Integracja technologii mobilnych i systemów klasy Enterprise

Wojciech Klicki Konrad Starzyk

Spis treści

Rozdział 1

Integracja w środowiskach Enterprise

Oprogramowanie Enterprise jest pojęciem dość szerokim, opisującym systemy przeznaczone dla przedsiębiorstw, których działanie odwzorowuje zachodzące procesy biznesowe. Niekiedy pojęcie to odnosi się do oprogramowania pisanego na zamówienie lub też do rozbudowanych pakietów wspierających określone czynności, np. kontakty z klientami lub księgowość. Rzadko się jednak zdarza, by istniał jeden system który potrafiłby spełnić wymagania klienta, a nawet jeżeli, z różnych przyczyn firma może nie chcieć go wdrożyć. Tak więc nawet w jednym przedsiębiorstwie często można się spotkać z sytuacją w której działa wiele niezależnych aplikacji, nierzadko pisanych przez różne firmy, które muszą się ze sobą komunikować w celu wymiany danych. Jeszcze do niedawna mówiąc o integracji mieliśmy na myśli właściwie wyłącznie serwery aplikacyjne. Obecnie obowiązującym trendem jest postępująca miniaturyzacja i wzrost mocy obliczeniowej urządzeń mobilnych, które udostępniają klientom usługi dostępne do tej pory wyłącznie za pośrednictwem komputera stacjonarnego. Tworząc aplikację na urządzenia przenośne, która

będzie współpracowała z istniejącymi już systemami, napotykamy jednak na problem komunikacji.

1.1 Istniejące podejścia do integracji

Rozważając możliwe sposoby integracji technologii mobilnych z rozbudowaną aplikacją klasy Enteprise musimy zwrócić uwagę na udostępniane z obu stron mechanizmy.

1.1.1 Transfer plików

W tym przypadku dane są zapisywane i odczytywane z plików. Niezbędna jest wspólna przestrzeń dyskowa lub sposób przesyłania (np. FTP). Z tego powodu musimy odrzucić takie rozwiązanie jako niedające się zaimplementować w środowisku mobilnym.

1.1.2 Wspólna baza danych

Dane są składowane we wspólnej bazie danych. Ze względu na zawodność transferu oraz długi czas przesyłania danych pomiędzy urządzeniem mobilnym a serwerem, a tym samym konieczność długiego oczekiwania, rezygnujemy z takiego rozwiązania.

1.1.3 RPC

Zdalne wywoływanie procedur (ang. Remote Procedure Call) jest mechanizmem który pozwala wykonywać procedury udostępniane przez zdalny system. Założeniem wywołań RPC jest synchroniczność, która wprawdzie nie oznacza, że musimy czekać na rezultat działania (możemy założyć, że wyłącz-

nie zlecamy wykonanie procedury, pobierając wynik potem),ale że zakładamy niezawodność komunikacji między urządzeniami.

1.1.4 Wiadomości

Komunikacja za pomocą wiadomości jest asynchroniczna, dodatkowo dobry system przesyłania wiadomości gwarantuje jej dostarczenie do adresata, lub informację o jej niedostarczeniu. Musimy pamiętać, że warunki w jakich możemy komunikować się z urządzeniem przenośnym są dalekie od doskonałych, a mimo to użytkownik oczekuje pełnej niezawodności systemu. To sprawia, że należy go tak zaprojektować, aby przerwy w komunikacji były jak najmniej odczuwalne. Systemy komunikacyjne takie jak RPC,RMI czy CORBA projektowane były dla warunków w których problemy komunikacyjne są sytuacją nadzwyczajną. Tymczasem w przypadku łączności bezprzewodowej utrata połączenia nie jest niczym czego nie moglibyśmy się spodziewać. W tym momencie idea przesyłania wiadomości wydaje się skutecznym rozwiązaniem tych problemów. Podczas tworzenia przykładowej implementacji napotkaliśmy jednak na problemy, które uniemożliwiły wykorzystanie go do integracji która jest przedmiotem tej pracy. Tym co ogranicza możliwość wykorzystania takiego

1.2 Zdalne wywoływanie procedur - Webserwisy

[tu coś trzeba by napisać]

1.3 Systemy oparte na przesyłaniu wiadomości

Systemy przesyłania wiadomości pozwalają na integrację systemów z uwzględnieniem specyfiki każdego z nich. Jednak jak każde rozwiązanie mają swoje mocne i słabe strony.

