Table of Contents

Wybrane zmiany w kolejnych wersjach serwera

WilfFly 8

- Pełne wsparcie dla Java EE 7
- Role Based Access Control
- Nowy serwer HTTP (Undertow)
- Poszerzenie możliwości CLI i Web Console
- Uproszczona konfiguracja gniazdek sieciowych (zredukowana ilość portów)
- Poprawki błędów

WildFly 9

- Możliwość użycia serwera WilfFly jako load balancera
- Poszerzenie możliwości CLI i Web Console
- Suspended mode
- Offline management
- Wsparcie dla HTTP/2

WildFly 10

- Nowy JMS provider (Apache Artemis MQ)
- Nowe możliwości w zakresie zarządzania pulami komponentów EJB
- Podniesienie wersji Hibernate
- Rezygnacja ze wsparcia JDK 7

Wprowadzenie

Wymagania systemowe

- Java Development Kit 8.x (zaleca się użycie najnowszej dostępnej wersji)
- Poprawnie ustawione zmienne środowiskowe **JAVA_HOME** i **PATH**

Instalacja

WildFly udostępniany jest pod adresem http://www.wildfly.org/downloads w postaci zwykłego archiwum **.zip** lub **.tar.gz** skąd należy go pobrać, a następnie rozpakować w preferowanym katalogu.

Uruchamianie serwera

Serwer aplikacji WildFly może działać w dwóch trybach **standalone** oraz **domian**. W obu przypadkach oferowane funkcjonalności są takie same. Różnica polega na sposobie konfiguracji i zarządzania - w drugim przypadku istnieje możliwość centralnego administrowania wieloma instancjami serwera.

Uruchomienie serwera odbywa się poprzez wykonanie skryptu startowego **standalone.sh** lub **domain.sh** (w zależności od trybu pracy) znajdującego się w katalogu **bin**.
Po pomyślnym zakończeniu procesu pod adresem http://localhost:8080 serwowana jest strona powitalna.

Istnieje możliwość uruchomienia serwera w trybie administracyjnym - serwer nie akceptuje żądań użytkowników oraz nie uruchamia wszystkich usług, jednak umożliwia na zmiany konfiguracji. W celu włączenia trybu administracyjnego należy użyć przy starcie parametru **--admin-only**.

Weryfikacja trybu pracy serwera z poziomu konsoli:

```
:read-attribute(name=running-mode)
```

Przeładowanie serwera ze zmianą trybu pracy:

```
reload --admin-only=true
```

Sprawdzenie trybu w jakim serwer został uruchomiony (nie zależnie od aktualnego stanu):

```
/core-service=server-environment:read-attribute(name=initial-running-mode)
```

Tworzenie konta administratora

Wykonywanie czynności administracyjnych wymaga utworzenia konta posiadającego odpowiednie uprawnienia. Konto można stworzyć za pomocą gotowego skryptu.

Nowy użytkownik powinien:

- być typu Management
- mieć możliwość podłączania się do innych procesów serwerów aplikacyjnych
- nie należeć do żadnej grupy

Warto zapamiętać wartość hasła zakodowaną w Base64, przyda się ona do późniejszej konfiguracji.

```
./add-user.sh
```

```
./add-user.sh -m -u admin -p P@ssw0rd
```

Użycie konsoli CLI

Administracja z poziomu linii poleceń jest możliwa po uruchomieniu konsoli za pomocą skryptu **jboss-cli.sh**. Po uruchomieniu należy podłączyć się do instancji serwera wydając polecenie **connect**.

Zatrzymywanie serwera

- użycie kombinacji klawiszy Ctrl-C
- wydanie odpowiedniej komendy z CLI
- zabicie procesu serwera

```
./jboss-cli.sh
[disconnected /] connect
Connected to localhost:9990
[localhost:9990 /] shutdown
```

```
./jboss-cli.sh
[disconnected /] connect 192.168.1.1
Username: nodeadmin
Password:
Connected to 192.168.1.1:9990
[192.168.1.1:9990 /] shutdown --restart=true
```

```
./jboss-cli.sh -c --command=shutdown
```

W nowszych wersjach serwera możliwe jest przejście do trybu suspended - aktualne żądania będą przetwarzane do końca, jednak serwer nie będzie akceptował nowych.

W trybie standalone:

```
:suspend(timeout=60)
```

W trybie domenowym:

```
:suspend-servers(timeout=60)
/host=master/server-config=server-one:suspend(timeout=60)
/server-group=main-server-group:suspend-servers(timeout=60)
```

Weryfikacja stanu i wznowienie pracy realizowane jest następująco:

```
:read-attribute(name=suspend-state)
/host=master/server=server-one:read-attribute(name=suspend-state)
:resume
```

Podobnie przeprowadzić moża również opóźnione wyłączenie serwera - tak, aby zdążył on przetworzyć jeszcze aktualne żądania klientów.

```
:shutdown(timeout=60)
:stop-servers(timeout=60)
/host=master/server-config=server-one:stop(timeout=60)
/server-group=main-server-group:stop-servers(timeout=60)
```

