Umstellung der WebApp auf Derby

Apache Ivy / Artifactory

Apache Ivy ist eine Erweiterung von dem Build-Management-Tool Apache Ant und ermöglicht das dynamische Nachladen und Einbinden von abhöngigen Java-Archiven (JAR's), während des Build-Prozesses. Diese Java Archive können aus privaten oder im Internet frei zur Verfügung stehenden Bibliotheken (sog. Repositories) bezogen werden. Wir verwenden im Rahmen von Timadorus das frei verfügbare Repository-Verwaltungssystem „Artifactory". In Artifactory ist es möglich verschiedene JAR's bereit zu stellen und deren Abhängigkeiten anzugeben, sodass in der Anwendung nur noch die eigentlich benötigte JAR angegeben werden muss und alle weiteren für dieses Archiv benötigten JAR's automatisch mit herunter geladen werden. Zum konfigurieren von Ivy bzw. dem einbinden der Artifactory JAR's benötigen wir zwei Dateien (ivysettings.xml und ivy.xml).

ivy/ivysettings.xml

In dieser Datei werden allgemeine Einstellungen definiert. Wir machen damit bekannt welches Repository von Ivy verwendet werden soll und wir sich Ivy innerhalb von Artifactory zurecht findet.

```
<ivysettings>

	<!-- Es werden Properties aus einer externen Datei geladen (falls
	diese vorhanden ist -->
	<properties file="${ivy.settings.dir}/ivysettings-file.properties" />

	<!-- Der Ort des Repositories wird bekannt gemacht -->
	<property name="org.repo.url"
 value="http://developer.timadorus.org/artifactory" />

	<!-- Es wird definiert welches Verfahren zur Auflösung der
	Abhängigkeiten verwendet werden soll (Werden später in dieser Datei
	definiert) -->
	<settings defaultResolver="ivy-and-maven" />

	<!-- Der Ordner in welchem Ivy seinen Cache einrichten soll wird
	definiert -->
	<caches defaultCacheDir="${ivy.cache.dir}" />

	<!-- Verfahren zur Auflösung der Abhängigkeiten -->
	<resolvers>

		<!-- Hier wird definiert wie die Abhängigkeiten von Ivy
	aufgelöst werden sollen -->
		<chain name="ivy-and-maven" returnFirst="true">

			<!-- Gibt an wie mit einem Maven „classifier" umgegangen
		werden soll -->
			<ibiblio name="timadorus-repo-maven"
		root="${org.repo.url}/remote-repos" m2compatible="true"
		pattern="[organisation]/[module]/[revision]/[artifact]-
		[revision](-[classifier]).[ext]" />

			<!-- Definiert Pfade damit Ivy sich innerhalb von
		Artifactory zurecht findet -->
			<url name="timadorus-repo" m2compatible="true"
		checkmodified="true">
```

```
                        <!-- Pfad zur Ivy.xml des gesuchten Archivs -->
                        <ivy
                    pattern="${org.repo.url}/repo/[organisation]/[module]/[re
                    vision]/ivy.xml" />

                        <!-- Pfad zum gesuchten Archiv -->
                        <artifact
                    pattern="${org.repo.url}/repo/[organisation]/[module]/[re
                    vision]/[artifact]-[revision].[ext]" />

                    </url>
            </chain>
      </resolvers>
</ivysettings>
```

ivy.xml

In dieser Datei definieren wir welche Version von Ivy wir verwenden. Es wird angeben für welche Organisation und welches Modul Ivy die Abhängigkeiten auflösen soll. Außerdem beschreiben wir hier welche Abhängigkeiten bestehen und aufgelöst werden sollen.

```
<ivy-module version="2.0">

      <!-- Name der Organisation und des Moduls -->
      <info organisation="timadorus" module="webapp" />

      <!-- Hier beschreiben wir die Abhängigkeiten -->
      <dependencies>

            <!-- GWT -->
            <dependency org="com.google" name="gwt" rev="2.0.4" />

            <!-- DataNucleus -->
            <dependency org="org.datanucleus" name="datanucleus-enhancer"
      rev="2.1.2" conf="*->default" />
            <dependency org="org.datanucleus" name="datanucleus-rdbms"
      rev="2.1.2" conf="*->default,provided"/>

            <!—Apache Derby -->
            <dependency org="org.apache.derby" name="derby" rev="10.6.2.1"
      conf="*->default" />
            <dependency org="org.apache.derby" name="derbyclient"
      rev="10.6.2.1" conf="*->default" />
            <dependency org="org.apache.derby" name="derbynet"
      rev="10.6.2.1" conf="*->default" />
            <dependency org="org.apache.derby" name="derbytools"
      rev="10.6.2.1" conf="*->default" />
      </dependencies>
</ivy-module>
```

Build-Prozess

Der Build-Prozess in unseren Projekten relaisieren wir mittels Apache Ant. Damit ist es möglich den Build soweit zu automatisieren, dass Tests (jUnit, Checkstyle, Selenium, …), Kompiliervorgang und Auflösen der Abhängigkeiten in einer festgelegten Reihenfolge ausgeführt werden. Dazu ist es nötig die auszuführenden Vorgänge in einer Datei zu definieren.

build.xml

In dieser Datei werden eben jene Vorgänge definiert, die während des automatischen Builds in welcher Reihenfolge ausgeführt werden sollen.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project basedir="." default="build" name="TimadorusWebApp"
xmlns:ivy="antlib:org.apache.ivy.ant">

	<!-- Beschreibt die Umgebung für die diese Datei verwendet wird -->
	<property environment="env" />

	<!-- Angaben zum Debuglevel -->
	<property name="debuglevel" value="source,lines,vars" />