1.3.1 Niezależność integrowanych systemów

Komunikacja między systemami jest asynchroniczna, co z jednej strony powoduje, że system musi obsługiwać bardziej skomplikowane scenariusze w których odpowiedź na żądanie może nie pojawić się od razu, z drugiej pozwala na działanie komponentom względnie niezależnie do czasu odzyskania komunikacji. W razie awarii jednego z systemów, drugi może cały czas wysyłać wiadomości do kanału komunikacyjnego, które zostaną odebrane po odzyskaniu sprawności przez odbiorcę.

1.3.2 Routing

Wiadomości mogą pokonywać skomplikowaną drogę zanim dotrą do miejsca przeznaczenia. System wysyłający wiadomość nie musi wiedzieć jak dostarczyć ją do odbiorcy, jedyne co musi zrobić, to wstawić ją do kolejki. Daje to duże możliwości zmian w konfiguracji systemu, bez ingerowania w aplikację.

1.3.3 Transformacja

System przesyłania wiadomości może dokonywać zmian w wiadomościach zgodnie z ustalonymi regułami. Pozwala to na dalsze uniezależnienie integrowanych systemów od siebie: każdy system może wysyłać i odbierać wiado-

mości we właściwym dla siebie, natywnym formacie, podczas gdy za tranformację odpowiada kanał komunikacyjny.

1.3.4 Czas reakcji

Przesyłanie i przetwarzanie wiadomości ustępuje wydajnością integracji na poziomie danych, gdzie nie pojawia się narzut związany z wyłuskiwaniem danych i opakowywaniem ich do ustalonego formatu. Istnieje jednak wiele zastosowań gdzie szybkość nie gra najważniejszej roli,a przesyłane są jedynie niewielkie (ale częste) porcje informacji. Nawet sytuacje w których należy przesłać ogromną ilość danych, np. podczas migracji systemów mogą zostać obsłużone w procesach batchowych wykonywanych poza godzinami normalnego użytkowania systemu.

1.4 Sposoby połączenia

W systemach komunikujących się za pomocą wiadomości można wyróżnić 3 różne sposoby połączenia komponentów.

1.4.1 Kanał Point-to-point

W tym przypadku wiele procesów może być podłączonych do obu stron kanału komunikacyjnego. Z jednej strony kilka równolegle działających procesów może wysyłać wiadomości, z drugiej strony procesy mogą je odbierać. Ponieważ jednak równoległe przetwarzanie tej samej wiadomości przez kilka procesów mogłoby być nie porządane, kanał komunikacyjny dba o to by jedna wiadomość mogła być pobrana przez jeden tylko przez jeden z nich. Pozwala to na znakomite zrównoleglenie przetwarzania wiadomości przez proste dodawanie odbiorców i podłączanie ich do kanału komunikacyjnego. [tu będzie

rysunek point-to-point]

1.4.2 Kanał Publish-Subscribe

Jeżeli chcemy udostępnić pewne informacje szerszej grupie procesów możemy utworzyć kanał typu Publish-Subscribe. W tym przypadku procesy rejestrują się w menedżerze kolejki wybierając temat (Topic) z którego chciałyby otrzymywać wiadomości. Zadaniem menedżera kolejki jest zapewnienie by wszystkie zarejestrowane procesy otrzymały wstawioną do kolejki wiadomość, oczywiście każdy z nich jeden raz. [tu będzie rysunek publish-subscribe]

1.5 Platforma Java Enterprise Edition

Platforma JEE wykorzystywana jest do budowania rozbudowanych, wielokomponentowych, najczęściej komercyjnych systemów. Poza standardową biblioteką oferowaną przez Javę Standard Edition, zawiera Servlety, Enterprise Java Beany czy parsery XML. Jednym z najważniejszych komponentów jest API umożliwiające przesyłanie wiadomości - Java Message Service.