Konfiguracja

Serwer może pracować w jednym z dwóch poniższych trybów:

- **standalone** - każda instancja jest niezależnym procesem. Pliki konfiguracyjne znajdują się w katalogu **JBOSS_HOME/standalone/configuration**
- **domain** - istnieje możliwość uruchomienia wielu instancji i zarządzanie nimi z jednego centralnego punktu (kontrolera domeny). Domena może obejmować wiele maszyn (fizycznych lub wirtualnych). Każda maszyna może hostować kilka instancji serwera aplikacyjnego. Pliki konfiguracyjne znajdują się w katalogu **JBOSS_HOME/domain/configuration**

Podstawowa struktura katalogów

Nazwa	Opis
bin	Skrypty i konfiguracja startowa, pozostałe komendy i narzędzia
docs	Przykłady konfiguracji, definicje XML Schema
domain	Konfiguracja serwera działającego w trybie domeny
standalone	Konfiguracja serwera działającego w trybie pojedynczej instancji
modules	Moduły i biblioteki będące częścią dystrybucji serwera
welcome-content	Główny katalog www, zawierający stronę startową

Metody konfiguracji

Web Console

Aplikacja webowa, będąca częścią dystrybucji serwera. Umożliwia zarządzanie podstawowymi komponentami, wdrażanie aplikacji, monitorowanie oraz inne. Pozwala na szybką i łatwą zmianę konfiguracji. Dedykowana głównie osobom rozpoczynającym pracę z serwerem. Konsola jest dostępna pod adresem http://localhost:9990

Command Line Interface (CLI)

Oparte o terminal, zaawansowane narzędzie umożliwiające administrację serwerem. Daje pełne możliwości konfiguracyjne niezależnie od trybu pracy i lokalizacji serwera.

Przykłady odczytu/zapisu konfiguracji:

```
/subsystem=datasources/data-source=ExampleDS:read-attribute(name=enabled)
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=enabled,value=false)

/core-service=management/management-interface=http-interface:write- attribute(name=con
sole-enabled,value=false)
```

Edycja plików XML

Niezalecane - może prowadzić do błędów. Możliwe tylko przy wyłączonym serwerze.

Podczas startu serwera tworzona jest kopia pliku konfiguracyjnego w pliku zawierającym w nazwie znacznik czasowy. Każda zmiana konfiguracyjna powoduje automatyczne utworzenie kopii zapasowej co umożliwia przywrócenie konfiguracji do oryginalnego stanu. Dodatkowo możliwe jest wykonywanie snapshotów, które również mogą posłużyć do przywrócenia konfiguracji.

```
:take-snapshot
:list-snapshots
:delete-snapshot(name=20151022-133109702standalone.xml)
standalone.sh --server- config=standalone_xml_history/snapshot/20151022-133109702stand
alone.xml
```

Dodatkowo w nowszych wersjach serwera istnieje możliwość śledzenia historii zmian.

```
/core-service=management/service=configuration-changes:add(max- history=10)
/core-service=management/service=configuration-changes:list-changes
```

Podstawowa struktura plików konfiguracyjnych

Server

Główny znacznik, zamyka w sobie wszystkie pozostałe elementy

Extensions

Większość funkcjonalności serwera jest dostarczana w ramach tzw. rozszerzeń, ładowanych w czasie startu serwera przed wdrożeniem jakichkolwiek aplikacji. Rozszerzenia muszą implementować interfejs *org.jboss.as.controller.Extension* i są przechowywane w katalogu

JBOSS_HOME/modules.

```
<extensions>
    <extension module="org.jboss.as.clustering.in nispan"/>
    <extension module="org.jboss.as.connector"/>
    <extension module="org.jboss.as.deployment-scanner"/>
    <extension module="org.jboss.as.ee"/>
    <extension module="org.jboss.as.ejb3"/>
    .......
</extensions>
```

Paths

Logiczne nazwy ścieżek do zasobów, które mogą być wykorzystane w ramach późniejszej konfiguracji np.:

- *jboss.home* - katalog główny serwera
- *user.home* - katalog domowy użytkownika
- *java.home* - katalog instalacji JDK
- *jboss.server.log.dir* - miejsce przechowywania logów
- ...

```
<path name="log.dir" path="/home/logs" />
<path name="logdata.dir" path="/logs" relative-to="jboss.server.data.dir"/>
```

Powyższe ustawienia mogą zostać nadpisane podczas uruchamiania serwera.

```
standalone.sh -Djboss.server.log.dir=/var/log
```

Interfaces

Adresy IP / nazwy hostów na których działa serwer. Domyślnie zdefiniowane interfejsy to:

- *management* - dostęp administracyjny - konsola webowa i CLI
- *public* - dostęp do publicznych usług serwera
- *unsecure* - komunikacja IIOP

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:127.0.0.1}"/>
    </interface>
    <interface name="unsecure">
        <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
    </interface>
</interfaces>
```

Powyższe ustawienia mogą zostać nadpisane podczas uruchamiania serwera.

