

Software Engineering 2

Praktikum 7 - Spring Framework

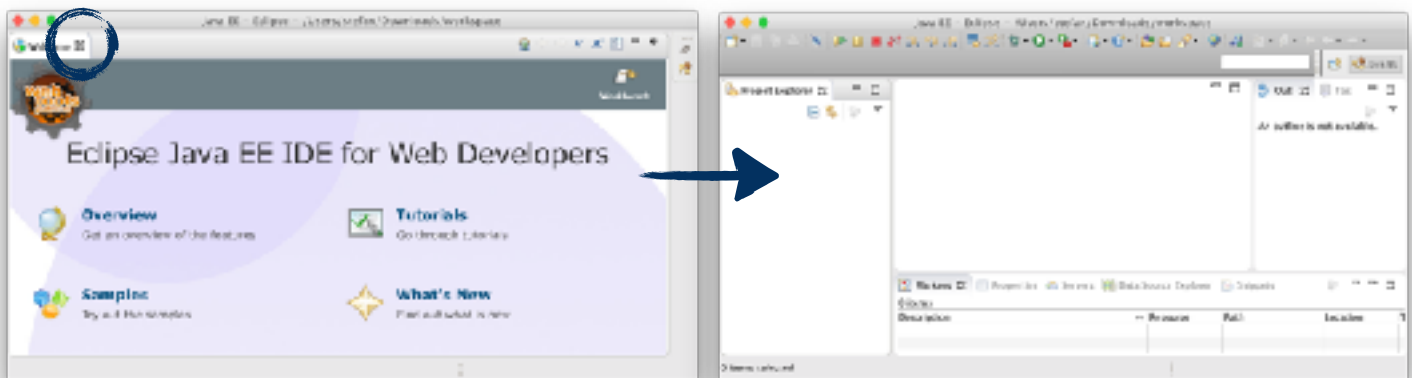
Aufgabe

Schritt 1 - Eclipse installieren

Installieren Sie ein aktuelles Java JDK (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>) und eine aktuelle Version des »Eclipse IDE for Java EE Developers« (<http://www.eclipse.org/downloads/eclipse-packages/>). Sie können auch eine bereits existierende Eclipse-Installation verwenden, z.B. die auf den Laborrechnern installierte Version!

Schritt 2 - Eclipse starten

Eclipse starten und auf Nachfrage einen neuen Workspace anlegen, z.B. auf dem Desktop. Achten Sie bei den Praktikumsrechnern darauf, den Workspace auf einem Netzwerklaufwerk oder USB-Stick anzulegen, damit Ihre Arbeiten nicht nur auf diesem Rechner verfügbar sind!



Schritt 3 - Maven Erweiterung installieren

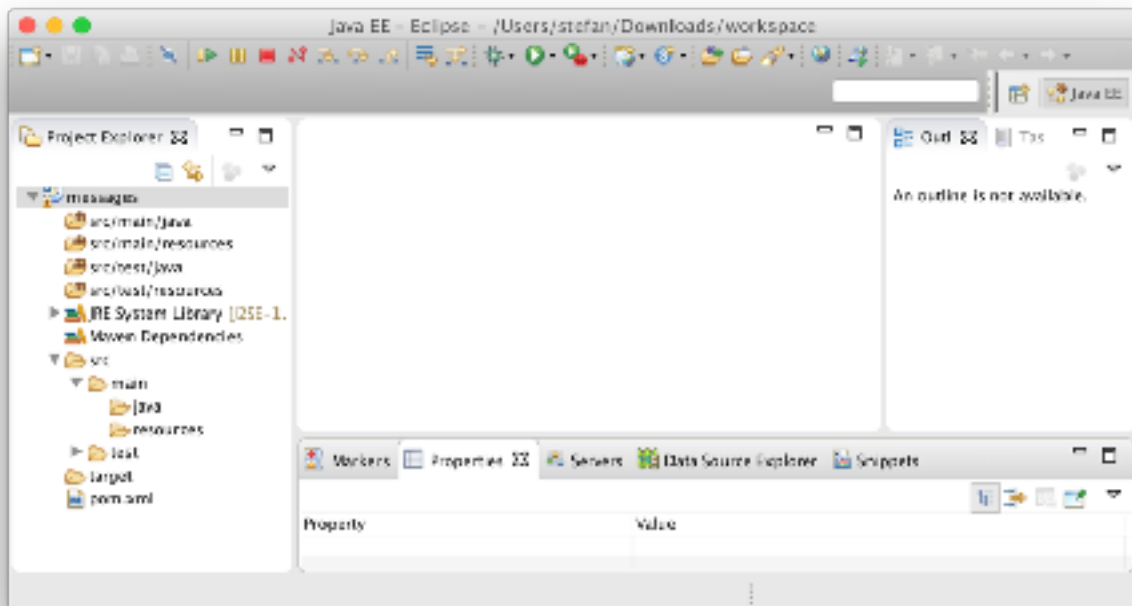
In der JEE Version von Eclipse ist die Maven Erweiterung bereits installiert (als »Maven Integration for Eclipse - m2e« im Menü Help → Installation Details → Features). Wenn Sie eine andere Eclipse-Version haben, können Sie »Maven Integration for Eclipse« über das Menü Help → Eclipse Marketplace... installieren.