1.5.1 Java Message Service

Java Message Service jest standardem oferującym aplikacjom Javowym możliwość tworzenia, wysyłania, odbierania i czytania wiadomości. JMS jest jedynie specyfikacją, pozwalającą na jednolite korzystanie z różnych systemów przesyłania wiadomości. Porównując konkretną implementację z bazą danych, można powiedzieć, że JMS oferuje podobne zunifikowane API do przesyłania wiadomości, jak JDBC zunifikowany interfejs dostępu do bazy danych. Istnieje wiele implementacji JMS, zarówno komercyjnych jak i dostępnych bezpłatnie. Istnieją wersje stand-alone, jak np IBM WebsphereMQ

czy Apache ActiveMQ, jak również systemy przesyłania wiadomości wbudowane w serwery aplikacyjne. Niestety nie wszystkie pozwalają na wymianę wiadomości z urządzeniami mobilnymi. Jednym z celów pracy będzie przeanalizowanie możliwości oferowanych przez poszczególne implementacje pod kątem urządzeń mobilnych.

1.5.2 Serwer aplikacyjny

Chociaż wykorzystanie serwera aplikacyjnego do uruchomienia aplikacji obsługujących kolejki wiadomości jest jedną z możliwości, obok lekkich kontenerów jak np. Spring, wybierzemy serwer JBoss jako popularne i sprawdzone, a jednocześnie darmowe rozwiązanie implementujące wszystkie usługi JEE.

Rozdział 2

Mechanizmy integracyjne w technologiach mobilnych

2.1 Szybka droga do mobilności ?

Na wstępie dyskusji o mobilnej części platformy integracyjnej zwrócimy uwagę na dostępne na rynku urządzenia, które posiadają preinstalowane systemy operacyjne pozwalające na dodawanie nie przewidzianych przez twórców, zewnętrznych rozwiązań. Tego typu systemy operacyjne muszą się charakteryzować udostępnieniem na zewnątrz, do instalowanych przez nas aplikacji pewnego API pozwalającego na realizację podstawowych usług takich jak transfer danych czy ich składowanie w pamięci urządzenia. Na chwile obecną (drugi kwartał 2008 roku) na rynku dostępne są urządzenia działające w oparciu wymienione poniżej platformy mobilne :

- WMD - Windows Media Devices oparte o system operacyjny Windows Mobile opracowany przez firmę Microsoft

- Symbian - platforma spotykana głównie na smartfonach oferowanych przez firmy takie jak Nokia czy Samsung
- Blackberry - marka wypromowana prze firmę Research in Motion, oferuje urządzenia z preistalowanym systemem operacyjnym opartym na wirtualnej maszynie Java
- Android - system operacyjny oparty na Linuxie, możliwy do rozbudowywania głównie przez aplikacje J2ME (w obecnej chwili system jest w fazie testów)
- pozostałe autorskie systemy operacyjne, udostepniające standardową maszynę wirtualną Java

Po dokładnym przyjrzeniu się powyższym platformom nasuwa się jeden stanowczy wniosek - poza platformami opartymi na J2ME, wykonanie jednej, spójnej aplikacji spełniającej założenia interoperacyjności jest praktycznie niemożliwe. Jedynym rozwiązaniem jest przygotowanie kilku instancji tej samej aplikacji, które niestety nie będą miały ze sobą niczego wspólnego poza przyjętymi założeniami o podstawowej funkcjonalności. Wynika to między innymi z tego, że każda platforma oferuje inne możliwości i przy okazji wprowadza inne ograniczenia. Co więcej tego typu różnicowanie występuje również w obrębie samych platform. Na przykład J2ME posiada profile związane na stałe z modelami urządzeń, które decydują o tym, jakie funkcjonalności dostępne są dla programów. Wskazane powyżej ograniczenia techniczne wymuszają przy próbie wejścia na rynek aplikacji mobilnych,już na etapie planowania produktu, wybranie konkretnej platformy docelowej na jakiej oferowane będzie nasze rozwiązanie. Wraz z sukcesem rozwiązania, możliwa będzie dalsza ekspansja na pozostałe platformy. Przed tą ostateczną ewaluacją wartości pomysłu na produkt, jest bardzo ryzykownym podjęcie próby równoległego

wprowadzenia go na różnych, niekompatybilnych platformach. Zwiększa to znacznie próg wejścia oraz utrudnia poprawną kontrolę nad stopniem dojrzałości produktu. Co więcej przy słabym zarządzaniu funkcjonalnościami może wystąpić problem zróżnicowania pomiędzy różnymi wersjami naszej aplikacji. Tego typu problemy wpłyną w sposób bezpośredni na postrzeganie naszego produktu przez klientów, w szczególności gdy będziemy mieli do czynienia z korporacjami, w obrębie których stosowane są rózne rozwiązania mobilne. W wypadku naszego systemu integracyjnego naturalnym wyborem będzie J2ME. Na chwilę obecną jest jedynym standardem, który daje jakiekolwiek nadzieje na uzyskanie dominacji na rynku. Co więcej platforma Blackberry, która jest stworzona właśnie w oparciu o ten standard jest jedną z najbardziej popularnych, w firmach, które chcą podjąć działania mające na celu umobilnienie, występujących w ich obrębie, procesów biznesowych.

