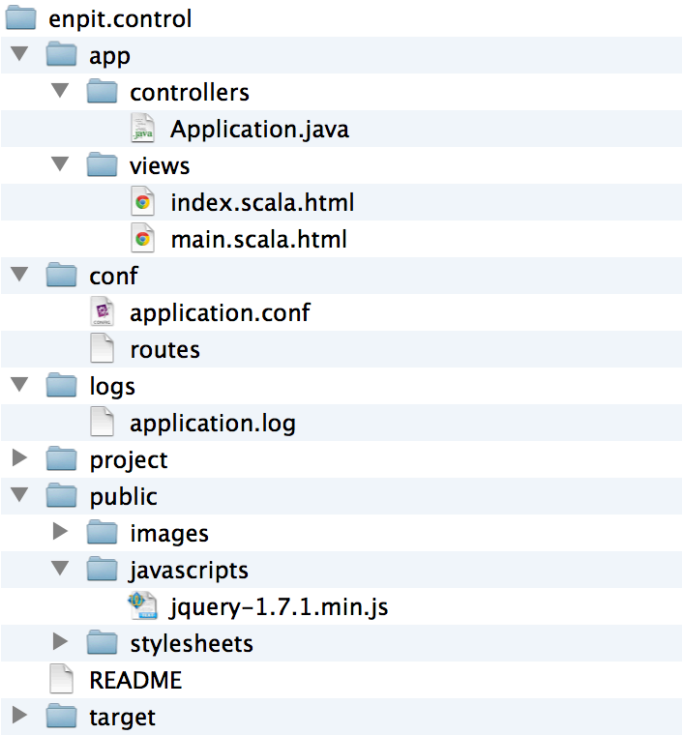

Abb. 3: Struktur / Basis-Dateien einer Play-Anwendung

Der Entwickler muss sich keine Gedanken über die Ablagestruktur für Modell-, Controllerklassen, Views, Konfigurationsdateien, Public-Resourcen wie Bild-, CSS- und JS-Dateien. Über die letzten Jahre hat sich für RAD-Webframeworks eine geordnete Struktur etabliert. Sogar die Logdatei und Build-Ausgabe haben im Entwicklungsmodus einen wohldefinierten Platz.

Konzepte und Techniken
Das Playframework unterstützt Entwicklungstechniken für nahezu jeglichen Bedarf. Folgende Tabelle gibt einen Überblick der verfügbaren Optionen für essentiellen Bereiche einer Webanwendung sowie des dazugehörigen Entwicklungsprozesses.

Bereich	Technik	Anmerkung
Modell (Fachdomäne)	POJO mit „Property-Feature“, keine Getter/Setter-Orgie	Getter und Setter werden zur Laufzeit generiert.
O/R-Mapping Persistenz, Transaktionshandling	Auswahl von - EBean - JPA oder - Anorm (Scala-native)	EBean löst JPA (Hibernate) als Default-Implementierung für O/R-Mapping in Play 2 ab. Warum?

Validierung	Konzeptionell ähnlich zu JSR-303 (Bean-Validation), jedoch Eigenimplementierung in Play 2	Laut Play!-Machern ist die eigene Implementierung umfangreicher und flexibler.
Controller	- RESTful Routing - RequestActions - Form-To-Modell-Mapping	Routing-Regeln werden mit einer Scala-DSL erstellt. Konfigurationsfehler werden somit zur Compilezeit aufgedeckt!
View / Client	- Scala basierte HTML(5)-Views - Modularisierung von Views in Form von Scala-Funktionen. - JQuery inkludiert - Integrierter LESS Precompiler - Integrierter CoffeeScript Precompiler - Integrierte Google Clojure JavaScript Compress Engine - „WebJars“	Im View und dem immer wichtigeren Client-Bereich (CSS3, JavaScript) setzt Play 2 neue Maßstäbe. Die Dynamik der Scala-Programmiersprache ermöglicht es HTML-Views als Scala-Methoden abzubilden. Fehler in den Views werden somit auch
Session-Handling	Stateless / Cookie (verschlüsselt)	Play Anwendungen sind zustandslos und besitzen daher keine aus Java EE vergleichbare HTTP-Session. Sessionbezogene Daten werden im Cookie gespeichert und haben daher eine Begrenzung von max. 32KB
Security	- Built-In Mechanismus: Annotationsbasiert - SocialSecure Plugin	
Logging	- Logback	Generisches Logging-Framework. Nachfolger des populären Log4J.
Push / Realtime-Streaming	- Comet Support - WebSocket Support - Server-Sent-Events Support	Serverseitiger Support durch innovatives (zumindest in der Java-Welt) Streamingkonzept aus der „funktional-orientierten Programmierwelt“: - Iteratee [E,A] (Consumer) - Enumerator [E] (Producer) - Enumeratee [From, To] (Transformation eines Streams)

