

SOA – laboratorium nr 1

Temat: Wprowadzenie do tworzenia aplikacji biznesowych w technologii JavaEE.

Poznanie środowiska serwerów Aplikacyjnych

Cel laboratorium:

1. Zaznajomienie się z środowiskiem potrzebnym do tworzenia Aplikacji webowych tworzonych w technologii JavaEE.
2. Poznanie i konfiguracja wybranego serwera aplikacyjnego.
3. Uruchomienie prostej aplikacji webowej na serwerze aplikacyjnym.

Wstęp.

Java EE (nazywana dawniej J2EE) wykorzystuje standardowy zestaw technologii do tworzenia aplikacji uruchamianych po stronie serwera. Technologia Javy EE zawiera:

servlety, Java Server Pages (JSP), Java Server Faces (JSF), Enterprise JavaBeans (EJB), Context Dependency Injection (CDI), Java Messaging Service (JMS), Java Persistence API (JPA), Java API for XML Web Services (JAX-WS) i Java API for RESTful Web Services (JAX-RS), a także kilka innych.

Istnieje kilka serwerów aplikacji umożliwiających programistom uruchamianie aplikacji zgodnych z Javą EE, jedne są komercyjne, inne ze źródłami otwartymi.

JBoss WildFly (dawna nazwa AS Application Server) to jedno z wiodących rozwiązań typu open source wykorzystywanych przez programistów. Choć trudno to dokładnie zmierzyć, jest wielce prawdopodobne, że stanowi on najczęściej wykorzystywany serwer aplikacyjny na rynku.

Inne popularne serwery aplikacyjne to Apache Tomcat oraz Glassfish.

Ćwiczenia

Pierwszym krokiem przed przystąpieniem do właściwej nauki serwera aplikacji jest instalacja na komputerze wszystkich elementów niezbędnych do uruchomienia tego serwera.

Serwer aplikacji wymaga tylko środowiska maszyny wirtualnej Javy.

W kwestii wymagań sprzętowych warto wiedzieć, że dystrybucja serwera wymaga jako minimum około 75 MB przestrzeni na dysku twardym i alokuje od 64 MB (minimum) do 512 MB pamięci RAM w przypadku serwera niezależnego.

Przygotowanie środowiska wymagać będzie realizacji pewnych kroków. Oto one:

- Instalacja Java Development Kit pozwalającego na uruchomienie serwera JBoss.
- Instalacja serwera JBoss - proponuje najnowszą aktualnie wersję WildFly 16
- Instalacja środowiska programistycznego Eclipse / Intelij / NetBeans (według własnych preferencji)

- Instalacja narzędzia Maven lub innego narzędzia do automatyzacji tworzenia oprogramowania.

Ćwiczenie 1. Instalacja i konfiguracja oprogramowania.

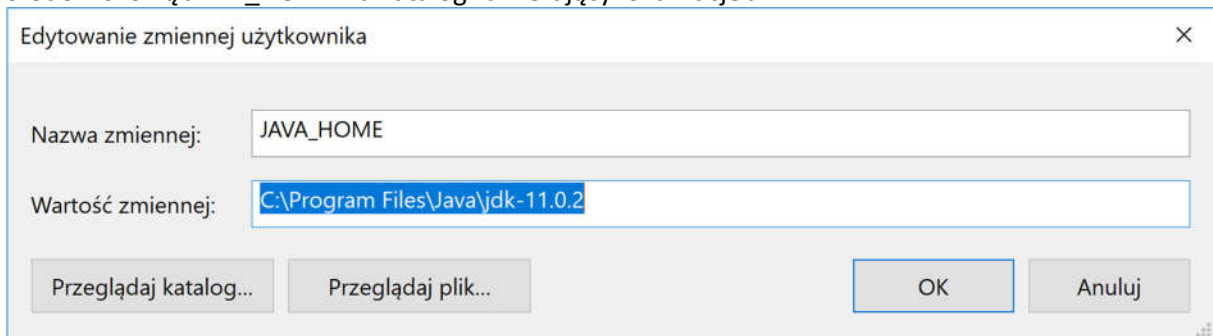
Jeśli korzystasz z komputera z laboratorium sprawdź czy wszystkie wymienione wcześniej komponenty są poprawnie zainstalowane. Jeśli czegoś brakuje – doinstaluj lub skonfiguruj.

Jeśli masz własny laptop – kolejno zainstaluj wymagane elementy (możliwe zamiana serwera JBoss na inny serwer Aplikacyjny podobnie jak i środowiska IDE).

Sprawdź czy na komputerze jest zainstalowana Java JDK (interesuje nas wersja minimum 1.8)

Instalacja JBoss WildFly 16 - Serwer aplikacji JBoss można pobrać bezpłatnie z witryny społeczności pod adresem <http://wildfly.org/downloads>. Sugeruje wybrać wersję WildFly 16.0. Instalacja polega tylko na rozpakowaniu zip w odpowiednim miejscu na dysku. Od tej chwili serwer jest gotowy do użycia.

Aby poprawnie pracować serwer musi być świadomy istnienia JDK Javy. W tym celu ustaw zmienną środowiskową JAVA_HOME na katalog zawierający lokalizację JDK.



Wykonaj test instalacji JBoss.