2.2 Jak się do tego zabrać ?

W momencie gdy mamy ustaloną już platformę docelową (J2ME) musimy przyjrzeć się temu co nam oferuje. Z punktu widzenia integracji najważniejsze dla nas są dwie cechy :

- możliwość do trwałego przechowywania danych,
- zdolności do komunikacji z systemami zewnętrznymi.

Pod pojęciem trwałego przechowywania danych kryje się niewrażliwość danych na przerwy w dostawie energii do urządzenia lub ewentualne przypadkowe restarty. Platforma J2ME oferuje nam tu standardowe rozwiązanie w postaci Persistance API dające dostęp do tzw. Record Store-ów pozwalających zapisywać dowolne dane w pamięci trwałej urządzenia. Każde urządzenie posiadające maszynę wirtualną java musi obowiązkowo posiadać tego

typu pamięć. API to jest bardzo prymitywne i nie pozwala na przechowywanie danych w sposób relacyjny, czy nawet nie pozwala na serializowanie obiektów. Jednak przy zastosowaniu odpowiedniego poziomu abstrakcji, poprzez opakowanie tych interfejsów w odpowiednie klasy jesteśmy w stanie uzyskać wygodny sposób na przechowywanie danych. Wymaga to niestety sporo dodatkowej pracy. Przy analizowaniu zdolności do komunikacji sprawy się znacznie komplikują. Istnieje kilka, dość znacząco różnych metod komunikacji, które możemy wykorzystać do synchronizacji danych z naszymi zewnętrznymi źródłami. U podstaw komunikacji leżą takie protokoły takie jak Direct TCP czy Http, które pozwalają na wymianę czystych nie przetworzonych danych binarnych, czy też tekstowych. Można je wykorzystać w sposób bezpośredni, jednak jest to niezwykle trudne i niewygodne. Z pomocą przychodzą nam protokoły komunikacyjne wyższego poziomu. Ich wadą jest to, że często nie są elementem standardowego wyposażenia urządzeń, na które będziemy pisali naszą aplikację. Nie jest to jednak problemem, ze względu na to, że u podstaw tychże protokołów wyższego poziomu zawsze leży protokół podstawowy, który jest dostępny bezpośrednio w standardowej bibliotece. Co więcej istnieją dziesiątki darmowych implementacji protokołów wyższego poziomu, które przy zerowym niemal koszcie, mogą zostać zaadoptowane do naszych potrzeb. Poniżej przedstawimy krótki przegląd rozwiązań wysokiego poziomu, które pozwalają na wygodną i łatwą do prześledzenia komunikację z systemami zewnętrznymi.

Istnieją cztery podstawowe sposoby, na które można zorganizować wysokopoziomową komunikację pomiędzy urządzeniem mobilnym a źródłem danych :

- Protokół binarny - najbardziej skomplikowana forma komunikacji, wymaga zaprojektowania własnego protokołu, a więc jest to metoda naj-

