Übungen zum Seminar

Anwendungsentwicklung mit dem Spring Framework

Inhaltsverzeichnis

Einleitung 3
Vorbereitung 4
Entwicklungsumgebung einrichten 5
Benötigte Software 5
Installation von Eclipse Plug-ins 6
SpringSource Tool Suite (Spring IDE Plug-in) Installation 6
Projekt einrichten 11
Manuelle Konfiguration 11
Übungen 12
Übung 0 – HelloWorld mit dem Spring Framework 13
Voraussetzungen 13
Abhängigkeiten 13
Ziel 13
Aufgaben 13
Übung 1 – IoC und DI 14
Voraussetzungen 14
Abhängigkeiten 14
Dauer 14
Ziel 14
Aufgaben 14
Übung 2 – DI mit Autowiring 16
Voraussetzungen 16
Abhängigkeiten 16
Dauer 16
Ziel 16
Aufgaben 16
Hinweis 17
Zusatzaufgaben 17
Annotationen 17
JavaConfig 17
Übung 3 – Spring AOP 19
Voraussetzungen 19
Abhängigkeiten 19
Dauer 19
Ziel 19
Aufgaben 19
Hinweis 20
Zusatzaufgaben 20
Übung 4 – Spring DAO Support 21
Voraussetzungen 21
Abhängigkeiten 21
Dauer 21

Ziel 21
Aufgaben 21
Hinweis 23
Zusatzaufgaben 23
Übung 5 - Transaktionssteuerung 25
Voraussetzungen 25
Abhängigkeiten 25
Dauer 25
Ziel 25
Aufgaben 25
Hinweise 26
Zusatzaufgaben 26
Übung 6 – Spring MVC 27
Voraussetzungen 27
Abhängigkeiten 27
Dauer 27
Ziel 27
Aufgaben 27
Übung 7 – Spring Remoting 31
Voraussetzungen 31
Abhängigkeiten 31
Dauer 31
Ziel 31
Aufgaben 31
Hinweis 32
Zusatzaufgaben 32
Übung 8 – RESTful Web Service mit Spring MVC 33
Voraussetzungen 33
Abhängigkeiten 33
Dauer 33
Ziel 33
Aufgaben 33
Übung 9 – Web-Anwendung mit Spring Security absichern 34
Voraussetzungen 34
Abhängigkeiten 34
Dauer 34
Ziel 34
Aufgaben 34
Abschlusstest 35
Anhang A 36
Maven und das Eclipse Plugin m2eclipse 37
m2eclipse Plug-in Installation – optional 37
Konfiguration mit dem m2eclipse Plugin 41
Vorbereiten des Plugins 41
Maven aus Eclipse mit m2eclipse Plugin starten 43
Spring Framework und Maven 45
Kontakt & Lizenzbedingungen 46

Einleitung

Bitte notieren Sie sich zu Beginn für die Übungen die Pfade zu folgenden Verzeichnissen:

- Netzlaufwerk auf dem die Software abgelegt ist:

  _____ [DOWNLOAD_PATH]
  Wenn auf dieses Verzeichnis im folgenden verwiesen wird, dann ist dies durch [DOWNLOAD_PATH] gekennzeichnet. Fragen Sie bitte den Dozenten nach dem Ordner auf dem Netzlaufwerk.
- Lokales Verzeichnis auf dem Schulungsrechner, wo die Software installiert ist:

  _____ [INSTALL_PATH]
  (Default: C:/schulung/software).
- Und das lokale Verzeichnis in dem sich der Eclipse Workspace befindet:

  _____ [WORKSPACE_PATH]
  (Default: C:/schulung/workspace).
- Der Quellcode für alle Übungen und Beispiele liegt auf github:
  https://github.com/tfr42/spring-course-examples
- Entweder das git Repository lokal clonen mit:
  `git clone https://github.com/tfr42/spring-course-examples.git`
  oder Download als ZIP-Datei vom github Repository und importieren aller Projekte als „Maven-Project“ über den Eclipse Import-Wizard.

Hinweis:
In den folgenden Screenshots sind teilweise Pfade zu sehen, die nicht mit denen der Installation auf Ihrem Schulungsrechner übereinstimmen. Bitte verwenden Sie bei Abweichungen die entsprechenden Pfade wie oben von Ihnen eingetragen.

Vorbereitung

Überprüfen Sie, ob alle benötigten Softwarepakete auf dem Schulungsrechner korrekt und vollständig im Verzeichnis [INSTALL_PATH] installiert sind.

Sollte eine Softwarekomponente fehlen, müssen Sie diese nachinstallieren. Bevor Sie mit der Installation der Softwarepakte beginnen, prüfen Sie, ob Sie auf dem Schulungsrechner lokale Administrationsrechte haben.

Die Installationsdateien für die einzelnen Softwarepakete finden Sie im Verzeichnis [DOWNLOAD_PATH]/Software.

Für die Installation von Eclipse Plugins über den Eclipse Update Manager ist eine Internet-Verbindung notwendig. Prüfen Sie die Einstellungen unter den Eclipse „Preferences > General > Network Connections“.

Entwicklungsumgebung einrichten

Benötigte Software

- Java SE 8[1]
- Installation von **Eclipse 4.6** (Eclipse IDE for Java EE Developers - Neon[2]) incl. der Plug-ins:
  - **Spring IDE 3.8.2**[3]
  - m2eclipe Plugin 1.7.0[4] (optional)
- **Spring Framework 4.3.4.RELEASE**[5]
- Apache Tomcat 7.x/8.x[6]
- GlassFish v3.0.1[7] (optional)
- Ant 1.8.1[8] (optional)
- Maven 3.0.5[9] (optional)
- SpringSource Tool Suite (Developer Edition) 3.8.4[10] (optional)

1 Oracle JDK - http://java.sun.com/javase/downloads/index.jsp
2 Eclipse IDE - http://www.eclipse.org/downloads/
3 Spring STS Update Site - https://spring.io/tools/sts/all
4 Eclipse m2eclipe - http://eclipse.org/m2e/
5 Spring Framework - http://projects.spring.io/spring-framework/
6 Apache Tomcat – http://tomcat.apache.org
7 Oracle GlassFish - https://glassfish.dev.java.net/
8 Apache Ant - http://ant.apache.org/bindownload.cgi
9 Apache Maven - http://maven.apache.org/download.html
10 SpringSource Tool Suite - https://spring.io/tools/sts/all

Installation von Eclipse Plug-ins

Die Installation von Eclipse Plug-ins wird über den Update-Manager vorgenommen.

SpringSource Tool Suite (Spring IDE Plug-in) Installation

Installationsanleitung für die STS: http://download.springsource.com/release/STS/doc/STS-installation_instructions-2.5.0.M3.pdf

Update-Site (STS):

- Eclipse 4.6 (Neon) - http://dist.springsource.com/release/TOOLS/update/e4.6
- Eclipse 4.5 (Mars) - http://dist.springsource.com/release/TOOLS/update/e4.5
- Eclipse 4.4 (Luna) - http://dist.springsource.com/release/TOOLS/update/e4.4
- Eclipse 4.3 (Kepler) - http://dist.springsource.com/release/TOOLS/update/e4.3
- Eclipse 3.8/4.2 (Juno) - http://dist.springsource.com/release/TOOLS/update/e3.8
- Eclipse 3.7 (Indigo) - http://dist.springsource.com/release/TOOLS/update/e3.7
- Eclipse 3.6 (Helios) - http://dist.springsource.com/release/TOOLS/update/e3.6
- Eclipse 3.5 (Galileo) - http://dist.springsource.com/release/TOOLS/update/e3.5

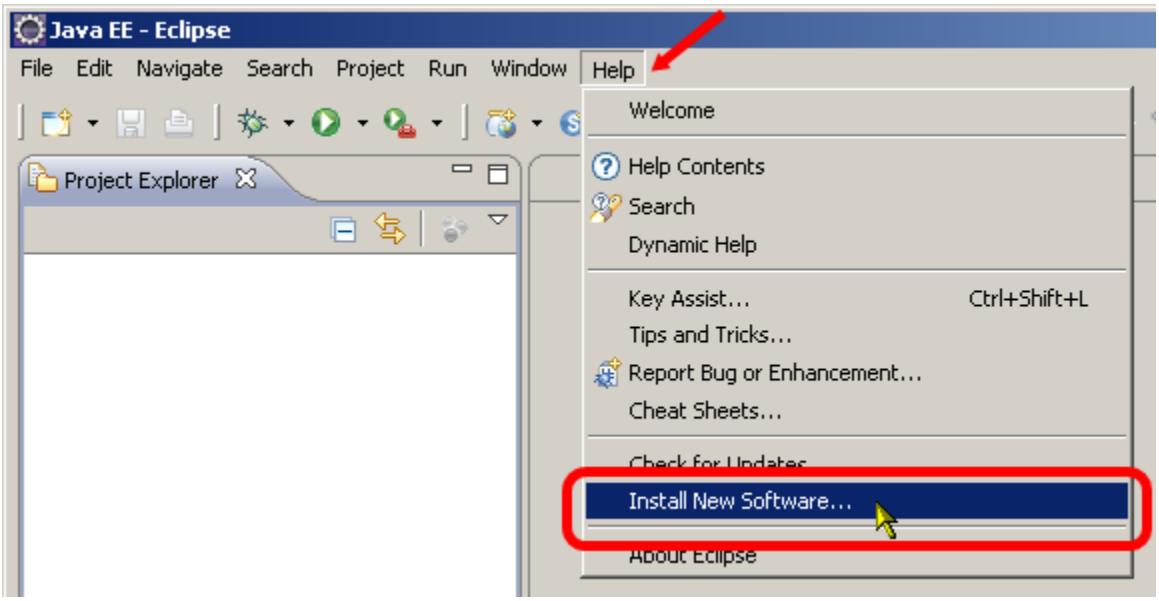

Abb.1.: Neue Software installieren

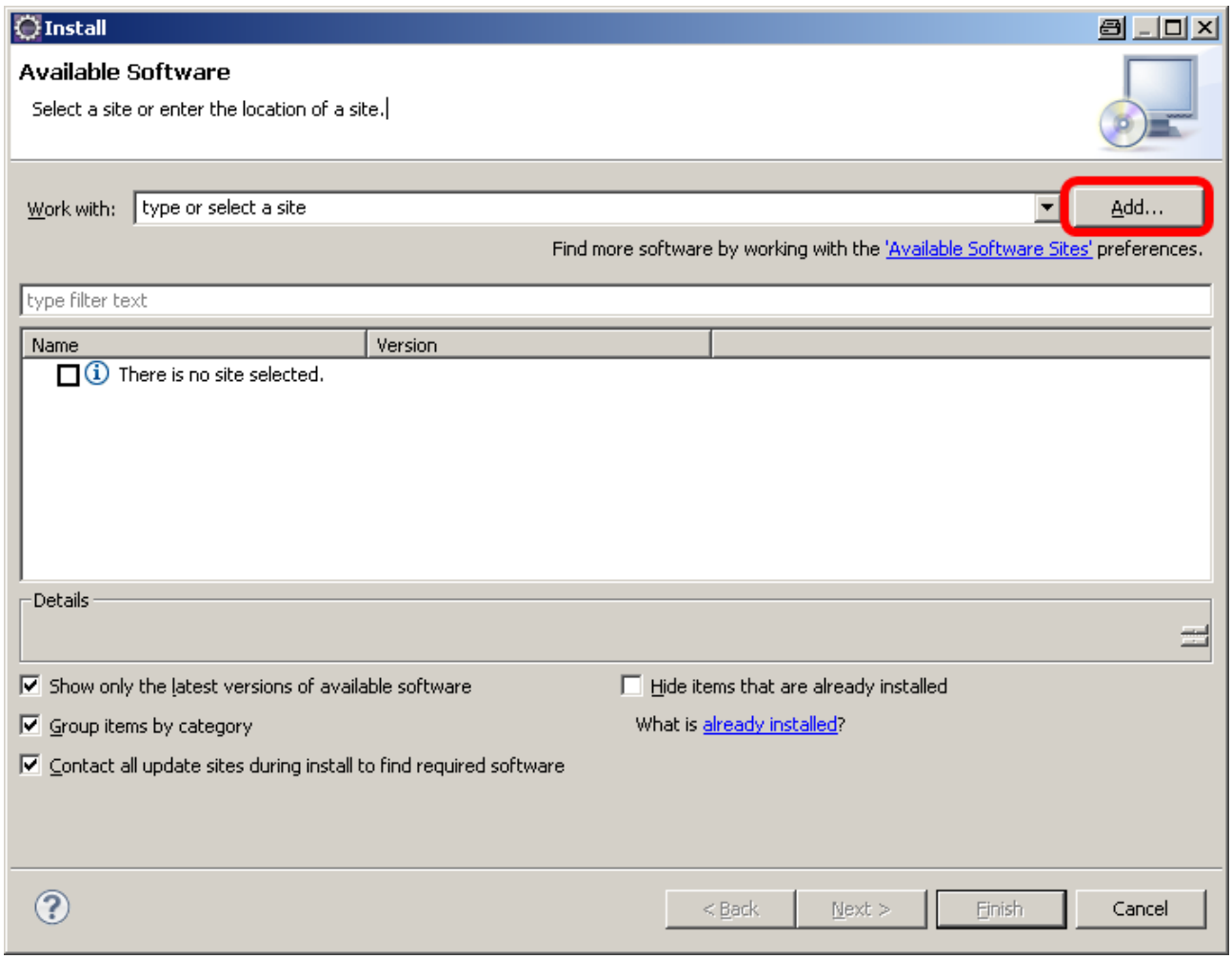

Abb.2.: Neue Update-Site hinzufügen

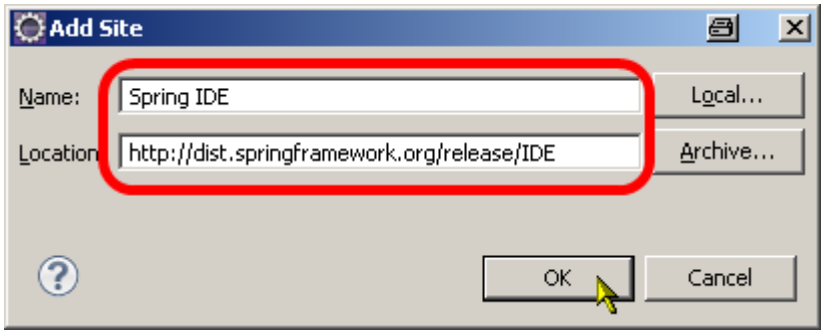

Abb.3.: URL für Spring IDE eingeben (je nach Eclipse Version die passende Update-Site!)

