Moduł 1: Wprowadzenie

Spring Framework

Spring powstawał przeszło 10 lat, początkowo jako alternatywa dla przerośniętej i nadmiernie skomplikowanej specyfikacji EJB 2.x. Jednakże na przestrzeni ostatnich 10lat pod nazwą Framework ewoluował i obecnie pod nazwą Spring rozumiemy szeroki zestaw narzędzi wspomagających pracę programisty. Narzędzi, których rdzeniem jest właśnie Spring Framework.

Spring Framework został zapoczątkowany w 2000 roku i w założeniu miał być remedium na problemy które Rod Johnson (autor i twórca narzędzia) napotykał w swojej codziennej pracy konsultanta i programisty. Pamiętajmy, iż początek 21 wieku to okres 'panowania' architektury J2EE ze wszystkimi jest wadami i bolączkami, dla których alternatywą miał być właśnie Spring. Podstawowymi założeniami narzędzia było:

- Skupienie się na obszarach które przez inne narzędzia traktują po macoszemu; platforma J2EE oferowała szereg doskonałych narzędzi w zakresie utrwalania (persistence), zdalnych wywołań (remoting), www (web frameworks). Jednakże integracja tych narzędzi była kwestią problematyczną. Spring oferował kompletne rozwiązanie umożliwiające połączenie różnych frameworków.
- Brak wymagań infrastrukturalnych. Aplikację opartą o Spring Framework od samego początku można było uruchomić na najzwyklejszej platformie Java - bez konieczności instalowania dodatkowych serwerów aplikacji.
- Łatwość użycia - w porównaniu z J2EE. Od samego początku Spring Framework po prostu działał, nie wymagając ponoszenia dodatkowych nakładów na konfigurację platformy.
- Testowalność, która między innymi wynika z paradygmatu POJO programming. Co więcej, w przeciwieństwie do większość serwerów aplikacji, Spring dostarczał narzędzia do testowania, umożliwiając zastępowanie pewnych elementów platformy namiastkami (ang. mock), dzięki czemu można było uruchomić aplikację bez konieczności wdrażania jej na środowisko testowe. Brak takiego wsparcia jest jednym z większych problemów na platformie JEE. Czasami można odnieść wrażenie, że w czasie projektowania i prac nad specyfikacja nikt nie pomyślał o sposobach testowanie tych rozwiązań, zostawiając problem w całości programistom i ich lepszym lub gorszym pomysłom. W przypadku Springa od pierwszych wydań widać pragmatyzm i doświadczenie autorów w zakresie testowania jednostkowego i integracyjnego.

Spring stał się alternatywą dla J2EE, do działania nie wymagał zaawansowanych serwerów aplikacyjnych i mógł być z powodzeniem wdrażany przy użyciu lekkich kontenerów takich jak Tomcat i Jetty. Spring od samego początku zapewnił daleko idącą modularność; zależnie od wymagań biznesowych umożliwiał zastosowanie tylko niektórych elementów, bez konieczności instalacji i używania przeładowanych serwerów aplikacji (oferujących cały zestaw narzędzi zgodnych ze specyfikacją EJB). Budując prostą stronę internetową niekoniecznie będzie potrzeba używania kolejek JMS, natomiast wsparcie dla transakcji może okazać się bardzo przydatne. Serwery aplikacji z reguły

monolityczne w swojej budowie byty, uniemożliwiające łatwe wyłączenie pewnych elementów - Spring pozwalał na to od samego początku.

Rozwój frameworku rozpoczął się od części obecnie znanej jako spring-mvc (implementacja wzorca Model View Controller) - stamtąd pochodzą najstarsze klasy narzędzia. Uważa się że to właśnie Spring jest protoplastą paradygmatu POJO programming - opisanego m.in. w książce Roda Johnsona: J2EE Development without EJB. Spring od lutego 2003 jest projektem otwartym (open-source).