Parallele / Verteilte Programmierung	- Nativer Support für die Akka Middleware (Aktor-basiertes Modell)	Ein Thema für sich.
Jobs / Scheduling	- Play Jobs API. Annotationen, entsprechende Hooks	Auch daran wurde gedacht! Einfach spitze!
Deployment / Runtime	- Built-In Netty-Server - WAR - Java EE App Server / WebContainer: JBoss, GlassFish, WebSphere, Geronimo, Tomcat, Jetty, Resin	Während der Entwicklung sind keine Redeployments notwendig. Fehler werden
Cloud	- Heroku (TOP) - OpenShift (TOP) - Cloudbees (WAR Deploy..) - und mehr!	Mittels Plugins direkt per CLI möglich, z.B. `$ git push heroku master` Ein Play-Heroku-Plugin für noch mehr Komfort ist in Arbeit.
Konfigurationen / Stage Modi	Konfiguration by Convention - conf/application.conf Stage Modi - development - production	Konfiguration erfolgt im HOCON-Format („Human Optimized Config Object Notation“) Play enthält von Haus aus Unterstützung für entsprechende Modi. Im Development-Modus startet der Server mit „auto-reload“ Feature, was bedeutet, dass bei jedem Request geänderter Quellcode automatisch übersetzt wird. Falls notwendig, wird der gesamte Server neugestartet. Außer einer etwas längeren Verzögerung bleibt diese radikale Hintergrundaktivität für den Play-Entwickler vollkommen transparent. Da kann man nur „Wow!“ sagen.

Build & Dependency Management, QA	- SBT (Simple Built Tool, Scala)
	- Maven Plugin verfügbar für Play 2 (http://cescoffier.github.com/maven-play2-plugin/maven/release/)
	
	`$ play compile`
	`$ play clean`
	`$ play clean-all`
		Mit Play 2 eingeführt. Standard Build-Tool in Scala-Projekten. SBT ist ein Thema für sich.

Debugging Support	`$ play debug`
	
	Dieser Befehl öffnet einen JPDA Debug Port.
	
	`Listening for transport dt_socket at address: 9999`
		Per Java Debugger aus einer IDE verbindung aufbauen und debuggen.

Test	- Specs2 - ScalaTest - JUnit - Selenium Testdaten (sog. evolution scripte) können ebenfalls bereitgestellt werden.	Specs2 Support ist per Default seit Play2 integriert. Falls der Testcode eine laufende Anwendung benötigt, kann dies mit der `FakeApplication()` simuliert werden
Projektdokumentation	Hier kann auf Bewährtes zugegriffen werden - Mit dem Maven-Plugin.	Standard-Reports, Test-Reports, API, usw. - Alles was das Maven-Ökosystem so anbietet.

<table>
<tr><td>Lastverteilung</td><td colspan="2">Aufgrund der stateless-Architektur von Play-Anwendungen ist eine Lastverteilung mit einem leichtgewichtigen HTTP-Server umsetzbar, z.B. mit lighttpd oder dem Standard Web-Server schlecht hin - Apache. Beispiel mit lighttpd:
o
<pre>server.modules = (
      "mod_access",
      "mod_proxy",
      "mod_accesslog"
)
…
$HTTP["host"] =~ "www.myapp.com" {
    proxy.balance = "round-robin" proxy.server =
("/" =>
        (("host" => "127.0.0.1", "port" =>
9000)))
}

$HTTP["host"] =~ "www.loadbalancedapp.com" {
    proxy.balance = "round-robin" proxy.server =
("/" => (
          ("host" => "127.0.0.1", "port" =>
9001),
          ("host" => "127.0.0.1", "port" =>
9002))
)
}</pre></td></tr>
<tr><td>Dokumentation</td><td>- Muss nicht lange gesucht werden: http://localhost:9000/@documentation
- API für Java http://localhost:9000/@documentation/api/java/index.html
- API für Scala http://localhost:9000/@documentation/api/scala/index.html</td><td>Die Dokumentation ist im dev-Modus ohne langes Suchen stets zur Hand! Einfach grandios!</td></tr>
<tr><td>Community</td><td>- Google Group play-framework (6883 Mitglieder)
- Issue Tracker unter https://play.lighthouseapp.com
- Twitter: @playframework
- ...</td><td></td></tr>
</table>

Durch Verwendung von Twitter-Bootstrap ist auch im Front-End einfach und schnell ein schicke Oberfläche gezaubert, wie in Abb. 4 dargestellt.