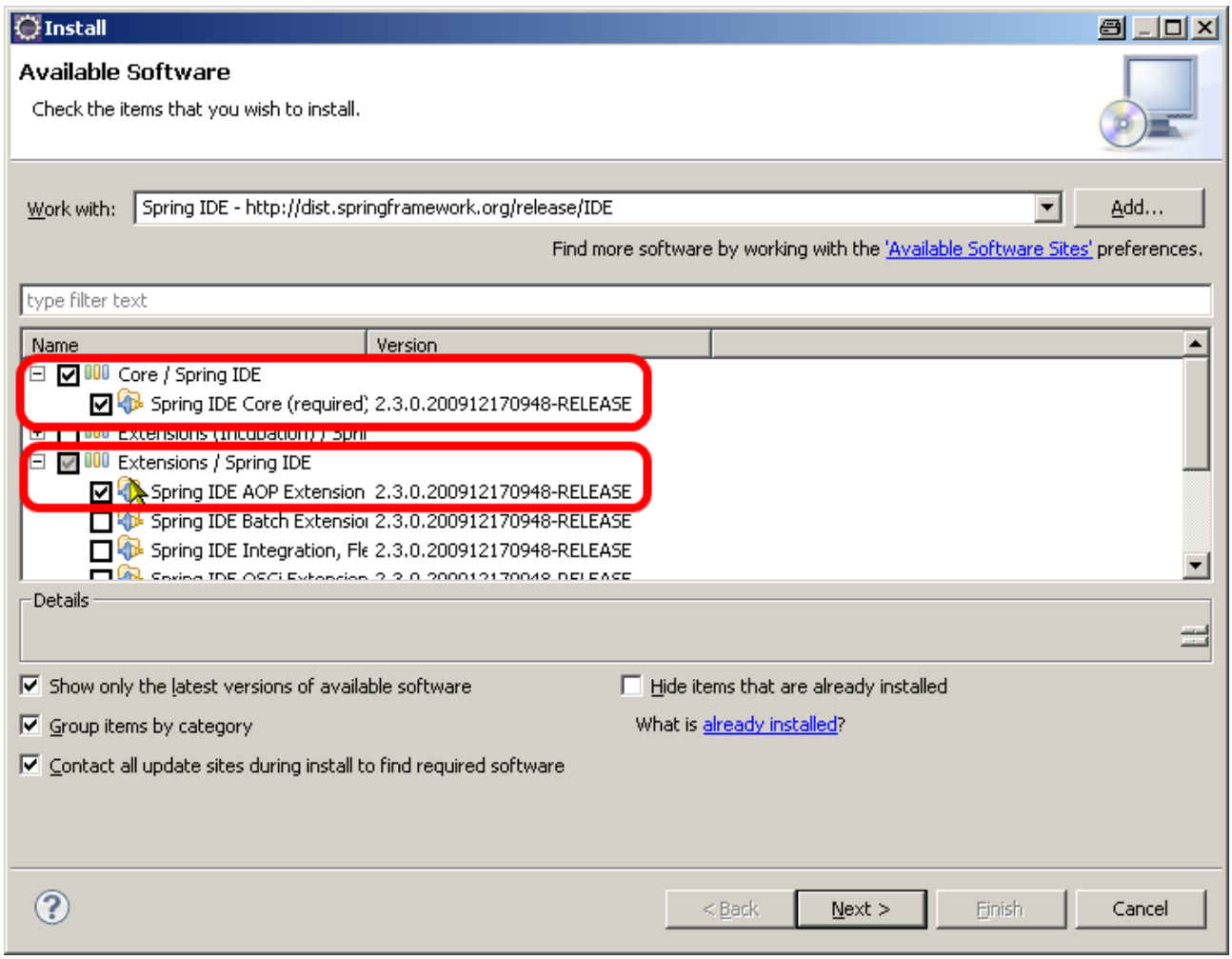

Abb.4.: Spring IDE Core auswählen

Alle folgenden Dialoge mit „Next“ bestätigen und am Ende die IDE mit „YES“ neu-starten:

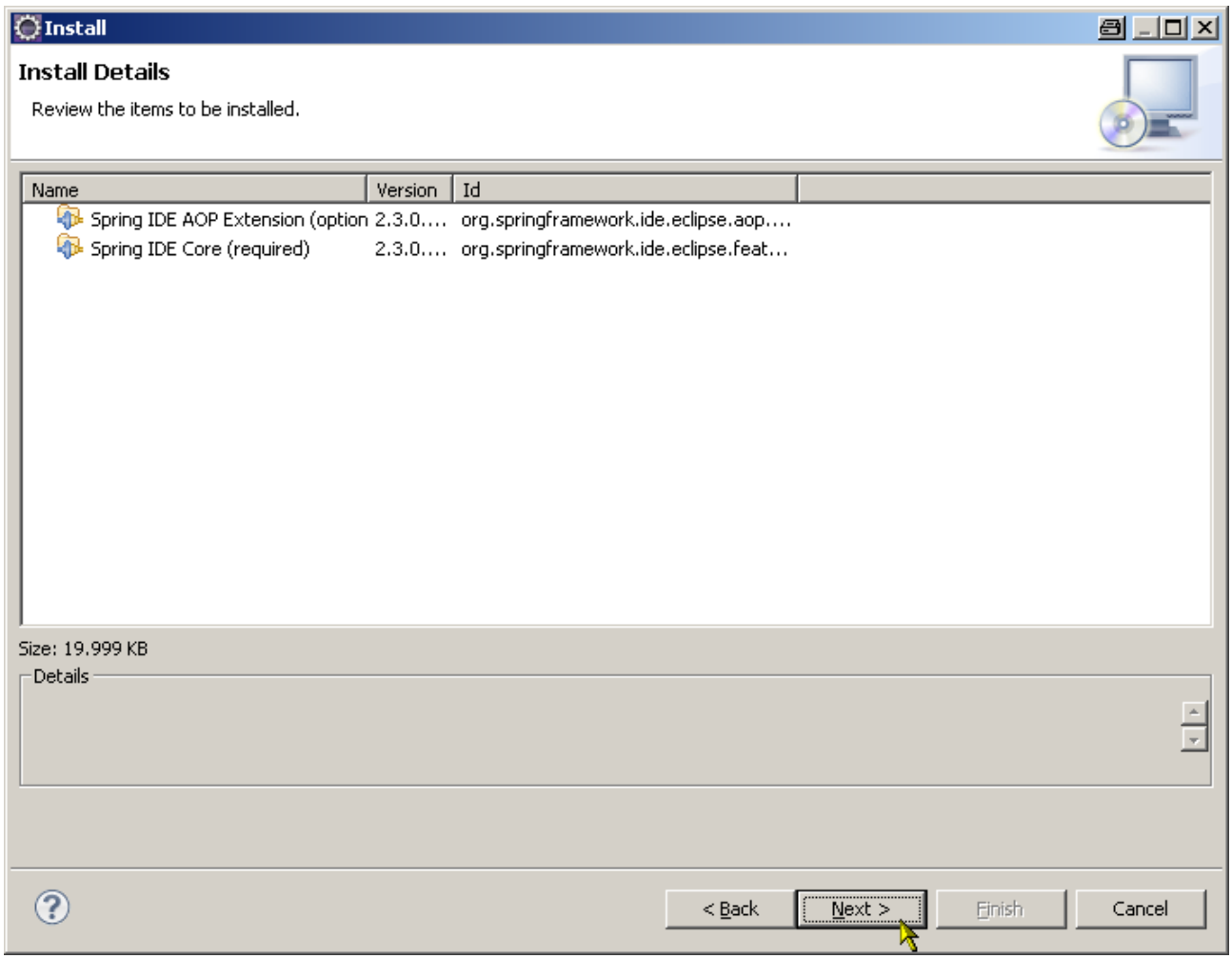

Abb.5.: Auswahl mit Next bestätigen

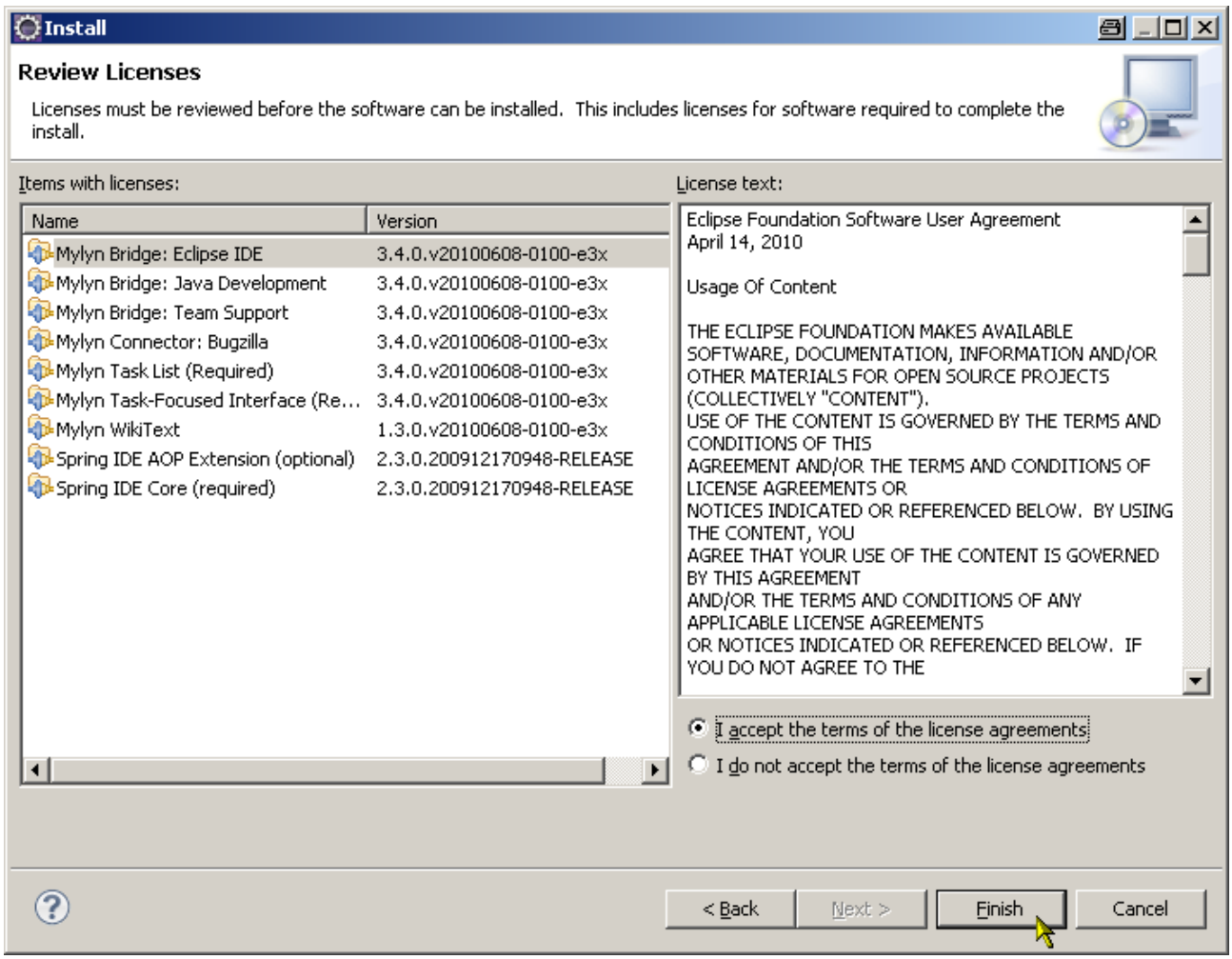

Abb.6.: Lizenzbedingungen akzeptieren

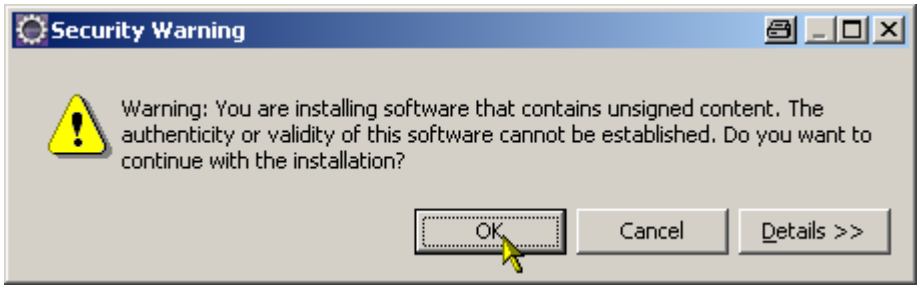

Abb.7.: Sicherheitshinweis mit OK bestätigen

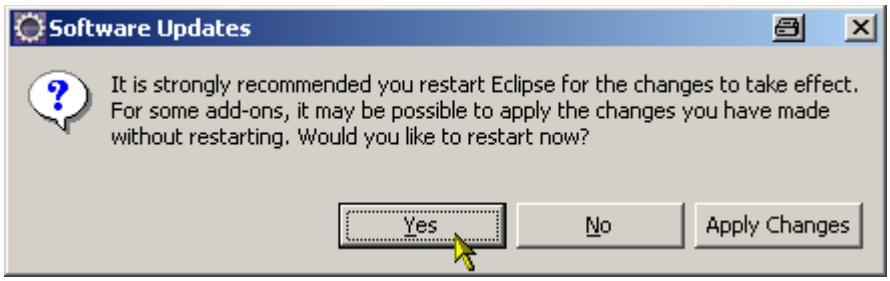

Abb.8.: Eclipse IDE neu starten

Herzlichen Glückwunsch - die Installation ist abgeschlossen!