Po instalacji serwera JBoss warto przeprowadzić prosty test, by sprawdzić, czy nie istnieją żadne poważne problemy z systemem operacyjnym lub dostępem do maszyny wirtualnej Javy. Aby przetestować instalację, trzeba przejść do folderu bin z instalacji JBoss i wykonać polecenie:

```
standalone.bat #Windows  
$ standalone.sh #Linux/Unix
```

Oto wygląd konsoli tuż po uruchomieniu serwera JBoss.

```
C:\windows\system32\cmd.exe
JAVA: "C:\Program Files\Java\jdk-11.0.2\bin\java"

JAVA_OPTS: "-Dprogram.name=standalone.bat -Xms64M -Xmx512M -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman --add-exports=java.base/sun.nio.ch=ALL-UNNAMED --add-exports=jdk.unsupported/sun.misc=ALL-UNNAMED --add-exports=jdk.unsupported/sun.reflect=ALL-UNNAMED --add-modules=java.se"

=====
12:36:51,050 INFO [org.jboss.modules] (main) JBoss Modules version 1.9.0.Final
12:36:51,633 INFO [org.jboss.msc] (main) JBoss MSC version 1.4.5.Final
12:36:51,641 INFO [org.jboss.threads] (main) JBoss Threads version 2.3.3.Final
12:36:51,748 INFO [org.jboss.as] (MSC service thread 1-1) WFLYSRV0049: WildFly Full 16.0.0.Final (WildFly Core 8.0.0.Final) starting
12:36:52,295 INFO [org.wildfly.security] (ServerService Thread Pool -- 27) ELY00001: WildFly Elytron version 1.8.0.Final
12:36:52,698 INFO [org.jboss.as.controller.management-deprecated] (Controller Boot Thread) WFLYCTL0028: Attribute 'security-realm' in the resource at address '/core-service=management/management-interface=http-interface' is deprecated, and may be removed in a future version. See the attribute description in the output of the read-resource-description operation to learn more about the deprecation.
12:36:52,722 INFO [org.jboss.as.controller.management-deprecated] (ServerService Thread Pool -- 8) WFLYCTL0028: Attribute 'security-realm' in the resource at address '/subsystem=undertow/server=default-server/https-listener=https' is deprecated, and may be removed in a future version. See the attribute description in the output of the read-resource-description operation to learn more about the deprecation.
12:36:52,767 INFO [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0039: Creating http management service using socket-binding (management-http)
12:36:52,782 INFO [org.xnio] (MSC service thread 1-7) XNIO version 3.6.5.Final
12:36:52,789 INFO [org.xnio.nio] (MSC service thread 1-7) XNIO NIO Implementation Version 3.6.5.Final
12:36:52,828 INFO [org.jboss.as.naming] (ServerService Thread Pool -- 62) WFLYNAM0001: Activating Naming Subsystem
12:36:52,831 INFO [org.jboss.as.security] (ServerService Thread Pool -- 68) WFLYSEC0002: Activating Security Subsystem
12:36:52,835 INFO [org.jboss.as.webservices] (ServerService Thread Pool -- 72) WFLYWS0002: Activating WebServices Extension
12:36:52,840 INFO [org.jboss.as.clustering.infinispan] (ServerService Thread Pool -- 49) WFLYCLINF0001: Activating Infinispan subsystem.
12:36:52,856 INFO [org.wildfly.extension.microprofile.opentracing] (ServerService Thread Pool -- 61) WFLYTRACEXT0001: Activating MicroProfile OpenTracing Subsystem
12:36:52,840 WARN [org.jboss.as.txn] (ServerService Thread Pool -- 70) WFLYTX0013: The node-identifier attribute on the /subsystem=transactions is set to the default value. This is a danger for environments running multiple servers. Please make sure the attribute value is unique.
12:36:52,876 INFO [org.jboss.as.connector] (MSC service thread 1-6) WFLYJCA0009: Starting JCA Subsystem (WildFly/IronJacamar 1.4.12.Final)
12:36:52,890 INFO [org.jboss.as.security] (MSC service thread 1-5) WFLYSEC0001: Current PicketBox version=5.0.3.Final
12:36:52,893 INFO [org.jboss.as.jsf] (ServerService Thread Pool -- 56) WFLYJSF0007: Activated the following JSF Implementations: [main]
```

```
C:\windows\system32\cmd.exe
12:36:53,024 INFO [org.jboss.as.mail.extension] (MSC service thread 1-7) WFLYMAIL0001: Bound mail session [java:jboss/mail/Default]
12:36:53,051 INFO [org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-7) WFLYJCA0010: Unbound data source [java:jboss/datasources/ExampleDS]
12:36:53,053 INFO [org.jboss.remoting] (MSC service thread 1-5) JBoss Remoting version 5.0.8.Final
12:36:53,183 INFO [org.jboss.as.ejb3] (MSC service thread 1-8) WFLYEJB0481: Strict pool slsb-strict-max-pool is using a max instance size of 128 (per class), which is derived from thread worker pool sizing.
12:36:53,183 INFO [org.jboss.as.ejb3] (MSC service thread 1-4) WFLYEJB0482: Strict pool mdb-strict-max-pool is using a max instance size of 32 (per class), which is derived from the number of CPUs on this host.
12:36:53,212 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 71) WFLYUT0014: Creating file handler for path 'C:\wildfly-16.0.0.Final\welcome-content' with options [directory-listing: 'false', follow-symlink: 'false', case-sensitive: 'true', safe-symlink-paths: '[]']
12:36:53,238 INFO [org.wildfly.extension.undertow] (MSC service thread 1-8) WFLYUT0012: Started server default-server.
12:36:53,261 INFO [org.wildfly.extension.undertow] (MSC service thread 1-4) WFLYUT0018: Host default-host starting
12:36:53,428 INFO [org.wildfly.extension.undertow] (MSC service thread 1-5) WFLYUT0006: Undertow HTTP listener default listening on 127.0.0.1:8080
12:36:53,466 INFO [org.jboss.as.patching] (MSC service thread 1-5) WFLYPAT0050: WildFly Full cumulative patch ID is: base, one-off patches include: none
12:36:53,481 WARN [org.jboss.as.domain.management.security] (MSC service thread 1-7) WFLYDM0111: Keystore C:\wildfly-16.0.0.Final\standalone\configuration\application.keystore not found, it will be auto generated on first use with a self signed certificate for host localhost
12:36:53,489 INFO [org.jboss.as.server.deployment.scanner] (MSC service thread 1-7) WFLYDS0013: Started FileSystemDeploymentService for directory C:\wildfly-16.0.0.Final\standalone\deployments
12:36:53,497 INFO [org.jboss.as.ejb3] (MSC service thread 1-1) WFLYEJB0493: EJB subsystem suspension complete
12:36:53,564 INFO [org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-6) WFLYJCA0001: Bound data source [java:jboss/datasources/ExampleDS]
12:36:53,564 INFO [org.wildfly.extension.undertow] (MSC service thread 1-2) WFLYUT0006: Undertow HTTPS listener https listening on 127.0.0.1:8443
12:36:53,619 INFO [org.jboss.ws.common.management] (MSC service thread 1-5) JBWS022052: Starting JBossWS 5.2.4.Final (Apache CXF 3.2.7)
12:36:53,725 INFO [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0212: Resuming server
12:36:53,729 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0060: Http management interface listening on http://127.0.0.1:9990/management
12:36:53,729 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin console listening on http://127.0.0.1:9990
12:36:53,738 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: WildFly Full 16.0.0.Final (WildFly Core 8.0.0.Final) started in 2938ms - Started 305 of 531 services (324 services are lazy, passive or on-demand)
```