Początkowa alternatywność wobec rozwiązań opartych o EJB przestała mieć obecnie znaczenie. Platforma Spring i JEE od pewnego czasu zbliżają się do siebie, jeżeli chodzi o łatwość tworzenia aplikacji oraz w kwestii produktywności programistów (JEE6 jest tutaj kamieniem milowym) - Spring wciąż jawi się jako rozwiązanie daleko bardziej modularne i lżejsze.

POJO Programming

POJO to skrót od Plain Old Java Object, czyli obiekt który nie implementuje żadnych specyficznych interfejsów (np. tych związanych z platformą, z EJB 2.x), obiekty nie związane z jakimkolwiek środowiskiem, serwerem, frameworkiem (ang. framework agnostic). Jedyną i najważniejszą odpowiedzialnością takich obiektów jest przechowywanie danych i/lub realizacja usług (funkcjonalności) biznesowych.

Poza daleko większą czystością kodu, takie obiekty są w naturalny sposób oderwane od środowiska w którym mają zostać wdrożone. Dzięki temu do testowania aplikacji nie jest wymagany kontener (serwer aplikacji), niekonieczna jest także baza danych. Co więcej – tworzone klasy mogą być użyte poza główną aplikacją, umożliwiając łatwe współdzielenie kodu pomiędzy różnymi środowiskami.

Elementy frameworka

Od samego początku Spring składał się z więcej niż jednego modułu.

Moduły to:

- Kontener wstrzykiwania zależności, czyli zarządzanie komponentami, ich tworzeniem, wykorzystaniem i niszczeniem,
- Kontekst aplikacyjny, będący warstwą abstrakcji umożlwiającą dostęp do zasobów dostarczanych przez serwer aplikacji,
- Aspekty, wsparcie dla programowania aspektowego m.in. poprzez interceptory,
- Zarządzanie transakcjami,
- Abstrakcja warstwy dostępu do danych poprzez obsługę Hibernate, iBatis, JDBC Template i kilka innych pomocnych narzędzi,
- Implementacja wzorca MVC (Model-Widok-Kontroler, ang. Model-View-Controller), generyczny framework webowy mogący działać niezależnie, integrując się z czystym JSP lub w połączeniu dowolnym innym narzędziem do generowania widoków (np. Velocity),
- Obsługa zdalnych wywołań, poprzez web service, RMI.

Spring obecnie

Na przestrzeni ostatnich 10 lat zmieniał się i rozwijał - tak jak rozwijały się technologie i inne narzędzie. Następował nie tylko rozwój samego frameworka, ale także szeregu narzędzi

wspierających programistę. Rod Johnson od zawsze podkreślał że Spring oparty jest na trzech filarach: elastyczność (portability), efektywność (productivity) oraz innowacyjność (innovation). Nie zmieniają się one na przestrzeni lat i pozostają aktualne zarówno w odniesieniu do pierwszych wersji narzędzia sprzed 10 lat, jak i wszystkich obecnych działań i inicjatyw SpringSource - czyli organizacji zajmującej się rozwojem frameworku.

Elastyczność na przestrzeni lat rozumiana była jako możliwość przenoszenia kodu aplikacji pomiędzy różnymi środowiskami (włącznie ze środowiskami testowymi): ta sama aplikacji działała poprawnie na różnych serwera aplikacji oraz w różnych kontenerach, gdzie Spring Framework stanowił warstwę abstrakcji oddzielającą aplikację od konkretnej implementacji środowiska. Przenaszalność aplikacji pomiędzy serwerami J2EE okazała się być iluzoryczna i odpowiednia abstrakcja dostarczana przez framework umożliwiała np. lokalne testowanie aplikacji w lżejszym środowisku niż docelowa platforma produkcyjna. Podobna sytuacja ma miejsce także obecnie, z tym że serwery aplikacji zostały zastąpione przez rozwiązania ‘w chmurze’ (cloud computing). Spring nadal pozostaje atrakcyjną warstwą abstrakcji, oddzielającą aplikację od specyficznych aspektów platformy (jak np. rozproszona baza danych).