Schritt 4 - Projekt anlegen

Menü File → New → Maven Project

»Create a simple project« ankreuzen → Next drücken

Group Id »de.hfu« → Artifact Id »messages« → Packaging »jar« → Finish drücken



Schritt 5 - XML-Dateien einfügen

Ersetzen Sie die Datei »pom.xml« im Hauptverzeichnis des Projekts durch die Version aus dem Veranstaltungsbereich von Felix. Damit wird ein wesentlicher Aspekt durch Maven vorbereitet:

- notwendige Bibliotheken — z.B. das Spring Framework — werden heruntergeladen und stehen dem Projekt zur Verfügung (»Maven Dependencies«)

Zusätzlich müssen Sie die Konfiguration für das Spring Framework anlegen. Kopieren Sie dazu aus Felix...

- ... die Datei »application.properties« in das Projektverzeichnis nach src/main/resources
- ... die Datei »RemotingConfiguration.java« in das Projektverzeichnis nach src/main/java in das Package de.hfu (muss vorher erstellt werden)

Schritt 6 - Klassen erstellen

Legen Sie in dem Package de.hfu zwei Java-Klassen an:

- MessagePrinter (erstmal leer)
- MessageApp (mit main-Methode)

Die Klasse MessagePrinter soll den MessageService per „Dependency Injection“ injiziert bekommen. Legen Sie dafür eine Instanzvariable vom Typ MessageService und eine Setter-Methode in der Klasse MessagePrinter an. Annotieren Sie an den richtigen Stellen mit @Component und @Autowired. (siehe Abschnitt »Inversion of Control« in der Vorlesung)

Legen Sie noch eine zusätzliche, beliebig benannte Methode im MessagePrinter an, die eine Liste aller Nachrichten aus dem MessageService ausliest und mittels `System.out.println()` in die Konsole ausgibt.

