# CHAPTER 11

# CONFIGURATION OF WEB SUBSYSTEM

| General description | |
|---|---|
| Meta | Configure connectors, servers and other functions of the web subsystem. |
| Goals | • Describe the functions of the Undertow web subsystem.<br>• Configure the components of the web subsystem.<br>• Configure JBoss EAP socket binding groups and network interfaces. |
| sections | • Exploration of the functions of the web subsystem (and questionnaire)<br>• Configuration of the web subsystem (and guided exercise) |
| Laboratory work | • Configuration of the web subsystem |

# Exploring the functions of the web subsystem

## Goals

After completing this section, a system administrator should be able to do the following:

• Describe the functions of the Undertow web subsystem.

## Explore the features

The JBoss Web Server (JWS) was the default web subsystem in JBoss EAP 6. This web subsystem is based on a fork of Apache Tomcat, which acts as an embedded Java EE web container.

In EAP 7, the default web subsystem was replaced by the Undertow project. Similar to JWS, Undertow can act as a Java EE web server and web container. Undertow was developed because some requirements of the Java EE 7 specification were not supported by JWS, such as:

• Web sockets – provide full duplex communication between two peers over TCP, and it is a new technology defined by W3C.

• JSON-P: Supports cross-domain requests, which avoids the restrictions imposed by the browsers.

• HTTP/2 – Improves performance and payload, compared to the HTTP protocol anterior.

Undertow is written in Java and implements several enhancements, namely:

• NIO Web Container: Undertow is faster and supports non-blocking I/O support provided by the XNIO API, a framework based on the new Java I/O (NIO) API.

• Load balancing: The Apache HTTPD server is no longer needed, because requests can now be distributed among other JBoss EAP instances with mod_cluster technology.

• Port reduction: Supports HTTP update, which reduces the amount of ports used by EAP to two. Port 990 is used for management and port 8080 is used for applications.

• Root Web Location – Defines the default web application to be deployed when the server receives a request.

• Session Management – Customize session management for HTTP sessions.

Within the Undertow subsystem, there are five main components to configure:

1. Buffered caches

2. Server

3. Servlet Container

4. Handlers

5. Filters

**Buffer Caches**

Buffer caches are responsible for caching static content.
A buffer cache consists of multiple caches (regions), with each region being divided into smaller buffers. The total amount of space used can be calculated by multiplying the buffer size times the number of buffers per region times the maximum number of regions. The buffer is defined in bytes, and the default size of a buffer cache is 10 MB.

**server**

The servers represent an instance of Undertow. It is possible to run multiple Undertow instances on a single EAP 7 instance. However, it is recommended to start a new EAP instance instead of creating a new Undertow instance, as it is an easy operation to perform and does not use many resources.

A listener handles a request according to the request protocol. Three types of listener are available for each server:

- HTTP: This type of listener is responsible for handling HTTP requests.
- HTTPS: This type of listener is responsible for handling HTTPS requests.
- AJP: This type of listener is responsible for handling AJP requests. The AJP protocol is responsible for the communication between the EAP instance and the load balancer.

By default, the default and full profiles only define the HTTP listener and the ha and full-ha profiles define the HTTP and AJP listeners.

**Servlet Container It is**

possible to configure servlet, JSP, and web-socket technologies using the servlet container. An EAP 7 instance can have multiple servlet containers, although most servers will only need a single servlet container.

**Handlers A**

handler adds functionality to the container. It is created using a Java class and can be used for any purpose, such as security, error page handling, metrics, or virtual host support. The handlers in Undertow are chained together.

Undertow defines synchronous or asynchronous handlers. Asynchronous handlers will provide better response time, as they will provide asynchronous responses, thus freeing up the servlet for other requests more quickly.

**Filters**

On a functional level, a filter is equivalent to a global valve used in earlier versions of JBoss EAP. A filter filters the request, allowing you to modify its content. Some common use cases for filters include setting headers or performing GZIP compression.