Projekt einrichten

Ein Projekt kann wahlweise mit Apache Maven[11] (siehe Kapitel Maven und das Eclipse Plugin m2eclipse im Anhang A) oder manuell im Eclipse Workspace eingerichtet werden.

Manuelle Konfiguration

Die Voraussetzung für die manuelle Erstellung und Konfiguration eines Projekts ist ein entpacktes Spring Framework Distributions-JAR, z.B. unter [INSTALL_PATH]/spring-framework-X.Y.Z. Wenn Sie die ZIP-Datei entpackt haben finden Sie

- im „src“ Unterverzeichnis die Java-Sourcen
- in „dist“ Unterverzeichnis die Distributions JARs und
- in „doc“ die Spring Framework API (JavaDoc) und die komplette Referenz Dokumentation in HTML und PDF-Format.

Seit Spring 3.0 sind alle Abhängigkeiten in einem separaten Download[12] ausgegliedert. Dies muss zusätzlich heruntergeladen und entpackt werden.

Die Spring JAR-Dateien finden Sie im Verzeichnis:

- [INSTALL_PATH]/spring-framework-X.Y.Z/dist/

Die Abhängigkeiten (Dependencies) finden Sie im Verzeichnis:

- [INSTALL_PATH]/spring-framework-X.Y.Z-dependencies/

Zu den einzelnen Übungen sind die Abhängigkeiten jeweils angegeben. Der Verzeichnisname und die Versionsnummer des Archivs sind aber nicht mit angegeben. Bitte verwenden Sie die Version wie im Kapitel „Benötigte Software“ angeben. Ist es ein Modul des Spring Frameworks, dann finden Sie die JAR-Datei im [INSTALL_PATH]/spring-framework-X.Y.Z/dist Verzeichnis. Ist es eine Abhängigkeit des Spring Frameworks, dann befindet sich die JAR-Datei in einem Unterverzeichnis von [INSTALL_PATH]/spring-framework-X.Y.Z-dependencies. Oder Sie können die fehlenden Abhängigkeiten über das Maven Central-Repository[13] herunterladen.

Bitte lesen Sie dazu auch das Kapitel „1.3.1 Dependency Management and Naming Conventions“[14] in der Spring Referenzdokumentation.

11 Eine kurze Einführung in Apache Maven finden Sie unter: http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

12 http://s3.amazonaws.com/dist.springframework.org/release/SPR/spring-framework-3.0.2.RELEASE-dependencies.zip

13 http://search.maven.org/

14 http://static.springsource.org/spring/docs/3.0.x/reference/overview.html#dependency-management

Übungen

Hinweis:
Bei jeder Aufgabe sind die fehlenden Anweisungen bzw. unvollständigen Konfigurationselemente im Quellcode mit „TODO“ markiert! Aktivieren Sie den Eclipse View „Tasks“.

Bitte lesen Sie jede Aufgabebeschreibung zu Beginn einer Übung sorgfältig durch, bevor Sie mit dem Lösen der Aufgabe beginnen. Wenn Sie innerhalb der vorgegebenen Zeit mit der Lösung fertig sind, probieren Sie zu jeder Aufgabe noch eine eigene, erweiterte Aufgabe aus. Dazu stehen am Ende einiger Übung Beispiele, wie die jeweilige Aufgabe auch anders gelöst werden kann.

Übung 0 – HelloWorld mit dem Spring Framework

Voraussetzungen

- Eclipse IDE ist installiert und gestartet
- Das komplette Projekt spring-course-examples ist als Maven Projekt importiert
- Das HelloWorldSpring Beispiel liegt unter [WORKSPACE_PATH]/spring-course-examples/lab/student/helloWorldSpring
- Spring Framework ist entpackt und die folgenden JAR-Dateien im Java Build Path

Abhängigkeiten

- log4j.jar
- commons-logging.jar
- spring-beans.jar
- spring-core.jar
- spring-context.jar

Ziel

HelloWorld Beispiel nachvollziehen

Aufgaben

Vollziehen Sie das "Hallo Welt"-Beispiel Schritt für Schritt nach. Was passiert in der Implementierung der Hauptmethode main(String[]) in der Klasse `HelloWorldRunnerWithSpring`?

Welche Objekte werden wann und wie erzeugt?

Testen Sie das Einlesen der Konfigurationsdatei „applicationContext.xml“ im Verzeichnis „src/main/resources“ auch mit anderen Werten für das Attribut „value“ vom XML-Element:

```
<property name="name" value="Anna Gramm" />
```

Führen Sie die Klasse **net.gfu.seminar.spring.helloworld.HelloWorldRunnerWithSpring** mit „Run as... > Java Application“ aus und schauen Sie sich das Ergebnis auf der Konsole an.

Legen Sie jetzt eine neue Klasse `SpecialGuest` an, die von `Guest` erbt und in der Sie die toString()-Methode überschreiben. Ändern Sie nun den Implementierungsblock der Methode **net.gfu.seminar.spring.helloworld.HelloWorldRunner.main(String[])** so ab, dass statt der Instanz der Klasse `Guest` eine Instanz der Klasse `SpecialGuest` erzeugt wird. Führen Sie die Klasse mit „Run as... > Java Application“ aus und überprüfen Sie das Ergebnis in der Konsole.

Jetzt passen Sie die XML-Konfigurationsdatei „src/main/resources/applicationContext.xml“ so an, dass statt der Klasse `Guest` Ihre neue Klasse `SpecialGuest` eingelesen wird.

Überprüfen Sie das Ergebnis in dem Sie die Klasse **net.gfu.seminar.spring.helloworld.HelloWorldRunnerWithSpring** wieder mit „Run as... > Java Application“ ausführen und schauen Sie sich das Ergebnis auf der Konsole an.

Übung 1 – IoC und DI

Voraussetzungen

- Sie haben Eclipse gestartet und die Spring IDE installiert
- Das Eclipse-Projekt „helloWorldSpring“ ist im Workspace vorhanden
- Es befinden sich die folgenden JAR-Dateien im Java Build Path

Abhängigkeiten

- junit:junit.jar
- log4j:log4j.jar
- commons-logging:commons-logging.jar
- Spring Framework
  - org.springframework:spring-asm.jar
  - org.springframework:spring-beans.jar
  - org.springframework:spring-context.jar
  - org.springframework:spring-core.jar
  - org.springframework:spring-expression.jar

Dauer

< 45 Minuten

Ziel

Einsatz von Dependency Injection (DI) mit dem Spring Inversion of Control (IoC) Container unter Verwendung von Spring XML Meta-Konfiguration.

Aufgaben

Konfigurieren Sie zu Beginn den Java Build-Path des Eclipse Projekts „helloWorldSpring“ und ergänzen Sie die Abhängigkeiten zu den Bibliotheken wie am Anfang der Übung angegeben, wenn Sie dies nicht bereits bei der vorherigen Aufgabe getan haben.

Extrahieren Sie ein Interface mit dem Namen `GreetingService` aus der Klasse `Greeting` mit der `welcome()`-Methode.

```
public interface GreetingService {

        public String welcome();

}
```

Die Klasse `Greeting` soll neben dem Default-Konstrukor noch zwei weitere Konstruktoren haben. Ergänzen Sie den Konstruktor mit dem Argumenttyp `Greeting(List<Guest>)`:

```
public Greeting() { ... }
public Greeting(Guest guest) { ... }

public Greeting(List<Guest> guests) {
  //TODO
}
```

Erzeugt man eine Instanz mit dem Konstruktor `Greeting(Guest)` soll beim Aufruf der `welcome()`-Methode genau ein Gast begrüßt werden. Wird der Konstruktor `Greeting(List<Guest>)` aufgerufen, sollen alle Gäste aus der Liste begrüßt werden. Passen Sie die Implementierung der `welcome()`-Methode entsprechend an.

Legen Sie jetzt eine JUnit-Testklasse `GreetingTest` an und überlegen Sie sich ein paar Testfälle für die `welcome()`-Methode.
Die Erzeugung der `BeanFactory` und das Abrufen der Instanz vom Typ `GreetingService` soll in der `setUp()`-Methode in der Testklasse `GreetingTest` erfolgen.

```
@Before
public void setUp() throws Exception {
  LOG.debug("Set up");
  ConfigurableApplicationContext beanFactory = new ClassPathXmlApplicationContext(
    new String[] { "applicationContext.xml" });
  service = beanFactory.getBean("welcome", GreetingService.class);
}
```

Legen Sie eine neue Spring-Konfigurationsdatei „testData.xml“ an, in die Sie alle Spring-Beans vom Typ `Guest` verschieben.

Verwenden Sie die Testdaten für die Unit-Tests in dem Sie eine Spring-Bean vom Typ `Greeting` anlegen und dieser eine Liste von Gästen (java.util.List<Guest>) als Konstruktor-Argument übergeben.

Sie sind mit der Übung fertig, wenn alle Unit-Tests ohne Fehler ausgeführt werden können.

Übung 2 – DI mit Autowiring

Voraussetzungen

- Übung 1 – IoC und DI ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringDI06

Abhängigkeiten

- wie Übung 1 – IoC und DI
- zusätzlich
  - org.springframework:spring-test.jar
  - cglib:cglib-nodep.jar

Dauer

< 30 Minuten

Ziel

Setzen Sie Autowiring für die Konfiguration ein.

Aufgaben

In dieser Übung setzen Sie das automatische Verschalten von Abhängigkeiten mit dem Spring Container ein.

Passen Sie die XML-Konfigurationsdatei „applicationContext.xml“ so an, dass Autowiring[15] deaktiviert ist. Setzen Sie das Attribut default-autowire im <beans>-Root-Element auf „no“:

```
<beans xmlns="http://www.springframework.org/schema/beans"
          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-3.0.xsd"
          default-autowire="no">
```

und entfernen Sie das Element

```
<constructor-arg ref="guest"/>
```

bzw.

```
<property name="guest" ref="guest"/>
```

für die Spring-Bean Definition „welcome“ aus der Spring XML-Konfigurationsdatei `applicationContext.xml`.

Bevor Sie mit der Übung fortfahren, führen Sie alle Unit-Tests aus. Prüfen Sie, ob die Unit-Tests weiterhin fehlerfrei ausgeführt werden können. Warum kommt es zu einem Fehler? Was bedeutet der Fehler?

Setzen Sie nun nacheinander folgende Werte für das Attribut default-autowire:

1. „byType“,
2. „byName“ und

[15] siehe Kapitel 3.4.5 in der Spring Referenzdokumentation http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/beans.html#beans-factory-autowire

3. „constructor“.

Lassen Sie nach der Änderung jeweils die Unit-Tests laufen und beobachten Sie die Ausgabe auf der Konsole. Welche der Autowiring Varianten passen?

Sie sind mit der Übung fertig, wenn in beiden Unit-Tests keine Fehler angezeigt werden!

Hinweis

Zusatzaufgaben

Es wird jetzt etwas schwieriger. Nicht mehr alle Schritte sind beschrieben. Versuchen Sie mit Hilfe der Spring Framework Referenzdokumentation und der API-Dokumentation sowie den Seminar-Unterlagen eine Lösung zu finden.
Ein Tipp: Notieren Sie sich die Schritte in Ihren Unterlagen!
Sollten Sie noch Zeit haben, versuchen Sie eine der folgenden Spring-Container Konfigurationen.

