Apache Tapestry
Alternatives Konzept zur Web-Anwendungsentwicklung

Jan Diller
Triestram & Partner GmbH
Bochum

Schlüsselworte

Web-Application, Tapestry

Einleitung

Tapestry ist ein komponentenbasiertes Framework zum Erstellen dynamischer, robuster, hoch skalierender Web-Anwendungen in Java.

Wer die Tapestry-Homepage besucht, kann diesen Satz dort so nachlesen. Klar, die Aussage ist Werbung. Man muss ja Interesse für solch ein Framework wecken. Dass es richtig ist, habe ich erfahren dürfen und die Erfahrung möchte ich teilen.

Meine erste Begegnung mit Tapestry hatte ich 2004 auf einer Konferenz. Ich verfolgte einen Tapestry-Vortrag mit Spannung, machte Tapestry doch alles so anders als die anderen Frameworks. Damals waren JSP State of the Art. Um es ein bisschen erträglicher zu machen, holte man sich Unterstützung von Struts. JSF gab es schon, jedoch erst in der Version 1.0 und wer arbeitet schon mit einer 1.0? Leider kam es damals nicht dazu, direkt mit einem Projekt zu starten. Tapestry geriet bei mir nicht in Vergessenheit, jedoch aus dem Fokus. So musste ich 8 Jahre warten, bis die Gelegenheit kam, ein Projekt auf der grünen Wiese neu zu gestalten.

Als ein Kollege mich fragte, ob ich seine Meinung teilen würde, dass wir es mal mit Tapestry versuchen sollten, war mein Interesse sofort geweckt. Und wir wurden nicht enttäuscht. In unserem Projekt sind wir erstaunlich schnell und gut vorangekommen. Geliefert haben wir eine robuste Anwendung, womit sich schon einmal eine Werbe-Aussage bestätigt hätte.

Abgrenzung zu anderen Technologien

Was unterscheidet Tapestry von anderen Web-Frameworks? Allen Frameworks gemein ist, dass sie auf der Servlet-API aufbauen. Kaum vorstellbar, dass ein Framework darauf verzichtet, käme es doch einem Verzicht auf einen Web-Servers gleich.

Auf der Servlet-Technologie basierend gibt es eine ganze Korona unterschiedlicher Frameworks mit unterschiedlichen Basis-Technologien. Die Frameworks Struts, Cocoon und WebWork ergänzten seinerzeit die JSP-Technologie. Die heutzutage beliebten Frameworks Mojarra, MyFaces, PrimeFaces usw. teilen sich die Gemeinsamkeit, dass sie auf JSF basieren.

Abseits hiervon stehen Frameworks wie Tapestry oder Wicket. Sie gehen ihren eigenen Weg und basieren weder auf JSP noch auf JSF.

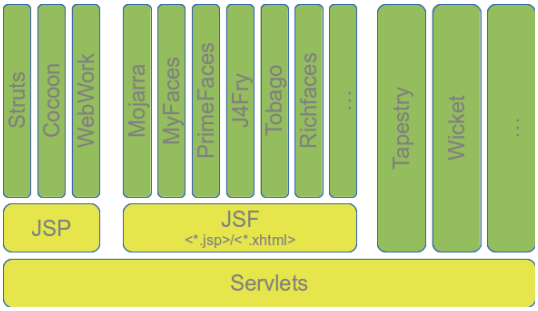

Abbildung 1: Technischer Unterbau populärer Frameworks

Da Tapestry nur auf der Servlet-Technologie basiert, besteht mit Ausnahme zur *web.xml* keine weitere Verpflichtung zur Konfiguration.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app
        PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <display-name>tpy Tapestry 5 Application</display-name>
    <context-param>
        <param-name>tapestry.app-package</param-name>
        <param-value>com.tp.doag2013</param-value>
    </context-param>
    <context-param>
        <param-name>tapestry.development-modules</param-name>
        <param-value>
            com.tp.doag2013.services.DevelopmentModule
        </param-value>
    </context-param>
    <context-param>
        <param-name>tapestry.qa-modules</param-name>
        <param-value>
            com.tp.doag2013.services.QaModule
        </param-value>
    </context-param>
    <filter>
        <filter-name>app</filter-name>
        <filter-class>org.apache.tapestry5.TapestryFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>app</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Abbildung 2: Basis web.xml von Tapestry

Eine Besonderheit in der *web.xml* fällt erst auf den zweiten Blick auf: Tapestry wird als Filter in die *web.xml* eingetragen. Der Grund hierfür kann der Tapestry Dokumentation entnommen werden. Es hat damit zu tun, dass Filter-Mapping technologisch auf einer niedrigeren Stufe als das Servlet Mapping ansetzt. Dadurch hat Tapestry quasi den gesamten technischen Ablauf eines Request-Response-Zykluses in seiner Hand.

