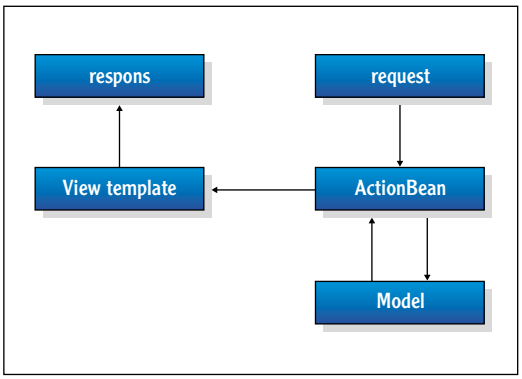
Stripes is een open source controller-framework dat nog het meest te vergelijken is met Struts of Spring Web-MVC. Stripes is makkelijker te configureren, beter onderhoudbaar, sneller aan te leren en maakt web development weer leuk. Daarnaast integreert Stripes met veelgebruikte frameworks als Hibernate, EJB3 en Spring.

Stripes: compleet framework voor webdevelopment

Jeroen van Wilgenburg
is Java Software Engineer bij AMIS. Hij is te bereiken op wilgenburg@amis.nl

Stripes biedt naast een controller-framework een aantal interessante extra's. Integratie met Javascript in de browser is een fluitje van een cent en er is een layout-taglibrary die vergelijkbaar is met Tiles en Sitemesh. Je kunt daarmee op een makkelijke manier je site 'in stukken knippen' en zo je header of menubalk maar een keer coderen. Je bent vrij in het kiezen van een view-technologie, er wordt een JSP-taglibrary meegeleverd en er is een handleiding voor Stripes in combinatie met Freemarker.

Struts is lange tijd het meest gebruikte webframework geweest maar heeft toch zijn beperkingen en is niet echt meegegroeid met de ontwikkelingen op java-gebied (bijvoorbeeld annotaties en generics). Veelgehoorde klachten zijn de af en toe gigantische configuratiebestanden en dat een simpele pagina al gauw in vijf bestanden aanwezig moet zijn en dat deze bestanden handmatig in sync gehouden moeten worden (bij bijvoorbeeld het renamen van classes of url's)



Stripes wil een oplossing voor deze beperkingen bieden en een webframework zijn dat het schrijven van webapplicaties weer makkelijk en vooral leuk moet maken.

Hello World

Voor Stripes zijn geen externe configuratiebestanden nodig. Na het toevoegen van een filter en servlet aan de web.xml kun je direct van Stripes gebruikmaken.

Een controller in Stripes is een java-klasse die de interface ActionBean implementeert. Deze interface schrijft een getter en setter voor het context-object voor. Naast het implementeren van deze methodes kunnen er methodes worden toegevoegd die acties in de applicatie uitvoeren (het versturen van een formulier, opzoeken van gegevens of het benaderen van een pagina). Voor elke ActionBean is een @DefaultHandler-annotatie op één van de methodes nodig voor het geval er moet worden teruggevallen op een standaardmethode (bijvoorbeeld als een methodenaam niet kan worden gevonden).

Een actiemethode heeft geen parameters en geeft een Resolution-object terug. Het Resolution-object is een interface en aan de hand van de gekozen implementatie wordt bepaald hoe naar de opgegeven url wordt gegaan. Standaard zul je een ForwardResolution gebruiken, verderop in dit artikel worden de andere Resolutions behandeld. Request of repsonse-objecten hoeven niet te worden meegegeven. Om bij de request of response te komen kun je de getRequest of getResponse op het context-object uitvoeren.

Een simple controller die een forward doet naar een jsp ziet er als volgt uit:

```
public class HelloWorldActionBean implements
ActionBean {

    private ActionBeanContext context;

    @DefaultHandler
    public Resolution hello() {
        return new ForwardResolution("/hello.jsp");
    }

    public void setContext(ActionBeanContext
    context) {
        this.context = context;
    }

    public ActionBeanContext getContext() {
        return context;
    }
}
```

