JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

DEZSYS-Z3 "SERVLET TUTORIAL"

Systemtechnik Labor
4CHIT 2015/16

Otahal Maro

Version 1

Note:

Betreuer: Borko

Begonnen am 15. April 2016

Beendet am 22. April 2016

Inhaltsverzeichnis

1 Einführung

Webframeworks helfen Entwickler schnell und einfach wiederkehrende Funktionalitäten nicht immer wieder neu programmieren zu müssen. Dabei gibt es in der Java Umgebung gut dokumentierte und weitverbreitete Umsetzungen wie zum Beispiel die JavaEE 7 und Spring Plattform.

Ziele

Diese Übung soll Einblick in die Verwendung der Plattformen geben. Dabei soll ein Tutorial der gewünschten Umgebung umgesetzt, dokumentiert und anschließend bewertet werden.

1.1 Voraussetzungen

- Grundlagen Java
- Grundlagen zu verteilten Systemen und HTTP
- Grundlegendes Verständnis von Webframeworks

1.2 Aufgabenstellung

Wählen Sie aus den jeweiligen Paketen eine Komponente aus und dokumentieren Sie die Umsetzung, Ausführung und Ihre subjektive Meinung zu den verwendeten Tools und Umgebungen.

IDEs: IntelliJ | eclipseEE | Netbeans

ApplicationServer: GlassFish | WildFly | Tomcat

Servlet Tutorial für Framework: JavaEE7 | Spring

Dokumentieren Sie etwaige Schwachstellen bzw. angegebene Einschränkungen. Sie können natürlich auch mögliche Erweiterungen einbauen. Nur ein einfachstes HelloWorld-Example sollte nur als Startpunkt genommen werden.

1.3 Quellen

"Java EE 7 Essentials"; A.Gupta; Aug 2013; O'Reilly Verlag; online: http://shop.oreilly.com/product/0636920030614.do"

Getting Started with Web Applications"online: https://docs.oracle.com/javaee/7/tutorial/webapp.htm#BNADR"

Serving Web Content with Spring MVC" online: http://spring.io/guides/gs/serving-web-content/"

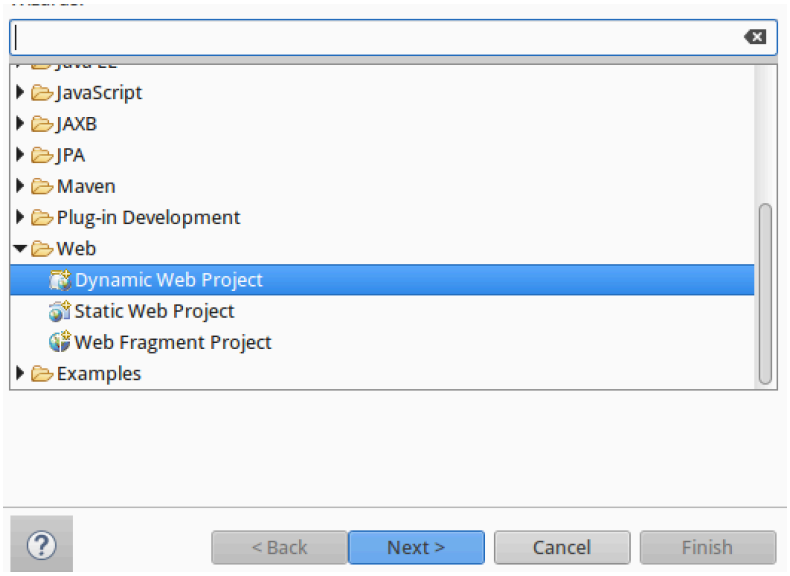
2 Ergebnisse

Für diese Aufgabenstellung wird Eclipse verwendet und ich habe das Crunchify Tutorial http://crunchify.com/simplest-spring-mvc-hello-world-example-tutorial-spring-model-view-controller-tips/ durchgeführt, welches bei mir einwandfrei funtioniert hat