Weitere Parameter sind die *<XY>Module*-Dateien. Mittels Module-Dateien findet die Konfiguration von Tapestry selbst statt. Es handelt sich hierbei um Java-Dateien, d.h. die Konfiguration erfolgt programmatisch. Ein weiterer wichtiger Eintrag ist das App-Package. Ein Package ist nichts anderes als das Äquivalent eines Pfades. Das App-Package definiert somit den Basis-Pfad der Anwendung. Wofür der wichtig ist, dazu im kommenden Kapitel mehr.

Convention over Configuration

Zwei Grundprinzipien liegen der Entwicklung von Tapestry zu Grunde. Eines davon ist das Prinzip, dass Konventionen gegenüber Konfigurationen zu bevorzugen sind. Statt zeitraubender Auslagerung von Abhängigkeiten in Konfigurations-Dateien favorisiert Tapestry den strikten Verzicht auf Konfigurationen. Die Verzeichnisstruktur folgt einer Konvention. Die Namen der Methoden in den Controllern folgen einer Konvention. Werden Services definiert, welche sich auf ein Interface beziehen, wird automatisch nach der Implementierung anhand des Interface Namens, ergänzt um den Suffix Impl, gesucht. Wer sich mit Tapestry beschäftigt wird noch viele Belege für die Verfolgung dieses Prinzips finden.

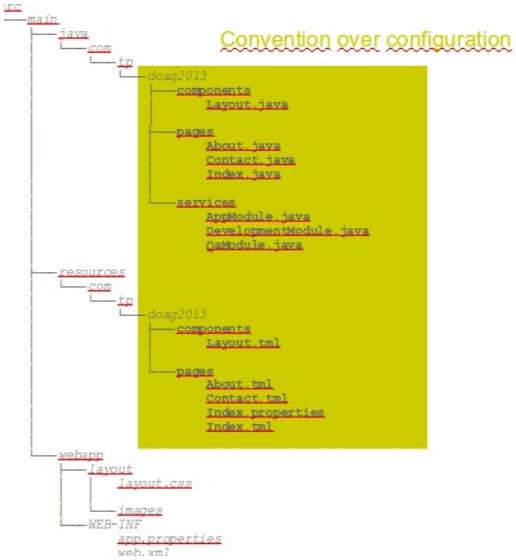

Abbildung 3: Convention over Configuration am Beispiel der Verzeichnisstruktur

Am einfachsten lässt sich dieses Prinzip an der Verzeichnisstruktur verdeutlichen. Unterhalb des Basis-Pfades befinden sich die drei Verzeichnisse *Components*, *Pages* und *Services*. Die Namen dieser Verzeichnisse sind reserviert. Es gibt weitere, z.B. *Mixins*, aber zur Verdeutlichung des Prinzips reichen diese drei aus. Im Verzeichnis *Components* befinden sich, wenig überraschend, Komponenten. Unter Komponenten versteht Tapestry unvollständige HTML-Konstrukte, also Fragmente, die entweder um weitere Inhalte ergänzt werden müssen, um vollständige Seiten zu ergeben oder solche, die in andere Seiten eingebunden werden können, um diese zu vervollständigen. Ein solches Fragment ist z.B. das Layout, aus dem mittels der Tapestry eigenen Template-Engine vollständige Seiten generiert werden.

Seiten befinden sich unterhalb des Verzeichnisses *Pages*. Eine Seite besteht zwingend aus ihrer Präsentation, der *TML*-Datei (.tml), und dem zugehörigen *Controller* (.java). TML und Controller bilden eine Einheit und kein Teil darf ohne den jeweils andern existieren. Seiten im *Pages*-Verzeichnis sind direkt aufrufbar. Unterverzeichnisse werden eins zu eins in die URL übernommen.

Im Verzeichnis *Services* liegen Dienste. Hier finden sich zum Beispiel die Dateien *AppModule*, *DevelopmentModule* und *QaModule*, die für die Konfiguration in verschiedenen Modi zuständig sind. Wer weder eine spezielle Entwicklungs- noch Qualitäts-Management Konfiguration benötigt, der kommt mit der *AppModule*-Datei hin. In dieser wird programmatisch die Konfiguration vorgenommen.