In web.xml is opgegeven dat alle url's die eindigen op .action naar de Stripes DispatcherServlet gaan. Omdat we niet in de handleiding gekeken hebben hoe de url voor deze klasse eruit ziet typen we iets willekeurigs in dat eindigt op .action. Er komt een foutmelding in beeld die duidelijk aangeeft wat er aan de hand is:

```
Could not locate an ActionBean that is bound to the
URL [/piet.action]. Commons reasons for this include
mis-matched URLs and forgetting to implement
ActionBean in your class. Registered ActionBeans
are: {/HelloWorld.action=class nl.amis.action.
HelloWorldActionBean}
```

Vaak is het altijd even een uitdaging om een Hello World-voorbeeld draaiend te krijgen met een nieuw framework. Met Stripes word je gelukkig goed geholpen. Vergeet alleen niet op een productiemachine deze feature dicht te zitten, niemand hoeft te weten waar de ResetAdminPasswordActionBean zich bevindt. Een ActionBean moet in een package zitten waarin de www, action of stripes voorkomt. Stripes gebruikt de packagenaam die komt na www, stripes en action als 'directories' op de server. Als we de klasse in nl.amis.action.hello hadden gestopt hadden we naar de url /hello/HelloWorld.action moeten gaan, in dit geval is het /HelloWorld.action.
Het kan zijn dat je de klassenamen niet wilt tonen aan de buitenwereld of gewoon kortere varianten wil. Dit kan door middel van een annotatie op de klasse. Met @UrlBinding("/hw.action") zorgen we ervoor dat /hw.action uitkomt bij de HelloWorldActionBean. HelloWorld.action is na het toevoegen van de annotatie ook niet meer benaderbaar.

De laatste stap om Hello World op het scherm te krijgen is een pagina hello.jsp aan te maken (de pagina die geretourneerd wordt in de methode hello). Typ in deze pagina het fameuze Hello World, deploy de applicatie en ga in de browser naar hw.action.

Taglibrary

Een andere belangrijke feature van Stripes is de taglibrary. Deze library biedt functionaliteiten voor layout, formulieren, url rewriting en een aantal andere zaken die verderop worden behandeld.

In de meeste web-frameworks is een configuratiebestand nodig om een url op een java-klasse te mappen. Het nadeel van deze manier is dat je vaak in deze bestanden zit te zoeken en ze vaak heel erg groot worden. Stripes staat toe om de naam van een klasse in de jsp te gebruiken zonder tussenkomst van een configuratiebestand. Als je <stripes:url beanclass="nl.amis.action.HelloWorldActionBean"/> toevoegt aan een jsp dan zal in de browser de url /hw.action verschijnen. Als je gaat refactoren kan deze werkwijze echter wel problemen opleveren, maar de meeste IDE's gaan hier tegenwoordig wel goed mee om. Het is in ieder geval beter dan url's in de xml-files opzoeken.

Bij de meeste webapplicaties zullen grote stukken van de interface worden hergebruikt, ook hier zijn weer frameworks voor gemaakt. Bij JSF is Facelets een erg mooie oplossing en bij Struts en Spring wordt veel gebruik gemaakt van Tiles. Het nadeel van Tiles is dat het niet erg intuïtief in elkaar zit en voor overkill zorgt. Wat Tiles kan is erg prettig, maar met de <stripes:layout/> tag kan het simpeler. De taglibrary is trouwens ook prima zonder de rest van Stripes te gebruiken, het toevoegen van het filter StripesFilter aan de web.xml is voldoende.

In onderstaand voorbeeld wordt een pagina in stukken geknipt en voorzien van de layout tags van Stripes. De site bestaat uit drie onderdelen: de balk bovenin (header), de balk met nieuwsberichten rechts (right) en de rest (content). Het idee van Stripes is dat je een template maakt (een gewone jsp) en op de plekken waar de inhoud kan veranderen, een <stripes:layout-component/> wordt ingevoegd.
Maak een bestand /WEB-INF/layout/default.jsp en maak een standaard-html pagina compleet met head en body. Zet nu voor <html> een <stripes:layout-definition> en sluit deze tag onderaan de pagina. Zet de volgende code in de <body>!

```
<div id="container" style="width: 750px;">
    <div id="header" style="float:left; width: 740px;
background: #AA273D; color: white">
        Welkom bij AMIS!
    </div>
    <div id="right" style="float:right; width: 200px;
background: #FBAD17;">
        <stripes:layout-component name="right"/>
    </div>
      <div id="content" style="float:left; width:
525px; background: #DDDDDD;">
        <stripes:layout-component name="content"/>
    </div>
</div>
```