```
./standalone.sh -Djboss.bind.address=192.168.1.1 -Djboss.bind.address.management=192.1
68.1.1
```

```
./standalone.sh -b 192.168.1.1 -bmanagement 192.168.1.1
```

Socket binding groups

Konfiguracja portów dla poszczególnych usług i powiązanie ich z wcześniej zdefiniowanymi interfejsami. Ta sekcja określna na jakich portach będzie nasłuchiwał serwer. Atrybut **jboss.socket.binding.port-offset** pozwala na przesunięcie wszystkich definicji portów o określoną wartość, co pozwala na uruchomienie kilku instancji serwera na jednej fizycznej maszynie.

```
<socket-binding-group name="standard-sockets" default-interface="public" port-off set=
"${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management" port="${jboss.manage
ment.http.port:9990}"/>
    <socket-binding name="management-https" interface="management" port="${jboss.manag
ement.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="8080"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    ....
</socket-binding-group>
```

Uwaga! WildFly używa portu 9990 dla wszystkich interfejsów administracyjnych (CLI i Web Console)

System properties

Pozwalają na definiowanie reużywalnych zmiennych. Mogą być definiowane podczas startu serwera lub na poziomie pliku konfiguracyjnego.

```
<system-properties>
    <property name="node" value="main"/>
</system-properties>
```

```
./standalone.sh -Dnode=main
```

```
./standalone.sh -P file.properties
```

Powyższe ustawienia mogą zostać nadpisane podczas uruchamiania serwera lub przez CLI.

```
standalone.sh -Djboss.bind.address=192.168.1.2
/system-property=jboss.bind.address:add(value=192.168.1.2)
```

Profiles

Konfiguracja podsystemów serwera np. kontener web może zawierać definicje wykorzystywanych konektorów. W przypadku serwera działającego w trybie standalone istnieje tylko jeden profil.

```
<profile>
    <subsystem xmlns="urn:jboss:domain:logging:2.0">
    ....
</profile>
```

Konfiguracja w trybie standalone

Jest oparta o jeden profil zawierający ustawienia wszystkich podsystemów serwera. Domyślnie w katalogu **JBOSS_HOME/standalone/configuration** znajdują się predefiniowane warianty konfiguracji:

- **standalone.xml** - Java EE Web profile
- **standalone-full.xml** - JEE Full profile
- **standalone-ha.xml** - rozszerza Web profile o możliwość klastrowania
- **standalone-full-ha.xml** - rozszerza Full profile o możliwość klastrowania

Uruchamianie serwera ze wskazaniem konkretnej konfiguracji:

```
./standalone.sh --server-config standalone-full.xml
```

```
./standalone.sh -c standalone-full.xml
```

Konfiguracja w trybie domain

Domena jest kolekcją grup serwerów. Grupa to logiczna jednostka organizacyjna, zawierająca określoną ilość hostów (instancji serwera WildFly). Każda grupa może definiować własne ustawienia takie np. parametry JVM, porty, interfejsy czy wdrażane aplikacje.

Każda domena składa się z:

- **Domain Controller** - przechowuje konfigurację współdzieloną przez węzły należące do domeny, stanowi punkt kontrolny domeny
- **Host Controller** - proces odpowiedzialny za komunikację z kontrolerem, propagację konfiguracji oraz dystrybucję aplikacji i zarządzanie instancjami serwerów
- **Application server nodes** - instancje serwera WildFly

Startując domenę można zauważyć dodatkowy proces tzw. **Process Controller**. Jego głównym zadaniem jest zarządzanie procesami. W jego przypadku nie ma możliwości dodatkowej konfiguracji.

Pliki konfiguracyjne w ramach domeny znajdują się w w katalogu **JBOSS_HOME/domain/configuration** i są to:

- **domain.xml** - główny plik konfiguracyjny, czytany tylko przez węzeł master, zawiera definicje wszystkich dostępnych profili
- **host.xml** - konfiguracja fizycznego hosta należącego do domeny,
- **host-master.xml** - konfiguracja niezbędna do uruchomienia master domain controller
- **host-slave.xml** - konfiguracja niezbędna do uruchomienia managed domain host controller

Domyślnie przy starcie używana jest konfiguracja z host.xml, jednak można wskazać inną:

```
domain.sh --host-config=host-master.xml
```

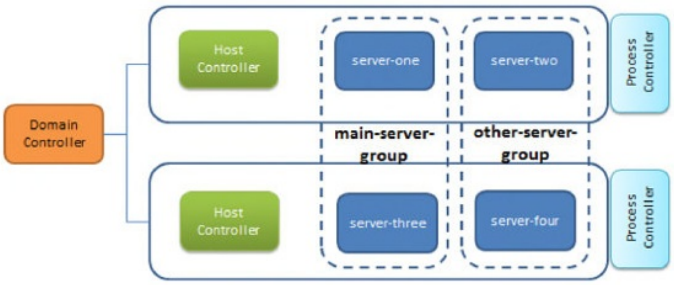
Domain Cotroller

Konfiguracja wszystkich serwerów działających w domenie znajduje się w pliku **JBOSS_HOME/domain/domain.xml** na kontrolerze domeny (plik nie jest wymagany na pozostałych węzłach). Zawiera ona definicje grup serwerów, ich powiązania z konkretnymi profilami, bindowania grup portów oraz inne.

Domyślnie zdefiniowane profile to:

- **default** - JEE Web profile
- **full** - JEE Full profile
- **ha** - rozszerza Web profile o możliwość klastrowania
- **full-ha** - rozszerza Full profile o możliwość klastrowania

Kontroler domeny jest odpowiedzialny za zarządzanie całą domeną jednak nie jest wymagany do działania pozostałych węzłów.



Przykład definicji grup logicznych zgodnych z powyższym diagramem:

```
<server-groups>
    <server-group name="main-server-group" profile="full">
        <jvm name="default">
            <heap size="64m" max-size="512m"/>
        </jvm>
        <socket-binding-group ref="full-sockets"/>
    </server-group>
    <server-group name="other-server-group" profile="full">
        <jvm name="default">
            <heap size="64m" max-size="512m"/>
        </jvm>
        <socket-binding-group ref="full-sockets"/>
    </server-group>
</server-groups>
```

Innym elementem konfiguracyjnym jest plik **host.xml**, który określa:

- serwery i ich przynależność do grup

- interfejsy sieciowe i ustawienia związane z bezpieczeństwem
- lokalizację kontrolera domeny
- nazwę hosta

W założonym przykładzie nie ma serwerów aplikacyjnych na fizycznej maszynie kontrolera domeny (dedykowany kontroler). Dlatego element definiujący serwery powinien pozostać pusty.

```
<servers/>
```

Z uwagi na to że kontroler domeny jest lokalny wskazujemy go przez element **local**.

```
<domain-controller>
    <local/>
</domain-controller>
```

Uruchomienie domeny odbywa się w standardowy sposób

```
./domain.sh -Djboss.bind.address.management=192.168.0.1
```

Host Controllers

Host Controller ładuje konfigurację z kontrolera i wykorzystuje lokalny plik **host.xml** do skonfigurowania serwerów działających na danym węźle. Pierwszą istotną kwestią jest określenie unikalnej nazwy hosta w ramach domeny.

```
<host name="host1" xmlns="urn:jboss:domain:4.0">
    ...
</host>
```

Następnie należy wskazać gdzie znajduje się kontroler domeny. Najlepiej wykorzystać zdefiniowane zmienne i podawać jego adres w momencie uruchamiania instancji. Ze względów bezpieczeństwa wymagane jest także podanie użytkownika wykorzystywanego w czasie połączenia.

```
<domain-controller>
    <remote host="${jboss.domain.master.address}" port="${jboss.domain.master.port:999
9}" username="wild yadmin" security-realm="ManagementRealm"/>
</domain-controller>
```

Dodatkowo musimy ustawić hasło zakodowane w postaci Base64 na poziomie znacznika **server-identities**. Uwierzytelnianie jest konieczne z uwagi na to, że kontroler domeny znajduje się na innej maszynie.

```
<management>
    <security-realms>
        <security-realm name="ManagementRealm">
            <server-identities>
                <secret value="RXJpY3Nzb24xIQ==" />
            </server-identities>
            ......
        </security-realm>
    </security-realms>
    ......
</management>
```

Ostatecznie należy skonfigurować poszczególne instancje serwerów w każdym **host.xml**. Flaga autostart określa czy serwery mają startować automatycznie po podniesieniu Host Controllera. Określenie przesunięcia portów zapobiega konfliktom na jednej fizycznej maszynie.

```
// Dla Host1
<servers>
    <server name="server-one" group="main-server-group"/>
    <server name="server-two" group="other-server-group" auto-start="false">
        <socket-bindings port-offset="150"/>
    </server>
</servers>
```

```
// Dla Host2
<servers>
    <server name="server-three" group="main-server-group"/>
    <server name="server-four" group="other-server-group" auto-start="false">
        <socket-bindings port-offset="150"/>
    </server>
</servers>
```

Uruchomienie obu serwerów:

```
./domain.sh -b 192.168.0.2 -Djboss.domain.master.address=192.168.0.1
./domain.sh -b 192.168.0.3 -Djboss.domain.master.address=192.168.0.1
```

Po prawidłowym zakończeniu procesu na konsoli kontrolera domeny powinna pojawić się informacja o dołączonych węzłach.

CLI

Uruchomienie CLI jest realizowane z użyciem skryptu **jboss-cli.sh**. Na początku należy podłączyć się do serwera za pomocą polecenia **connect**. Opcjonalnie powyższe kroki można wykonać wykonać w trybie wsadowym.

```
 ./jboss-cli.sh –-connect 192.168.10.1 –-user=admin1234 –-password=password1234!
```

Inną ciekawym przełącznikiem jest **--file** umożliwiający uruchamianie gotowych skryptów CLI.