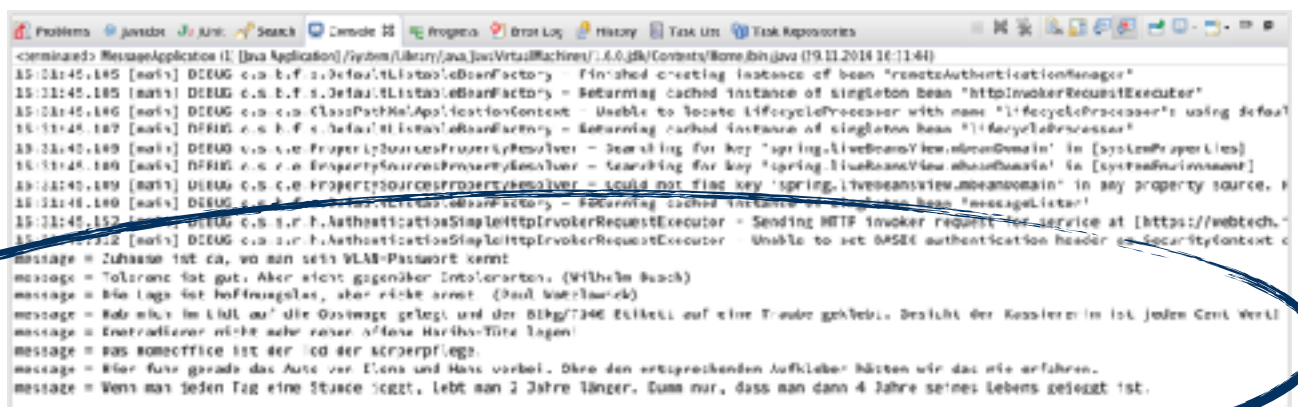
Schritt 8 - Main-Methode implementieren

Die Klasse MessageApp muss nur aus einer statischen `main()`-Methode bestehen, in der der Spring Kontext erzeugt wird (siehe Folie »Spring Context«). Aus dem Spring Context holen Sie Ihren MessagePrinter und rufen darauf Ihre selbst erstellte Methode zur Ausgabe aller Nachrichten auf. Annotieren Sie die Klasse MessageApp mit der Annotation `@SpringBootApplication`.

Schritt 9 - Anwendung ausführen

Rechtsklick mit der Maus auf Ihre Klasse MessageApp → Run as → Java Application

Die Anwendung wird gestartet, im Hintergrund kontaktiert Spring den MessageService auf dem Server und holt die Nachrichten, die Ausgabe in der Konsole sieht ungefähr so aus:



```
<terminated> MessageApplication [L [Java Application] /system/Library/Java/JavaVirtualMachines/1.6.0_80/Contents/Home/bin/java (19.11.2014 16:11:44)
15:11:45.185 [main] DEBUG c.s.b.f.s.DefaultListableBeanFactory - Finished creating instance of bean "remoteAuthenticationManager"
15:11:45.185 [main] DEBUG c.s.b.f.s.DefaultListableBeanFactory - Returning cached instance of singleton bean "httpInvokerSecurityExecutor"
15:11:45.186 [main] DEBUG c.s.c.c.ClassPathXmlApplicationContext - Unable to locate LifecycleProcessor with name "LifecycleProcessor" using default
15:11:45.187 [main] DEBUG c.s.b.f.s.DefaultListableBeanFactory - Returning cached instance of singleton bean "LifecycleProcessor"
15:11:45.189 [main] DEBUG c.s.c.c.PropertySourcesPropertyResolver - Searching for key 'spring.liveBeansView.mbeanDomain' in [systemProperties]
15:11:45.189 [main] DEBUG c.s.c.c.PropertySourcesPropertyResolver - Searching for key 'spring.liveBeansView.mbeanDomain' in [systemEnvironment]
15:11:45.189 [main] DEBUG c.s.c.c.PropertySourcesPropertyResolver - Could not find key 'spring.liveBeansView.mbeanDomain' in any property source.
15:11:45.190 [main] DEBUG c.s.b.f.s.DefaultListableBeanFactory - Returning cached instance of singleton bean "messagePrinter"
15:11:45.192 [main] DEBUG c.s.s.s.f.AuthenticationSimpleHttpInvokerRequestExecutor - Sending HTTP invoker request for service at [https://webtech.
15:11:45.192 [main] DEBUG c.s.s.s.f.AuthenticationSimpleHttpInvokerRequestExecutor - Unable to set BASIC authentication header on SecurityContext c
message = Zuhause ist da, wo man sein WLAN-Passwort kennt
message = Toleranz ist gut. Aber nicht gegenüber Intoleranten. (Willy Reich)
message = Die Tage ist hoffnungslos, aber nicht arm. (Paul Watzlawick)
message = Kolchak im LIDL auf die Gustavstraße gelegt und der BIKK/7346 Etikett auf eine Tüte geklebt. Besitzt der Kassierer im ist jedem Cent wertl
message = Entschleunigung nicht mehr von der offenen Hardware-Tüte legen!
message = Das Komettische ist der Tod der Körperpflege.
message = Hier fahr gerade das Auto von Elena und Hans vorbei. Dann den entsprechenden Aufkleber hätten wir das nie erfahren.
message = Wenn man jeden Tag eine Stunde isst, lebt man 2 Jahre länger. Wenn man, dass man dann 4 Jahre seines Lebens isst.
```