Annotationen

Für die Annotations-basierte Container Konfiguration[16] müssen Sie die applicationContext.xml wie folgt anpassen:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">
  <context:annotation-config/>
  <context:component-scan base-package="net.gfu.seminar.spring "/>
</beans>
```

Die Klasse „`Guest`“ müssen Sie mit `@Component` annotieren, die Klasse „`Greeting` “ mit `@Service` oder `@Component` (beide Annotationen befinden sich im Package `org.springframework.stereotype`). Den Konstruktor für die Klasse `Greeting` annotieren Sie mit `@org.springframework.beans.factory.annotation.Autowired` oder `@javax.annotation.Resource`.

Passen Sie nun die Implementierungen der beiden `setUp()`-Methoden in den Test-Klassen an.

JavaConfig

Für die JavaConfig-basierte Container Konfiguration[17] benötigen Sie eine neue Klasse, z.B. ApplicationConfig, in der die beiden Spring-Beans „welcome“ und „guest“ definiert werden.

[16] siehe Kapitel 3.9 in der Spring Referenzdokumentation http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/beans.html#beans-annotation-config
[17] siehe Kapitel 3.11 in der Spring Referenzdokumentation http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/beans.html#beans-java

```
@Configuration
public class ApplicationConfig {
  @Autowired private Guest guest;

  @Bean public Guest guest() {
          return new Guest("Hans", "Dampf");
  }

  @Bean public GreetingService welcome() {
          return new Greeting(guest);
  }
}
```

In den `setUp()`-Methoden der beiden Test-Klassen müssen Sie einen AnnotationConfigApplicationContext erzeugen:

```
AnnotationConfigApplicationContext beanFactory = new AnnotationConfigApplicationContext(
                                        ApplicationConfig.class);
```

Übung 3 – Spring AOP

Voraussetzungen

- Die Übung 2 – DI mit Autowiring ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringAOP01

Abhängigkeiten

- Wie Übung 2 – DI mit Autowiring
- Zusätzlich
  - org.springframework:spring-aspects.jar
  - aopalliance:aopalliance.jar
  - org.aspectj:aspectjweaver.jar

Dauer

< 45 Minuten

Ziel

Implementieren eines Aspekts mit Spring AOP.

Aufgaben

Fügen Sie zu dem „Hello World“-Beispiel einen Tracing-Aspekt hinzu. Der Tracing-Aspekt soll alle Methodenaufrufe auf das Spring-Bean “welcome” protokollieren.

Zur Konfiguration des Aspekts verwenden Sie den `<aop:*>`-Namespace.

Legen Sie dazu eine neue Klasse mit dem Namen “`net.gfu.seminar.spring.helloworld.SimpleTraceAdvice`” an. Das Trace-Advice soll so konfiguriert werden, dass vor und nach dem Methodenaufruf eine Ausgabe in der Console oder einer Log-Datei erfolgt. Legen Sie dazu drei Methoden an:

1. public void enter(JoinPoint joinPoint) {;}
2. public void exit(StaticPart staticPart, Object returnValue) {;}
3. public void fail(StaticPart staticPart, Exception cause) {;}

Erstellen Sie eine neue XML-Konfigurationsdatei “tracingAdvice.xml” im Verzeichnis “src/test/resources” mit dem Spring IDE Wizard (siehe auch Übung 1 – IoC und DI). Importieren Sie zusätzlich den Namespace `<aop:*>`[18].
In der Spring XML-Konfigurationsdatei sollten dann folgende Namespaces deklariert sein:

[18] XML Schema für Spring AOP: http://www.springframework.org/schema/aop/spring-aop-3.0.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:aop="http://www.springframework.org/schema/aop"
          xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">

<!-- Konfiguration des Tracing Advices -->
</beans>
```

Verwenden Sie für die Konfiguration des Aspekts die AOP-Konfigurationselemente `<aop:before>`, `<aop:after-returning>` und `<after-throwing>`.

Führen Sie jetzt die Unit-Tests aus und beobachten Sie die Ausgabe auf der Console. Sie sind mit der Übung fertig, wenn Sie die Ausgaben des Advices auf der Console sehen.

Hinweis

Zusatzaufgaben

Erstellen Sie eine AOP-Konfiguration unter Verwendung des `<aop:advisor>`-Elements[19] für eine weitere Aspekt-Klasse die eine der folgenden Schnittstellen implementiert: `MethodBeforeAdvice`, `AfterReturningAdvice` und/oder `ThrowsAdvice`. Die Spring Advice API ist in der Spring Referenzdokumentation[20] beschrieben

Im nächsten Schritt konfigurieren Sie einen weiteren Aspekt über Annotationen[21]. Verwenden Sie die Annotationen @Aspect für die Aspekt-Klasse, @Before, @AfterReturning, @AfterThrowing und @Pointcut.

Überlegen Sie sich welche technischen Anforderungen (querschnittlichen Belange/Cross-Cutting Concern) man noch mit Aspekten umsetzen kann.

[19] Kapitel 7.3.6 Advisors in http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/aop.html#aop-schema-advisors

[20] Kapitel 8.3 Spring Advice API: http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/aop-api.html#aop-api-advice-types

21 Annotation Tutorial: http://download-llnw.oracle.com/docs/cd/E17409_01/javase/tutorial/java/javaOO/annotations.html

Übung 4 – Spring DAO Support

Voraussetzungen

- Die Übung 3 – Spring AOP ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringJDBC01

Abhängigkeiten

- Wie Übung 3 – Spring AOP
- zusätzlich
  - org.springframework:spring-jdbc.jar
  - org.springframework:spring-transaction.jar
  - commons-dbcp:commons-dbcp.jar
  - commons-pool:commons-pool.jar
  - hsqldb:hsqldb.jar

Dauer

< 60 Minuten

Ziel

Implementieren Sie ein Data Access Object (DAO)[22] mit den DaoSupport Klassen des Spring Frameworks.

Aufgaben

Implementieren Sie ein neues DAO für den Zugriff auf eine relationale Datenbank mit JDBC[23] und unter Verwendung der Klassen JdbcTemplate und JdbcDaoSupport aus dem Spring Framework.

Ergänzen Sie den Build-Path um die Abhängigkeiten für den JDBC-Treiber und die Modules des Frameworks wie oben angegeben!

Legen Sie ein neues Interface `GuestDao` mit den folgenden Methoden an:

1. int create(Guest guest);
2. Guest findById(Long id);
3. List<Guest> findByName(String name);
4. List<Guest> findAll();
5. Guest update(Guest guest);
6. void remove(Guest guest);

Dann legen Sie die Klasse Guest**Jdbc**Dao an und leiten diese Klasse von der `JdbcDaoSupport` Klasse aus dem Spring Framework ab:

[22] DAO Design Pattern: http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html
23 JDBC Tutorial: http://download-llnw.oracle.com/docs/cd/E17409_01/javase/tutorial/jdbc/index.html

```
public class GuestJdbcDao extends JdbcDaoSupport implements GuestDao {
 @Override
 public int create(Guest guest) {
  final String sql = "INSERT INTO GUESTS (firstname,lastname) VALUES (?,?)";
  final Object[] args = new Object[] { guest.getFirstName(),
            guest.getLastName()};
  int updatedRows = this.getJdbcTemplate().update(sql, args);
  return updatedRows;
 }
 // … more DAO methods for read, update, delete and finder methods
}
```

Implementieren Sie die Methoden für CRUD[24] mit Hilfe der entsprechenden Template-Klasse aus dem Spring Framework. Überlegen Sie sich, wie Sie das GuestJdbcDao als Spring-Bean definieren und welche Properties bzw. Konstruktor-Argumente übergeben werden müssen.

Legen Sie jetzt eine neue Spring Konfigurationsdatei “persistenceLayer.xml” an und definieren Sie dort die Spring-Beans für die Datenbankverbindung:

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
   <property name="driverClassName" value="${jdbc.driverClassName}"/>
   <property name="url" value="${jdbc.url}"/>
   <property name="username" value="${jdbc.username}"/>
   <property name="password" value="${jdbc.password}"/>
</bean>
```

Ersetzen Sie die Platzhalter ${jdbc.*} für HSQL mit einer In-Memory Datenbank:

```
jdbc.driverClassName=org.hsqldb.jdbcDriver
jdbc.url=jdbc:hsqldb:mem:testdb
jdbc.username=sa
jdbc.password=
```

Alternativ mit einer Dateisystem-basierten Datenbank (In-Process/standalone/single user):

```
jdbc.driverClassName=org.hsqldb.jdbcDriver
jdbc.url=jdbc:hsqldb:file:testdb
jdbc.username=sa
jdbc.password=
```

oder als Server-basierte Datenbank (multi-user):

```
jdbc.driverClassName=org.hsqldb.jdbcDriver
jdbc.url=jdbc:hsqldb:hsql://localhost/testdb
jdbc.username=sa
jdbc.password=
```

Sie können die Platzhalter über den Property-Placeholder ersetzen. Speichern Sie dazu die Schlüssel-Wert-Paare in einer Properties-Datei z.B. jdbc.properties ab und ergänzen Sie folgendes Element in der Spring Konfiguration:

```
<context:property-placeholder location="classpath:/jdbc.properties" />
```

Um den HSQLDB-Server zu starten müssen Sie von der Eingabeaufforderung folgenden Befehl ausführen:

24 CRUD alias CDUR steht für Create, Read, Update, Delete Operationen

```
[WORKSPACE_PATH]/helloWorldSpring>java -cp lib/hsqldb.jar;src/main/resources
org.hsqldb.Server
```

Die HSQLDB Server Manager GUI-Anwendung starten Sie mit:

```
[WORKSPACE_PATH]/helloWorldSpring>java -cp lib/hsqldb.jar
org.hsqldb.util.DatabaseManagerSwing
```

Mehr zur Konfiguration von HSQLDB finden Sie unter http://hsqldb.org/doc/guide/ch01.html.

Passen Sie nun die XML Konfigurationsdatei "applicationContext.xml" so an, dass die neue Klasse `GuestJdbcDao` zum Einsatz kommt. Sie müssen einen weiteren Konstruktor in der Klasse `Greeting` anlegen und ggfs. auch die Implementierungen der einzelnen Methoden anpassen.

DDL für die Tabelle GUESTS:

```
CREATE TABLE GUESTS(ID INTEGER generated by default as identity (start with 1) NOT NULL
PRIMARY KEY,FIRSTNAME VARCHAR(50),LASTNAME VARCHAR(50))
```

Und

```
DROP TABLE guests if exists
```

Die Initialisierung der Datenbank kann beim Starten der Anwendung durch das Spring Framework erfolgen. Definieren Sie ein Spring-Bean im <jdbc>-Namespace[25]:

```
<jdbc:initialize-database data-source="dataSource" ignore-failures="ALL">
        <jdbc:script location="classpath:drop_hsql_schema.sql" />
        <jdbc:script location="classpath:create_hsql_schema.sql" />
</jdbc:initialize-database>
```

Und legen Sie die zwei Dateien mit den DDL-Statements an.

Ergänzen Sie jetzt die Unit-Test Klasse um

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "classpath:/applicationContext.xml" })
```

und

```
@Autowired
private Greeting greeting;
```

Sie sind mit der Übung fertig, wenn in beiden Unit-Tests keine Fehler angezeigt werden!

Eine Anpassung der Unit-Tests sollte nicht notwendig sein.

Hinweis

Zusatzaufgaben

Implementieren Sie je ein DAO unter Verwendung der folgenden Technologien:

a) Mit iBATIS[26] und unter Verwendung der Klassen SqlMapClientTemplate und SqlMapClientDaoSupport aus dem Spring Framework.

b) Mit Hibernate[27] und unter Verwendung der Klassen HibernateTemplate und HibernateDaoSupport aus dem Spring Framework.

c) Mit JPA 1.0[28] und unter Verwendung der Klassen JpaTemplate und JpaDaoSupport aus dem Spring Framework (Hibernate-EntityManager als Provider).

[25] http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd
[26] iBatis: http://ibatis.apache.org/
27 Hibernate Tutorial: http://docs.jboss.org/hibernate/core/3.3/reference/en/html/tutorial.html
28 JPA Tutorial: http://download.oracle.com/docs/cd/E17410_01/javaee/6/tutorial/doc/bnbpy.html

Sie benötigen dann folgende zusätzliche Abhängigkeiten:

- mysql:mysql-connector-java.jar (für MySQL DB wenn diese anstatt HSQLDB verwendet werden soll)
- org.springframework:spring-orm.jar (nur für Hibernate und JPA)
- org.hibernate:hibernate-core.jar (sowie alle transitiven Abhängigkeiten für Hibernate)
- javax.persistence:persistence-api.jar (nur für JPA)
- org.hibernate:hibernate-entitymanager.jar (nur für JPA)

Übung 5 - Transaktionssteuerung

Voraussetzungen

- Die Übung 4 – Spring DAO Support ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringTX01

Abhängigkeiten

- Wie Übung 4 – Spring DAO Support

Dauer

< 30 Minuten

Ziel

Konfigurieren Sie einen Transaktions-Manager für den Datenbankzugriff und verwenden Sie die deklarative Transaktionssteuerung von Spring.