Dit bestand is nu het hoofdtemplate, dat in andere pagina's kan worden gebruikt.
Maak nu een bestand genaamd hello.jsp met de volgende inhoud:

```
<%@ taglib prefix="stripes"
uri="http://stripes.sourceforge.net/stripes.tld" %>
<stripes:layout-render    name="/WEB-INF/layout/
default.jsp">
    <stripes:layout-component name="right">
        <ul>
            <li>20:45 Hello World 1.1 bugfix</li>
           <li>20:00 Hello World in productie</li>
        </ul>
    </stripes:layout-component>
    <stripes:layout-component name="content">
        Hello World
    </stripes:layout-component>
</stripes:layout-render>
```

De <stripes:layout-component/>-tag vervangt nu de <stripes:layout-component/> tags in de hoofdtemplate.

Please fix the following errors:

1. The value (honderd) entered in field Bewoners Leeftijd must be a valid number
2. The minimum allowed value for Bewoners Leeftijd is 50

Voornaam	Achternaam	Leeftijd
Jan	Klaassen	honderd
Piet	Hendriksen	83
Hendrik	Jansen	34
Klaas	Pietersen	67

In de praktijk zul je nog een gedeeltelijk gevulde template maken en deze template in de uiteindelijke pagina gebruiken. Het is toegestaan een <stripes:layout-render> binnen een <stripes:layout-definition> te gebruiken. Alleen als je de <stripes:layout-component> tag gebruikt, wordt deze gebonden aan de layout-definition (wat je niet wil).
Om dit op te lossen dien je een EL expressie te gebruiken. Stel dat je <stripes:layout-component name=înestedcontentî/> wilt gebruiken moet je dit door ${nestedcontent} vervangen. Dit is een wat minder elegante oplossing, maar doordat je gebruik maakt van de layouttags is de jsp overzichtelijker dan hij vroeger was en valt dit wel netjes in te passen. De layouttags zijn overigens prima zonder de Stripes ActionBeans te gebruiken.

Formulieren en validatie

Voor de formulieren in jsp zijn tags beschikbaar die vergelijkbaar zijn met tags van Spring en Struts. In de ActionBean zijn alle velden die toegankelijk gemaakt zijn met behulp van getters en setters direct beschikbaar. Met het name-attribuut op de <stripes:submit/> kun je aangeven welke methode in de ActionBean wordt uitgevoerd.

Met Stripes is het mogelijk om een lijst van gegevens (een List-object op de ActionBean) in één keer te bewerken. Op zich is dit niet bijzonder (indexed properties in Struts), maar het gemak waarmee het gaat wel. Validaties zijn nog steeds net zo simpel als met normale formulieren. Het valideren in Stripes gaat met de @Validate-annotatie. Het is ook mogelijk om dieper in de objectboom te valideren met behulp van @ValidateNestedProperties. In het volgende voorbeeld worden Bewoner-objecten in de lijst gevalideerd, een achternaam is verplicht en een leeftijd moet minimaal 50 zijn. Was het bewoners-object geen lijst geweest maar een Bewoner object, dan had de validatie er hetzelfde uitgezien.

```
@ValidateNestedProperties({
@Validate(field = "achternaam", required = true),
@Validate(field = "leeftijd", minvalue = 50)}
)
private List<Bewoner> bewoners = new
ArrayList<Bewoner>();
```

Vullen we nu een aantal bewoners in met fouten erin dan zal dat op de volgende wijze worden getoond:

Nergens is aangegeven dat er in het veld leeftijd geen letters mogen komen, maar omdat Stripes weet dat het een Integerveld is, wordt hier automatisch op gevalideerd (ongeacht of er validatieregels zijn ingesteld). De validatie wordt alleen op de server uitgevoerd. De validatie-annotaties zijn op runtime beschikbaar, dus zou het relatief eenvoudig moeten zijn om ook validatie op de client te bouwen.