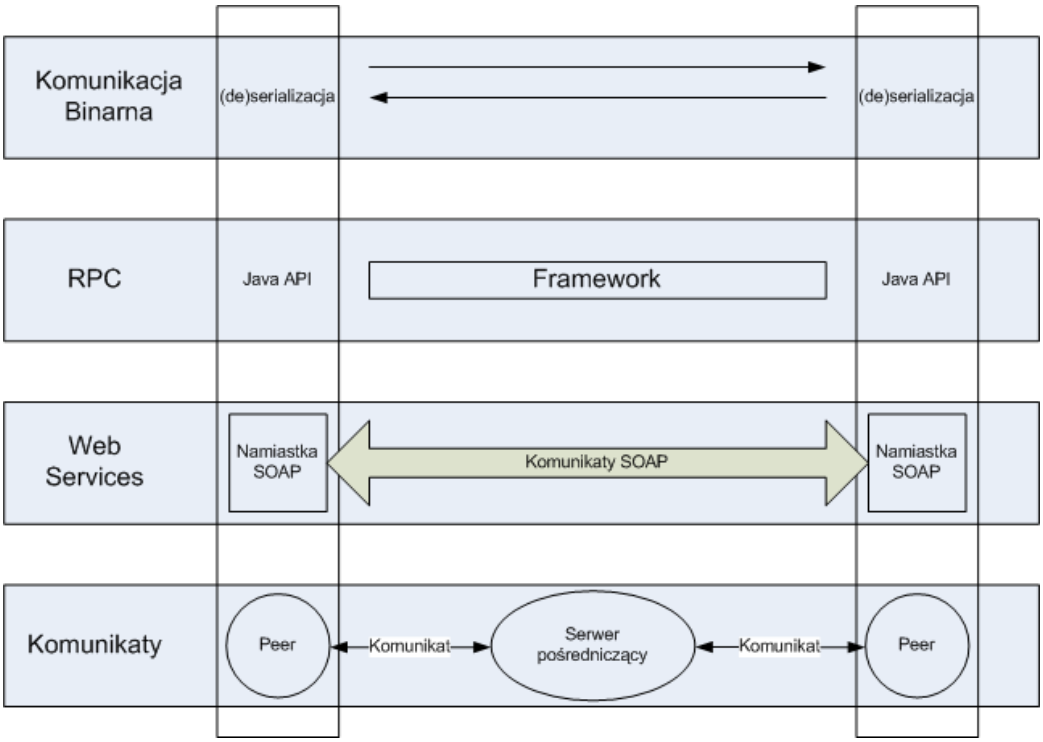
bardziej podatna na błedy programisty/projektanta.

- RPC - wykorzystanie narzędzi do tworzenia standardowej komunikacji opartej o zdalne wywoływanie procedur, dużo prostsze i odporniejsze na błedy niż poprzednia metoda, jednak wymaga bardzo bliskiego związania serwera z klientem, co zmniejsza interoperacyjność rozwiązania.

- Web Services - bardzo popularna metoda pozwalająca na zbudowanie komunikacji pomiędzy dowolnymi systemami w oparciu o standaryzowane, niezależne od końcowych systemów komunikaty XML. Budowa aplikacji w oparciu o Web Servces jest bardzo prosta i częściowo zautomatyzowana, jeśli zastosujemy odpowiednie frameworki. Największa zaletą jest tu całkowita interoperacyjność. Rozwiązanie to posiada niestety również wady w postaci zwiększonego ruchu w sieci oraz zwiększenia wymagań wobec sprzętu, na którym je wykorzystujemy.

- JMS - wprowadzenie elementu pośredniczącego pomiędzy urządzeniem a źródłem danych. Komunikacja polega na przesyłaniu wiadomości, które w razie awarii jednego z sytemów mogą zostać przechowane na kolejce komunikatów, która znajduje się na systemie pośredniczącym w komunikacji.

Z powyższego zestawinia można wyróżnić dwie technologie - Web Services oraz JMS. Obie pozwalają na wysoką interoperacyjność rozwiązania, co w sytuacji gdy mamy przed sobą perspektywę implementacji naszego rozwiązania na dziesiątkach typów urządzeń jest kluczowym zagadnieniem do rozpatrzenia. Wybór pomiędzy nimi sprowadza się do ustalenia czy niezbędna dla naszego projektu jest obecność systemu pośredniczącego, który jest w stanie zwiększyć niezawodność komunikacji. Czynnikiem decyducjącym tu jest ilość źródeł danych oraz sposób ich wykorzystywania przez użytkowników. Jeśli



Metody komunikacji

okazałoby się, że przez system będzie przechodziło dużo komunikatów, które będa mogły doprowadzić do przeciążenia systemów źródłowych, element pośredniczący buforujący komunikaty może okazać się niezbędny. Jednak w wypadku gdy każdy użytkownik będzie korzystał z danych, które dostępne są przez niezależne kanały - czyli systemy źródłowe nie będą posiadały wąskiego gardła, należałoby rozważyć skorzystanie z web services, które są znacznie tańszym rozwiązaniem niż JMS. Wynika to z tego, że większość implementacji WS dostępna jest za darmo w postaci Open Source lub jest już wbudowana w systemy źródłowe, a co więcej nie wymaga dodatkowego serwera buforującego komunikaty.

