Konferencja Optymalny znaczy najlepszy

czyli, co nam dają nowe wersje oprogramowania?

Maciej Zrobek

Ktoś WAS widział w kolejce po MQ

18-19 listopada 2010
Naruszewo, hotel Szkockie Ranczo

AGENDA

- WebSphere Application Server V7: nowe rozwiązania charakterystyczne dla z/OS
- WebSphere MQ for V7: Publish-Subscribe

Co nowego w WebSphere Application Server for z/OS V7?

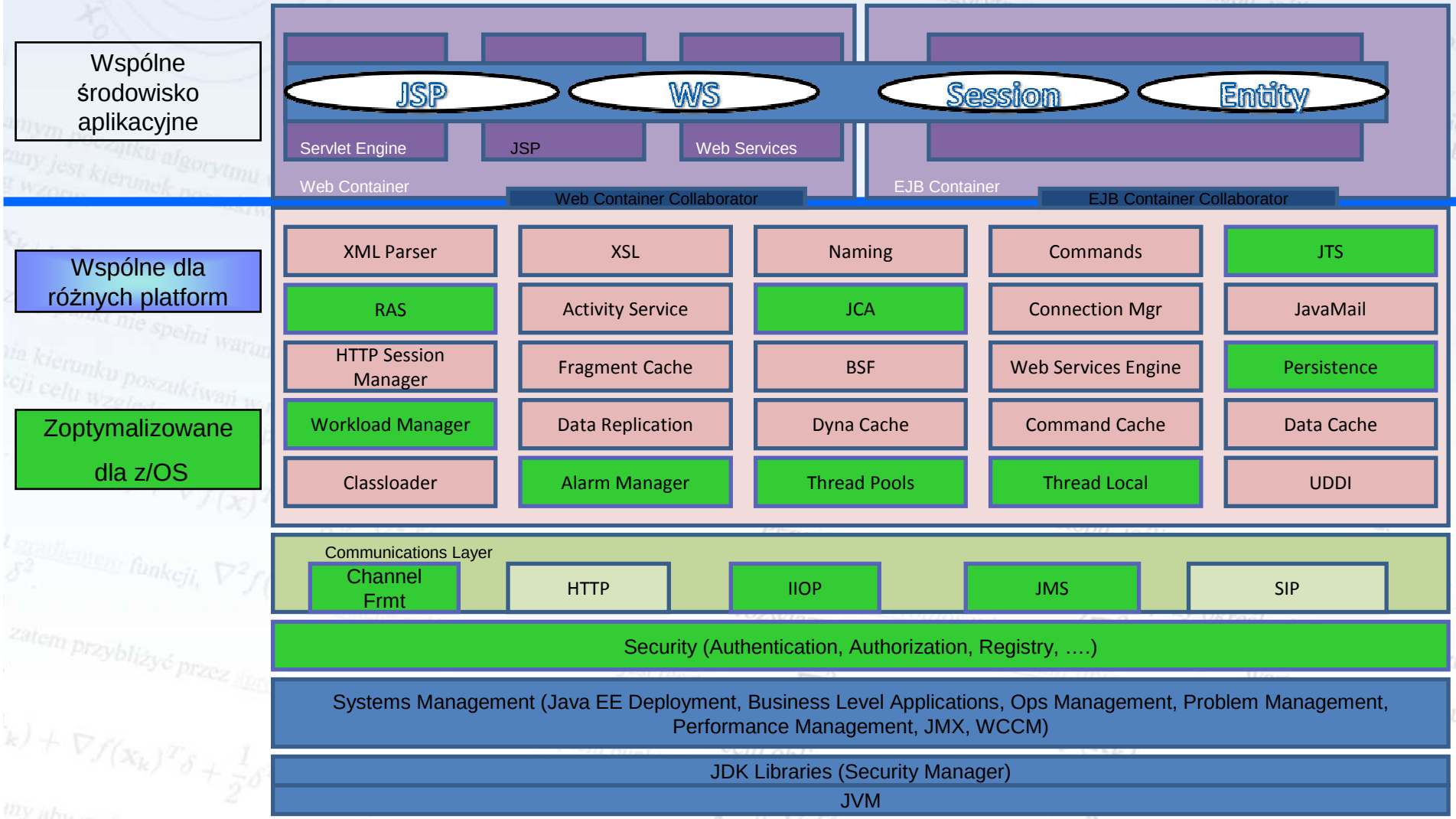
WebSphere Application Server (WAS) for z/OS

Połączenie wiodącego środowiska wykonawczego SOA i platformy z/OS

- WAS jest produktem dostępnym na wiele platform
  - Wszystkie funkcje i interfejsy aplikacyjne są dostępne na wszystkie platformy
  - Korzyści to wspólny model programowania, wspólne funkcje administracyjne itd.

- **Różnica** leży w integracji WAS z platformą z/OS
  - WAS w szczególny sposób wykorzystuje możliwości z/OS i Parallel Sysplex
  - … w sposób przeźroczysty dla aplikacji

WebSphere Application Server: wszędzie ten sam, ale nie taki sam dla z/OS



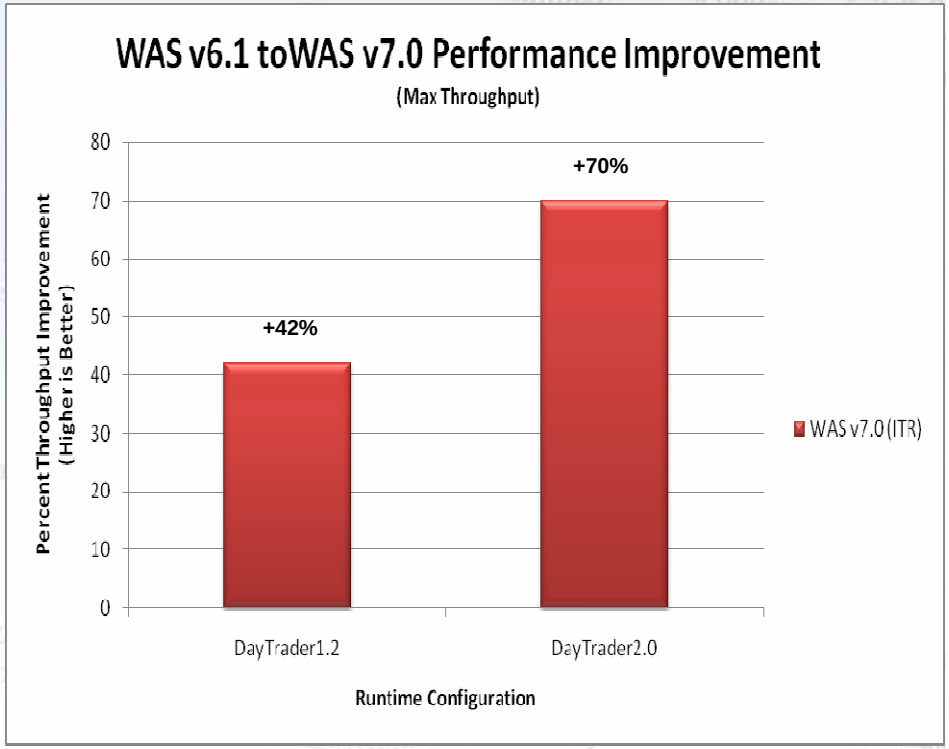
WAS V7 – nowe wersje zgodności ze standardami