Wskazane polecenie uruchamia niezależną instancję serwera JBoss.

Jeśli konieczne jest dostosowanie właściwości początkowych serwera aplikacji, trzeba otworzyć plik `standalone.conf` (lub plik `standalone.conf.bat` w przypadku systemu Windows) i znaleźć fragment odpowiedzialny za wymagania pamięciowe. Oto odpowiedni fragment dla systemu Linux.

```
if [ "x$JAVA_OPTS" = "x" ]; then
  JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -Dorg.jboss.resolver.
  warning=true -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.
  server.gcInterval=3600000"
fi
```

Domyślnie serwer używa minimalnie 64 MB pamięci operacyjnej na stos i maksymalnie 512 MB RAM. To odpowiednie wartości na początek. Jeśli jednak na serwerze mają być uruchamiane bardziej rozbudowane aplikacje Javy EE, najprawdopodobniej potrzeba będzie minimum 1 GB pamięci, a czasem nawet 2 GB.

Warto sprawdzić, czy uruchomiony serwer jest dostępny przez interfejs sieciowy, wskazując w przeglądarce ogólnie znany domyślny adres strony początkowej serwera aplikacji: <http://localhost:8080>.

Jeśli wszystko jest w porządku powinien pokazać się w oknie przeglądarki następujący ekran:



Serwer WildFly działa trybach — samodzielnym i domenowym.

W trybie **samodzielnym** każda instancja serwera stanowi niezależny proces. Pliki konfiguracyjne trybu samodzielnego znajdują się w folderze `standalone/configuration`.

W trybie **domenowym** możliwe jest uruchomienie wielu serwerów aplikacji, ale zarządzanie nimi z jednej, centralnej lokalizacji. Domena może włączyć wiele maszyn fizycznych (lub wirtualnych). Każda maszyna może zawierać kilka instancji Serwera znajdujących się pod kontrolą procesu sterującego maszyną. Pliki konfiguracyjne dotyczące trybu domenowego znajdują się w folderze `domain/configuration`.

Ćwiczenie 2. Przetestuj zarządzanie serwerem z wiersza poleceń. Wykonaj zatrzymanie i ponowne uruchomienie serwera. Przetestuj za pomocą przeglądarki czy serwer jest aktywny czy nie.

Połączenie z serwerem przy użyciu wiersza poleceń.

Osoby korzystające z poprzednich wydań serwera aplikacji przypominają sobie zapewne narzędzie `twiddle`, które służyło do odpytywania ziaren zainstalowanych na serwerze aplikacji.

Narzędzie zostało zastąpione przez bardziej wyrafinowany interfejs nazwany Command Line Interface (CLI); jest on dostępny w folderze JBOSS_HOME/bin.

Wystarczy uruchomić skrypt jboss-cli.bat (lub jboss-cli.sh w systemie Linux) i można zarządzać serwerem aplikacji z poziomu wiersza poleceń.

Zatrzymanie serwera JBoss

Prawdopodobnie najprostszym sposobem zatrzymania serwera jest wysłanie odpowiedniego sygnału za pomocą kombinacji klawiszy Ctrl+C.