The following types of filters can be defined as:

- custom-filter
- error-page

- expression-filter
- gzip
- mod-cluster
- request-limit
- response-header
- rewrite

## References

**For more information, see JBoss EAP**

> **Configuration Guide: Configuring the Web Server (Undertow)** https://access.redhat.com/documentation/en/red-hat-jboss enterprise-application-platform/version-7.0/configuration -guide/#configuring_the_web_server_undertow

# Quiz: Functions of the web subsystem

**Match the following items with their equivalents in the table.**

| | | | | |
|---|---|---|---|---|
| HTTP update | buffer cache | | | |
| Servlet Container | Filter | JSON-P | Listeners | |
| handlers | asynchronous handlers | server | | |
| Sockets web | | | | |

| Component | Description |
|---|---|
| Allows you to modify a request to the server. | |
| Responsible for caching of static content. | |
| The servers represent an instance of Undertow. | |
| Add functionality to a container. | |
| Responsible for activating protocols, such as HTTP/ HTTPS/AJP. | |
| Supports full duplex communication over TCP. | |

| Component | Description |
| --- | --- |
| They improve response times, since they deploy the servlet more quickly. | |
| They allow you to configure web technologies, such as JSPs, servlets, and websockets. | |
| Adds support for cross-domain requirements. | |
| Reduces the number of ports required to access EAP 7. | |

# Solution

**Match the following items with their equivalents in the table.**

| Component | Description |
|---|---|
| Allows you to modify a request to the server. | Filter |
| Responsible for caching of static content. | buffer cache |
| The servers represent an instance of Undertow. | server |
| Add functionality to a container. | handlers |
| Responsible for activating protocols, such as HTTP/ HTTPS/AJP. | Listeners |
| Supports full duplex communication over TCP. | Sockets web |
| They improve response times, since they deploy the servlet more quickly. | asynchronous handlers |
| They allow you to configure web technologies, such as JSPs, servlets, and websockets. | Container of Servlet |
| Adds support for cross-domain requirements. | JSON-P |
| Reduces the number of ports required to access EAP 7. | Update of HTTP |

# Web subsystem configuration

## Goals

**After completing this section, a system administrator should be able to do the following:**

**• Configure the components of the web subsystem.**

## Root Web Application Configuration

**Any application deployed to EAP will be published to a standard address. For example, an application named app.war will be available by default at http://<server>:8080/app. The /app path is defined as a context path, according to Java EE. If the HTTP request does not provide a context path, EAP will return a welcome application implemented in the Undertow subsystem.**

**The Undertow subsystem is responsible for providing the welcome application to JBoss.**

```
<server name="default-server">
  <http-listener name="default" socket-binding="http" redirect-socket="https" /> <host name="default-host"
  alias="localhost">
    <location name="/" handler="welcome-content"  <filter-ref                    (1)  />
    name="server-header" /> <filter-ref name="x-
    powered-by-header" /> </host> </server> ... More
  Undertow

configuration ... <handlers>

  <file name="welcome-content" path="${jboss.home.dir}/welcome-content" /> </handlers>
```

(1) **The location tag defines the default app to display.**

**Notice that the root path (/) uses a handler called welcome-content, which is a file handler that defines the location of the JBoss welcome application.**

**The app can be replaced in two ways:**

**• Changing the welcome-content file handler.**

**• Changing the default-web-module attribute on the default-host.**

**Changing the file handler welcome-content.**
**The first way to replace the default application is to change the file handle named welcome-content so that it has a new path that contains the application.**

**It is possible to replace the welcome-content file handler using the tool CLI:**

```
/subsystem=undertow/configuration=handler/file=welcome-content:\
write-attribute(name=path,value="/path/to/application" )                    (1)
```

1. Defines the directory where the Java EE application is stored. The application must be available on the file system on all hosts that belong to the domain, so this method is not recommended.

use

A reload is required for the changes to apply.

```
/:reload-servers
```

To achieve the same goal using the managed domain, simply add the level / profile=full-ha to the beginning of the CLI command.