Efektywność początkowo rozumiana była poprzez zwolnienie programisty z konieczności użycia skomplikowanego modelu J2EE i zastąpieniem go zwinnym (lekkim) podejściem oferowanym przez Spring Framework (ang. lightweight). Natomiast w dniu dzisiejszym to przede wszystkim szerokie wsparcie dla programisty we wszystkich aspektach związanych z wytwarzaniem oprogramowania:

- Spring Roo do powtarzalnego generowania kodu aplikacji na przykład na podstawie bazy danych,
- Zintegrowane środowisko programistyczne: Spring Tool Suite - narzędzie oparte o środowisko Eclipse lecz wychodzące daleko poza nie, dodając wsparcie dla szeregu narzędzi związanych ze Springiem.

Od zawsze założeniem związanym ze Springiem było upraszczanie wszystkie co możliwe (make things simpler) - i to nie zmieniło się do dnia dzisiejszego. Chodzi przecież o to aby tworzyć oprogramowanie rozwiązujące konkretne problemy, dostarczające wartość dodaną, skupiać się na celach biznesowych a nie być rozpraszanym przez kwestie związane z serwerem aplikacji, bądź przez specyficzne aspekty platformy.

Na koniec innowacja, czyli przede wszystkim zaproponowanie nowego modelu POJO programming który z powodzeniem zastąpił mocno skomplikowane podejście znane z J2EE - z czasem podejście to zostało szerzej zaadaptowane i stało się także elementem specyfikacji EJB 3.x. SpringSource to obecnie także Groovy czyli jeden z funkcyjnych języków obecnych na platformie JVM oraz Grails (Groovy on Rails) - odpowiedź świata Javy na fenomen Ruby on Rails. Mogłoby się wydawać że są to już rzeczy bardzo subtelnie powiązane ze Spring Framework - nic bardziej mylnego. Omawiany framework jest kluczowym elementem każdego z tych rozwiązań i umiejętność posługiwania się nim leży u podstaw zrozumienia całego szeregu narzędzi dostarczanych przez SpringSource

Spring Framework a EJB

Jak już podkreśliłem, Spring projektowany był jako alternatywa wobec EJB. Można jednak powiedzieć, że wraz z biegiem czasu Spring i EJB stały się rozwiązaniami komplementarnymi; każdy z frameworków ma swoje wady i zalety, wymusza pewne specyficzne decyzje architektoniczne. Spring

z małego i lekkiego narzędzia rozrósł się o masę modułów i dodatków. Z kolei twórcy EJB dostrzegli ociężałość technologii i każde kolejne wydanie specyfikacji EJB powodowało, że platforma JEE stawała się lżejsza i elastyczniejsza. Nie da się jednak pominąć kilku istotnych różnic pomiędzy tymi dwoma technologiami:

Modularność: Java Enterprise Edition (czyli platforma, która wykorzystuje EJB jako komponenty biznesowe) jest całością i nie ma możliwości skorzystania tylko z niektórych elementów. JEE łączy się z konkretnym zestawem API z których należy korzystać: EJB (komponenty), JPA (dostęp do bazy danych), JMS (kolejny, asynchroniczność), JSF (warstwa prezentacji). Co więcej, wszystkie te elementy są obecne w każdym serwerze aplikacji zgodnym z JEE, nawet jeżeli rozwiązanie które budujemy nie ma potrzeby z nich korzystać.

Podejście Spring'owe jest diametralnie inne; Spring umożliwia dowolne łączenie narzędzi i bibliotek. Nacisk stawiany jest na ograniczenie ilość wykorzystywanych modułów, które włączane są do aplikacji dopiero gdy zachodzi taka potrzeba. Co więcej, Spring nie wymusza korzystania z konkretnego API i pozostawia dowolność programiście (architektowi). Nie ma znaczenie czy dostęp do bazy danych zostanie zrealizowany poprzez JPA (Hibernate) czy poprzez iBatis lub z wykorzystaniem bezpośredniej komunikacji po JDBC.