Wurden die Konventionen einmal verinnerlicht, ist ein schnelles Einarbeiten in bestehende Projekte möglich, da keine x-tausend Konfigurationsdateien zu lesen sind. Natürlich spart es auch Zeit bei der Entwicklung, müssen sie doch erst gar nicht geschrieben werden.

Inversion of Control

Das zweite Grundprinzip von Tapestry ist die *Inversion of Control*. Tapestry versteht darunter die Möglichkeit, Dienste zu registrieren und diese einfach einem entsprechenden Konsumenten, z.B. dem Controller, zu injizieren. Dienste, die in die Controller injiziert werden, beeinflussen die Event-Kette, was in gewisser Art und Weise einer Umkehrung des Kontrollflusses gleichkommt, daher „Inversion of Control".

Dieses Prinzip ist keine nachträgliche Erweiterung für den Benutzer des Frameworks, es ist zentrales Element von Tapestry und findet bei sämtlichen Abläufen Anwendung. Wenn somit die Notwendigkeit besteht, grundlegende Mechanismen zu überschreiben, erreicht man das indem man den entsprechenden Dienst überschreibt oder erweitert.

Getting started

Für den zügigen Einstieg ins produktive Arbeiten mit Tapestry existieren zwei Möglichkeiten. Am schnellsten kommt der zum Ziel, welcher beide auf geeignete Weise verbindet.

Möglichkeit eins besteht darin, sein Projekt durch die Verwendung von Maven innerhalb kürzester Zeit aufzusetzen.

Hierzu stellt Tapestry einen eigenen Maven Archetype-Catalog bereit. Ist Maven installiert kann man ihn einfach aufrufen.

- *mvn archetype:generate -DarchetypeCatalog=http://tapestry.apache.org*

Das Ergebnis ist eine kleine, lauffähige Anwendung mit vorbereiteter POM. So kann man die Anwendung direkt mit dem folgenden Befehl starten.

- *mvn clean install jetty:run*

Die zweite Möglichkeit besteht in einem Tutorial mit dem Namen Jump-Start auf der Homepage von Tapestry. Dieses kann online verwendet werden, steht aber auch zum Download bereit. Der Download kann auf JBoss, Tomcat, Jetty und anderen App- bzw. Web-Servern betrieben werden. Die Aufbereitung zur Einspielung erfolgt über ein beigefügtes Ant-Script.

In diesem Tutorial werden alle Prinzipien und Komponenten gut erklärt. Wenn es etwas zu kritisieren gibt, dann allerhöchstens, dass manche Beispiele zu erschöpfend und zu detailliert beschrieben sind. Sie ufern teilweise so aus, dass der Blick fürs Wesentliche verloren geht. Dem Tutorial können alle wichtigen Dinge für das eigene Projekt entnommen werden. Die meisten Sachen sind in Tapestry recht knapp zu halten, weshalb man hier mit Copy&Paste zügig vorankommt.