Changing the default-web-module attribute

The second way is by using an application that is deployed on the standalone server or managed domain.

```
/subsystem=undertow/server=default-server /host=default-host :\
write-attribute(name=default-web-module ,value=myapp.war )
```

1. EAP 7 provides a default server named default-server. Servers are discussed later in this chapter.
2. EAP 7 also provides a default host called default-host. Hosts are discussed later in this chapter.
3. The default-web-module attribute is responsible for defining an application that will appear if a context path is not specified.
4. Define which deployed application should be displayed if a path is not specified. context.

Disable the default web application In some cases, it is required to remove the Welcome to JBoss application, but without having another application as the default root web application. This can be achieved by removing the location (/) entry for default-host:

```
/subsystem=undertow/server=default-server/host=\ default-host/
location=\/:remove
```

After the server is reloaded, any request using the root context will be directed to a 404 error page.

# Demo: Configuring the root web location

1. Open a terminal window from the workstation virtual machine (Applications > Favorites > Terminal) and run the following command:

```
[student@workstation ~]$ demo root-web setup
```

The previous command:

- **It will verify that EAP was installed.**
- **Verify that the Configure Controllers guided exercise has been run JDBC.**
- **It will verify that EAP is not running.**
- **You will download the files required for this demonstration.**

**2. It will start the EAP instance using the base directory located at /home/student/ JB248/labs/standalone/:**

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./standalone.sh \
-Djboss.server.base.dir=/home/student/JB248/labs/standalone/
```

**3. It will open a new terminal window and run the following commands to start the connecting the CLI tool to your EAP instance:**

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./jboss-cli.sh -c
```

**use**

**The credentials are not required because the security subsystem has not been configured to request the credentials for local connections.**

4. Deploy the ***welcome.war*** **application available in the /home/student/JB248/ folder labs/root-web:**

```
[standalone@localhost:9990 /] deploy \
/home/student/JB248/labs/root-web/welcome.war
```

**5. Open the web browser on the workstation and navigate to http://localhost:8080/welcome to test if the application was successfully deployed. You should see the welcome app:**

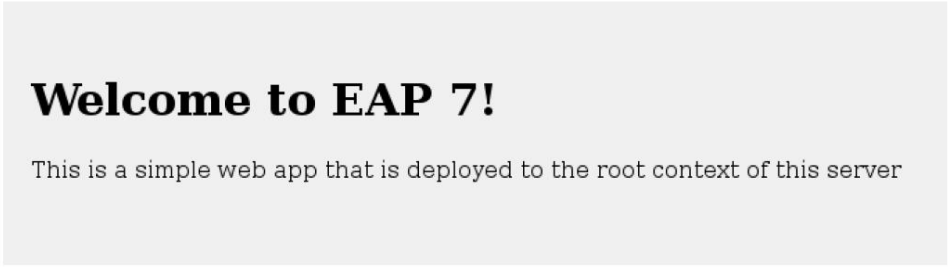


***Figure 11.1: Personalized welcome page***

6. Notice that the welcome context path is required to be sent after the port to access the application. If the context path is not specified, the root application will be deployed. By default, the root application is the Welcome to JBoss application. Access the JBoss welcome page by navigating to the URL http://localhost:8080:

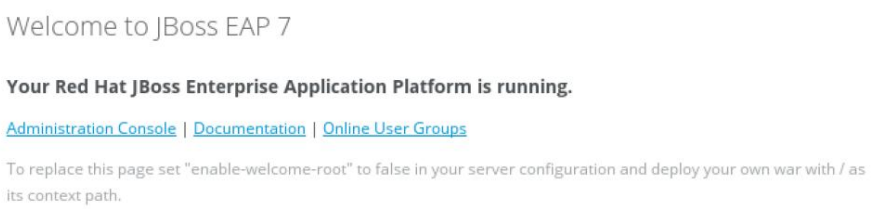


*Figure 11.2: Default welcome page provided by EAP*

7. It is possible to remove the default root app. To do this, go back to the window terminal in which the CLI tool is running and run the following command:

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/
host=default-host/location=\/:remove
```

The above command will remove the / location of default-host. The location / is the root application.

8. A server reload is required to determine if the root application has been removed. Update the EAP instance configuration with the reload command:

```
[standalone@localhost:9990 /] reload
```

9. Open the web browser and navigate to the root application using the URL http:// localhost:8080. You should see a 404 - Not Found error message.

10. It is possible to define the application welcome.war as root context. in the subsystem Undertow you must have a different application running as the root web application, also known as default-web-module for default-server. In this step, you will update it using the CLI.

```
[standalone@localhost:9990 /] /subsystem=undertow/server=\ default-server/
host=default-host:\ write-attribute(name=default-
web-module,value=welcome.war)
```

11. To update the settings with the new root app, a reload is required.

```
[standalone@localhost:9990 /] reload
```

12. Open a web browser and navigate to the root application using the URL http:// localhost:8080. You should now see the welcome application running without the context path.

**13. Exit the CLI:**

```
[standalone@localhost:9990 /] exit
```

**14. Stop the EAP instance by pressing Ctrl+C in the terminal window that is running EAP.**

**This concludes the demo.**

# Configure servers with listeners

**The Undertow subsystem can start multiple web servers or web containers on a single EAP instance.**

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    ...
    <server name="default-server">
        <http-listener name="default" redirect-socket="https" socket-binding="http" /> <host name="default-host"
        alias="localhost">
            <location name="/" handler="welcome-content" /> <filter-ref
            name="server-header" /> <filter-ref name="x-
            powered-by-header" /> </host> </server>


    ...
</subsystem>
```

**Each server tag starts an instance of Undertow. However, it is recommended to start a new EAP instance instead of creating a new Undertow instance, as it is easy to do and does not consume many resources.**

**A listener handles a request according to the request protocol. Three types of listener are available for each server:**

**• HTTP: This type of listener is responsible for handling HTTP requests.**

**• HTTPS: This type of listener is responsible for handling HTTPS requests.**

**• AJP: This type of listener is responsible for handling AJP requests. The AJP protocol is responsible for the communication between the EAP instance and the load balancer.**

**By default, the default and full profiles only define the HTTP listener and the ha and full-ha profiles define the HTTP and AJP listeners.**

**Each server can define more than one listener for each listener type:**

```
/subsystem=undertow/server=default-server/http-listener=\ new-http:add(socket-
binding=http-n)
```

**Notice the new XML configuration:**

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    ...
    <server name="default-server">
        <http-listener name="default" redirect-socket="https" socket-binding="http" />
```

```
        <http-listener name="new-http" socket-binding="http-n" /> <host name="default-
        host" alias="localhost">
            <location name="/" handler="welcome-content" /> <filter-ref
            name="server-header" />
            <filter-ref name="x-powered-by-header" /> </host> </
        server>

    ...
</subsystem>
```

**A different socket-binding must be defined to avoid a port conflict. In the above configuration, the web application will be available for HTTP requests on two different sockets: http or http-n.**

**The host configuration within the servers element is responsible for creating virtual hosts. Virtual host pool web applications based on DNS names. Each server can configure a set of hosts:**

```
/subsystem=undertow/server=default-server/host=\ version-
host:add(alias=[version.myapp.com])
```

**A default web application can be published to a host that listens on multiple addresses:**

```
/subsystem=undertow/server=default-server/host=\ version-
host:write-atribute(name=default-web-module, value=version.war)
```

**In the example above, by default, requests to http:// version.myapp.com will return the application version.war. The following XML will reflect the above modification:**

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    ...
   <server name="default-server">
       <http-listener name="default" redirect-socket="https" socket-binding="http" /> <http-listener name="new-
       http" socket-binding="http-n" /> <host name="default-host" alias="localhost">

           <location name="/" handler="welcome-content" />
           <filter-ref name="server-header" />
           <filter-ref name="x-powered-by-header" /> </host> <host

       name="version-host" default-web-module="version.war"
  alias="version.myapp.com"/>
   </server>
    ...
</subsystem>
```