Specyfikacja	*V6.1*	*V7*
IBM Java SDK	SDK 5	SDK 6
Java Platform, Enterprise Edition (Java EE) specification	J2EE 1.4	Java EE 5
Java Platform, Standard Edition (Java SE) specification	J2SE 5	J2SE 6
Java Servlet specification	2.4	2.5
EJB	2.1	3.0
Portlet specification	1.0	2.0
JDBC	3.0	4.0
Java API for XML Web Services (JAX-WS) specification	2.0	2.1
Web Services Atomic Transaction (WS-AT)	1.0	1.1
Web Services Business Activity (WS-BA)	1.0	1.1
Web Services Coordination (WS-COOR)	1.0	1.1
Web Services for Java Platform, Enterprise Edition (JSR 109)	1.1	1.2
Web Services Policy Namespace	n/a	1.5
Web Services Addressing - Metadata	n/a	1.0
Web Services Atomic Transaction Version	n/a	1.0 and 1.1
Web Services Reliable Messaging Policy Assertion Version	n/a	1.1
WS-SecurityPolicy	n/a	1.2
WS-MakeConnection Version 1.0	n/a	1.0
JavaBeans Activation Framework (JAF)	1.0.2	1.1
OASIS WS-SecureConversation	n/a	1.3
OASIS WS-Trust	n/a	1.3
Java Naming and Directory Interface (JNDI) Specification	SE 5	SE 6
Java Transaction API (JTA) specification	1.0.1	1.1

Uwaga! Standardy, dla których wersja zgodności nie zmieniła się w stosunku do V6.1 nie są ujęte w tej tabeli

WAS for z/OS: wzrost wydajności pomiędzy V6.1 i V7

- **DayTrader 1.2: Wzrost wydajności pomiędzy WebSphere Application Server for z/OS V6.1 i V7**
  - Wzrost wynosi 42%:
    - Ulepszone JDK
    - Optymalizacja komunikacji pomiędzy Servant Region i Controller Region
    - Optymalizacja kodu WebSphere Application Server V7
- **DayTrader 2.0 (EJB3): Wzrost wydajności EJB3 pomiędzy WAS V6.1 z EJB3 FP i V7**
  - WAS V7 jest o 70% szybszy niż V6.1 z EJB3 FP
    - Znacząca optymalizacja kodu EJB3/OpenJPA



*** Wyniki uzyskane na maszynie z9 EC
*** Wyniki mogą różnić się w zależności od aplikacji

* All data is pre-GA WAS V7 Info
GA information will be distributed with Performance Report

WAS V7 – nowe funkcje **specyficzne dla z/OS**

Funkcja	Wartość biznesowa	Korzyści techniczne
Wsparcie dla Fast Response Cache Acceleration (FRCA)	Polepszona **wydajność i zredukowany czas wykonania dla żądań, których odpowiedzi korzystają z cache.**	**Warstwa transportu wykorzystywana jako cache dla zawartości statycznej, serwletów oraz stron JSP.**
High Availability Manager (HAM) oparty na Cross-System Coupling Facility (XCF)	Zredukowany **narzut przy wykorzystaniu High Availability Manager dla z/OS.**	**Kiedy WAS V6.1 nie przetwarzał transakcji narzut związany z ciągłą detekcją awarii DCS powodował nieakceptowalny narzut** **W WAS V7 „heartbeat messages" są przesyłane poprzez Coupling Facilty, bez użycia TCP/IP**
Uwalnianie zawieszonych wątków (Thread Hang Recovery)	Zwiększone **możliwości recovery oraz niezawodność.**	**Serwer może podjąć próbę „uwolnienia" wątku Java, który się zawiesił**
Większa unifikacja zadań związanych z instalacją i konfiguracją	Ulepszona **łatwość użycia produktu**	**Łatwiejsza instalacja i zarządzanie produktem.** **Eliminacja źródeł problemów.**

WAS V7 – nowe funkcje **specyficzne dla z/OS**

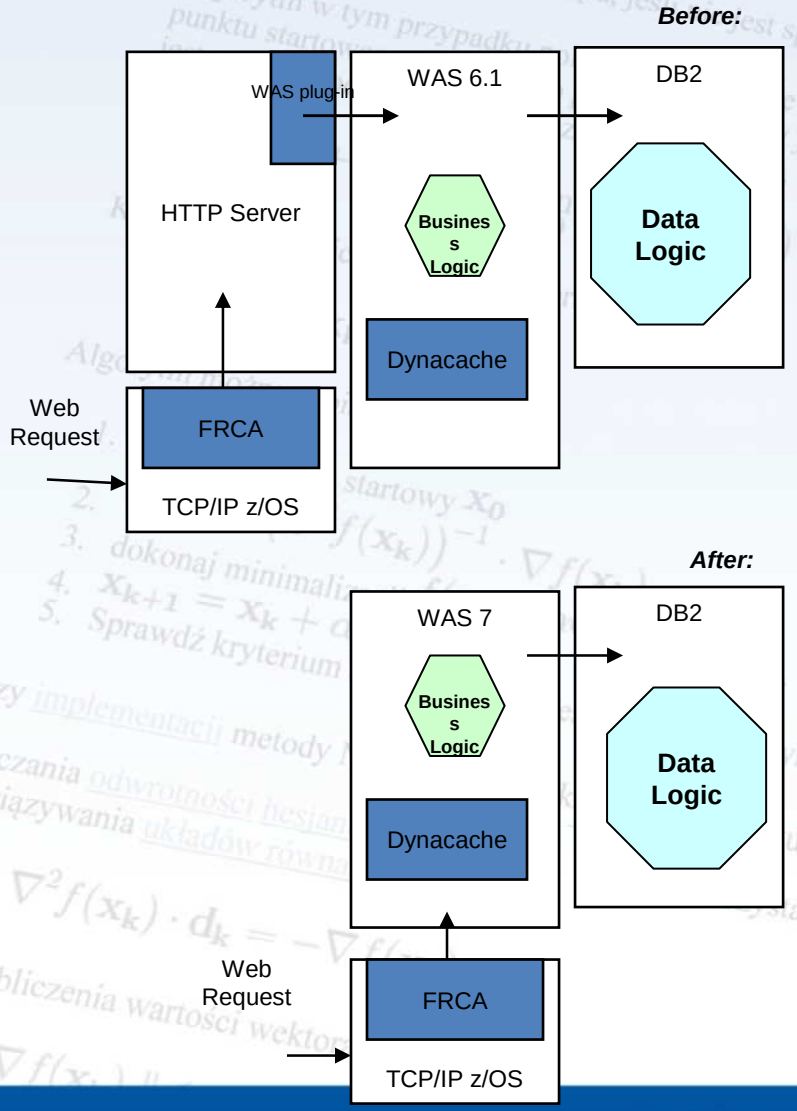
Funkcja	Wartość biznesowa	Korzyści techniczne
Wsparcie dla Fast Response Cache Acceleration (FRCA)	Polepszona **wydajność i zredukowany czas wykonania dla żądań, których odpowiedzi korzystają z cache.**	**Warstwa transportu wykorzystywana jako cache dla zawartości statycznej, serwletów oraz stron JSP.**
High Availability Manager (HAM) oparty na Cross-System Coupling Facility (XCF)	Zredukowany **narzut przy wykorzystaniu High Availability Manager dla z/OS.**	**Kiedy WAS V6.1 nie przetwarzał transakcji narzut związany z ciągłą detekcją awarii DCS powodował nieakceptowalny narzut** **W WAS V7 „heartbeat messages" są przesyłane poprzez Coupling Facilty, bez użycia TCP/IP**
Uwalnianie zawieszonych wątków (Thread Hang Recovery)	Zwiększone **możliwości recovery oraz niezawodność.**	**Serwer może podjąć próbę „uwolnienia" wątku Java, który się zawiesił**
Większa unifikacja zadań związanych z instalacją i konfiguracją	Ulepszona **łatwość użycia produktu**	**Łatwiejsza instalacja i zarządzanie produktem.** **Eliminacja źródeł problemów.**