```
./jboss-cli.sh jboss-cli.bat –-file=myscript.cli
```

Komendy być także uruchomiane w trybie nieinteraktywnym po podaniu ich z przełącznikiem **--command** lub **--commands**.

```
./jboss-cli.sh --commands="connect,deploy Utility.jar“
```

Ze względów administracyjnych możliwe jest uruchomienie CLI w trybie w którym możliwe będzie zarządzanie konfiguracją, jednak sam serwer i jego usługi nie będą działały.

```
./standalone.sh --admin-only
reload --admin-only=false
```

Konsola aktywnie wspiera administratora podpowiadając zarówno poszczególne elementy ścieżki jak i możliwe operacje do wykonania po naciśnięciu klawisza tab.
Każde polecenie CLI składa się z trzech części:

- adresu poprzedzonego /
- nazwy operacji poprzedzonej :
- opcjonalną listą parametrów podanych w nawiasach, po przecinku, w postaci klucz=wartość

Adres może składać się z wielu członów, a każdy człon ma postać /node-type=node-name

```
/subsystem=undertow/server=default-server/http-listener=default/:read-resource()
/subsystem=undertow/server=default-server/http-listener=default/:read-attribute(name=e
nabled)
/socket-binding-group=standard-sockets/socket-binding=http/:write-attribute(name=port,
value=8280)
/subsystem=naming/binding=myname/:add(binding-type=simple,cache=false,value=value)
/subsystem=logging:read-children-resources(child-type=log-file)
```

Każdy z zasobów może posiadać dodatkowe, specyficzne operacje

```
/subsystem=naming/:jndi-view
```

CLI można także uruchomić w trybie graficznym

```
./jboss-cli.sh –-gui
```

Istnieje możliwość uruchamiania poleceń w trybie wsadowym. Jeśli jakakolwiek operacja się nie powiedzie pozostałe zostaną wycofane.

```
[standalone@localhost:9990 /] batch
[standalone@localhost:9990 /#] deploy MyApplication.jar
[standalone@localhost:9990 /#] /system-property=myprop:add(value=myvalue)
[standalone@localhost:9990 /#] run-batch
```

Istnieje możliwość pokazania / wylistowania dodanych dodanych komend.

```
standalone@localhost:9990 /] list-batch
```

A także możliwość wstrzymania, a później wznowienia operacji batch np. kiedy trzeba w między czasie wykonać niezależnie inną operację.

```
[standalone@localhost:9990 / #] undeploy myproject.war
#1 undeploy myproject.war
[standalone@localhost:9990 / #] holdback-batch
```

Wznowienie dodawania poleceń odbywa się przez ponowne wykonanie komendy **batch**. Co więcej istnieje także możliwość robienia savepointów

```
[standalone@localhost:9990 /# ] holdback-batch step1
...
[standalone@localhost:9990 /] batch step1
```

Instrukcje w trybie wsadowym mogą być składowane i uruchamiane z pliku.

```
run-batch --le=myscript.cli --verbose
```

Wszystkie dokonane zmiany konfiguracji są automatycznie zapisywane i można je odtworzyć tak jak przy systemach kontroli wersji (katalogi standalone_xml_history i domain_xml_historyw katalogu z konfiguracją).

Zawartość:

- standalone.initial.xml - oryginalna konfiguracja, zapisana po pierwszym poprawnym uruchomieniu serwera. Ten plik nie jest nadpisywany
- standalone.boot.xml - plik nadpisywany po każdym poprawnym uruchomieniu serwera
- standalone.last.xml - plik nadpisywany po każdej zmianie konfiguracji

Oprócz tego mamy katalog **current** który jest czyszczony przy starcie i zawiera 100 ostatnich zmian konfiguracji. Przy każdym uruchomieniu serwera folder jest kopiowany z znacznikiem czasowym i trzymany przez 30 dni.
Ostatnim folderem jest **snapshot** gdzie trzymane są zrzuty konfiguracji wykonane jawnie z CLI.

```
:take-snapshot
:list-snapshots
:delete-snapshot(name="20131108-171642235standalone.xml")
```

Zawieszanie serwera (pozwala na przetworzenie istniejących żądań bez przyjmowania nowych). Klienci dostają status 503. W czasie kiedy serwer jest zawieszony nadal można zarządzać konfiguracją i wykonywać deploy/undeploy.

```
:suspend
:read-attribute(name=suspend-state)
:resume
```

Przykłady komend konfiguracyjnych

Interfejsy sieciowe

Zmiana adresu interfejsu administracyjnego (zaleca się żeby robić to przez parametry startowe).

```
/interface=management/:write-attribute(name=inet-address,value=${jboss.bind.address.ma
nagement:192.168.10.1 })
```

Zmiana adresu interfejsu publicznego.

```
/interface=public/:write-attribute(name=inet-address,value=${jboss.bind.address:192.16
8.10.1})
```

Usostępnienie interfejsu na wszystkich adresach IP.