## use

**A jboss-web.xml file should be created in the WEB-INF folder in the version.war application with the following code:**

```
<jboss-web>
   <context-root>/</context-root> <virtual-
   host>version-host</virtual-host> <server-instance>default-
   server</server-instance>
</jboss-web>
```

**If the new host is no longer required, remove it with:**

```
/subsystem=undertow/server=default-server/host=\ version-
host:remove()
```

# Configure SSL connections

**HTTPS is a protocol that encrypts requests and responses over the network. This protocol protects sensitive information that is exchanged between the web browser and the server.**

**To configure this protocol, a certificate file is required to encrypt and decrypt the information exchanged. A keystore must be created to store a certificate securely. The Java Development Kit (JDK) provides the keytool command that can create a keystore, as well as a self-signed certificate if a trusted CA certificate is not provided.**

**To create a new keystore within a self-signed certificate, use the following command:**

```
# keytool -genkeypair -alias appserver -storetype jks -keyalg RSA -keysize 2048 - keystore /path/identity.jks
```

**The following parameters were specified: •**
**genkeypair: responsible for defining that a new self-signed certificate should be created.**

**• alias: defines the name of the certificate within the keystore.**

**• storetype: defines the type of key store.**

**• keyalg – Specifies the algorithm to use to generate the certificate.**

**• keysize – Specifies the size of each key to be generated.**

**• keystore: Defines where the keystore should be stored.**

**The above command requires two passwords. The first is the password to open the identity keystore and the second is to access the appserver certificate. Use the same value for both passwords to enable the HTTPS protocol in EAP 7.**

**Each host in the managed domain must have the keystore in its file system.**

**After creating the keystore, a new security domain must be created for each host:**

```
/host=servera/core-service=management/security-realm=HTTPSRealm:add()
```

**After adding the security domain, it must be configured to load the keystore created:**

```
/host=servera/core-service=management/security-realm=HTTPSRealm/server-identity=\ ssl:add(keystore-path=/
path/identity.jks,keystore-password=changeit,alias=appserver)
```

**A reload is required for the changes to apply.**

```
/:reload-servers
```

**To activate the HTTPS protocol, a new HTTPS listener must be created on the default server. The listener must use the created security domain:**

```
/profile=full/subsystem=undertow/server=default-server/https-listener=https:\ add(socket-
binding=https,security-realm=HTTPSRealm)
```

### Important

**Remember to configure the firewall to open the HTTPS port. The default port for HTTPS is 8443.**

# Other components

**One of the components of Undertow is the buffer cache.**

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    <buffer-cache name="default" buffer-size="1024" buffers-per-region="1024" max  regions="10"

    ...
</subsystem>
```

**It is possible to update the buffer size using the CLI:**

```
/subsystem=undertow/buffer-cache=default:\ write-
attribute(name=buffer-size,value=2048)
```

**If a cache is not required, it can be removed:**

```
/subsystem=undertow/buffer-cache=default:remove
```

**Servlet Container**

**A servlet container instance is defined by the servlet-container tag:**

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    ...
```

```
    <servlet-container name="default"> <jsp-config/
        > <websockets/>
        </servlet-
    container>
    ...
</subsystem>
```

**To check the available attributes that can be set for a servlet container, use the CLI read-resource operation:**

```
/subsystem=undertow/servlet-container=default:read-resource
```