Caching

- Czym jest caching i do czego nam to potrzebne?
  - Sposób na wykorzystanie pamięci w celu zaoszczędzenia czasu CPU lub przetwarzania
  - Badania wykazały, że wykorzystanie cache w pamięci może mieć duży wpływ na przepustowość przetwarzania aplikacji webowych
    - http://www.ibm.com/developerworks/websphere/techjournal/0405_hines/0405_hines.html
- Caching jest dzisiaj używany w wielu miejscach konfiguracji środowiska aplikacji webowych, np.:
  - WebSphere Application Server
    - Funkcja Dynacache pozwala na caching danych wyjściowych serwletów, komend administracyjnych WAS, stron JSP oraz odpowiedzi usług sieciowych
    - Dynacache jest wspierane na wszystkich platformach
  - IBM HTTP Server
    - Caching plików statycznych
    - Wspierane na wszystkich platformach

Fast Response Cache Acceleration (FRCA)

- Co to jest FRCA?
  - Technologia cachingu zintegrowana w stosie TCP/IP systemu z/OS
  - FRCA jest wspierane przez IBM HTTP Server, ale tylko dla zawartości statycznej
  - WAS 7.0 wspiera bezpośrednie wykorzystanie FRCA jako rozszerzenie istniejącego mechanizmu Dynacache
    - FRCA składuje zarówno zawartość statyczną, jak i dynamiczną, taką jak serwlety i JSP
- Wartość dla klienta
  - Polepszenie czasu odpowiedzi na żądania webowe
  - Zmniejszone obciążenie CPU
  - Uproszczona administracja



WAS V7 – nowe funkcje **specyficzne dla z/OS**

Funkcja	Wartość biznesowa	Korzyści techniczne
Wsparcie dla Fast Response Cache Acceleration (FRCA)	Polepszona **wydajność i zredukowany czas wykonania dla żądań, których odpowiedzi korzystają z cache.**	**Warstwa transportu wykorzystywana jako cache dla zawartości statycznej, serwletów oraz stron JSP.**
High Availability Manager (HAM) oparty na Cross-System Coupling Facility (XCF)	Zredukowany **narzut przy wykorzystaniu High Availability Manager dla z/OS.**	**Kiedy WAS V6.1 nie przetwarzał transakcji narzut związany z ciągłą detekcją awarii DCS powodował nieakceptowalny narzut** **W WAS V7 „heartbeat messages" są przesyłane poprzez Coupling Facilty, bez użycia TCP/IP**
Uwalnianie zawieszonych wątków (Thread Hang Recovery)	Zwiększone **możliwości recovery oraz niezawodność.**	**Serwer może podjąć próbę „uwolnienia" wątku Java, który się zawiesił**
Większa unifikacja zadań związanych z instalacją i konfiguracją	Ulepszona **łatwość użycia produktu**	**Łatwiejsza instalacja i zarządzanie produktem.** **Eliminacja źródeł problemów.**

High Availability Manager oparty na komunikatach „Heartbeats”

- High Availability Manager (HAM)
  - Funkcja wewnątrz WebSphere Application Server.
  - Wykrywa sytuacje kiedy element klastra ulegnie awarii oraz podejmuje akcje przywracające poprzedni stan (recovery)
- High Availability Manager był oryginalnie zaimplementowany za pomocą komunikatów TCP/IP „heartbeats”
  - Dla efektywnego wykrywania awarii „heartbeats” powinny być częste i regularne
  - „Heartbeats” zużywają zasoby nawet jeśli serwer nic nie robi
  - „Heartbeats” zużywają więcej zasobów wraz z rozrostem topologii serwera
- Jak zredukować narzut ze strony High Availability Manager:
  - Wyłączyć HAM tam, gdzie nie jest on potrzebny,
    - Jednakże niektóre funkcje WAS, takie jak np. usługi replikacji danych wymagają, aby HAM był aktywny
  - Zastosować bardziej efektywną metodologię wykrywania awarii:
    - Wykorzystanie funkcji z/OS, jaką jest Cross System Coupling Facility (XCF)
    - Wykorzystanie powiadomień, a nie „heartbeats”
    - XCF wysyła powiadomienie do członków grupy kiedy nastąpi zmiana statusu któregoś z nich

High Availability Manager Oparty na XCF

- Co to jest DCS?
  - WAS Distribution and Consistency Services (DCS)
    - Dystrybucja informacji pomiędzy zbiorem węzłów WAS (w ramach jednej komórki) należących do grupy (*core group*)
    - Wykrywanie awarii powyższych węzłów
    - Stanowi infrastrukturę wykorzystywaną przez HAM
  - High Availability Manager on z/OS począwszy od wersji 7:
    - Może nadal używać “Heartbeats”
    - Może opcjonalnie wykorzystywać XCF jeśli wszystkie elementy grupy są w wersji 7
- Wartość dla klienta:
  - XCF oferuje wyższą jakość odtwarzania stanu poprzedniego (*recovery*)
  - Zredukowane zużycie CPU