In de @Validate kunnen we ook het attribuut on gebruiken, hiermee kun je aangeven voor welke acties de validatie wel/niet moet worden uitgevoerd. Bij het wissen van records is het bijvoorbeeld niet nodig om een valide formulier te hebben. Door een actie te prefixen met een uitroepteken geef je aan dat bij deze actie de validatie juist niet uitgevoerd moet worden.

```
@Validate(required=true, on="!delete") private
String achternaam;
```

Zodra versie 1.5 uit is, is het ook mogelijk om @DontBind te gebruiken. Validatie en binding zullen dan worden overgeslagen.
De validatieregels van Stripes zijn redelijk uitgebreid, maar in de praktijk zul je toch altijd net even wat anders willen. Dit is opgelost door de @ValidateMethod-annotatie. Deze annotatie kan op een methode in de ActionBean worden gezet en wordt uitgevoerd nadat de andere validaties gedaan zijn. Standaard wordt deze methode niet uitgevoerd als er al andere validatiefouten zijn opgetreden. Wil je dit wel, dan kan het attribuut when=ValidationState.ALWAYS worden toegevoegd. Aan de validatiemethode kan je optioneel een ActionErrors-object meegeven om de errors in beeld te brengen. Het is echter mooier om dit met de methode getContext().getValidationErrors() te doen.

Het volgende stuk code voert een check uit of Pietersen niet jonger dan 60 is:

```
@ValidationMethod(when = ValidationState.ALWAYS)
public void valideer() {
    ValidationErrors errors = getContext().getVali-
dationErrors();
    Bewoner bewoner = getBewoner();

    if (bewoner.getLeeftijd() < 60 && bewoner.get-
Achternaam().equals("Pietersen")) {
      SimpleError error = new SimpleError("Pietersen
mag niet jonger dan 60 zijn");
        errors.add("bewoner.leeftijd", error);
    }
}
```

Resolutions

We kunnen niet meer om AJAX heen, elk webframework biedt wel iets aan voor Ajax. Met Stripes is het mogelijk om in plaats van een jsp een objectboom te retourneren die als JSON-object wordt aangeboden. JSON wordt veel gebruikt als alternatief voor Xml (de X in Ajax) vanwege de compacte notatie en het direct bruikbaar zijn in Javascript. Om JSON-objecten terug te geven gebruiken we in plaats van een ForwardResolution een JavaScriptResolution.

Als we de hello methode uit het eerdere voorbeeld vervangen door een JavaScriptResolution die een lijst van Employee-objecten teruggeeft dan zien we in een browser een stuk Javascriptcode terugkomen. Als we op dit stuk code een eval in Javascript doen dan is het volgende object direct in Javascript beschikbaar:

Als je verder in de JavaScriptResolution duikt zul je zien dat intern de JavaScriptBuilder gebruikt wordt om van Java Javascript te maken, deze klasse kun je ook prima met andere frameworks gebruiken om JSON op een hele makkelijke manier in je browser te krijgen.

De RedirectResolution lijkt op het eerste gezicht niet zo bijzonder. Kijken we echter in de javadoc dan vinden we de flash-methode. Als deze methode wordt aangeroepen in het return-statement dan is de meegegeven ActionBean direct beschikbaar in de pagina waarnaar geredirect wordt. Dit is erg handig omdat je normaalgesproken na een redirect een nieuw requestobject krijgt en je de data van het oude requestobject kwijt bent.

```
0   Object deptIlo=10 functionIlame=ACROBAAT id=30 name=ADRIAAN
    deptNo          10
    functionName    "ACROBAAT"
    id              30
    name            "ADRIAAN"
1   Object deptIlo=10 functionIlame=BOEF id=14 name=B2
2   Object deptIlo=10 functionIlame=CLOWN id=12 name=BASSIE
3   Object deptIlo=40 functionIlame=VELDWACHTER id=26
4   Object deptIlo=30 functionIlame=DIRECTEUR id=22
5   Object deptIlo=30 functionIlame=INDIAAN id=20 name=KLUKKLUK
6   Object deptIlo=20 functionIlame=MATROOS id=16 name=KOKKI
7   Object deptIlo=20 functionIlame=MATROOS id=10 name=PEPPI
8   Object deptIlo=30 functionIlame=CLOWN id=18 name=PIPO
9   Object deptIlo=40 functionIlame=INTERIEURVERZORGSTER id=28
10  Object deptIlo=40 functionIlame=ZWERVER id=24 name=SWIEBERTJE
```

Ten slotte is er nog de StreamingResolution. Met deze Resolution krijg je alle vrijheid, je kunt hem bijvoorbeeld gebruiken om grafieken te genereren en deze als gif terug te sturen. Bij de StreamingResolution kun je het contenttype en een eventuele bestandsnaam opgeven (voor het geval er een opslaan als dialoog in de browser moet komen.