**The following result is expected:**

```
{
        "outcome" => "success",
        "result" => { "allow-
                non-standard-wrappers" => false, "default-buffer-
                cache" => "default", "default-encoding" =>
                undefined, "default-session-timeout" => 30,
                "directory-listing" => undefined, "disable-
                caching-for-secured-pages" => true, "eager-
                filter-initialization" => false, "ignore-flush" => false, "max-
                sessions" => undefined, "proactive-authentication"
                => true, "session-id-length" =>
                30, "stack-trace-on-error" => "local-
                only", "use-listener-encoding" => false, "mime-
                mapping" => undefined, "setting"
                => { "jsp" => undefined, "websockets" => undefined
                },
                "welcome-file" => undefined
        }
}
```

**Note that it is possible to define the default buffer cache with the default buffer-cache attribute.**

**The session can be managed in a servlet container. Notice that there are three properties available for the configuration:**

- **default-session-timeout: the default session timeout (in minutes) for all applications deployed in the container.**
- **max-sessions: The maximum number of sessions that can be active at the same time.**
- **session-id-length: The number of characters of the generated session ID. Longer login IDs are more secure.**

**To set the default session to one hour, use the following operation:**

```
/subsystem=undertow/servlet-container=default:write-attribute(name=default-session  timeout,value=60)
```

**It is also possible to configure JSP technology for the servlet container. To list the available attributes for the JSP configuration, use the following CLI command:**

```
/subsystem=undertow/servlet-container=default/setting=jsp:read-resource
```

**handlers**

**EAP 7 offers two types of handlers:**

- **File manager**
- **Reverse proxy handler**

**File handlers offer static files. Each file handle must be attached to a location on a virtual host.**

**To create a new file handler, use the following CLI command:**

```
/subsystem=undertow/configuration=handler/file=photos\ :add(path=/var/
 photos, directory-listing=true)
```

**Append the new file handle to a location on a virtual host.**

```
/subsystem=undertow/server=default-server/host=default-host/location=photos\ :add(handler=photos)
```

**This content will be served using ip:port/context/photos.**

**The reverse proxy handler allows EAP to offer a high performance reverse proxy.**
**This topic is discussed later in this course.**

**Filters**

**By default, JBoss EAP defines two filters:**

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    ...
   <filters>
      <response-header name="server-header" header-value="JBoss-EAP/7" header  name="Server"/
> <response-header
      name="x-powered-by-header" header-value="undertow/1" header  name="X-Powered-By"/> </filters> </
subsystem>
```

**These filters will set headers on the response. The first will define a header named Server with the value JBoss-EAP/7. The second will define a header named X-Powered-By with the value undertow/1.**

**These filters apply to all requests on the host default-host:**

```
<subsystem xmlns="urn:jboss:domain:undertow:3.0">
    ...
   <server name="default-server">
       ...
      <host name="default-host" alias="localhost">
          <location name="/" handler="welcome-content" />
```

```
            <filter-ref name="server-header" /> <filter-ref
            name="x-powered-by-header" /> </host> </server>


    ...
</subsystem>
```

**To include a new filter, use the following CLI command:**

```
/subsystem=undertow/configuration=filter/gzip=gzip:add()
```

## I/O subsystem

**The I/O subsystem is responsible for configuring the XNIO workers. An XNIO worker manages several types of threads.**

**The Undertow subsystem depends on the I/O subsystem. An Undertow listener defines a worker:**

```
/subsystem=undertow/server=default-server/http-listener=\ default:read-
attribute(name=worker)
```

**The following result is expected:**

```
{
        "outcome" => "success",
        "result" => "default"
}
```

**The worker named default is defined in the I/O subsystem. Use the following command to list the worker attributes:**

```
/subsystem=io/worker=default:read-resource
```

**The following result is expected:**

```
{
        "outcome" => "success",
        "result" => { "io-
                threads" => undefined, "stack-size"
                => 0L, "task-keepalive"
                => 60, "task-max-threads" =>
                undefined
        }
}
```

**The XNIO worker instance is an abstraction layer of the Java NIO APIs. Note that you can define the number of I/O threads to use by setting the io-threads attribute. You can define the maximum number of threads for a task by setting the task-max threads attribute.**