WAS V7 – nowe funkcje **specyficzne dla z/OS**

Funkcja	Wartość biznesowa	Korzyści techniczne
Wsparcie dla Fast Response Cache Acceleration (FRCA)	Polepszona **wydajność i zredukowany czas wykonania dla żądań, których odpowiedzi korzystają z cache.**	**Warstwa transportu wykorzystywana jako cache dla zawartości statycznej, serwletów oraz stron JSP.**
High Availability Manager ((HAM) oparty na Cross-System Coupling Facility (XCF)	**Zredukowany narzut przy wykorzystaniu High Availability Manager dla z/OS.**	**Kiedy WAS V6.1 nie przetwarzał transakcji narzut związany z ciągłą detekcją awarii DCS powodował nieakceptowalny narzut** **W WAS V7 „heartbeat messages" są przesyłane poprzez Coupling Facilty, bez użycia TCP/IP**
Uwalnianie zawieszonych wątków (Thread Hang Recovery)	Zwiększone **możliwości recovery oraz niezawodność.**	**Serwer może podjąć próbę „uwolnienia" wątku Java, który się zawiesił**
Większa unifikacja zadań związanych z instalacją i konfiguracją	Ulepszona **łatwość użycia produktu**	**Łatwiejsza instalacja i zarządzanie produktem.** **Eliminacja źródeł problemów.**

Zawieszone wątki Java

- Co to jest zawieszony wątek?
  - Zawieszenie wątku ma miejsce gdy żądanie na serwerze czeka na jakieś zdarzenie zewnętrzne, które jednak nie następuje lub następuje bo bardzo długim czasie
    - Przykład: wywołanie funkcji TCP/IP skierowane do zewnętrznego systemu nie kończy się ze względu na problemy z siecią
  - Zawieszone wątki blokują zasoby serwera, a zatem ich akumulacja z czasem prowadzi do degradacji wydajności
  - Sposobem na poradzenie sobie z zawieszonymi wątkami jest restart procesu, w którym te wątki żyją
    - Na platformie *distributed* oznacza to restart całego serwera
    - Pod z/OS oznacza to restart *servant region* z abendem EC3. Ponieważ serwer może składać się z kilku SR, nie musi to oznaczać niedostępności serwera.

Odzyskiwanie zawieszonych wątków na z/OS

- Co to jest odzyskiwanie zawieszonych wątków i do czego jest nam to potrzebne?
  - Nowa funkcja WAS V7 dla z/OS pozwalająca na „uwolnienie” zawieszonego wątku
  - Restart serwera nie jest już konieczny
    - Serwer kontynuuje swoją aktywnośc
    - Pozostała „niewinna” część zadań na serwerze pozostaje nienaruszona
    - Możemy również określić dopuszczalny procent zawieszonych wątków w serwerze, poniżej którego nie nastąpi restart serwera
  - Omawiany tu mechanizm podjąć próbę zatrzymania wątków zapętlonych (*runaway threads)*
    - Ten typ wątków jest jeszcze gorszy ponieważ nie dość, że blokuje zasoby, to jeszcze „zjada” czas procesora

WAS V7 – nowe funkcje **specyficzne dla z/OS**

Funkcja	Wartość biznesowa	Korzyści techniczne
Wsparcie dla Fast Response Cache Acceleration (FRCA)	Polepszona **wydajność i zredukowany czas wykonania dla żądań, których odpowiedzi korzystają z cache.**	**Warstwa transportu wykorzystywana jako cache dla zawartości statycznej, serwletów oraz stron JSP.**
High Availability Manager ((HAM) oparty na Cross-System Coupling Facility (XCF)	**Zredukowany narzut przy wykorzystaniu High Availability Manager dla z/OS.**	**Kiedy WAS V6.1 nie przetwarzał transakcji narzut związany z ciągłą detekcją awarii DCS powodował nieakceptowalny narzut** **W WAS V7 „heartbeat messages" są przesyłane poprzez Coupling Facilty, bez użycia TCP/IP**
Uwalnianie zawieszonych wątków (Thread Hang Recovery)	Zwiększone **możliwości recovery oraz niezawodność.**	**Serwer może podjąć próbę „uwolnienia" wątku Java, który się zawiesił**
Większa unifikacja zadań związanych z instalacją i konfiguracją	Ulepszona **łatwość użycia produktu**	**Łatwiejsza instalacja i zarządzanie produktem.** **Eliminacja źródeł problemów.**

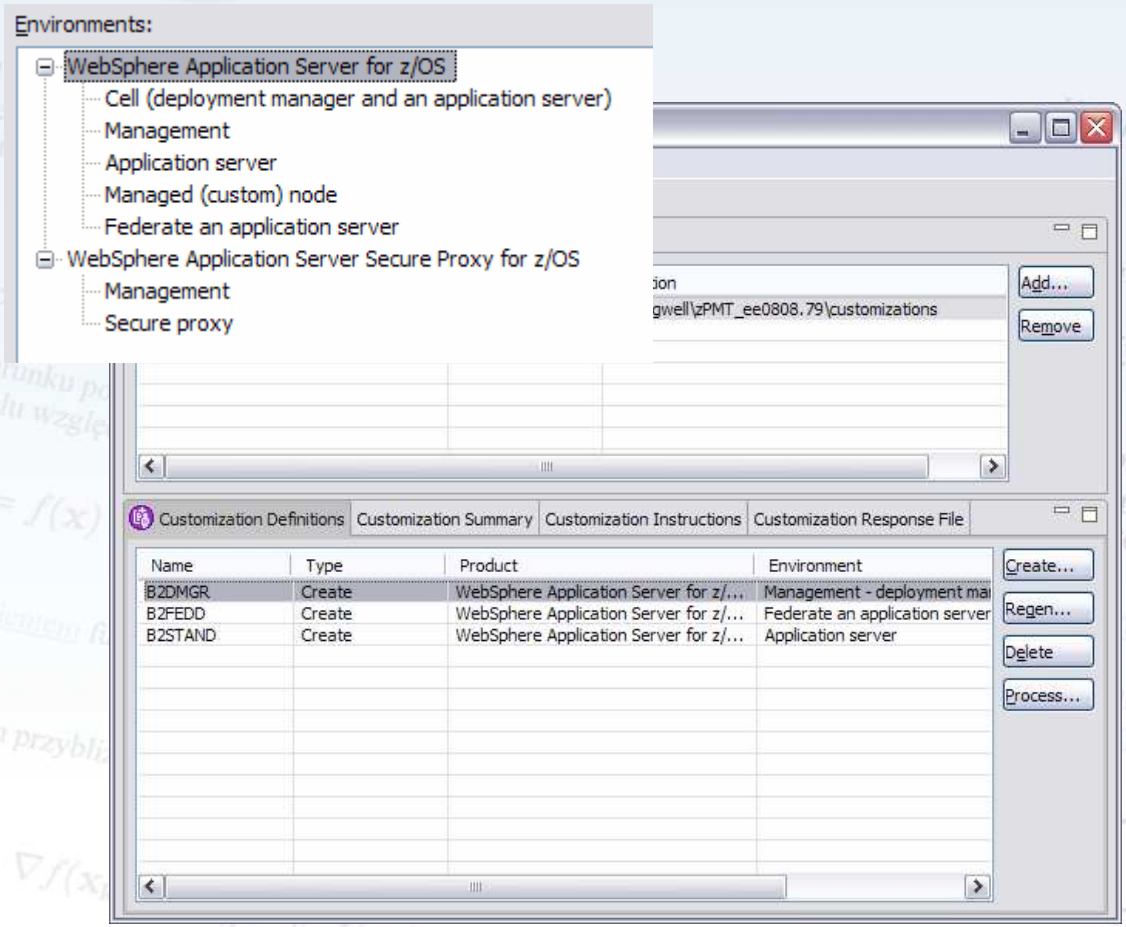
Większa unifikacja zadań związanych z instalacją i konfiguracją