Jeśli jednak proces JBoss został uruchomiony w tle lub działa na innym komputerze, do jego zatrzymania można użyć interfejsu CLI i polecenia shutdown.

```
[disconnected /] connect
Connected to localhost:9999
[localhost:9999 /] :shutdown
```

Zatrzymanie JBoss na zdalnym systemie

Zatrzymanie serwera aplikacji uruchomionego na zdalnym serwerze wymaga jedynie przekazania do CLI adresu serwera zdalnego, a także — ze względów bezpieczeństwa — nazwy użytkownika oraz hasła (tworzenie użytkowników zostało opisane w następnym rozdziale).

```
[disconnected /] connect 192.168.1.10
Authenticating against security realm: ManagementRealm
Username: admin1234
Password:
Connected to 192.168.1.10:9999
[192.168.1.10:9999 /] :shutdown
```

Ponowne uruchomienie JBoss

Interfejs wiersza poleceń zawiera wiele interesujących poleceń. Jednym z nich jest możliwość przeładowania konfiguracji Serwera Aplikacyjnego lub ich części za pomocą polecenia reload. Użycie polecenia w głównej ścieżce węzła serwera zapewnia przeładowanie pełnej konfiguracji usługi.

```
[disconnected /] connect
Connected to localhost:9999
[localhost:9999 /] :reload
```

Cwiczenie 3. Zapoznaj się z poniższym opisem mechanizmu zarządzania serwerem aplikacji. Spróbuj samodzielnie wykonać opisane zagadnienia.

Zarządzanie serwerem aplikacji

JBoss WildFly zapewnia trzy różne sposoby konfiguracji i zarządzania serwerami: są to

- interfejs webowy,
- klient wiersza poleceń (poznany w ćwiczeniu 2)

- zestaw plików konfiguracyjnych XML.

Niezależnie od wybranego sposobu edycji, cała konfiguracja jest zawsze synchronizowana pomiędzy poszczególnymi widokami i ostatecznie trafia do plików XML.

Zarządzanie JBoss WildFly przy użyciu interfejsu webowego

Interfejs webowy jest aplikacją GWT (Google Web Toolkit), która może służyć do zarządzania zarówno trybem samodzielnym, jak i trybem domenowym JBoss WildFly. Domyślnie jest uruchomiona w systemie lokalnym na porcie 9990, ale można to zmienić przy użyciu właściwości `jboss.management.http.port` znajdującej się w pliku konfiguracyjnym serwera (`standalone.xml` lub `domain.xml`).

```
<socket-binding-group name="standard-sockets" defaultinterface="public">
  <socket-binding name="management-http" interface="management"
    port="{jboss.management.http.port:9990}"/>
  .....
</socket-binding-group>
```

Aktualna ekran webowej konsoli zarządczej wygląda tak. Jak widać jesteśmy w trybie error – ze względu na brak stworzonego konta użytkownika.



Serwer JBoss jest domyślnie zabezpieczony. Domyślny mechanizm bezpieczeństwa korzysta z mechanizmu loginu i hasła. Powodem, dla którego serwer jest domyślnie zabezpieczony (wymaga uwierzytelnienia), jest możliwość przypadkowego udostępnienia interfejsu zarządzania pod publicznym adresem IP. Z tego samego powodu w dystrybucji nie stosuje się domyślnego użytkownika i hasła.

Dane użytkowników znajdują się w pliku właściwości o nazwie `mgmt-users.properties`, w folderach `standalone/configuration` lub `domain/configuration`, w zależności od trybu działania serwera. Plik zawiera informacje na temat nazwy użytkownika, wstępnie przygotowany skrót nazwy użytkownika, a także nazwę mechanizmu uwierzytelniania oraz hasło.

Do modyfikacji pliku i dodawania użytkowników służą narzędzia `add-user.sh` lub `add-user.bat`, które poza samym dodaniem użytkowników generują również odpowiednie skróty. Trzeba uruchomić skrypt i postępować zgodnie ze wskazówkami na ekranie.

Aby utworzyć nowego użytkownika, należy wprowadzić kilka wartości.

- **Typ użytkownika** (What type of user do you wish to add?) — wybieramy typ Management User, ponieważ użytkownik będzie zarządzał całym serwerem aplikacji.

- **Nazwa mechanizmu uwierzytelniania (Realm)** musi odpowiadać nazwie mechanizmu użytej w konfiguracji, więc jeśli w konfiguracji nie występuje inna nazwa, pozostawiamy domyślną nazwę **ManagementRealm**.
- **Nazwa użytkownika (Username)** — nazwa dodawanego użytkownika
- **Hasło (Password)** — hasło dla użytkownika.

```
C:\windows\system32\cmd.exe

What type of user do you wish to add?
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
(a):

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : grogus
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.

- The password should be different from the username
- The password should not be one of the following restricted values {root, admin, administrator}
- The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[ ]:
About to add user 'grogus' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'grogus' to file 'C:\wildfly-16.0.0.Final\standalone\configuration\mgmt-users.properties'
Added user 'grogus' to file 'C:\wildfly-16.0.0.Final\domain\configuration\mgmt-users.properties'
Added user 'grogus' with groups to file 'C:\wildfly-16.0.0.Final\standalone\configuration\mgmt-groups.properties'
Added user 'grogus' with groups to file 'C:\wildfly-16.0.0.Final\domain\configuration\mgmt-groups.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.
yes/no? yes
To represent the user add the following to the server-identities definition <secret value="SXp1bmlhXzcyciIQ==" />
Press any key to continue . . .
```