Częstotliwość zmian: Spring żyje, nowe wersje biblioteki pojawiają się z dużą regularnością; rozwojem frameworka zarządza jedna firma: SpringSource i ona jest odpowiedzialna za to jakie nowe elementy pojawiają się w narzędziach. Wciąż jednak framework, jako taki, pozostaje narzędziem o otwartym kodzie źródłowym, opartym na otwartej licencji

Alternatywnie, specyfikacja EJB rodzi się w bólach, nowe wersje pojawiają się raz na kilka lat. Przez ten czas programiści zmuszani są do korzystania z rozwiązań coraz bardziej archaicznych lub zaczynają korzystać z rozszerzeń oferowanych przez poszczególnych producentów serwerów aplikacji (ang. *verndor specific extensions*). Powoduje to jednak przywiązanie do konkretnego producenta / serwera aplikacji. Czasem nawet do konkretnej wersji serwera.

Przenaszalność: Spring jest całością samą w sobie, decydując się na użycie tego frameworka, używamy także dostarczanego przez Spring kontenera – pomostu pomiędzy aplikacją a serwerem (środowiskiem) gdzie aplikacja jest uruchamiana. W przypadku EJB sprawa jest bardziej skomplikowana. Oparcie platformy JEE o szereg standardów pozwalało (w teorii) na dowolne przenoszenie aplikacji pomiędzy serwerami aplikacji dostarczanymi przez różnych producentów. W praktyce jednak (co podkreśliłem już w poprzednim punkcie), specyfikacja pozostawiła stosunkowo szerokie pole niedookreślenia, pozostawiając dużą dowolność producentom serwera. Poprzez to niezmiernie ciężko spotkać aplikację niezależną od tzw. *vendor extensions*, czyli elementów ściśle zintegrowanych z konkretnym serwerem aplikacji.

Podstawowym wnioskiem płynącym z takiego porównania jest to że te same wymagania jesteśmy w stanie zrealizować zarówno wykorzystując EJB (platformę JEE) oraz Springa (wraz z otaczającymi modułami). Spring oferuje bogactwo opcji, a co za tym idzie konieczność podejmowania szeregu decyzji projektowych, decyzji wymagających doświadczenia oraz praktyki. Natomiast w przypadku EJB zmuszenia jesteśmy do skorzystania z konkretnego zestawu narzędzi; jeżeli te narzędzia są odpowiednie do naszych potrzeb – tym lepiej, nie mamy bowiem możliwości zmian. Z powodu takiej

a nie innej konstrukcji platformy, zwolnieni zostaliśmy z konieczności podejmowania szeregu nietrywialnych decyzji projektowych i zaistniałą sytuację musimy zaakceptować taką jak jest.

Program szkolenia

Tak jak już pisałem, Spring Framework stanowi rdzeń wszystkich rozwiązań dostarczanych przez SpringSource. Dodatkowo, sam w sobie, jest bardzo potężnym narzędziem do tworzenia aplikacji w języku Java. W niniejszym szkoleniu skupimy się na najważniejszych aspektach frameworku, które zostaną podzielone na następujące moduły:

- Głównym elementem Spring Framework jest kontener który realizuje proces wstrzykiwania zależności (odwrócenie kontroli - ang. Inversion of Control). W jaki sposób uruchomić kontener, jak go skonfigurować, dodać komponenty i w jaki sposób opisać zależności pomiędzy komponentami: poprzez adnotacje, poprzez konfigurację w pliku XML lub popraz skrypt konfiguracyjny.
  Komponenty dodane do kontenera są przez niego zarządzanie. Omówiony zostanie cykl życia komponentów; w jaki sposób komponenty są tworzone, zarządzane, w jaki sposób programista może wpłynąć na poszczególne działania kontenera.
- W kolejnej części dokładniej zostanie umówione wstrzykiwanie zależności. Jest to jeden z istotniejszych elementów Spring dlatego warto poświęcić więcej uwagi i dokładnie przeanalizować sposoby budowania zależności, różnych możliwości tworzenia obiektów itd.
- Aplikacje wymagają sprawdzenia poprawności wprowadzanych danych (walidacji). Spring w tym zakresie implementuje jeden ze standardów Javy (JSR-303); dokładnie zostanie omówione jak działa walidacja, w jaki sposób ją skonfigurować i wykorzystać w aplikacji.
- Kluczowym elementem większości systemów informatycznych jest baza danych. W tej kwestii Spring nie odkrywa koła na nowo (nie narzuca jakiegokolwiek, własnego sposobu komunikacji z bazą danych) tylko umożliwia połączenie poprzez najchętniej przez nas wykorzystywane narzędzie; czy to Hibernate, czy iBatis, czy bezpośrednie JDBC. W kolejnym module przyjrzymy się dokładnie w jaki sposób powinno to zostać zrealizowane w aplikacji opartej o Spring Framework
- Kolejny moduł będzie dotyczył już tematów trudniejszych, rzadziej spotykanych - mianowicie aspekty; deklaratywne rozbudowywanie elementów systemu o dodatkowe funkcjonalności, bez konieczności ingerencji w sam kod klasy / moduły. Programowanie w oparciu o aspekty jest nagminnie wykorzystywane w projektowaniu i budowie szeregu narzędzi i frameworków - rzadziej spotyka się je w codziennych projektach. Spring niejako wprowadził te zagadnienia pod strzechy i bardzo mocno ułatwił ich użycie. W tym modelu przyglądniemy się jak konfigurować i używać aspektów.
- Przez całe szkolenie komunikowaliśmy się z naszą aplikacją poprzez zwykły interfejs tekstowy, poprzez konsolę. Nie jest to oczywiście jedyny sposób; Spring oferuje cały szereg narzędzi umożliwiających publikację aplikacji biznesowej poprzez jeden lub wiele przyjaznych kanałów. W module pokażę w jaki utworzyć aplikację WWW, z wykorzystaniem dostarczanego przez Spring narzędzia Spring MVC lub co zrobić aby wykorzystać alternatywny framework jak np. Struts czy JSF. Równie łatwo komponenty biznesowe opublikować jako web service - i to także zostanie pokazane w module.
- Na sam koniec, choć w zasadzie od tego należałoby zacząć, omówimy sposoby pisania testów jednostkowych. Jak już zaznaczyłem na samym początku, Spring stanowi doskonałą

abstrakcję oddzielającą moduły biznesowe od specyficznych aspektów platformy na której aplikacja jest wdrażana. Dzięki tej abstrakcji nie ma najmniejszych problemów aby platformą wdrożenia były narzędzia do testów jednostkowych. Dodatkowo, paradygmat *POJO programming* nie wymusza implementacji specyficznych interfejsów, dostarczanych przez platformę (jak miało to miejsce w EJB 2.x) i poprzez to znakomicie ułatwiał testowanie.

Środowisko programistyczne

Wszystkie przykłady zaprezentowane w niniejszym szkoleniu zostały stworzone przy użyciu Spring Tool Suite - czyli środowiska opartego o platformę Eclipse, dostarczanego przez SpringSource. Wszystkie przykłady z powodzeniem będą działać w dowolnym innym środowisku, przy czym nieocenioną zaletą STS jest szereg ułatwień związanych z frameworkiem - na przykład gotowe już wzorce dla szeregu aplikacji opartych o Spring Framework. Wykorzystamy tę właściwość aby rozpocząć pierwszy projekt; tworząc nowy “Spring Template Project”. Z szeregu dostępnych wzorców wybieramy najprostszy: Sample Spring Utility Project, który będzie zawierał dwa, najważniejsze z naszego punktu widzenia, elementy: spring-context (czyli kontekst aplikacyjny) oraz spring-test (czyli zestaw narzędzi do testowania aplikacji). Co więcej - tak utworzony projekt jest w pełni funkcjonalną aplikację Spring Framework.

W kolejnym module...

Mam nadzieję, że powyższe wprowadzenie zarysowało jakiego typu narzędziem jest Spring framework i jakie może być jego zastosowaniem. W kolejnej części pokażę jak rozpocząć pracę z frameworkiem, jak utworzyć projekt, uruchomić, przedstawię podstawowe kroki potrzebne do napisania aplikacji opartej o Springa.