Aufgaben

Ergänzen Sie einen Transaktions-Manager in der Spring Konfigurationsdatei "persistenceLayer.xml" und deklarieren Sie die Transaktionssteuerung über die Spring-Konfiguration für das in Übung 4 – Spring DAO Support angelegte DAO.

```
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
</bean>
```

Verwenden Sie die XML-basierte Konfiguration und den <tx:*/> Namespace:

```
<beans xmlns="http://www.springframework.org/schema/beans"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:context="http://www.springframework.org/schema/context"
          xmlns:tx="http://www.springframework.org/schema/tx"
          xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
 http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
 http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">
...
</bean>
```

Fügen Sie ein <tx:advice> und ein <aop:config> Element hinzu.

Sie sind mit der Übung fertig, wenn in beiden Unit-Tests keine Fehler angezeigt werden. Beobachten Sie die Ausgabe auf der Console, ob die Transaktionen für die Methodenaufrufe erzeugt werden.

Hinweise

Zusatzaufgaben

Für die Service Klasse deklarieren Sie ebenfalls Transaktionen für alle öffentlichen Methoden.

Sie können wählen zwischen:

a) Konfiguration über das TransactionProxyFactoryBean

b) Transaktionssteuerung durch Annotationen.

Übung 6 – Spring MVC

Voraussetzungen

- Die Übung 5 - Transaktionssteuerung ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringMVC01
- Ein Java EE Server (Web-Profile) ist installiert (z.B. Apache Tomcat oder GlassFish)

Abhängigkeiten

- Wie Übung 5 - Transaktionssteuerung
- zusätzlich
  - spring-mvc.jar
  - spring-webmvc.jar

Dauer

< 90 Minuten

Ziel

Implementieren Sie eine dynamische Web-Anwendung basierend auf Spring MVC.

Aufgaben

Wechseln Sie in die “Java EE Perspective” und erstellen Sie eine neue Server Definition für einen Java EE Server.

Legen Sie ein neues Dynamisches Web-Projekt „helloWorldMvc“ in Eclipse an. Fügen Sie im Java Build Path unter „Projects“ das Projekt “helloWorld” hinzu. Unter “Java EE Module Dependencies” wählen Sie neben den Projekt “helloWorld” auch alle Abhängigkeiten für das Projekt “ helloWorld ” aus.

Legen Sie nun eine JSP Datei “add.jsp” im Verzeichnis „WebContent/guest“ an. Deklarieren Sie folgende Tag-Bibliotheken:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
```

Ergänzen Sie im Body ein Form-Element:

```
<form:form modelAttribute="addGuestForm" method="post">
  <table>
   <tr>
            <td>Vorname:</td>
            <td><form:input path="firstname" /><form:errors path="firstname"/></td>
  </tr>
  <tr>
            <td>Nachname:</td>
            <td><form:input path="lastname" /><form:errors path="lastname"/></td>
  </tr>
  <tr>
            <td></td>
```

```
            <td><input type="reset" value="Reset" /> <input type="submit" value="Invite"
/> </td>
                        </tr>
                        <tr>
          <td colspan="2" align="center">${msg}</td>
  </tr>
 </table>
</form:form>
```

Zusätzlich legen Sie im Verzeichnis „WebContent/guest“ eine Datei „welcome.jsp“ mit folgendem Inhalt im Body-Element:

```
<h1 align="center">${welcome}</h1>
```

Als Modellklasse für den View „guest/add.jsp“ legen Sie eine Klasse `AddGuestForm` an, mit den Properties firstname, lastname, entsprechend der JavaBean-Konvention. Die Klasse muss das java.io.Serializable Interface implementieren.

Legen Sie nun die Klasse “`GreetingController`” an und konfigurieren Sie über Annotationen den Pfad für das RequestMapping auf „/greeting“. Für die Methode setupForm() definieren Sie das RequestMapping so, dass auf ein HTTP GET ein neues `AddGuestForm` Objekt erzeugt wird:

```
@RequestMapping(value="/add", method = RequestMethod.GET)
public String setupForm(Model model) {
          model.addAttribute(new AddGuestForm());
          return "/guest/add";
}

@RequestMapping(value="/welcome", method = RequestMethod.GET)
public String welcome(Model model) {
          model.addAttribute("welcome", service.welcome());
          return "/guest/welcome";
}
```

Über HTTP POST soll ein `AddGuestForm` verarbeitet werden:

```
@RequestMapping(method = RequestMethod.POST)
public ModelAndView processForm(@ModelAttribute AddGuestForm addGuestForm,
                                BindingResult result) {
          ModelAndView mav = new ModelAndView();
          if (result.hasErrors()) {
                    mav.setViewName("/guest/add");
          } else {
                    service.addGuest(new GuestImpl(addGuestForm.getFirstname(),
addGuestForm.getLastname()));
                    mav.addObject("welcome", service.welcome());
                    mav.setViewName("/guest/welcome");
          }
          return mav;
}
```

Ergänzen Sie das Feld für den GreetingService und entsprechende Methoden bzw. einen Konstruktor, um die Referenz über DI zusetzen.

Ergänzen Sie in der web.xml die Konfiguration des Dispatcher-Servlets:

```
<context-param>
          <param-name>contextConfigLocation</param-name>
          <param-value>/WEB-INF/webContext.xml</param-value>
  </context-param>
  <!-- Init Spring -->
  <listener>
          <listener-
class>org.springframework.web.context.request.RequestContextListener</listener-class>
  </listener>
  <!-- read the XmlWebApplicationContext for spring -->
  <listener>
          <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <servlet>
     <servlet-name>controller</servlet-name>
     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
     <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
     <servlet-name>controller</servlet-name>
     <url-pattern>*.do</url-pattern>
  </servlet-mapping>
```

Legen Sie die Spring-Konfigurationsdatei für das GuestDao und den GreetingService in der Datei webContext.xml an. Die javax.sql.DataSource soll über JNDI verschaltet werden.

```
<bean id="jndiDatasource" name="dataSource"
          class="org.springframework.jndi.JndiObjectFactoryBean">
          <property name="jndiName" value="java:comp/env/jdbc/DefaultDS"/>
</bean>
```

Für die Konfiguration des Spring MVC Controller Servlets legen Sie die Datei controller-servlet.xml an.

```
<!-- Configures the @Controller programming model -->
  <mvc:annotation-driven />

  <!-- Scans the classpath of this application for @Components to deploy as beans -->
  <context:component-scan base-package="net.gfu.seminar.spring.helloworld" />

  <bean id="jspViewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
      value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/" />
    <property name="suffix" value=".jsp" />
  </bean>
```

Fügen Sie nun noch eine DataSource im Java EE Server hinzu. Bei Apache Tomcat kann dies über die Datei META-INF/context.xml konfiguriert werden.

```
<Resource name="jdbc/DefaultDS" auth="Container"
            type="javax.sql.DataSource" username="sa" password=""
            driverClassName="org.hsqldb.jdbcDriver"
url="jdbc:hsqldb:hsql://localhost/testdb"
            maxActive="8" maxIdle="4"/>
```

Starten Sie die Web-Anwendung in dem Sie folgende URL im Browser öffnen:
http://localhost:8080/helloWorldMvc/greeting/add.do

Übung 7 – Spring Remoting

Voraussetzungen

- Die Übung 5 - Transaktionssteuerung ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringRMI01

Abhängigkeiten

- Wie Übung 5 - Transaktionssteuerung

Dauer

< 30 Minuten

Ziel

Exportieren Sie die Service Klasse als RMI-Service und rufen Sie den Service remote via RMI auf.

Aufgaben

Das Spring Bean “greeting” als RMI-Service exportieren und in einem Test-Client über den Proxy verwenden.

Legen Sie eine neue Unit Test Klasse “GreetingServiceRemoteTest” als Kopie von “GreetingServiceTest” an.

Erstellen Sie dann eine Spring Konfigurationsdatei mit den Spring-Bean Definitionen für den Service und Client. Diese Konfigurationsdatei verwenden Sie zusätzlich für den neuen GreetingServiceRemoteTest.

Service:

```
<bean id="rmiGreetingServiceExporter"
class="org.springframework.remoting.rmi.RmiServiceExporter">
    <property name="serviceName" value="GreetingService" />
    <property name="service" ref="welcome" />
    <property name="serviceInterface"
value="net.gfu.seminar.spring.helloworld.GreetingService" />
    <property name="registryPort" value="1099" />
</bean>
```

Client:

```
<bean id="rmiGreetingServiceProxy"
class="org.springframework.remoting.rmi.RmiProxyFactoryBean">
    <property name="serviceUrl"
value="rmi://${server.ip}:${server.port}/GreetingService"/>
    <property name="serviceInterface"
value="net.gfu.seminar.spring.helloworld.GreetingService"/>
</bean>
```

Für den lokalen Aufruf über das RMI-Protokoll müssen Sie die Platzhalter ${server.ip} und ${server.port} durch localhost und 1099 ersetzen.

Sie sind mit der Übung fertig, wenn in dem neuen Unit-Test keine Fehler angezeigt werden!

Hinweis

Zusatzaufgaben

Erzeugen Sie zusätzlich jeweils einen Client-Proxy und Service-Exporter für die Protokolle

a) HTTP und

b) SOAP (mit JAX-WS)

Übung 8 – RESTful Web Service mit Spring MVC

Voraussetzungen

- Die Übung 6 – Spring MVC ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringREST

Abhängigkeiten

- Wie Übung 6 – Spring MVC

Dauer

<45 Minuten

Ziel

Erstellen Sie einen RESTful Web Controller mit dem Spring MVC Framework.
Rufen Sie den RESTful Service dann mit dem Spring RestTemplate auf.

Aufgaben

Fügen Sie in dem GreetingController eine neue Methode hinzu,

```
@RequestMapping(value = "/to/{name}", method = RequestMethod.GET)
public @ResponseBody String getTextMessage(@PathVariable String name) {
          return "Hello, " + name + "!";
}
```

Und ergänzen Sie die web.xml mit einem weiteren Servlet-Mapping:

```
<servlet-mapping>
     <servlet-name>controller</servlet-name>
     <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Installieren Sie die Web-Anwendung innerhalb eines Servlet-Containers und starten Sie den Server.
Prüfen Sie, ob Sie die neue Methode über die URL:
http://localhost:8080/helloWorldMvc/rest/greeting/to/Hans
aufrufen können. Achten Sie auf den richtigen Web Context (Context Root der Web-Anwendung).
Implementieren Sie einen neuen Unit Test, der die Service Klasse via HTTP/REST aufruft.
Nutzen Sie dabei das org.springframework.web.client.RestTemplate. Um eine HTTP GET Anfrage mit dem RestTemplate an eine URL zu schicken, wählen Sie eine der getForObject() Methoden aus.

Übung 9 – Web-Anwendung mit Spring Security absichern

Voraussetzungen

- Die Übung 6 – Spring MVC ist abgeschlossen und alle Unit-Tests lassen sich **ohne** Fehler ausführen.
- Oder Sie arbeiten mit der Musterlösung [WORKSPACE_PATH]/spring-course-examples /lab/trainer/helloWorldSpringSEC01

Abhängigkeiten

- Wie Übung 6 – Spring MVC
- Spring Security

Dauer

<60 Minuten

Ziel

Absichern der Web-Anwendung mit Spring Security.

Aufgaben

Ergänzen Sie die Konfiguration des DelegatingFilterProxy in der web.xml. Der Filter sollte auf das URL-Pattern /* konfiguriert sein. Jetzt muss Spring Security noch konfiguriert werden. Dazu legen Sie eine neue Spring XML-Konfigurationsdatei „security-context-config.xml“ an.

```
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"
    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-3.0.xsd
                http://www.springframework.org/schema/security
                http://www.springframework.org/schema/security/spring-security-3.0.xsd">

    <http use-expressions="true">
        <intercept-url pattern="/**" access="permitAll" />
        <form-login />
        <logout/>
    </http>

    <authentication-manager>
        <authentication-provider>
            <user-service>
              <user password="pass" name="efall" authorities="ROLE_USER"/>
              <user password="kg" name="agramm" authorities="ROLE_USER"/>
              <user password="luft" name="hdampf" authorities="ROLE_USER,ROLE_ADMIN"
            </user-service>
        </authentication-provider>
    </authentication-manager>
</beans:beans>
```

Der Zugriff auf die Ressource „/greeting/add*“ soll nur für authentifizierte Benutzer möglich sein. Dies wird wie folgt konfiguriert:

```
<intercept-url pattern="/greeting/add*" access="isAuthenticated()" />
```

Abschlusstest

Bitte beantworten Sie folgende Fragen:

1. Was bedeutet Inversion of Control (IoC)?
2. Welche unterschiedlichen Typen von Dependency Injection gibt es?
3. Was sind die Vorteile von Dependency Injection?
4. Welche sind die Vorteile vom Spring Framework?
5. Nennen Sie einige Features von Spring.
6. Nennen Sie mindestens 4 Module vom Spring Framework.
7. Was ist ein ApplicationContext?
8. Welche Typen von Konfiguration unterstützt das Spring Framework?
9. Nennen Sie mindestens eine Implementierung der Schnittstelle ApplicationContext.
10. Wie ist der Lebenszyklus einer Spring-Bean in einem Spring-Container (BeanFactory)?
11. Welche Typen von Autowiring unterstützt das Spring Framework?
12. Mit welchen Annotationen kann das Spring TestContext Framework in einen JUnit-Test eingebunden werden?
13. Welche zusätzlichen Annotationen bietet das Spring TestContextFramework an?
14. Was ist ein Aspekt?
15. Welchen Typ von Weaving unterstützt Spring AOP?
16. Was ist der Unterschied zwischen einem JoinPoint und einem Pointcut?
17. Welche Typen von Advices gibt es?
18. Welche Typen von Transactions-Management unterstützt das Spring Framework?
19. Wie heisst die Annotation für die Transaktionssteuerung?
20. Was ist das JdbcTemplate?
21. Welche Methode ist in der Schnittstelle RowMapper definiert?
22. Welche Technologien werden von Spring MVC unterstützt?
23. Wie heisst die Implementierung des FrontControllers aus dem Spring MVC Framework?
24. Welche Scopes gibt es in Spring MVC zusätzlich?
25. Mit welcher Annotation kann man aus folgender RESTful HTTP GET Anfrage den Pfad-Abschnitt „9“ auslesen? http://localhost:8080/helloworld/welcome/guest/9
26. Welche Protokolle unterstützt Spring Remoting?
27. Welche Arten von Authentifizierung unterstützt Spring Security?