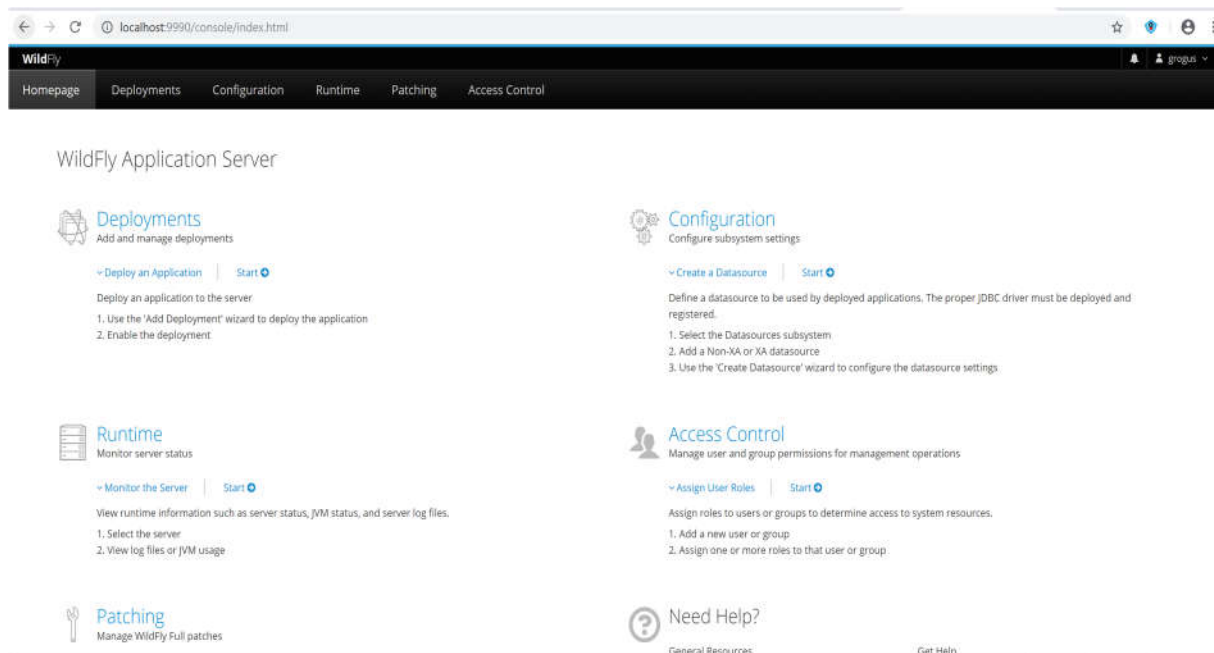
Uruchomienie konsoli webowej

Po dodaniu przynajmniej jednego użytkownika można przystąpić do uruchomienia konsoli webowej dostępnej pod domyślnym adresem <http://<host>:9990/console>.

W naszym przypadku proszę wpisać : ***http://localhost:9990***

Pojawi się prośba o podanie nazwy użytkownika oraz hasła. Wpisujemy w polach Nazwa użytkownika i Hasło dane użyte przy tworzeniu nowego użytkownika.

Po zalogowaniu nastąpi przekierowanie do głównego ekranu konsoli administracyjnej. Konsola działająca w trybie samodzielnym serwera wygląda tak jak na ekranie poniżej.



składa się z kilku głównych elementów - z których najważniejsze to Deployments, Configuration oraz Runtime.

Zakładka *Configuration* zawiera wszystkie pojedyncze podsystemy stanowiące część modułów serwera. Najważniejsze z nich to moduł do konfiguracji dostępu do baz danych oraz zarządzania kolejkami komunikatów.

Zakładka Deployments pozwala dodać nowe pakiety typu ear lub war do katalogu deployment.

Zakładka Runtime pomaga w zarządzaniu wdrożeniami aplikacji i sprawdzaniu charakterystyki serwera.

Ćwiczenie 4. Zapoznaj się z zawartością poszczególnych zakładki.

Ćwiczenie 5. Uruchomienie przykładowej aplikacji na serwerze.

W katalogu test znajdziesz przykładową aplikację, którą możesz wykorzystać do przetestowania działania serwera. Jest to plik WildFlyTest_war.

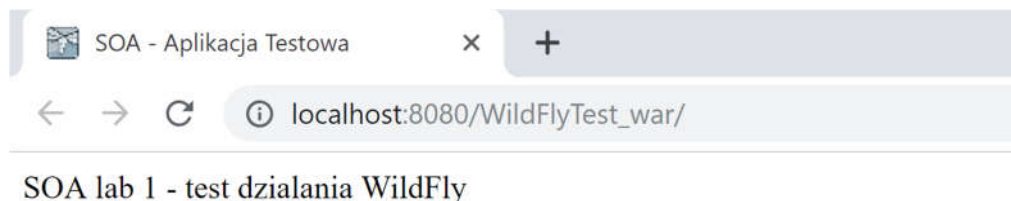
Istnieje kilka sposobów w jaki możesz dokonać deploy aplikacji na serwer.

1. Pierwszy z nich to ręczne umieszczenie pliku war w katalogu standalone\deployments. Wystarczy przekopiować plik do katalogu standalone\deployments.