```
/interface=public/:write-attribute(name=inet-address,value=${jboss.bind.address:0.0.0.
0})
```

Wszystkie powyższe zmiany wymagają przeładowania konfiguracji

```
reload
```

Porty

Określenie / modyfikacja portów

```
 ./standalone.sh -Djboss.http.port=8280
```

```
/socket-binding-group=standard-sockets/socket-binding=http/:write-attribute(name=port,
value=${jboss.http.port:8180})
```

Dodawanie ścieżki

```
/path=logpath/:add(path=/home/logs)
/path=mydir/:add(relative-to=jboss.server.base.dir)
```

Properties

Praca ze zmiennymi

```
/system-property=mykey/:add(value=value)
/system-property=mykey/:read-resource(recursive=false)
/system-property=mykey/:remove
/core-service=platform-mbean/type=runtime:read-attribute(name=system-properties)
```

Zarządzanie na poziomie domeny

Podłączenie do kontrolera domeny

```
./jboss-cli.sh --connect controller=192.168.0.1:9990
```

Po naduszeniu tab pokazują się nowe opcje do zarządzania domeną. Dla nas istotne są:

- *profile* - pozwala na zarządzanie profilami zdefiniowanymi w domenie
- *host* - operacje wykonywane na poszczególnych hostach (start/stop/restart/reload)
- *server-group* - wykonywanie operacji na grupach serwerów

Przykłady użycia

Modyfikacja ustawień profilu polegająca na zmianie ustawień timeoutu dla SLSB

```
/profile=full/subsystem=ejb3/strict-max-bean-instance-pool=slsb-strict-max-pool:write-
attribute(name=timeout,value=100)
```

Wykonanie restartu server-one dostępnego na host master

```
 /host=master/server-config=server-one:restart
```

Dodanie nowego serwera

```
 /host=master/server-config=server-five:add(auto-start=false, socket-binding-port-o se
t=400, group=main-server-group)
 /host=master:read-children-names(child-type=server-config)
 /host=master/server-config=server-five:start
```

Usuwanie zatrzymanego serwera.

```
  /host=master/server-config=server-five:remove
```

Sprawdzanie informacji o wybranych parametrach serwera.

```
/host=master/server=server-one/subsystem=datasources/data-source=ExampleDS/statistics=
pool:read-resource(include-runtime=true)
```

Zarządzanie Host Controllerem (po zatrzymaniu należy ponownie odpalić domain.sh).

```
/host=slave:reload
/host=slave:shutdown
```

Operowanie na grupie serwerów.

```
/server-group=main-server-group:restart-servers
/server-group=main-server-group:reload-servers
```

Wdrażanie aplikacji

File system deploy

Działa tylko w trybie standalone. Sprowadza się do skopiowania archiwum do katalogu **deployments**. Domyślnie archiwa powinny być wdrażane automatycznie, a aplikacje rozpakowane (exploded) nie.

Skaner wykrywający aplikacje może działać w dwóch trybach:

- **auto-deploy** - automatyczne wykrywanie nowych elementów oraz zmian znacznika czasowego skutkujące wdrożeniem
- **manual-deploy** - wdrożenie odbywa się z użyciem tzw. pliki markerów

```
cp example.war /usr/wildly-10.0.0.Final/standalone/deployments
```

```
cp -r Example.ear $JBOSS_HOME/standalone/deployments
$ touch $JBOSS_HOME/standalone/deployments/Example.ear.dodeploy
```

Gdyby wdrożenie się nie powiodło utworzony zostanie plik z rozszerzeniem **.failed**.

Konfiguracja ustawień skanera:

```
/subsystem=deployment-scanner/scanner=default:read-resource
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-enabled,value=
false)
/subsystem=deployment-scanner/scanner=default:write-attribute(name=auto-deploy-explode
d,value=true)
/subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-interval,value
=10000)
/subsystem=deployment-scanner/scanner=default:write-attribute(name=path,value=/path/to
/deployments)
```

CLI

Tryb standalone

```
deploy /home/user1/myproject.war
undeploy myproject.war
deploy /home/user1/myexplodedapp.war --unmanaged
deploy -f myproject.war
deploy --url=https://dropbox.com/9766485/helloworld.war --name=helloworld.war
deploy -l
```

Tryb domain

```
deploy application.war --all-server-groups
deploy application.war --server-groups=main-server-group
undeploy application.war --all-relevant-server-groups
undeploy application.war --server-groups=main-server-group
undeploy application.war --server-groups=main-server-group --keep-content
```

Plugin do Maven

```
<plugin>
    <groupId>org.wildfly.plugins</groupId>
    <artifactId>wildfly-maven-plugin</artifactId>
    <version>${version.wildfly.maven.plugin}</version>
</plugin>

mvn clean install wildfly:deploy
mvn wildfly:undeploy


<plugin>
    <groupId>org.wildfly.plugins</groupId>
    <artifactId>wildfly-maven-plugin</artifactId>
    <version>${version.wildfly.maven.plugin}</version>
    <configuration>