- Co się zmieniło?
  - Przed wersją 7:
    - WAS for z/OS był dostarczany zarówno jako biblioteki MVS, jak i system plików HFS
    - Powodowało to konieczność synchronizacji dwóch zestawów kodu wykonywalnego
  - Od wersji 7:
    - Moduły wykonywalne są dostarczane wyłącznie w katalogu `.../AppServer/lib` wraz z resztą środowiska
    - Nie ma potrzeby zmian w LPA/LNKLST oraz w STEPLIB (zarówno w JCLu, jak i w `setupCmdLine.sh`)
- Wartość dla klientów:
  - Zmniejszona złożoność instalacji i zarządzania produktem
  - Eliminacja możliwości przypadkowej niezgodności pomiędzy bibliotekami wykonywalnymi i binariami HFS

Nowe narzędzie konfiguracyjne

Nie ma już paneli ISPF…

Stare narzędzie zPMT oparte na AST ustąpiło miejsca znacznie „lżejszemu” WebSphere Configuration Tool (WCT)



WCT wygląda podobnie jak zPMT dla wersji 6.1

WCT nie jest już jednak częścią Application Server Toolkit, a to oznacza, że jest mniejsze i szybsze

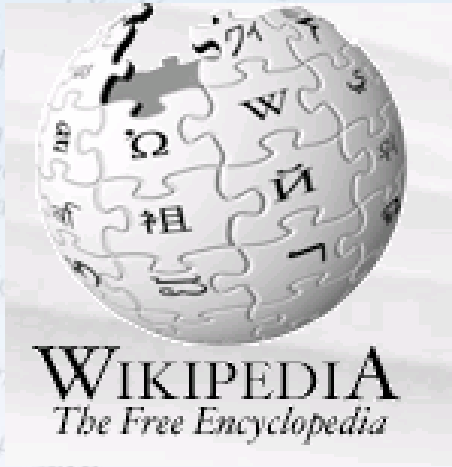
Dostępne scenariusze kastomizacji nowych środowisk WAS V7.0, jak i migracji istniejących w V6.1

Migracja do WAS V7

- Możliwe scenariusze:
  - V5.1 -> V7.0
  - V6.0 -> V7.0
  - V6.1 -> V7.0
- Migracja odbywa się poprzez uruchomienie jobów wygenerowanych przez WCT
- Serwery nie muszą być wyłączone podczas migracji
- Nowy *Deployment Manager* dopuszcza węzły z V5.1, 6.0 i 6.1
- Proces migracji jest dokładnie opisany w *White Paper* **WP101329**

Co nowego w WebSphere MQ for z/OS V7: Publish/Subscribe

Czym jest *Publish/Subcribe*?



- *Publish/Subscribe* (lub **pub/sub**) jest asynchronicznym paradygmatem komunikacji za pomocą komunikatów, w którym klienci wysyłający komunikaty (**publishers**) nie są zaprogramowani do wysyłania komunikatów do specyficznych odbiorców (**subscribers**). Zamiast tego komunikaty są opisywane przez klasy, bez wiedzy na temat tego, jacy (i czy w ogóle) odbiorcy zamierzają je odbierać.

- **Odbiorcy wyrażają zainteresowanie dla jednej lub więcej klas komunikatów i otrzymują wyłącznie komunikaty ich interesujące bez wiedzy o tym, czy są jacyś klienci, którzy je wysyłają.** To oddzielenie wysyłających i odbiorców pozwala na większą skalowalność i na bardziej dynamiczną topologię sieci.

Przykłady komunikacji Pub/Sub w kontraście do *Point-to-Point*

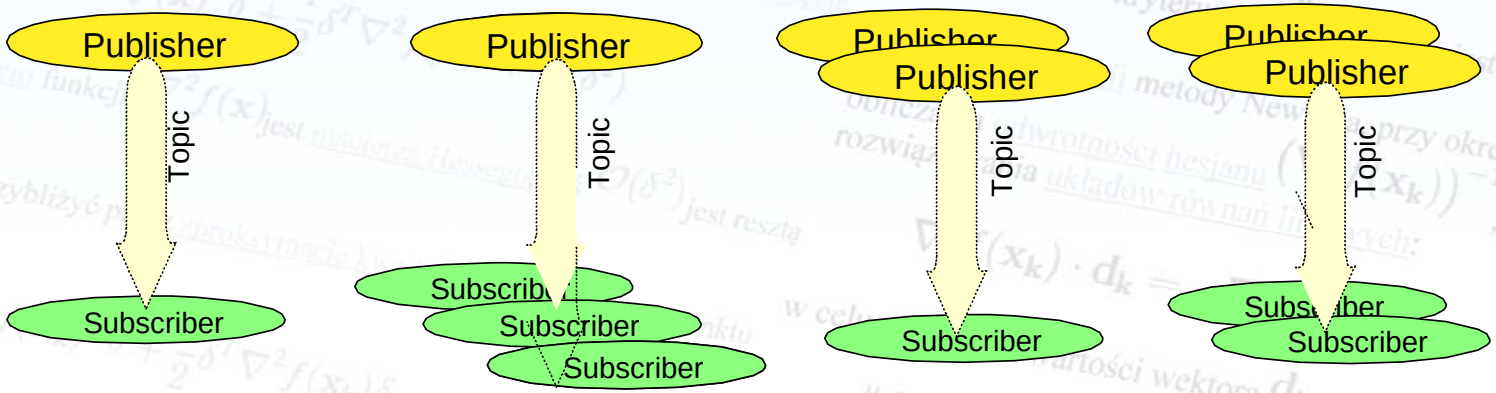
Point-to-Point	Publish/Subscribe
List – wysyłany jest tylko do jednej, konkretnej osoby	**Audycja Radiowa** – tylko ci, co dostroją się do stacji odbiorą audycję
E-mail – może być wysłany do wielu osób, ale zawsze wiemy, kto go otrzyma	**Czasopismo** – na podstawie informacji na okładce sięgną po nie tylko ci, których zainteresuje ich treść
Komunikacja poprzez kolejkę – jeżeli komunikat jest umieszczany w kolejce, dotrze on do tylko jednego konsumenta	**Tablica odlotów samolotów** – może prezentować wszystkie odloty, z danego terminala lub dla danej linii lotniczej