Mehr unter https://www.springmockexams.com und http://javaetmoi.com/2016/01/spring-core-4-2-certification-mock-exam/

Anhang A

Maven und das Eclipse Plugin m2eclipse

m2eclipse Plug-in Installation – optional

Für folgende Eclipse Versionen muss das m2e Plugin nachinstalliert werden.

Die Projektseite ist: http://eclipse.org/m2e/

Update-Site (m2e):

- Eclipse 3.7 (Indigo) – Version 1.1 - http://download.eclipse.org/releases/indigo
- Eclipse 3.6 (Helios) – Version 1.0 - http://download.eclipse.org/technology/m2e/releases
- Eclipse 3.5 (Galileo) – Version 0.12 - http://m2eclipse.sonatype.org/sites/m2e

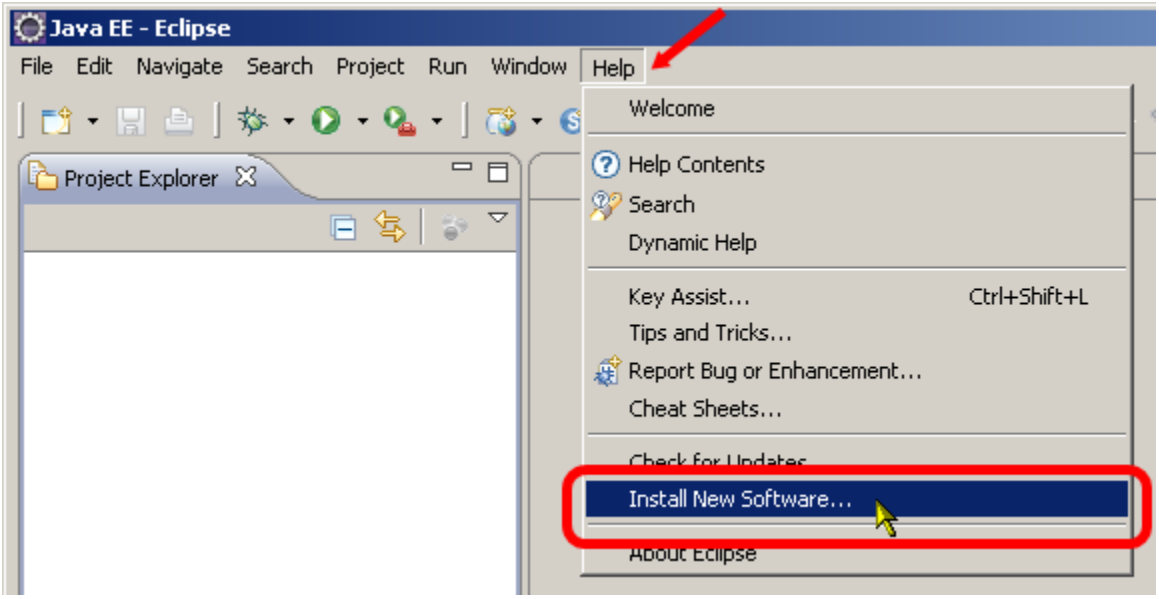

Abb.9.: Neue Software installieren

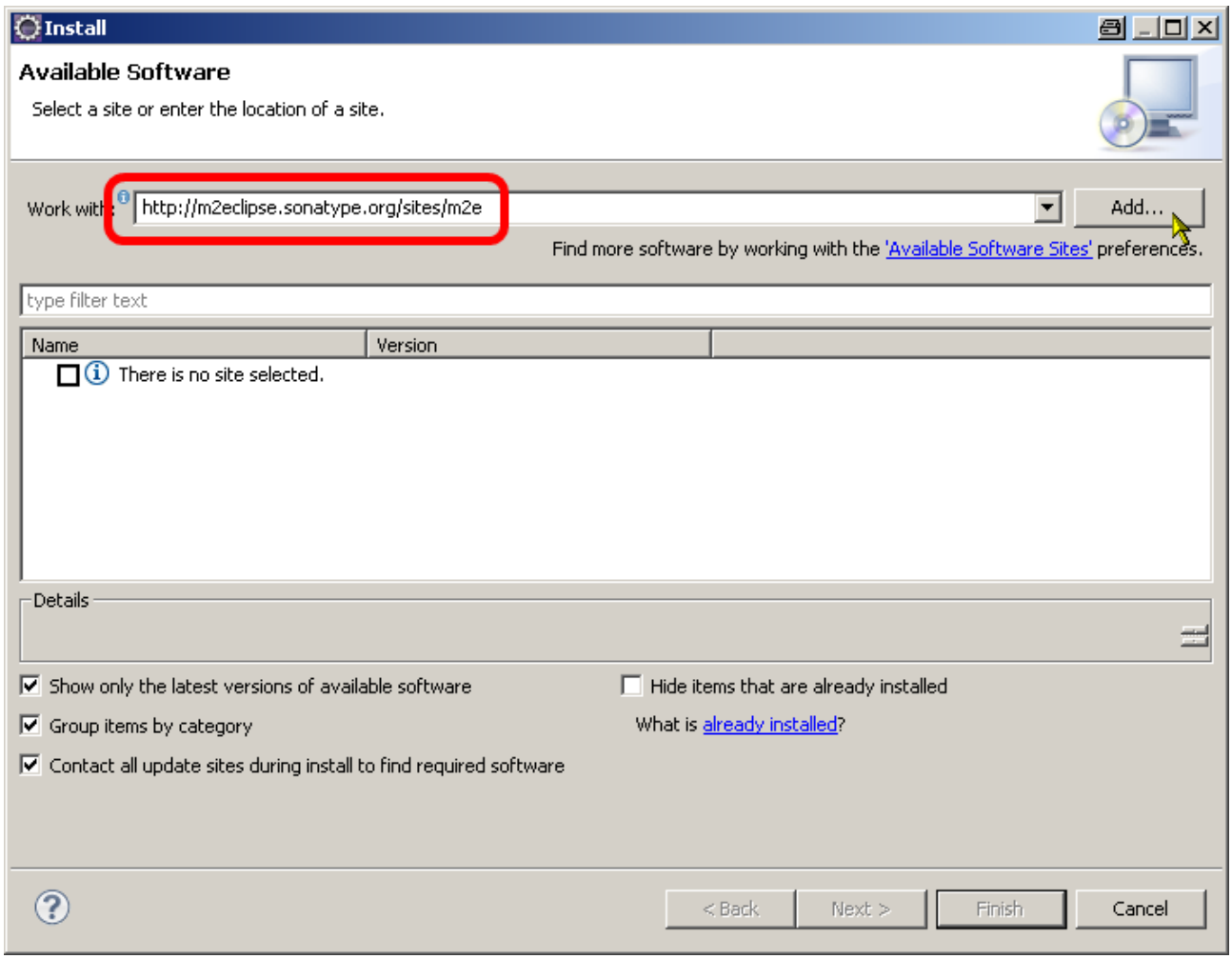

Abb.10.: Neue Update Site hinzufügen

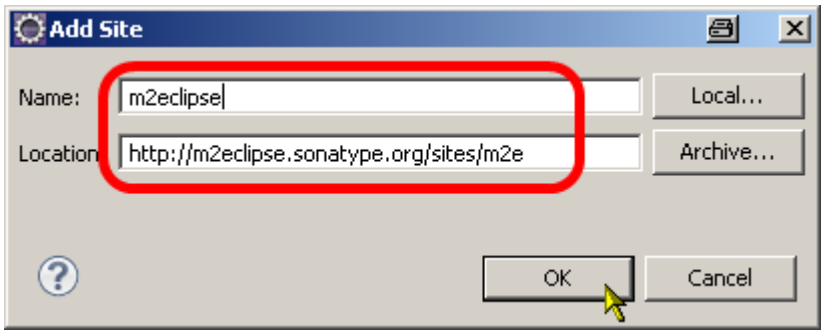

Abb.11.: URL für m2e-Plugin eingeben (URL ist von Eclipse IDE Version abhängig!)