<domain>
            <server-groups>
                <server-group>main-server-group</server-group>
            </server-groups>
        </domain>
    </configuration>
</plugin>
```

Źródło danych

CLI

Tryb standalone

```
module add --name=com.mysql --resources=/var/mysql-connector-java-5.1.24-bin.jar --dep
endencies=javax.api,javax.transaction.api

/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module- name=com
.mysql)

data-source add --jndi-name=java:/MySqlDS --name=MySQLPool --connection-url=jdbc:mysql
://localhost:3306/mysqlschema --driver-name=mysql --user-name=jboss --password=jboss

/subsystem=datasources/data-source=MySQLPool:test-connection-in-pool
```

W przypadku XA Datasource ostatni krok wygląda tak

```
xa-data-source add --name=MySqlDSXA --jndi-name=java:/MySqlDSXA --driver-name=mysql --
xa-datasource-class=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource --user-name=jboss
--password=jboss --xa-datasource-properties=[{ServerName=localhost}, {DatabaseName=mys
qlschema}]
```

Tryb domeny

Należy zainstalować moduł na każdym Host kontrolerze

```
module add --name=com.mysql --resources=/var/mysql-connector-java-5.1.24-bin.jar --dep
endencies=javax.api,javax.transaction.api
```

Dalej działamy na kontrolerze domeny i instalujemy źródło w profilu

```
/profile=full-ha/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-
 module-name=com.mysql)

data-source add --jndi-name=java:/MySqlDS --name=MySQLPool --connection-url=jdbc:mysql
://localhost:3306/mysqldb --driver-name=mysql --user-name=jboss --password=jboss --pro
file=full-ha
```

Jako zasób

Może być wrzucone bezpośrednio lub w ramach aplikacji (WEB-INF, META-INF)
Wady: Brak możliwości zarządzania z poziomu CLI i konsoli

```
cp mysql-connector-java-5.1.24-bin.jar /usr/share/wildfly-10.0.0.Final/standalone/depl
oyments

// plik my-datasource-ds.xml
<datasources xmlns="http://www.jboss.org/ironjacamar/schema">
    <datasource jndi-name="java:/MySqlDS" pool-name="MySQLPool">
          <connection-url>jdbc:mysql://localhost:3306/mysqlschema</connection-url>
          <driver>mysql-connector-java-5.1.24-bin.jar</driver>
          <pool>
              <max-pool-size>30</max-pool-size>
          </pool>
          <security>
              <user-name>jboss</user-name>
              <password>jboss</password> </security>
    </datasource>
</datasources>
```

Konfiguracja

```
/subsystem=datasources/data-source=MySqlDS:write-attribute(name=min-pool-size,value=10
) /subsystem=datasources/data-source=MySqlDS:write-attribute(name=max-pool-size,value=
50)

/subsystem=datasources/data-source=MySqlDS:write-attribute(name=statistics- enabled,va
lue=true)
/subsystem=datasources/data-source=MySqlDS/statistics=pool:read-resource(include- runt
ime=true)
```

Bezpieczeństwo

Domena bezpieczeństwa - konfiguracja uwierzytelniania, autoryzacji oraz innych elementów związanych z bezpieczeństwem. Implementuje Java Authentication and Authorization Service (JAAS) declarative security.

Domyślnie zdefiniowane są następujące **Security Realm**.

- **ManagementRealm** - umożliwia bezpieczny dostęp do interfejsów administracyjnych - CLI i Web Console
- **ApplicationRealm** - związany z bezpieczeństwem wdrożonych aplikacji

W obu przypadkach dodawanie nowych użytkowników odbywa się z pomocą skryptu **add-user.sh**.

ManagementRealm

Domyślnie opiera się o prosty mechanizm uwierzytelniania i autoryzacji zakładający konfigurację opartą o pliki **mgmt-users.properties** i **mgmt-groups.properties** znajdujące się w katalogu z konfiguracją.

```
<security-realm name="ManagementRealm">
    <authentication>
        <local default-user="$local" />
        <properties path="mgmt-users.properties" relative-to="jboss.server.conKg.dir"
/>
    </authentication>
    <authorization map-groups-to-roles="false">
        <properties path="mgmt-groups.properties" relative-to="jboss.server.conKg.dir"
 />
    </authorization>
</security-realm>
```

W przypadku CLI uwierzytelnianie zachodzi automatycznie jeśli próba dostępu jest realizowana z lokalnej maszyny.

ApplicationRealm

Wykorzystywany do zabezpieczania aplikacji na poziomie webowym jak i komponentów EJB.