Droga w stronę luźnych powiązań (*Loose Coupling)*

- Architektura monolityczna
  - Cała praca wykonywana jest na jednym komputerze!
- *Client-Server*
  - Praca jest rozdzielona, ale komponenty rozwiązania muszą znać topologię architektury i połączeń oraz muszą być dostępne jednocześnie.
- Komunikacja poprzez kolejki (*Message Queuing*)
  - Współpracujące komponenty mogą działać z różnymi prędkościami. Praca zostanie zakolejkowana w przypadku przestoju lub wyłączenia komponentu. Połączenia oparte są na nazwach kolejek.
- Publish/Subscribe
  - Współpracujące komponenty wymieniają dane poprzez infrastrukturę, która identyfikuje „**temat**” danych

Luźne powiązania poprzez Pub/Sub

- **Publikujący są luźno powiązani z odbiorcami i nie muszą nawet wiedzieć o ich istnieniu.**
- **Przy nacisku położonym na temat komunikatów publikujący i odbiorcy nie muszą nic wiedzieć na temat topologii systemu. Każdy z nich może działać niezależnie od innych.**
- **Systemy pub/sub nie tylko rozdzielają publikujących i odbiorców w sensie lokalizacji, lecz również w sensie czasowym**



Implementacja Pub/Sub w WebSphere MQ 7

- Centralną koncepcją jest **temat** (**TOPIC STRING).**
- Komunikaty są „publikowane” w tematach.
  - Każdy fakt publikacji odbywa się na rzecz dokładnie jednego tematu.
- Odbiorcom dostarczane są komunikaty, które zostały opublikowane w tematach, do których są on subskrybowani
  - Odbiorcy mogą się subskrybować do wielu tematów i mogą używać masek (*wildcards*) w nazwach tematów.
- Tematy są zorganizowane w logiczną hierarchię.
- Tematy mogą być tworzone na drodze administracyjnej, lub poprzez fakt ich użycia przez Publish/Subscribe.
- Fizycznie tematy opisywane są za pomocą łańcuchów (o długości do 10K znaków) podobnych do ścieżek w systemie plików, np.:

```
deli/fresh/fruit/orange
deli/fresh/#
deli/fresh/fruit/+
```

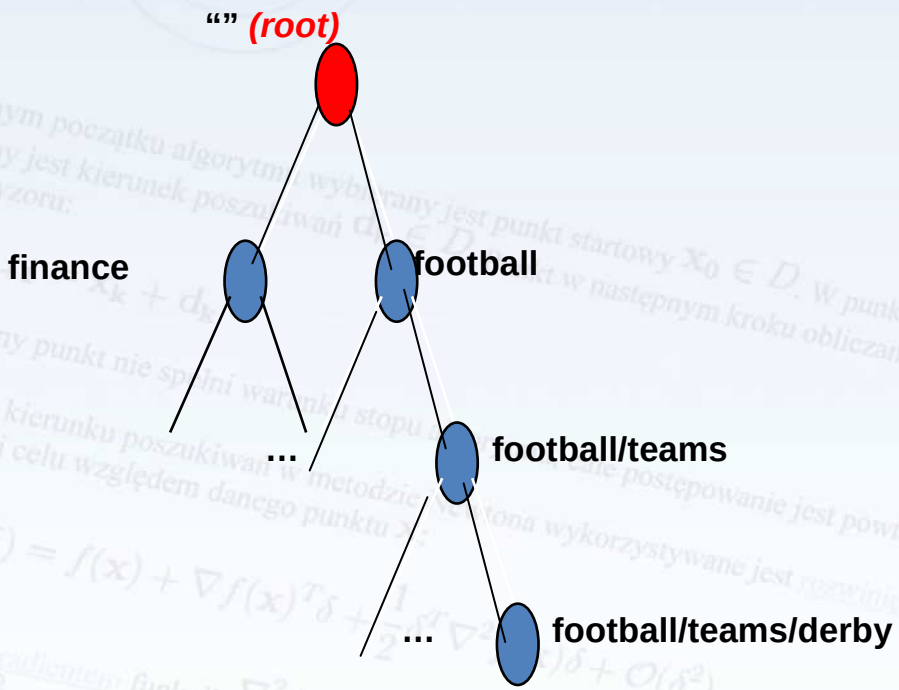
- „#” oznacza **0 lub więcej** poziomów w hierarchi tematów
- „+” oznacza **dokładnie 1** poziom w hierarchi tematów

Aplikacje Publish/Subscribe

Administracja Pub/Sub – tematy jako obiekty

- Tematy Pub/Sub mogą być reprezentowane przez obiekty MQ (tak, jak np. kolejki), którymi można administrować
- Obiekt reprezentują cy temat (*Topic Object*) ma nazwę podlegającą takim samym zasadom, jak nazwy kolejek (nie musi ona mieć nic wspólnego z TOPIC STRING)
- TOPIC STRING jest jedną z, ale nie jedyną właściwością obiektu
- Zarządzanie bezpieczeństwiem oparte jest na *Topic Objects*