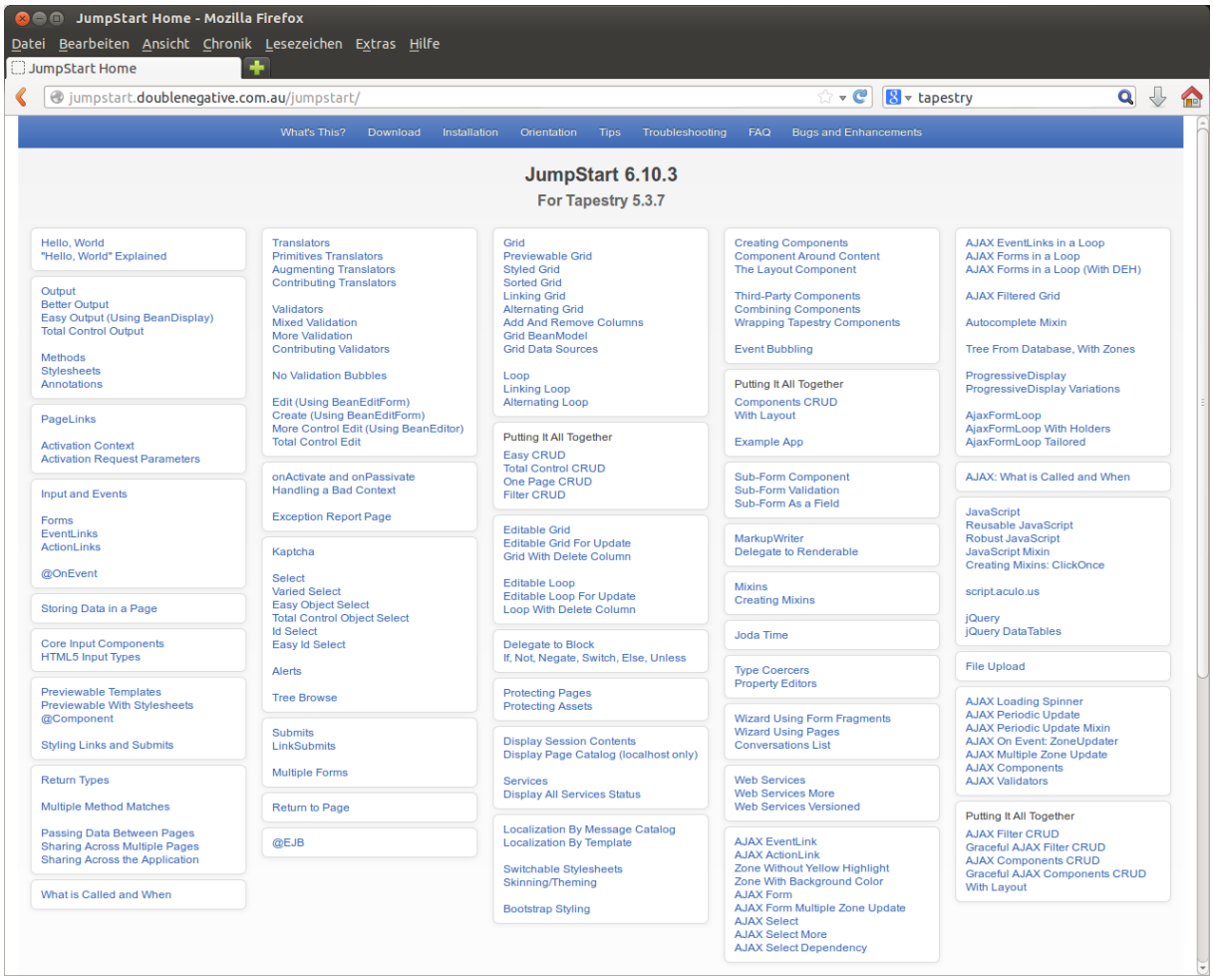

Abbildung 4: Tapestry JumpStart

Die Verknüpfung beider Möglichkeiten besteht darin, sein Projekt mittels Maven aufzusetzen und sich bei der Entwicklung immer wieder der reichlichen Beispiele aus dem Tutorial zu bedienen.

Dokumentation

An dieser Stelle sei ein Wort zum allgemeinen Zustand der Dokumentation erlaubt: Tapestry ist vorbildlich dokumentiert. Wer sich in die Gedankenwelt, die hinter dem Framework steckt, einarbeiten möchte, findet auf der Homepage fast alle hierfür nötigen Information. Es wird deutlich, wie wichtig es den Entwicklern gewesen ist, ihre Vorstellung von Software an den Nutzer weiter zu geben.

Fazit

Dieser Beitrag reicht nicht aus, alle Aspekte von Tapestry zu beleuchten. Er soll vielmehr Lust darauf machen, sich in der nächsten Projektpause oder während einer internen Weiterbildungsphase damit zu beschäftigen. Dass es für den produktiven Einsatz geeignet ist, habe ich erfahren dürfen. Wer also mehr tun muss, als seine Projektpause sinnvoll zu gestalten und sich vielleicht aus gutem Grund nach einer Alternativen zum gegenwärtigen Mainstream umschaut, der sei dazu ermutigt, sich Tapestry

einmal genauer anzusehen.

Bei Tapestry handelt es sich um ein vorbildlich dokumentiertes Framework, welches einem alle Werkzeuge liefert, möglichst schnell produktiv ins kommende Projekt zu starten.

Kontaktadresse:
Jan Diller
Triestram und Partner GmbH
Kohlenstraße 55
D-44795 Bochum

Telefon:	+49 (0) 234 943 750
Fax:	+49 (0) 234 452 206
E-Mail	j.diller@t-p.com
Internet:	http://www.t-p.com