```
<security-realm name="ApplicationRealm">
    <authentication>
        <local default-user="$local" allowed-users="*" />
        <properties path="application-users.properties" relative-to="jboss.server.conK
g.dir" />
        </authentication>
    <authorization>
    <properties path="application-roles.properties" relative-to="jboss.server.conKg.di
r" />
    </authorization>
</security-realm>
```

Proces uwierzytelniania zachodzi tak samo jak w poprzednim przypadku, z tą różnicą że dane uwierzytelniające znajdują się w pliku **application-user.properties**. Proces autoryzacji przeprowadzany jest na podstawie pliku **application-roles.properties**.

Security Domains

Konfiguracja modułów logowania oraz mechanizmów mapowania ról. Znajduje się w podpakiecie **security**.

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
    <security-domains>
        <security-domain name="other" cache-type="default">
            <authentication>
                <login-module code="Remoting" flag="optional">
                    <module-option name="password-stacking" value="useFirstPass" />
                </login-module>
                <login-module code="RealmDirect" flag="required">
                    <module-option name="password-stacking" value="useFirstPass" />
                </login-module>
            </authentication>
        </security-domain>
    </security-domains>
</subsystem>
```

Domyślnie skonfigurowana i zarazem podstawowa konfiguracja domeny nazywa się other. Definiuje ona moduły **Remoting** i **RealmDirect**. Pierwszy używany jest wewnętrznie podczas gdy potrzebne jest uwierzytelnianie na poziomie zdalnych połączeń (zwykle wywołania EJB). Drugi używany jest kiedy np. logujemy się z poziomu konsoli webowej.

Domyślnie dostępne moduły bezpieczeństwa

- RealmDirect - oparty o pliki tekstowe
- Database - oparty o bazę relacyjną

- LDAP - oparty o bazę LDAP

High availability

HA + Apache + mod_proxy jako loadbalancer

Procedura konfiguracyjna

- Ściągnij Apache z modułem mod_proxy http://www.jboss.org/mod_cluster/downloads
- Zainstaluj serwer Apache

```
sudo tar xvfz mod_cluster-1.2.6.Final-linux2-x64.tar.gz -C /
```

> Dla ubuntu dodatkowo:
>
> Usunąć bibliotekę /opt/jboss/httpd/lib/libapr1
>
> sudo apt-get install libaprutil1 libaprutil1-dev libapr1 libapr1-dev

- Uruchom serwer Apache (przeglądarka powinna wyświetlić It's works!)

```
sudo /opt/jboss/httpd/sbin/apachectl start
```

- Ustaw nazwę dla każdego serwera - atrybut name na poziomie znacznkia server
- Uruchom instancje serwera WildFly w konfiguracji trybie ha lub full-ha.
- Przetestuj działanie load balancera http://localhost:6666/mod_cluster_manager
- Zmodyfikuj plik http.conf podając adres ip balancera oraz port

HA + WildFly jako loadbalancer

IMPORTANT: this config is not suitable for production use, as it only uses a single network interface for production use the socket bindings used by modcluster (management and advertise) should be only exposed to the internal network, not a public IP

Cleanup existing servers that are enabled by default

```
:stop-servers(blocking=true)
/host=master/server-config=server-one:remove
/host=master/server-config=server-two:remove
/host=master/server-config=server-three:remove
/server-group=main-server-group:remove
/server-group=other-server-group:remove
```

Add the IP address 192.168.1.4 to the host aliases (replace with your local IP)

```
/profile=ha/subsystem=undertow/server=default-server/host=default-host:write-attribute
(name=alias, value=[192.168.1.4])
```

Set the mod_cluster security key

```
/profile=ha/subsystem=modcluster/mod-cluster-config=configuration:write-attribute(name
=advertise-security-key, value=mypassword)
```

Add backend server group

```
/server-group=backend-servers:add(profile=ha, socket-binding-group=ha-sockets)
```

Add backend servers, using the ha profile so mod_cluster support is included

```
/host=master/server-config=backend1:add(group=backend-servers, socket-binding-port-off
set=100)
/host=master/server-config=backend2:add(group=backend-servers, socket-binding-port-off
set=200)
```

Start the backend servers

```
/server-group=backend-servers:start-servers
```

Add system properties (so we can tell them apart)

```
/host=master/server-config=backend1/system-property=server.name:add(boot-time=false, v
alue=backend1)
/host=master/server-config=backend2/system-property=server.name:add(boot-time=false, v
alue=backend2)
```

Deploy the demo to our backend servers

```
deploy ~/workspace/modcluster-example/load-balancing-demo/target/clustering-demo.war --server-groups=backend-servers
```

Now set up the default profile to include the mod_cluster load balancer

```
/profile=default/subsystem=undertow/configuration=filter/mod-cluster=modcluster:add(management-socket-binding=http, advertise-socket-binding=modcluster, security-key=mypassword)
/profile=default/subsystem=undertow/server=default-server/host=default-host/filter-ref=modcluster:add
/socket-binding-group=standard-sockets/socket-binding=modcluster:add(multicast-port=23364, multicast-address=224.0.1.105)
/server-group=load-balancer:add(profile=default, socket-binding-group=standard-sockets)
/host=master/server-config=load-balancer:add(group=load-balancer)
/server-group=load-balancer:start-servers
```

Now lets add another server

```
/host=master/server-config=backend3:add(group=backend-servers, socket-binding-port-offset=300)
:start-servers
/host=master/server-config=backend3/system-property=server.name:add(boot-time=false, value=backend3)
```

And remove one

```
/host=master/server=backend1:stop
```