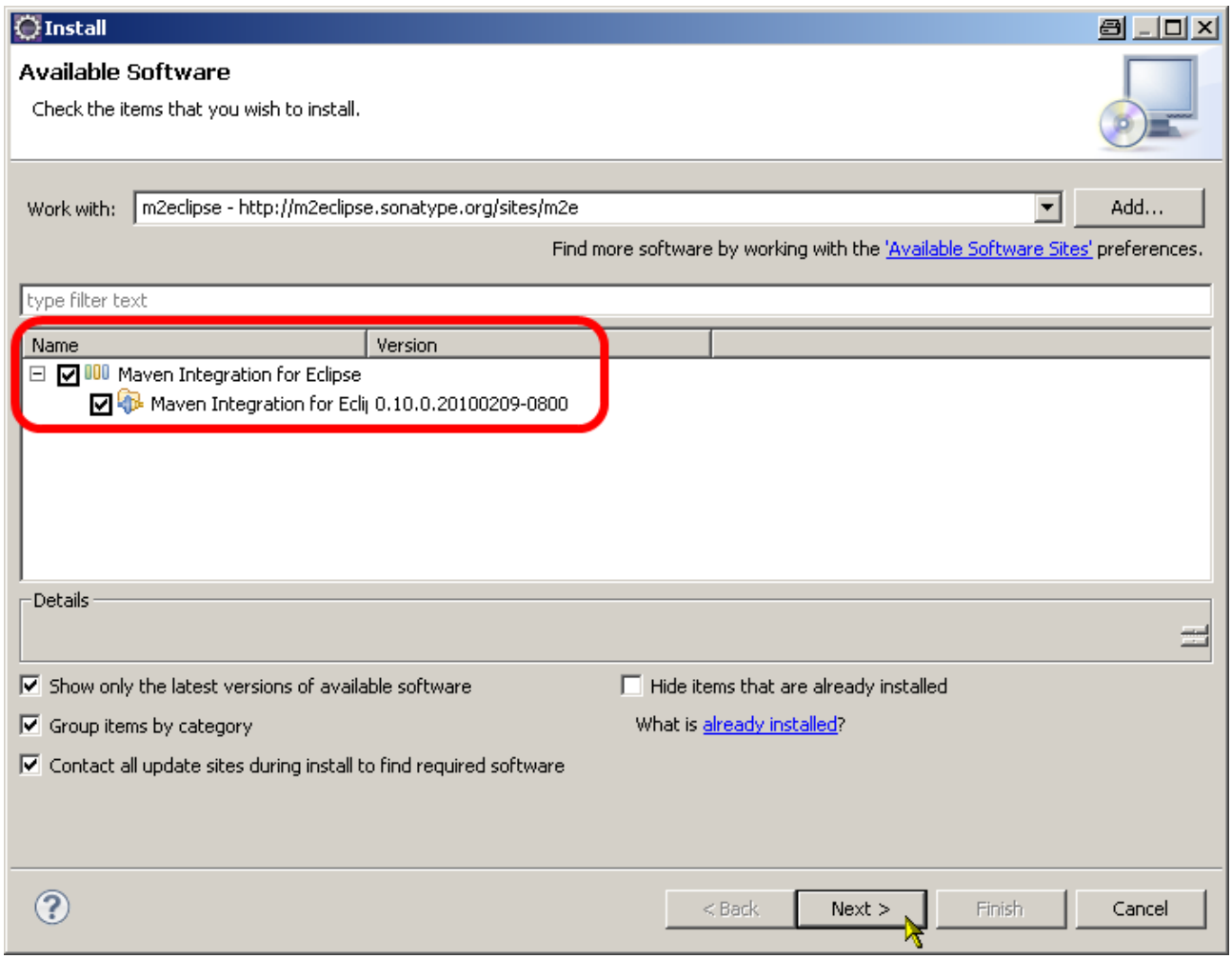

Abb.12.: Plugin auswählen

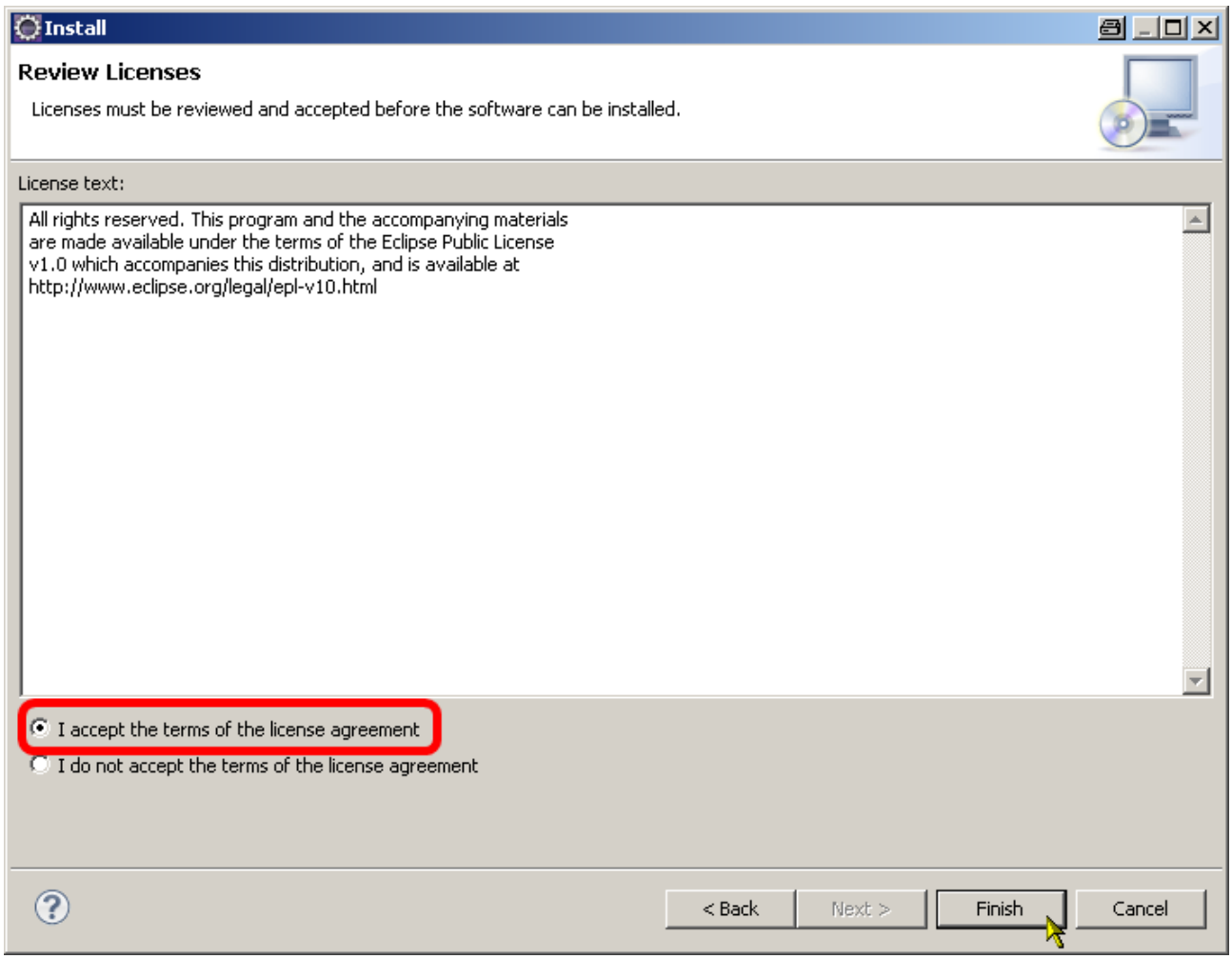

Abb.13.: Lizenzbedingungen akzeptieren

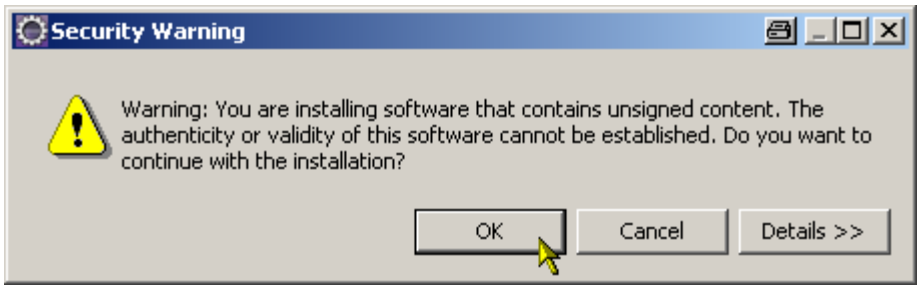

Abb.14.: Warnung mit OK bestätigen

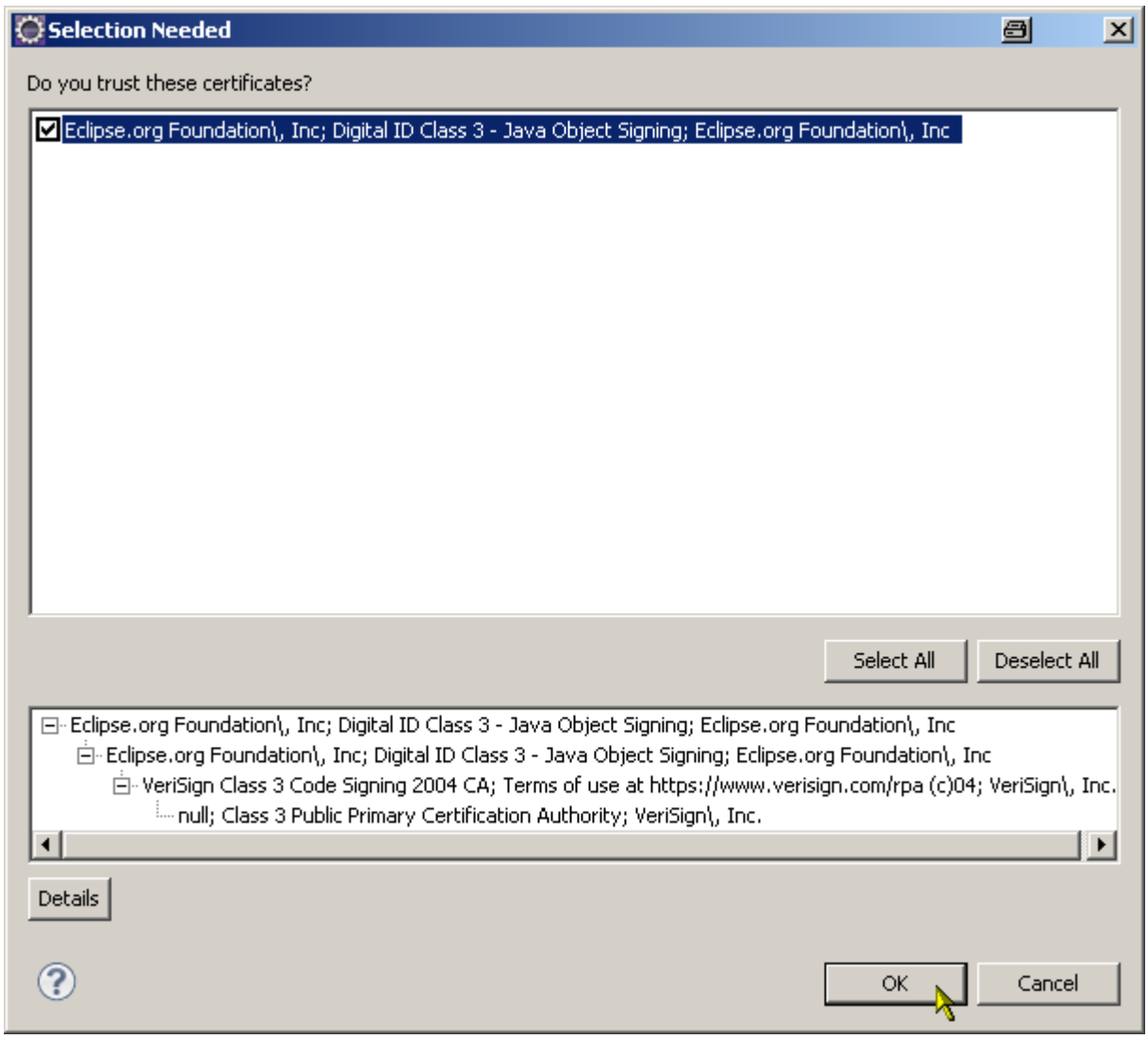

Abb.15.: Eclipse Foundation Zertifikat annehmen

Mit einem Neu-Start der Eclipse IDE ist die Installation abgeschlossen.

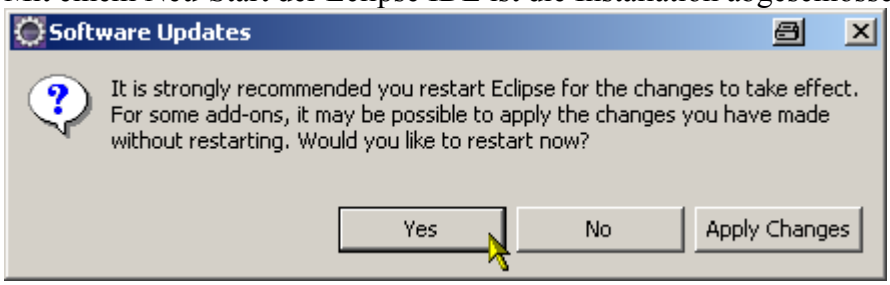

Abb.16.: Eclipse IDE neu starten

Konfiguration mit dem m2eclipse Plugin

Für die Konfiguration mit Apache Maven ist die Installation des m2eclipse Plugins (siehe m2eclipse Plug-in Installation – optional) notwendig!

Vorbereiten des Plugins

Unter „Window > Preferences > Maven“ kann das m2eclipse Plugin konfiguriert werden:

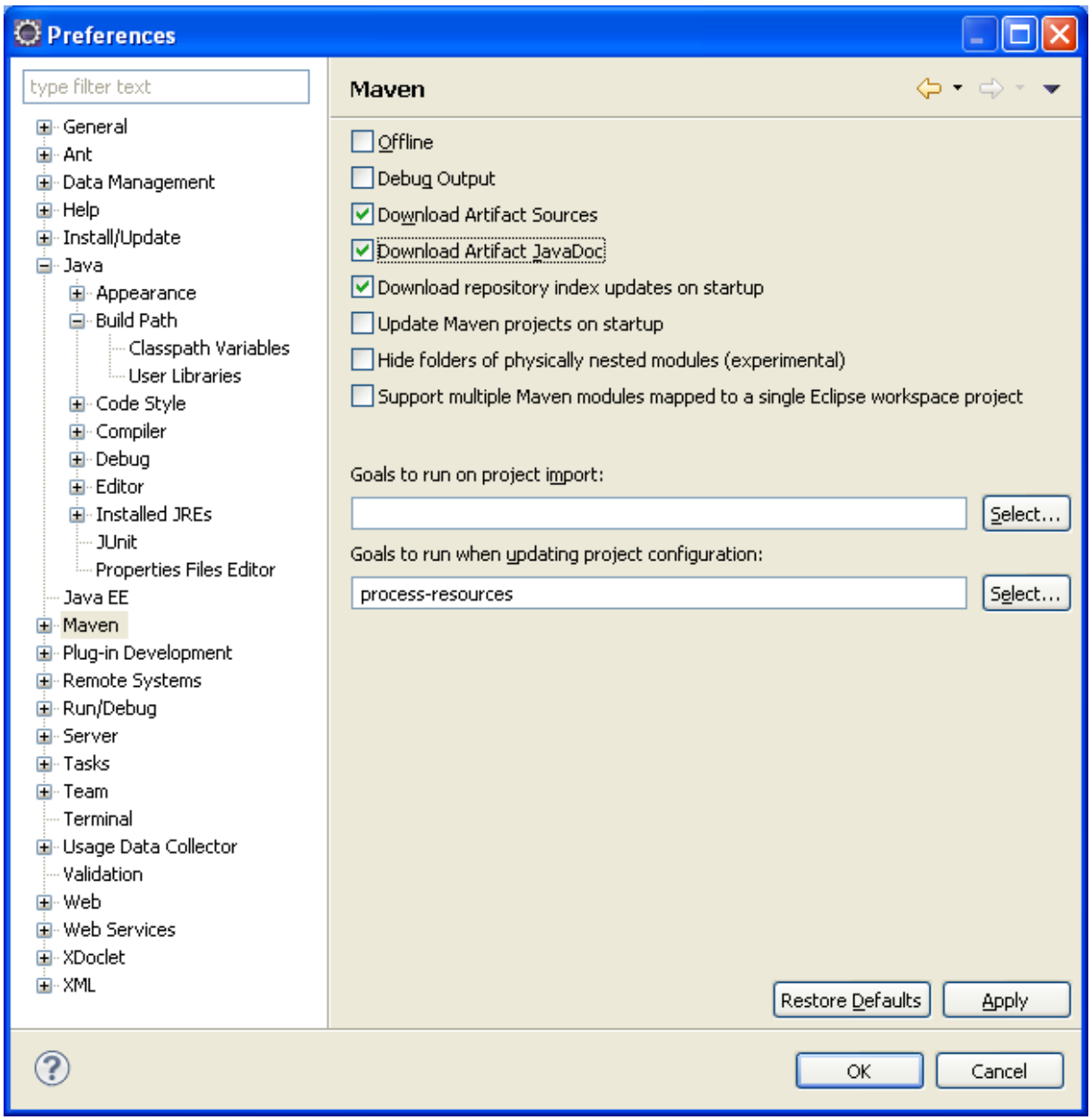

Abb.17.: Aktivieren des automatischen Download der Java Source und der JavaDoc Dateien

Ändern und aktualisieren der Benutzer-spezifischen „settings.xml“-Datei.