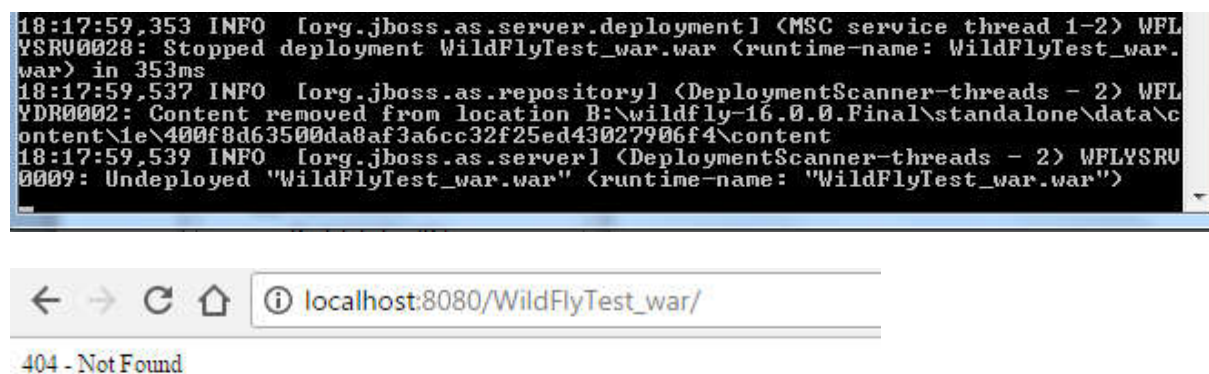
Efekt kopiowania widać na konsoli tekstowej serwera aplikacyjnego

```
YDR0001: Content added at location B:\wildfly-16.0.0.Final\standalone\data\content\1e\400f8d6350dda8af3a6cc32f25ed43027906f4\content
18:01:40.992 INFO [org.jboss.as.server.deployment] (MSC service thread 1-2) WFLY
YDR0027: Starting deployment of "WildFlyTest_war.war" (runtime-name: "WildFlyTest_war.war")
18:01:56.728 INFO [org.infinispan.factories.GlobalComponentRegistry] (MSC service thread 1-3) ISPN000128: Infinispan version: Infinispan 'Infinity Minus ONE' 9.4.8.Final
18:01:59.871 INFO [org.jboss.as.clustering.infinispan] (ServerService Thread Pool -- 6) WFLYCLINF0002: Started client-mappings cache from ejb container
18:02:00.644 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 77) WFLYUT0021: Registered web context: '/WildFlyTest_war' for server 'default-server'
18:02:01.384 INFO [org.jboss.as.server] (DeploymentScanner-threads - 1) WFLYCRU0010: Deployed "WildFlyTest_war.war" (runtime-name: "WildFlyTest_war.war")
```


Wystarczy teraz wpisać w przeglądarce a otrzymujemy dowód działania naszego serwera.



Aby wykonać undeploy wystarczy usunąć pliki z katalogu i wszystko wraca do normy.



Jest to jednak sposób nie zalecany. Stosujemy go tylko w ostateczności.

2. Innym sposobem wdrożenia aplikacji jest użycie narzędzia CLI (Command Line Interface), które można uruchomić za pomocą pliku jboss-cli.bat (lub jboss-cli.sh w przypadku użytkowników systemu Linux). Sposób ten działa tylko na wdrożone aplikacje. Wykorzystując ten sam plik co w punkcie 1 uruchom go z poziomu linii komend.

Jak nietrudno się domyślić, do wdrożenia aplikacji niezbędne będzie użycie polecenia deploy. Polecenie bez argumentów powoduje wyświetlenie listy aktualnie wdrożonych aplikacji.

```
[localhost:9999 /] deploy  
WildFlyTest_war.war
```

Jeśli pierwszym parametrem polecenia będzie archiwum z zasobami (na przykład lokalizacja pliku .war), zostanie ono od razu wdrożone na lokalnym, samodzielnym serwerze.

```
[standalone@localhost:9999 /] deploy ../WildFlyTest_war.war
```

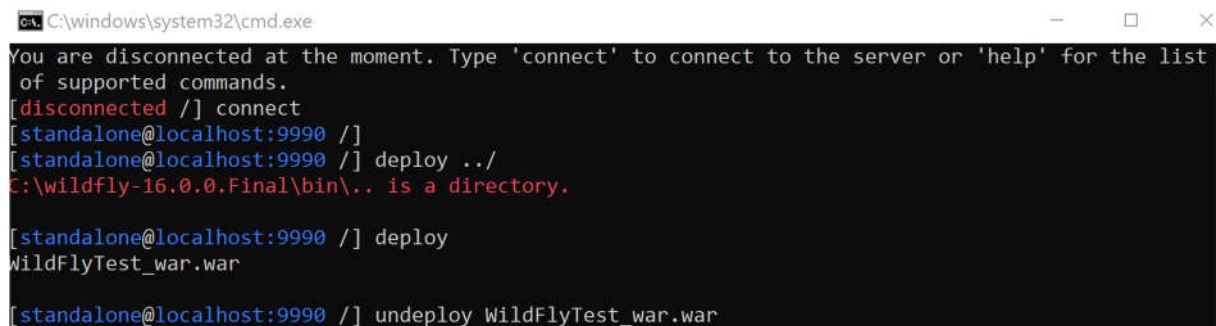
Jak można się domyślić z powyższego zapisu, narzędzie CLI używa standardowo folderu, w którym zostało uruchomione, czyli JBOSS_HOME/bin. Nic jednak nie stoi na przeszkodzie, by do wskazywania

lokalizacji archiwów używać pełnych ścieżek; obsługa przez narzędzie funkcji automatycznego uzupełniania (klawisz Tab) dodatkowo ułatwia całe zadanie.

```
[standalone@localhost:9999 /] deploy c:/deployments/WildFlyTest_war.war  
' WildFlyTest_war.war' deployed successfully.
```