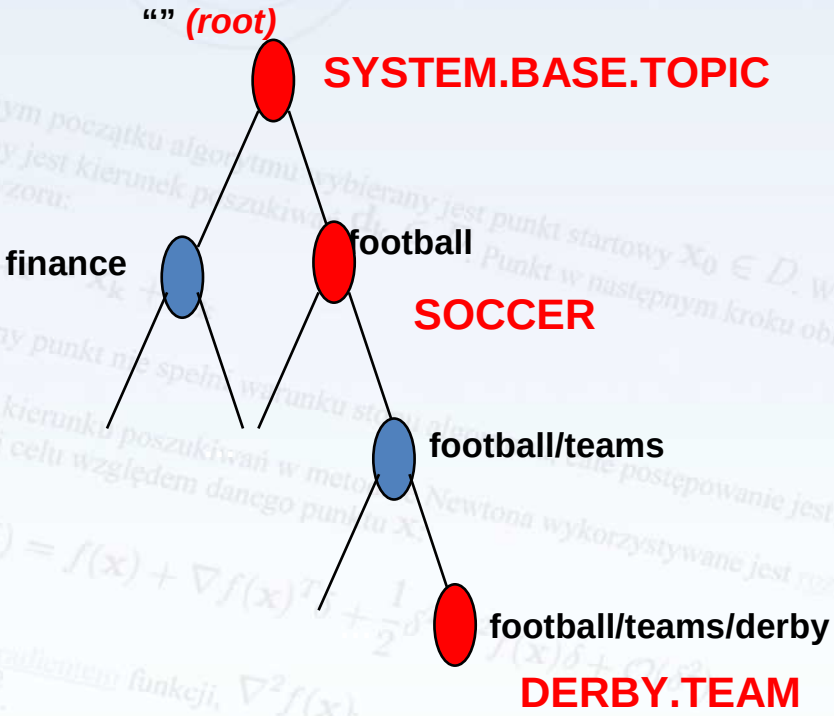
Koncepcja drzewa tematów



- **Administered Node**
- **Non-administered Node**

- Drzewo tematów jest wewnętrzną reprezentacją hierarchii tematów
- Topologia drzewa implikowana jest kompletnym zestawem TOPIC STRINGS w użyciu
- Nie musi być zapewnione mapowanie „jeden do jednego" pomiędzy obiektami reprezentującymi tematy i węzłami drzewa

Koncepcja drzewa tematów



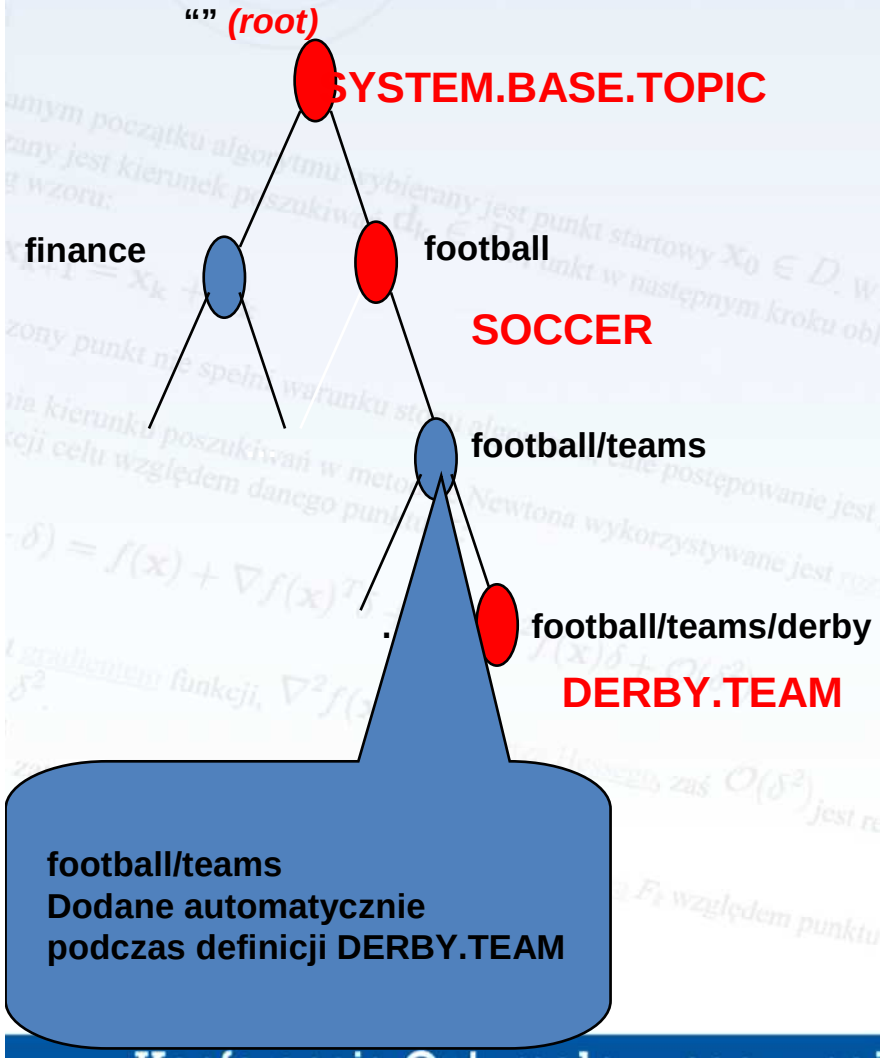
Administered Node

Non-administered Node

- Węzły, którym odpowiadają obiekty mogą być rozumiane jako węzły zarządzane (*Administered Nodes)*
- Takie węzły są **jedynymi** stałymi częściami hierarchii

DEFINE TOPIC(SOCCER) TOPICSTR('football')

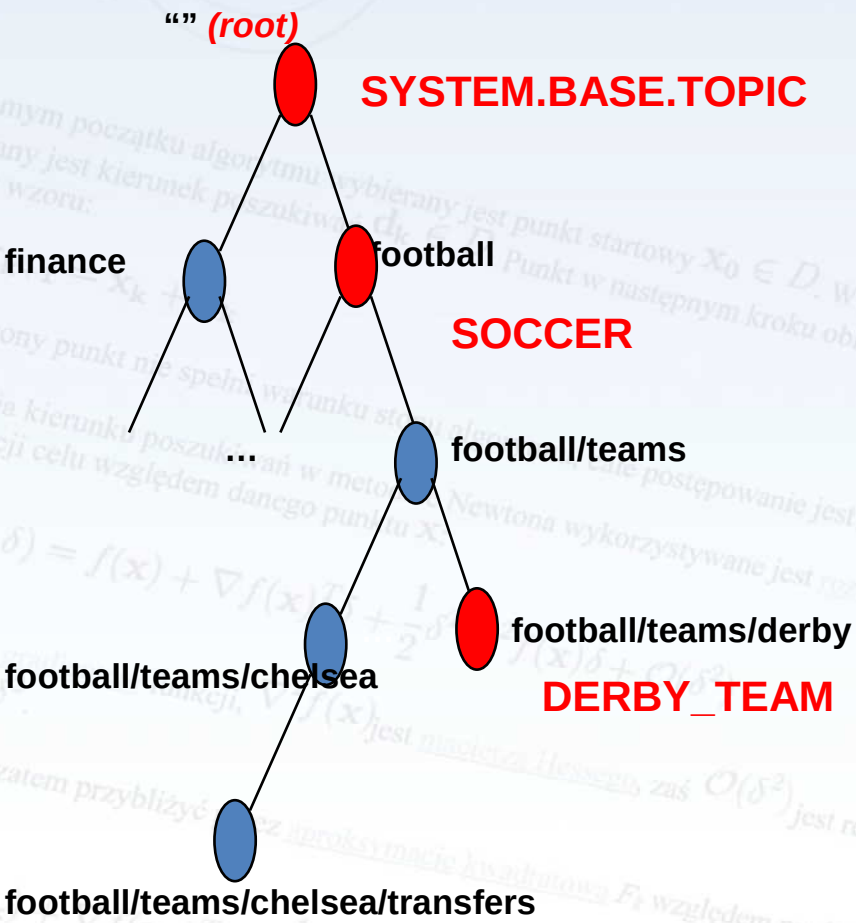
Koncepcja drzewa tematów



- **Administered Node**
- **Non-administered Node**

- Brakujące części hierarchii zostaną automatycznie uzupełnione przy definicji węzła zarządzanego