	<!-- Pfad zu den Java Quelldateien -->
	<property name="sourceDir" value="src" />

	<!-- Pfad zum Zielordner der WebApp -->
	<property name="to" value="war/WEB-INF" />

	<!-- Pfad zum Zielordner der Klassenkompilierung der WebApp -->
	<property name="destination" value="${to}/classes" />

	<!-- Pfad zu dem Ordner in welchen die JAR's kopiert werden -->
	<property name="destinationLib" value="${to}/lib" />

	<!-- Angaben zur Version -->
	<property name="target" value="1.6" />
	<property name="source" value="1.6" />

	<!-- Pfad zum Ivy Arbeitsverzeichnis -->
	<property name="ivy.dir" value="${basedir}/ivy" />

	<!-- Pfad zur Datei mit den Ivy Einstellungen -->
	<property name="ivy.settings.file" value="${ivy.dir}/ivysettings.xml"
/>

	<!-- Pfad zum Cache von Ivy -->
	<property name="ivy.cache.dir" value="${ivy.dir}/cache" />


	<!-- Definition der im Classpath enthaltenen Dateien -->
	<path id="TimadorusWebApp.classpath">

		<!-- Quell und Ziel Java-Dateien -->
		<pathelement location="${destination}" />
		<pathelement location="${sourceDir}" />

		<!-- Alle JAR's aus dem Lib-Ordner des WebApp -->
		<fileset dir="${destinationLib}">
			<include name="*.jar" />
		</fileset>
	</path>
```

```
	<!-- Task:resolve | Löse Abhängigkeiten mit Ivy auf -->
	<target name="resolve" description="-> handle dependencies with ivy">
		<!-- Cache von Ivy löschen um Neues herunterladen zu erzwingen
	-->
		<!--ivy:cleancache /-->

		<!-- Ivy Einstellungen einbinden -->
		<ivy:settings file="${ivy.settings.file}" />

		<!-- Abhängigkeiten auflösen -->
		<ivy:retrieve />
	</target>


	<!-- Task:clean | Löschen von Zielordnern -->
	<target name="clean">
		<delete dir="${destination}" />
		<delete dir="${destinationLib}" failonerror="false" />
	</target>


	<!-- Task:init | Erstellen von Zielordnern, Umkopieren von Dateien
und Libraries -->
	<target name="init">

		<!-- Erstellen eines Ordners -->
		<mkdir dir="${destination}" />

		<!-- Umkopieren von Einstellungsdateien aus src/ -->
		<copy includeemptydirs="false" todir="${destination}">
			<fileset dir="src">
				<exclude name="**/*.launch" />
				<exclude name="**/*.java" />
			</fileset>
		</copy>

		<mkdir dir="${destinationLib}" />

		<!-- Umkopieren der JAR's -->
		<copy includeemptydirs="false" todir="${destinationLib}">
			<fileset dir="lib" />
		</copy>
	</target>


	<!-- Task:java-compile | Kompilieren der Javaklassen (führt vorher
die Tasks clean und init aus) -->
	<target name="java-compile" depends="clean,init">

		<!-- Aufruf zum Kompilieren -->
		<javac debug="true" debuglevel="${debuglevel}"
destdir="${destination}" source="${source}" target="${target}">

			<!-- Quellordner -->
			<src path="src" />

			<!-- Zugehöriger Classpath (Definition siehe oben) -->
			<classpath refid="TimadorusWebApp.classpath" />
		</javac>
	</target>
```

```
	<!-- Task:enhance | Führt ein Byte-Enhancement auf die kompilierten
Javaklassen aus -->
	<target name="enhance" description="DataNucleus enhancement">

		<!-- Gib den Enhancement-Task bekannt -->
		<taskdef name="datanucleusenhancer"
classpathref="TimadorusWebApp.classpath"
classname="org.datanucleus.enhancer.tools.EnhancerTask" />

		<!-- Führt die Enhancement-Task aus -->
		<datanucleusenhancer failonerror="true" verbose="true">

			<!-- Pfad zu den Klassen die enhanced werden sollen -->
			<fileset dir="${destination}">
				<include name="**/*.class" />
			</fileset>
			<classpath refid="TimadorusWebApp.classpath" />
		</datanucleusenhancer>
	</target>


	<!-- Task:gwt-compile | Kompiliert die Javaklassen in Javascript-Code
-->
	<target name="gwt-compile">

		<!-- Ruft das Makro auf in welchem das Kompilieren definiert
	ist -->
		<gwtCompile module="org.timadorus.webapp.TimadorusWebApp"
classpathref="TimadorusWebApp.classpath" />
	</target>

	<!-- Makro:gwtCompile | Definiert wie die Klassen mit GWT kompiliert
werden sollen -->
	<macrodef name="gwtCompile">

		<!-- Parameter die beim Aufruf angegeben werden -->
		<attribute name="module" />
		<attribute name="classpathref"
default="TimadorusWebApp.classpath" />

		<!-- Sequenz für die Ausführung des Makros -->
		<sequential>

			<!-- Ruft den GWT-Compiler unter Angabe des Entrypoints
auf -->
			<java classpathref="@{classpathref}"
classname="com.google.gwt.dev.Compiler" fork="true">
				<arg value="@{module}" />
			</java>
		</sequential>
	</macrodef>


	<!-- Task:build | Definiert den Ablauf der Taskaufrufe für den Build
-->
	<target name="build" depends="resolve,java-compile,enhance,gwt-
compile" />
</project>
```

Quellen:

http://de.wikipedia.org/wiki/Apache_Ivy [Zugriff: 2010-11-28 20:45]

http://www.jfrog.org/products.php [Zugriff: 2010-11-28 21:42]