Domyślnie wdrożenie powoduje również aktywację aplikacji, więc jest ona od razu dostępna dla użytkowników. Jeśli chce się jedynie wdrożyć aplikację i aktywować ją później, konieczne jest przekazanie opcji `--disabled`.

```
[standalone@localhost:9999 /] deploy ../ WildFlyTest_war.war --disabled  
' WildFlyTest_war.war' deployed successfully.
```



```
C:\windows\system32\cmd.exe  
You are disconnected at the moment. Type 'connect' to connect to the server or 'help' for the list  
of supported commands.  
[disconnected /] connect  
[standalone@localhost:9990 /]  
[standalone@localhost:9990 /] deploy ../  
C:\wildfly-16.0.0.Final\bin\.. is a directory.  
[standalone@localhost:9990 /] deploy  
WildFlyTest_war.war  
[standalone@localhost:9990 /] undeploy WildFlyTest_war.war
```

Aby uaktywnić już wdrożoną aplikację, ponownie trzeba użyć polecenia wdrożenia bez opcji `--disabled`, a zamiast lokalizacji archiwum podać nazwę wdrożenia.

```
[standalone@localhost:9999 /] deploy --name= WildFlyTest_war.war  
' WildFlyTest_war.war' deployed successfully.
```

Ponowne wdrożenie aplikacji wymaga użycia w poleceniu wdrożenia dodatkowego znacznika.

Używamy opcji `-f`, by wymusić ponowne wdrożenie.

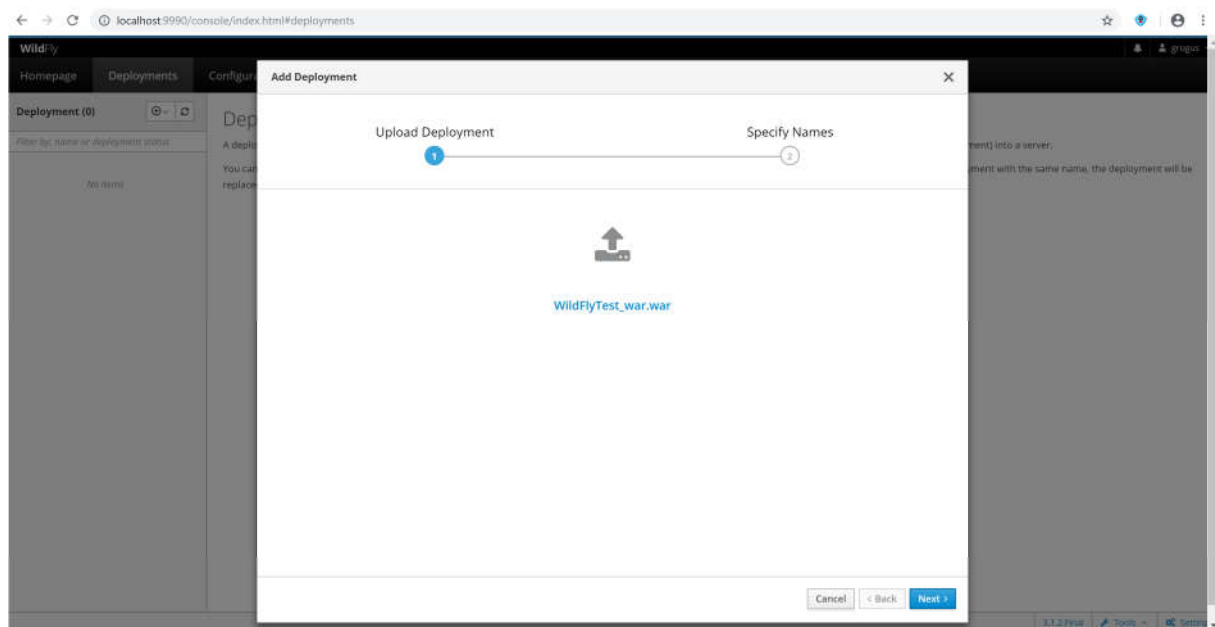
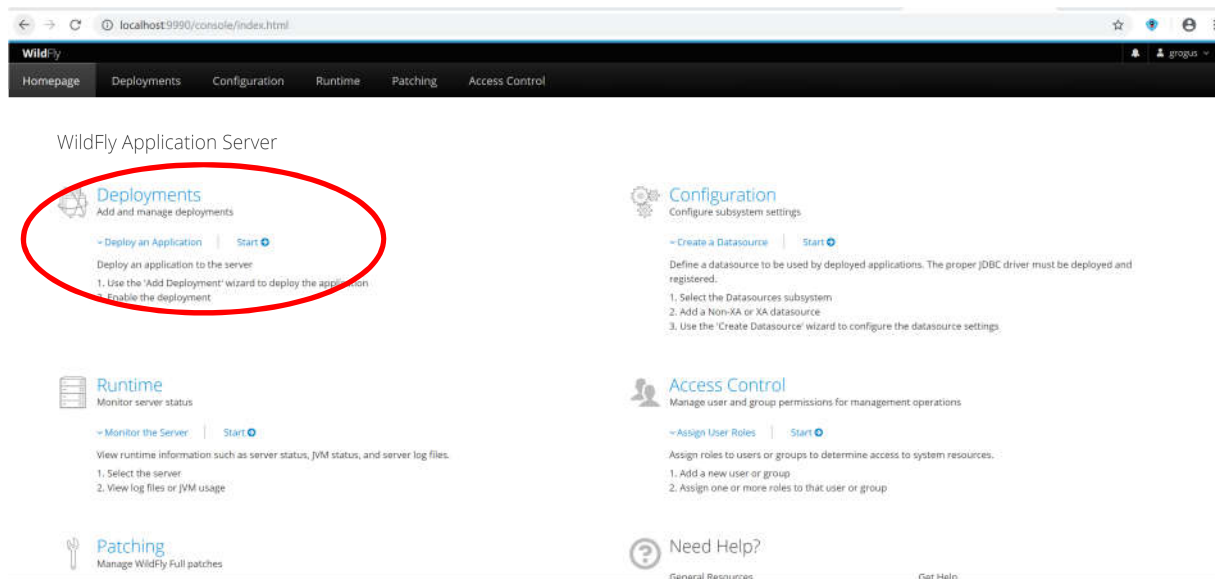
```
[localhost:9999 /] deploy -f ../ WildFlyTest_war.war  
' WildFlyTest_war.war' re-deployed successfully.
```