Koncepcja drzewa tematów



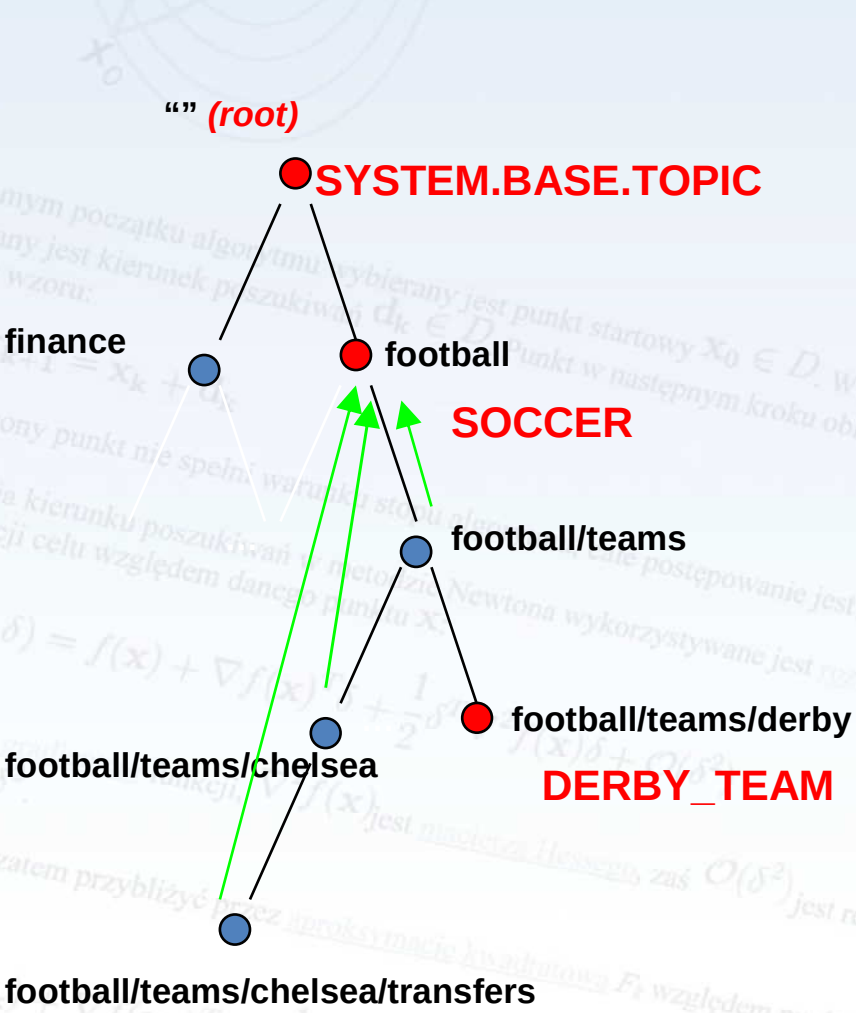
Administered Node

Non-administered Node

- Publikacja lub subskrypcja do węzła, którego jeszcze nie ma w hierarchii spowoduje rozszerzenie drzewa
- Takie węzły są tymczasową częścią hierarchii i zostaną automatycznie usunięte gdy nie będzie dla nich już żadnych publikujących lub odbiorców

MQSUB('/football/teams/chelsea/transfers')

Koncepcja drzewa tematów



- Administered Node
- Non-administered Node

- Węzły, z którymi związany jest obiekt to węzły zarządzane.
- Węzły, które zostały automatycznie wygenerowane „dziedziczą" właściwości pierwszego węzła zarządzanego powyżej w hierarchii, np. football/teams/chelsea przejmuje właściwości węzła zarządzanego SOCCER
- Jeśli wyżej w hierarchii drzewa nie można znaleźć żadnego węzła zarządzanego domyślne właściwości zostaną przejęte z SYSTEM.BASE.TOPIC

Koncepcja subskrypcji

- Kiedy subskrybujemy się do tematu możemy określić miejsce (kolejkę), do którego mają być dostarczane komunikaty
  - *Queue Manager* utworzy miejsce dostarczenia albo możemy podać własne
- Subskrypcja może być trwała (*durable*) lub nie
  - Jeśli nasza subskrypcja nie jest określona jako trwała to komunikaty będą dostarczane do nas tylko jeśli będziemy podłączeni
  - Komunikaty dla subskrypcji trwałych są do nas dostarczane przy następnym połączeniu
  - Subskrypcje trwałe muszą mieć unikalne nazwy
- Obiekty odpowiadające subskrypcjom mogą być tworzone programistycznie lub administracyjnie.

Administracja Pub/Sub – subskrypcje jako obiekty

- Zapewniają dostarczenie do wybranej kolejki MQ wszystkich komunikatów publikowanych na temacie
- Pozwalają aplikacjom typu *Point-to-point* na podłączenie się do tematu (aplikacje te odbierają komunikaty z kolejki, a nie z tematu)
  - Przykładowe właściwości:
    - TOPICSTR - n.p. TOPICSTR('/entertainment/tv/#')
    - DEST – nazwa kolejki, do której mają być przesyłane wszystkie komunikaty publikowane w temacie: n.p. DEST(MYQ)
    - DESTQMGR – nazwa *Queue Managera*, do którego mają kierowane komunikaty DESTQMGR(RMT.QMGR)

Pub/Sub - podsumowanie

- Pub/Sub to koncepcja komunikacji opartej na komunikatach, która zakłada luźne powiązanie pomiędzy producentami (publikującymi) komunikatów i ich odbiorcami
- Implementacja Pub/Sub w MQ V7 oparta jest na koncepcji tematów, które tworzą strukturę drzewiastą
- Węzły w drzewie tematów mogą być administrowane (permanentne) lub tymczasowe
- Aplikacje działające według tradycyjnej koncepcji *Point-to-point* mogą również podłączać się do tematów Pub/Sub poprzez wykorzystanie subskrypcji zarządzanych

Więcej informacji o Pub/Sub:
http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wmq_v7/wmq/7.0/Pubsub.html

Dziękuję za uwagę!