Schritt 1

Zuerst wird ein Projekt erstellt, welches ein Dynamisches Webproject sein muss.

```
New -> Project -> Dynamic Web Project
```

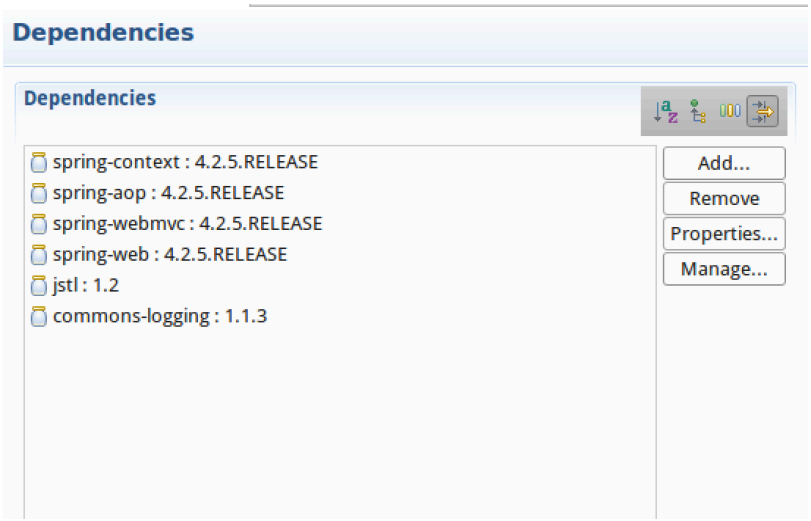


Schritt 2

Darauffolgend wird das Projekt in ein Maven Projekt konvertiert, um alle Spring MVC Abhängigkeiten in das Projekt zu konvertieren.

```
Rechtsklick -> Configure -> Convert to Maven Project
```

Zunächst wird das pom.xml geöffnet und die jar dependencies hinzugefügt.



Danach wird eine Spring Configuratuin `Bean` file unter `/WebContent/WEB-INF/crunchify-servlet.xml` erstellt.

Der Source Code sieht wie gefolgt aus:

```
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

	<context:component-scan base-package="com.crunchify.controller" />

	<bean id="viewResolver"
		class="org.springframework.web.servlet.view.UrlBasedViewResolver">
		<property name="viewClass"
			value="org.springframework.web.servlet.view.JstlView" />
		<property name="prefix" value="/WEB-INF/jsp/" />
		<property name="suffix" value=".jsp" />
	</bean>

</beans>
```

Das `context:component-scan` erlaubt Spring alle Komponenten vom Package `com.crunchify.controller` zu laden.

Schritt 3

Des Weiteren wird ein neues `web.xml` File unter `/WebContent/WEB-INF/web.xml` erstellt.

Der Source-Code sieht wie gefolgt aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
 <display-name>CrunchifySpringMVCTutorial</display-name>
 <welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>

  <servlet>
    <servlet-name>crunchify</servlet-name>
    <servlet-class>
       org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>crunchify</servlet-name>
    <url-pattern>/welcome.jsp</url-pattern>
    <url-pattern>/welcome.html</url-pattern>
    <url-pattern>*.html</url-pattern>
  </servlet-mapping>

</web-app>
```

Wenn der Dispatcher-Servlet Server initialisiert wurde, sucht er nach einem `-servlet.xml` File, im WEB-INF Ordner. In diesem Fall sucht er nach dem `crunchify-servlet.xml` File.

Schritt 4

Danach wird eine neue Controller Klasse unter dem Package `com.crunchify.controller` erstellt.

Der File-Name ist: `CrunchifyHelloWorld.java`

Der Source-Code sieht wie gefolgt aus:

```
package com.crunchify.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

/*
 * author: Crunchify.com
 *
 */

@Controller
public class CrunchifyHelloWorld {
```

```
	@RequestMapping("/welcome")
	public ModelAndView helloWorld() {

		String message = "<br><div style='text-align:center;'>"
				+ "<h3>********** Hello World, Spring MVC Tutorial</h3>This message is coming
from CrunchifyHelloWorld.java **********</div><br><br>";
		return new ModelAndView("welcome", "message", message);
	}
}
```

Wenn Spring das Package scannt, wird es als ein Controller Bean erkannt, um die Anforderungen zu verarbeiten.
Das @Requestmapping sagt Spring das dieser Controller alle Anforderungen mit /welcome im URL Pfad verarbeiten soll.
Die helloWorld() Methode redturnt ModelandView.

Darauffolgend wird noch ein `.jsp` File `/WebContent/index.jsp` erstellt, welches wie gefolgt aussieht.

```
<html>
<head>
<title>Spring MVC Tutorial Series by Crunchify.com</title>
<style type="text/css">
body {
	background-image: url('http://crunchify.com/bg.png');
}
</style>
</head>
<body>
	<br>
	<div style="text-align:center">
		<h2>
			Hey You..!! This is your 1st Spring MCV Tutorial..<br> <br>
		</h2>
		<h3>
			<a href="welcome.html">Click here to See Welcome Message... </a>(to
			check Spring MVC Controller... @RequestMapping("/welcome"))
		</h3>
	</div>
</body>
</html>
```

Und dann noch das `welcome.jsp`

```
<html>
<head>
<title>Spring MVC Tutorial by Crunchify - Hello World Spring MVC
	Example</title>
<style type="text/css">
body {
	background-image: url('http://crunchify.com/bg.png');
}
</style>
</head>
<body>${message}

	<br>
	<br>
	<div style="font-family: verdana; padding: 10px; border-radius: 10px; font-size: 12px; text-align:center;">

		Spring MCV Tutorial by <a href="http://crunchify.com">Crunchify</a>.
		Click <a
			href="http://crunchify.com/category/java-web-development-tutorial/"
```

```
                target="_blank">here</a> for all Java and <a
                href='http://crunchify.com/category/spring-mvc/' target='_blank'>here</a>
            for all Spring MVC, Web Development examples.<br>
        </div>
</body>
</html>
```

Schritt 5

Das Projekt wird als Maven build ausgeführt.

Rechtsklick -> Run as -> Maven Build

```
[INFO] Installing /home/user/workspace/CrunchifySpringMVCTutorial/target/CrunchifySpr
[INFO] Installing /home/user/workspace/CrunchifySpringMVCTutorial/pom.xml to /home/us
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4.323 s
[INFO] Finished at: 2016-04-21T19:41:48+02:00
[INFO] Final Memory: 16M/39M
```

➔ Build Succes

Und anschließend der Tomcat Server gestartet.

Rechtsklick -> Start

Schlussendlich wird die URL aufgerufen:
http://localhost:8080/CrunchifySpringMVCTutorial/

Und die Website sollte dies Anzeigen

3 Eclipse

Das Eclipse Projekt ist ein Open Source Projekt, das sich der Entwicklung einer robusten und umfassenden Entwicklungsplattform verschrieben hat, die für den kommerziellen Einsatz tauglich ist. Das Grundgerüst Projektes bildet die Eclipse Plattform, die eine erweiterbare auf Java basierende IDE darstellt. Sie zeichnet sich neben ihren eigenen Features durch einen starken Integrationscharakter aus, der das Einbinden externer Tools ermöglicht.

Mir persönlich gefällt inteliJ besser als Eclipse, da es auf meinem Betriebssystem flüssiger läuft und mir die Oberfläche besser gefällt. Jedoch ist inteliJ kostenpflichtig, bzw. für Schüler 1 Jahr gratis.

[1] http://www.oio.de/public/warum-eclipse.htm

[2] GIT: https://github.com/motahal/DEZSYS-Z3-SERVLET-TUTORIAL-.git

Zeitaufzeichnung

Aufgabe	**Dauer**
Tutorial	2 h
Eclipse Recherchieren	1 h
Protokoll	1 h
Gesamt:	**4 h**