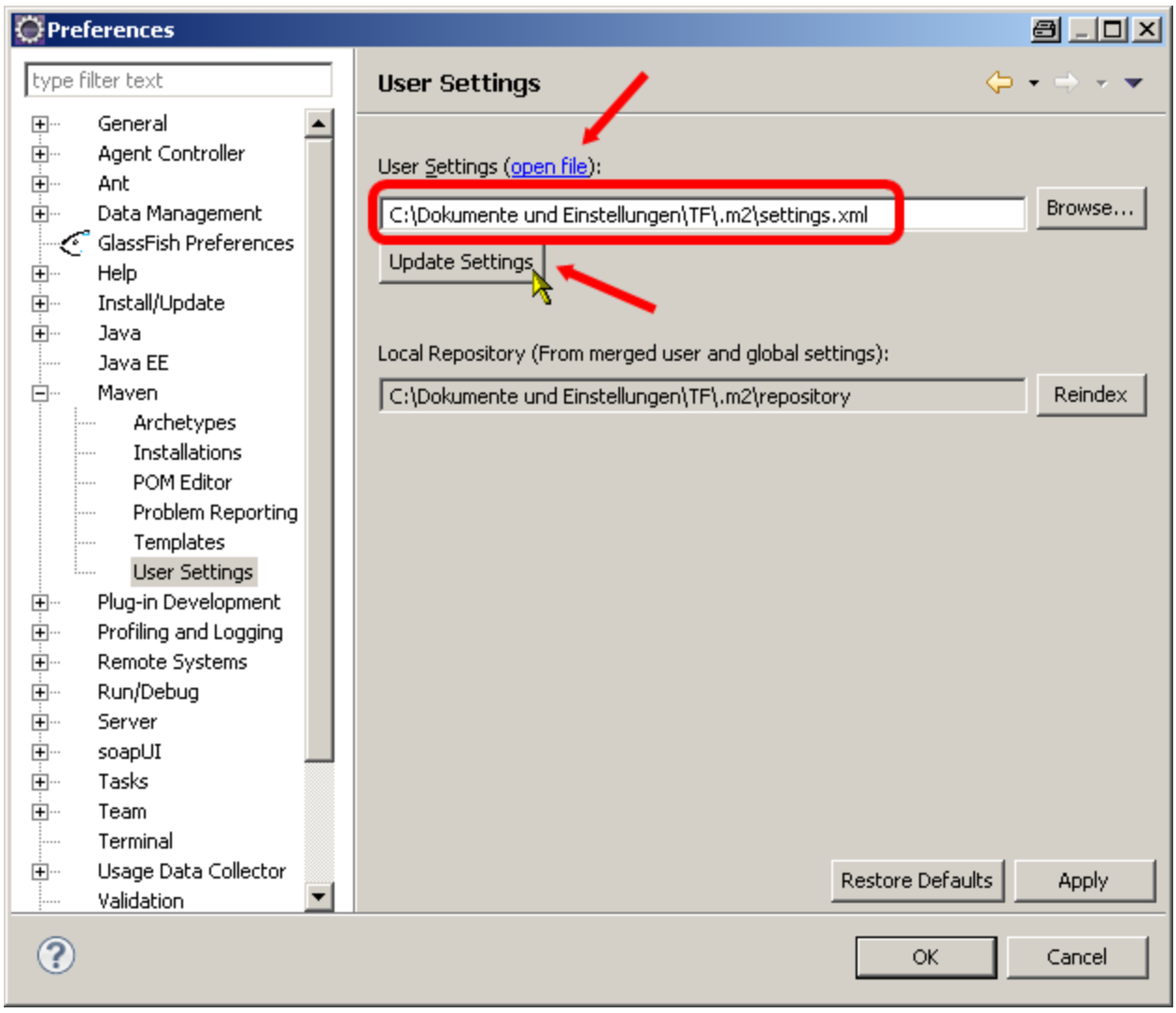

Abb.18.: User Settings aktualisieren und innerhalb der Eclipse IDE öffnen

Maven aus Eclipse mit m2eclipse Plugin starten

Bei den folgenden Übungen können Sie entscheiden, ob Sie Maven über das Command Line Interface (CLI) aufrufen oder über das m2eclipse Plugin innerhalb der Eclipse IDE.

In die Java Perspektive wechseln: 'Window' > 'Open Perspective' > 'Java' und im Fenster 'Package Explorer' > rechte Maustaste auf Projekt > 'Run As' > 'Maven build …' aufrufen.

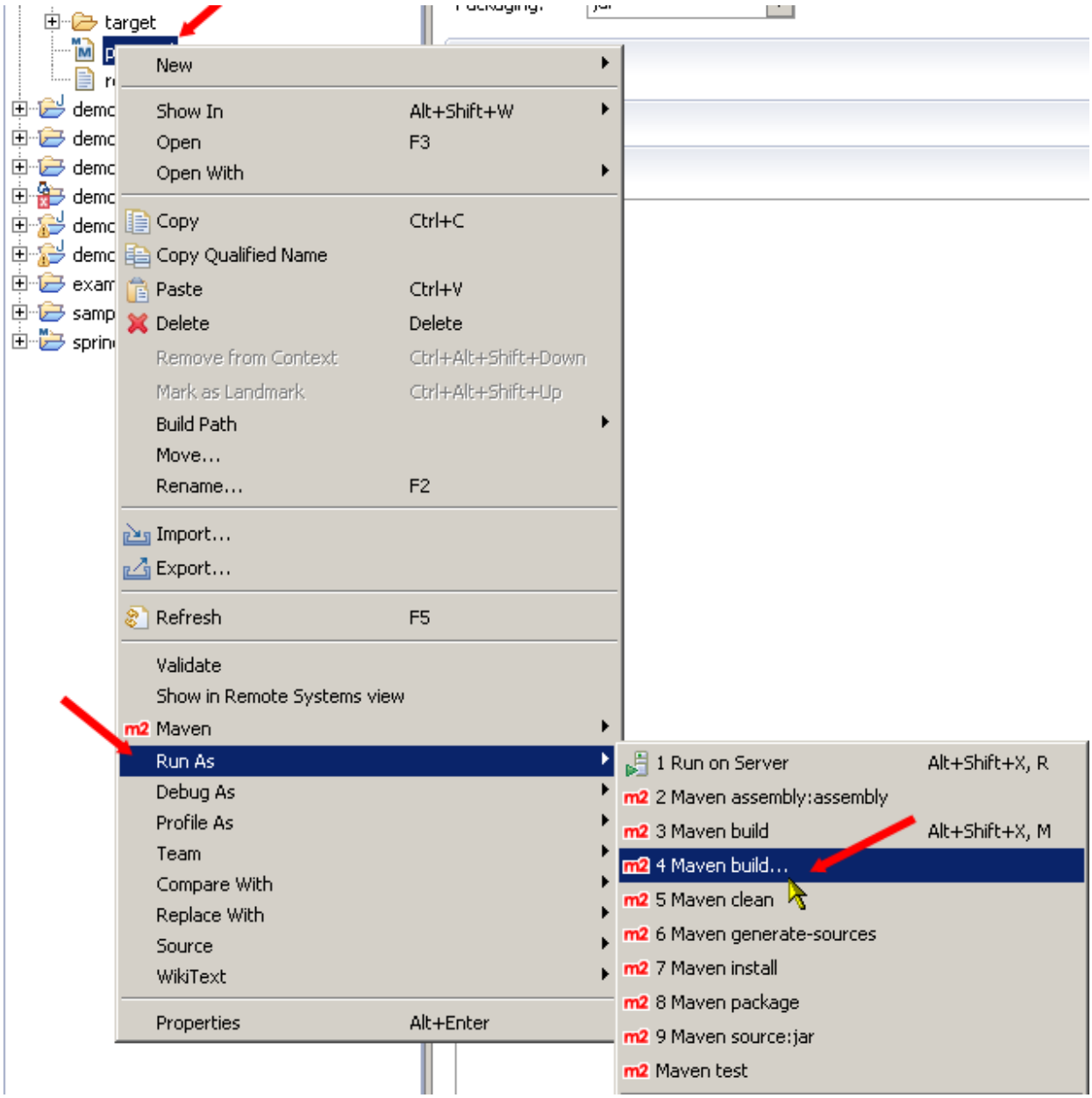

Abb.19.: Run As – Maven build ...

In der Konfiguration dann das Maven Goal angeben, wie z.B. das Maven Eclipse Plugin mit dem Goal 'eclipse:eclipse':

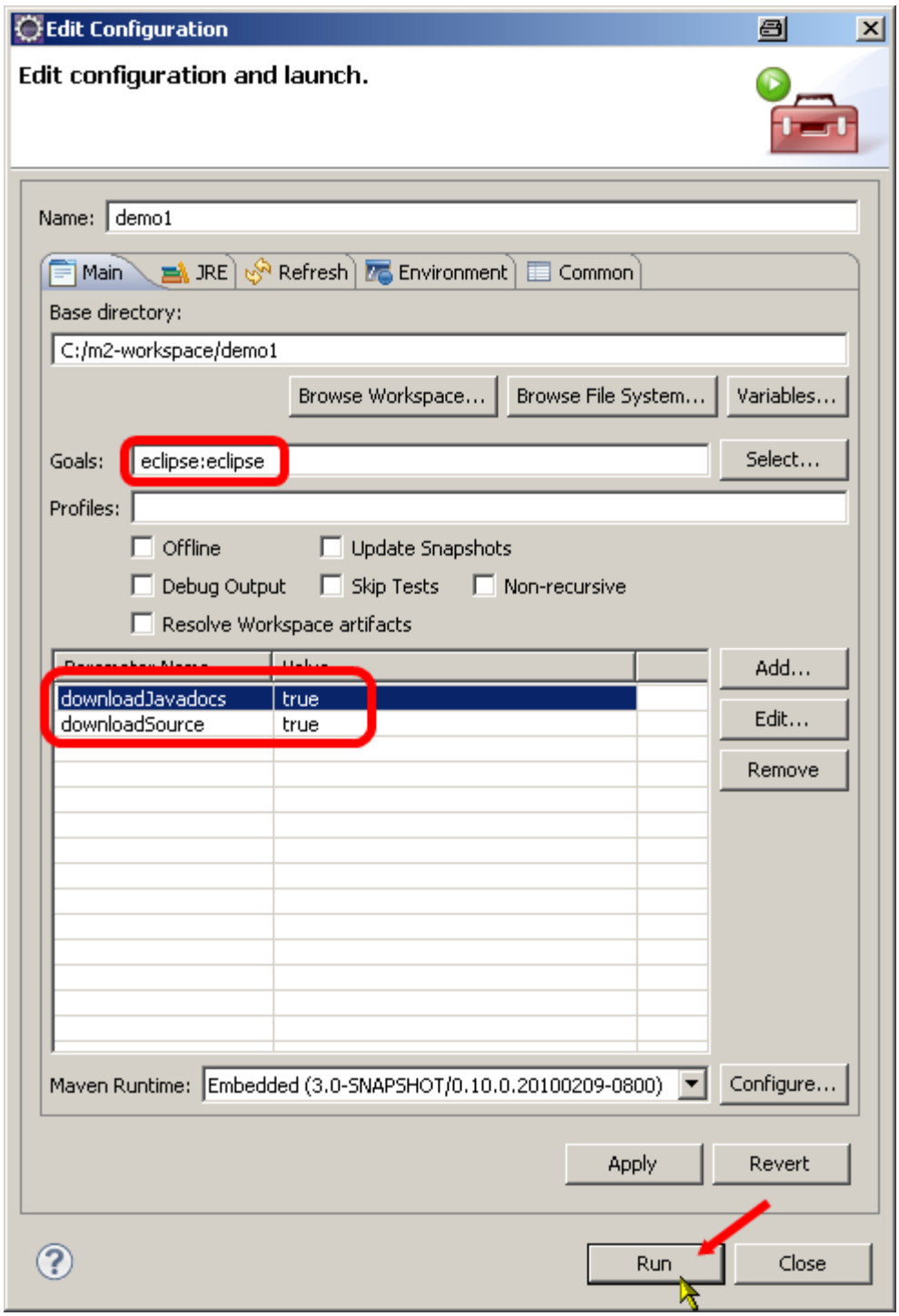

Abb.20.: Aufruf des Goals eclipse:eclipse

Durch den Aufruf des Goals "`eclipse:eclipse`" werden die Eclipse Projektdateien „.classpath" und „.project" erzeugt.
Alternative können Sie auch die Funktion „Project > Update All Maven Dependencies" in der Eclipse IDE aufrufen.

Spring Framework und Maven

Weitere Informationen zu Maven und Spring finden Sie im SpringSource Team Blog[29].

[29] http://blog.springsource.com/2011/01/17/green-beans-getting-started-with-maven-and-spring/

Kontakt & Lizenzbedingungen

Autor:

Torsten Friebe, mailto:tfriebe@gmx.net