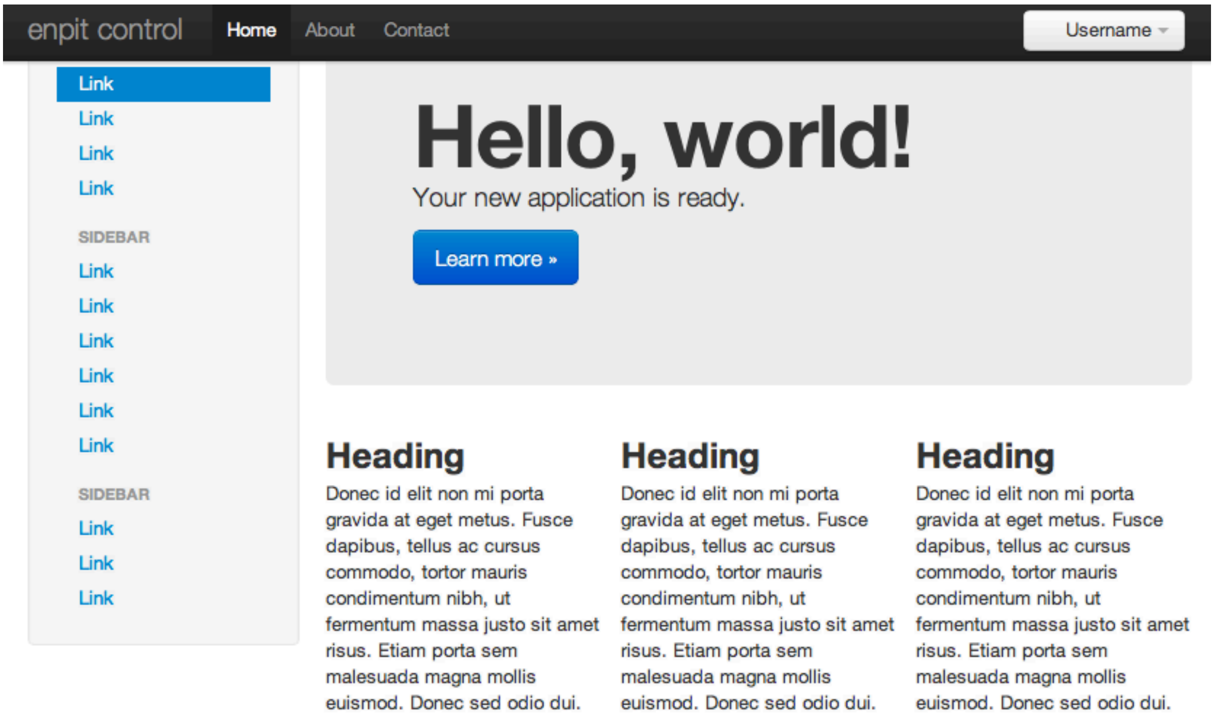

Abb. 4: UI auf Basis von Twitter-Bootstrap

Zusammenfassung

Die Architektur von Play! ist sehr durchdacht und für Echtzeit- und hochskalierbare Webanwendungen ausgelegt. In großen Teilen sind die verwendeten Konzepte mit denen von z.B. Grails vergleichbar.

Das Playframework erfährt u.a. aufgrund des wirklich umfangreichen und kompletten „Stacks“ einen deutlich höheren Hype, nicht zuletzt durch die Unterstützung der Scala-Macher von Typesafe Inc.

Wer nah an CSS(3), JavaScript, HTML(5) bleiben möchte und ganz nebenbei noch neue Programmiersprachen (Scala, CoffeeScript, LESS) erlernen möchte, der ist beim Play-Framework genau richtig! Die exzellenten Möglichkeiten, neuste Techniken wie WebSockets und SSE auf verblüffend einfache und geniale Art undWeise einzusetzen, rufen wahre Freuden bei der Entwicklung hervor!

Kontaktadresse:

Dipl.-Inform. Andreas Koop
enpit consulting OHG
Theodor-Heuss-Str. 17c
D-33102 Paderborn

Telefon: +49 (0) 5251 20277 92
E-Mail andreas.koop@enpit.de
Internet: www.enpit.de