Integratie met andere frameworks

Een belangrijk aspect van een framework is de samenwerking met bestaande frameworks. Meestal werken verschillende frameworks redelijk samen, maar het is altijd fijn als twee losse frameworks in combinatie meer waard zijn dan hun som. Stripes biedt extra's in combinatie met Spring, Hibernate en EJB3.

Om Stripes te laten samenwerken met Spring is het nodig om in de web.xml bij de init-params aan te geven dat je gebruik wilt maken van de SpringInterceptor (meegeleverd met Stripes). Zodra er een SpringInterceptor is kun je gebruik maken van de @SpringBean-annotatie. In de waarde van de annotatie geef je de naam van de Spring bean op die je wilt injecteren. Het is zelfs mogelijk de waarde weg te laten, in dit geval zorgt Stripes er via auto-wire by name/type voor dat de juiste bean wordt geïnjecteerd.

```
@SpringBean("locationService")
public void setLocationService(LocationService
ls) {
    this.ls  = ls;
}
```

De integratie met EJB3 gaat op een vergelijkbare manier. De eerste stap is een init-param toevoegen en daarna kun je gebruik maken van de @EJBBean annotatie.

Ten slotte is er nog Stripernate die de integratie met Hibernate makkelijker maakt. Met Stripernate kunnen domeinobjecten voor Stripes direct in Hibernate worden gebruikt en zullen validatiefouten van Hibernate aan Stripes worden doorgegeven. Stripernate is ontwikkeld door Aaron Porter en is geen onderdeel van Stripes.

Conclusie

Met Stripes heb je een vrij compleet framework in huis waarmee je erg snel en simpel applicaties kunt ontwikkelen. Stripes biedt mogelijkheden om controllers te maken, serverside validatie, eenvoudige templating, integratie met andere frameworks en een aantal classes waarmee unittesten van Stripes mogelijk is.
Bij het ontwerp van Stripes is niet alleen rekening gehouden met de bruikbaarheid van het framework zelf, maar ook met de uitbreidbaarheid. Op dit moment integreert Stripes met populaire frameworks als Spring, Hibernate en EJB. Hierdoor is de overstap naar Stripes gemakkelijk te maken en zul je veel code kunnen hergebruiken.

Er is door de makers van Stripes goed gekeken naar de nadelen en omslachtige elementen van andere frameworks. Het controllergedeelte levert duidelijke code op waarbij ongebruikte parameters tot een minimum worden beperkt.
Stripes maakt gebruik van configuration by exception. Dit houdt in dat er pas configuratie nodig is zodra je afwijkt van de standaard. Code die bij elkaar hoort zul je op één plek aantreffen. Validatie, url-mappings en controllercode staan bij elkaar in één klasse.
Dankzij deze vereenvoudigingen is het erg duidelijk wat er van request tot response gebeurt.

Stripes is geschikt voor de wat simpelere applicaties met een eenvoudige navigatiestructuur en waarbij ingewikkelde zaken als bijvoorbeeld het componentgebaseerd ontwikkelen van JSF of de conversaties van Seam niet nodig zijn. Stripes kan ook een goed framework zijn bij applicaties waar veelvuldig gebruik wordt gemaakt van Ajax. Bij dit soort applicaties wil je vaak volledige controle over wat er op de server gebeurt en deze controle heb je met Stripes.

Stripes is snel te leren en er zijn inmiddels plugins voor Netbeans en IntelliJ beschikbaar.
Op de site van Stripes is erg veel informatie te vinden, vaak voorzien van duidelijke voorbeelden. Een boek over Stripes (Stripes: ... and Java web development is fun again) verkeert in de afrondende fase. Het zal rond half oktober verkrijgbaar zijn. «

Links

Homepage van Stripes: http://www.stripeframework.org
Stripernate: http://www.mongus.com/Topics/Web%20Development/Stripernate/
Artikel over Stripernate: http://www.theserverside.com/tt/articles/article.tss?l=Stripernate
Boek over Stripes: http://www.pragprog.com/titles/fdstr/stripes