Usunięcie wdrożenia realizuje polecenie `undeploy`, które jako parametr przyjmuje nazwę aktualnie wdrożonej aplikacji.

```
[localhost:9999 /] undeploy WildFlyTest_war.war  
' WildFlyTest_war.war' undeployed successfully.
```

3. Kolejnym sposobem jest użycie Konsoli Webowej.

Jak pamiętasz konsola posiada specjalną zakładkę do tego celu:



localhost:9990/console/index.html#deployments:path=deployment-dply-wildflytestwarwar

WildFly

Homepage Deployments Configuration Runtime Patching Access Control

Deployment (1)

Filter by: name or deployment status

WildFlyTest_war.war View

WildFlyTest_war.war

The deployment WildFlyTest_war.war is enabled and active. [Disable](#)

Main Attributes

Name:	WildFlyTest_war.war
Runtime Name:	WildFlyTest_war.war
Context Root:	/WildFlyTest_war
Enabled, Managed, Exploded:	
Status:	OK
Last enabled at:	3/3/19, 4:33 PM
Last disabled at:	n/a

```
16:33:38,982 INFO [org.jboss.as.repository] (External Management Request Threads -- 1) WFLYDR0001: Content added at location C:\wildfly-16.0.0.Final\standalone\data\content\1e\400f8d63500da8af3a6cc32f25ed43027906f4\content
16:33:38,989 INFO [org.jboss.as.server.deployment] (MSC service thread 1-4) WFLYSRV0027: Starting deployment of "WildFlyTest_war.war" (runtime-name: "WildFlyTest_war.war")
16:33:39,268 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 81) WFLYUT0021: Registered web context: '/WildFlyTest_war' for server 'default-server'
16:33:39,332 INFO [org.jboss.as.server] (External Management Request Threads -- 1) WFLYSRV0010: Deployed "WildFlyTest_war.war" (runtime-name: "WildFlyTest_war.war")
16:34:37,499 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 81) WFLYUT0022: Unregistered web context: '/WildFlyTest_war' from server 'default-server'
16:34:37,524 INFO [org.jboss.as.server.deployment] (MSC service thread 1-4) WFLYSRV0028: Stopped deployment WildFlyTest_war.war (runtime-name: WildFlyTest_war.war) in 25ms
16:34:37,571 INFO [org.jboss.as.server] (External Management Request Threads -- 1) WFLYSRV0009: Undeployed "WildFlyTest_war.war" (runtime-name: "WildFlyTest_war.war")
16:34:37,575 INFO [org.jboss.as.repository] (External Management Request Threads -- 1) WFLYDR0002: Content removed from location C:\wildfly-16.0.0.Final\standalone\data\content\1e\400f8d63500da8af3a6cc32f25ed43027906f4\content
```

Efekt jest taki sam jak w przykładach powyżej.

4. Zastosowanie Mavena.

Jako że jesteś specjalistą od Mavena spróbuj samodzielnie wykonać deploy przykładowej aplikacji na serwer za pomocą Mavena

5. Ostatni sposób to deploy z poziomu Środowiska IDE: NetBeans, Eclipse czy IntelliJ. Tym sposobem zajmiemy się na następnych zajęciach. Niewątpliwie na naszych zajęciach będzie to podstawowy sposób zarządzania i uruchamiania aplikacji na serwerze.

Ćwiczenie 5. W celu lepszego poznania WildFly sugeruje zapoznanie się z następującym materiałem: http://docs.wildfly.org/16/Admin_Guide.html

Jeśli chodzi o specyfikację JavaEE, która będzie podstawą naszych kolejnych zajęć to podstawowym źródłem informacji będą pozycje:

<https://javaee.github.io/tutorial/toc.html>

lub

<https://docs.oracle.com/javaee/7/JEETT.pdf>

Na następne zajęcia proszę przygotować się w oparciu o

<https://javaee.github.io/tutorial/webapp.html#BNADR>