Rozdział 3

Uniwersalny szablon komunikacyjny

3.1 Istniejące sposoby dostępu do informacji z urządzenia mobilnego

3.1.1 Mobilne przeglądarki

3.1.2 Aplikacje mobilne

3.2 Idea szablonu

3.2.1 Możliwości

3.2.2 Dlaczego to jest nowatorskie - jak by to napisać?

3.2.3 Możliwe zastosowania

Ankietowanie

Zdalne podejmowanie decyzji

3.2.4 Architektura

Aplikacja serwerowa

Aplikacja mobilna

Rozdział 4

Mobilny system informacyjny

Przykładem integracji systemu klasy Enterprise z platformą mobilną jest aplikacja dostarczająca spersonalizowane informacje. Ideą stojącą za stworzeniem takiego systemu jest zagospodarowanie czasu użytkowników nie mogących aktywnie wyszukiwać interesujących ich wiadomości. Z takimi sytuacjami mamy do czynienia podczas oczekiwania na środki komunikacji miejskiej, a także w czasie utrudnień w ruchu samochodowym. W takim momencie można zaproponować lekturę najnowszych informacji w skróconej formie. Dodatkowo czytelnik w prosty sposób może określić, czy aktualnie pokazywane wiadomości interesują go czy też nie. W ten sposób uzyskamy możliwość analizy zainteresowań użytkowników, tak aby w przyszłości przekazywać im jedynie informacje które mogą ich zainteresować.

4.1 Funkcjonalność systemu

Funkcjonalność systemu będzie oparta na następujących założeniach:

- Aplikacja na urządzeniu mobilnym wyświetla krótkie, dostosowane do użytkownika informacje

- Informacje zmienieniają się automatycznie, tak aby ograniczyć potrzebę samodzielnego ich wyszukiwania
- Istnieje możliwość określenia czy aktualnie pokazywana wiadomość jest dla nas interesująca, co będzie miało wpływ na dobór kolejnych wiadomości
- Możliwość robienia zakładek na wybranych wiadomościach, co spowoduje wysłanie odnośników do nich na skrzynkę pocztową, w celu późniejszego przeczytania

4.2 Wymagania systemu

4.2.1 Infrastruktura

Potrzebne będzie połączenie z Internetem, dostępne z urządzenia mobilnego jak i z serwera dostarczającego informacje. Dodatkowo dla komfortowego przeglądania wiadomości, urządzenie mobilne powinno posiadać ekran o odpowiednio dużej rozdzielczości (co najmniej 320x240).

4.2.2 Oprogramowanie

Urządzenie mobilne będzie musiało posiadać maszynę wirtualną Java. Aplikacja na serwerze będzie działała pod kontrolą kontenera Apache Tomcat.

4.3 Wstępna analiza projektu

Przypuszczamy, że system który planujemy stworzyć może znaleźć realne zastosowanie i odnieść sukces rynkowy.

	Zalety	Wady
Wewnętrzne	Wykorzystanie popularnej platformy Niski koszt rozwiązania	Powolna lub przerywana transmisja Zbyt mały ekran i mała ergonomia użytkowania
Zewnętrzne	Długi czas spędzany w korkach lub w środkach komunikacji w dużych miastach Podatność ludzi krótkie, podane w ciekawej formie informacje	Obawa ludzi przed korzystaniem z internetu mobilnego, jako ciągle zbyt drogiego

4.3.1 Podobne systemy

Inspiracją, która stoi za naszym pomysłem jest system wyświetlania wiadomości na stacjach i w wagonach metra. Tym co chcielibyśmy w nim udoskonalić to możliwość dostarczenia spersonalizowanych, dostosowanych do każdego czytelnika informacji.

Bibliografia

[1] Michael Juntao Yuan: *Enterprise J2ME Developing Mobile Applcations*, Prentice Hall Pennsylvania 2004

[2] Gregor Hophe,Bobby Woolf: *Enterprise Integration Patterns*, The Addison-Wesley Signaure Series 2003

[3] : *Spring Tutorial*, http://www.techfaq360.com/tutorial/spring/