

# **WildFly - The artist formerly known as JBoss AS**

---

Harald Pehl · Red Hat · <http://hpehl.info>

# Harald Pehl

- Senior Software Engineer, Red Hat Inc.
- Management API & Admin Console
- Scala, Akka & Functional Programming

# The Great Java Application Server Debate with Tomcat, JBoss, GlassFish, Jetty and Liberty Profile

May 21, 2013    Simon Maple    27 comments



## Part V – ...And The Best Application Server Award Goes To...

In case you were wondering, we did finally decide that one application server among those tested proved to win over the others...JBoss wins the award!



“*If we had to pick a **winner**, it would be **JBoss**. The only application server in the group whose score **never dropped below a 4***

– [zeroturnaround.com](http://zeroturnaround.com)

“ *JBoss consistently performs very well in each category which is why it also shines in the developer profiles exercise*

— zeroturnaround.com

## Most Often Used Java EE containers (700 respondents)

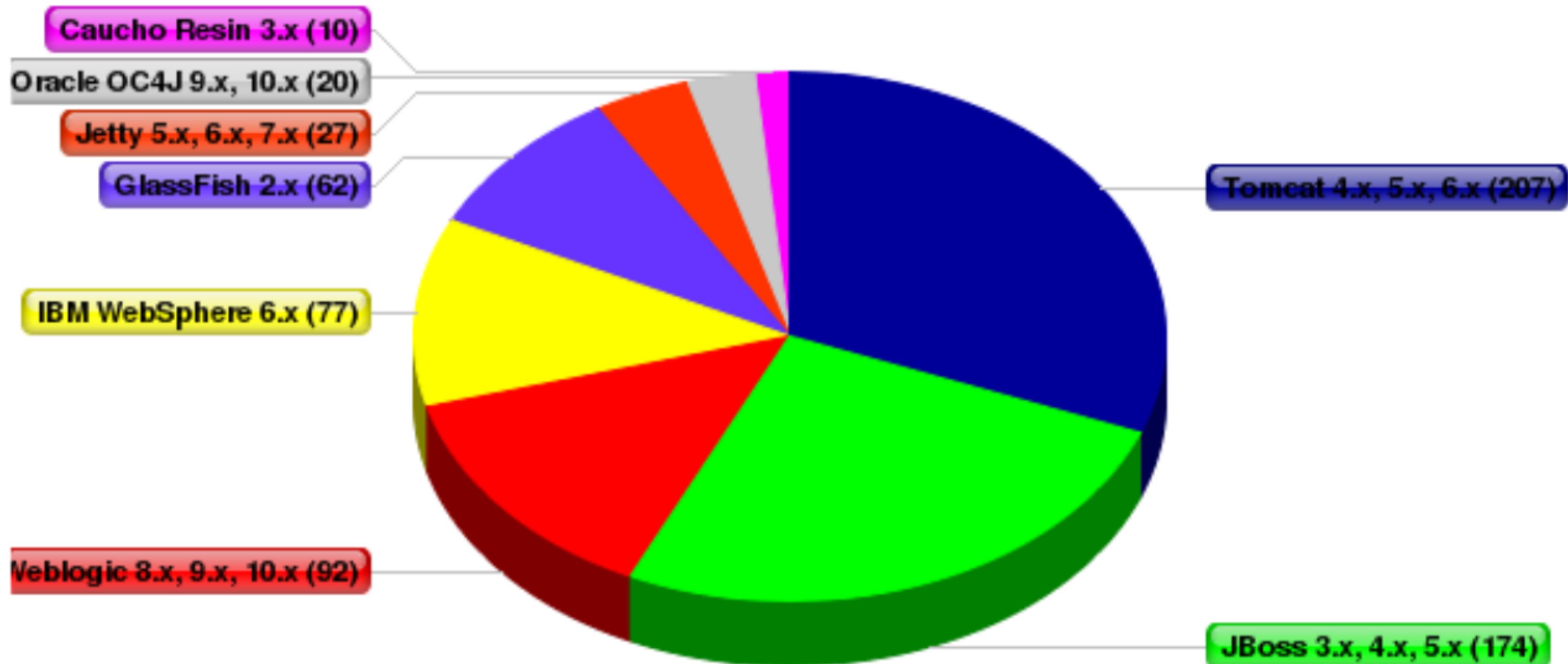


Figure 1. zeroturnaround 2009

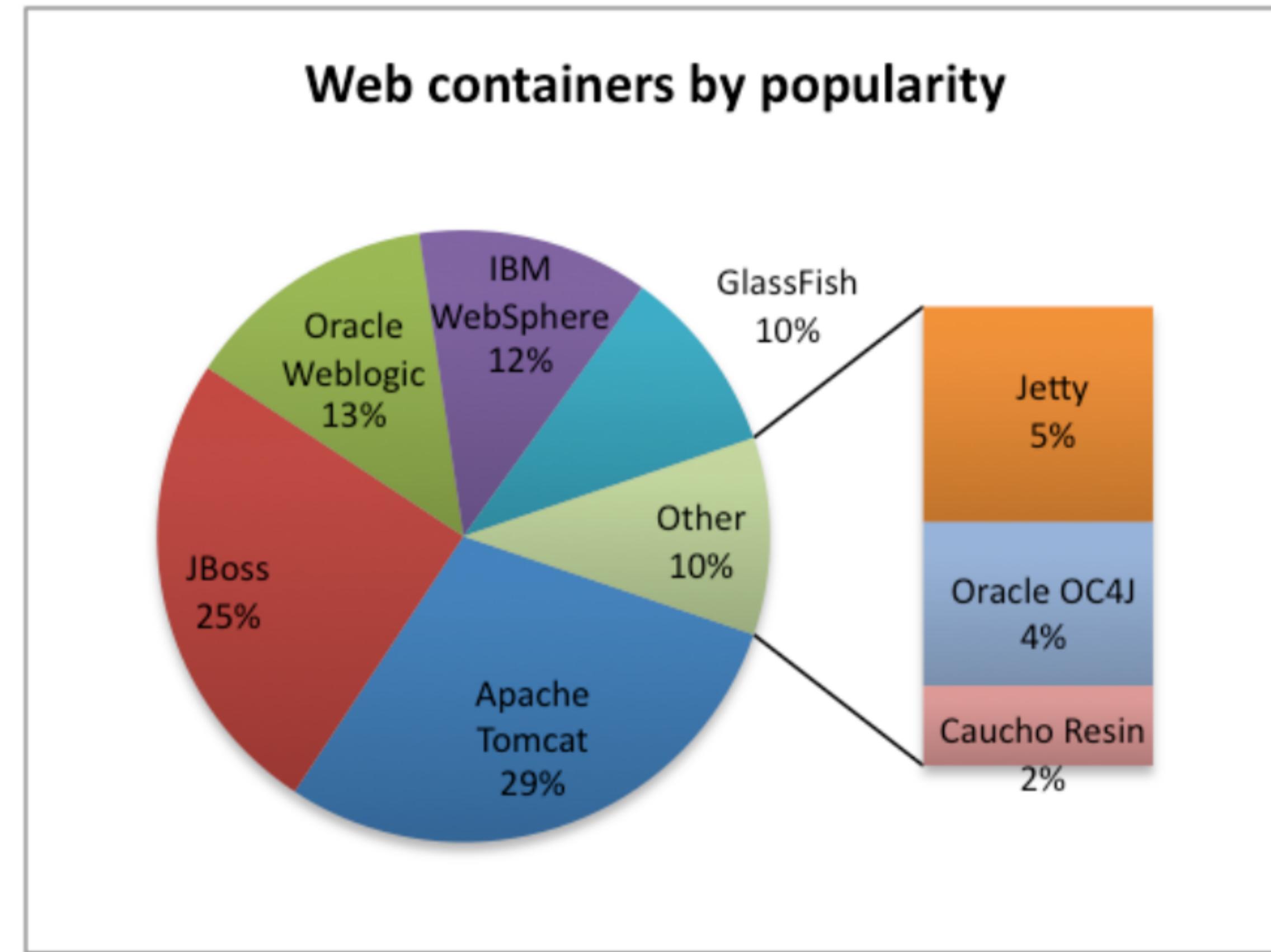


Figure 2. zeroturnaround 2010

# Container Popularity

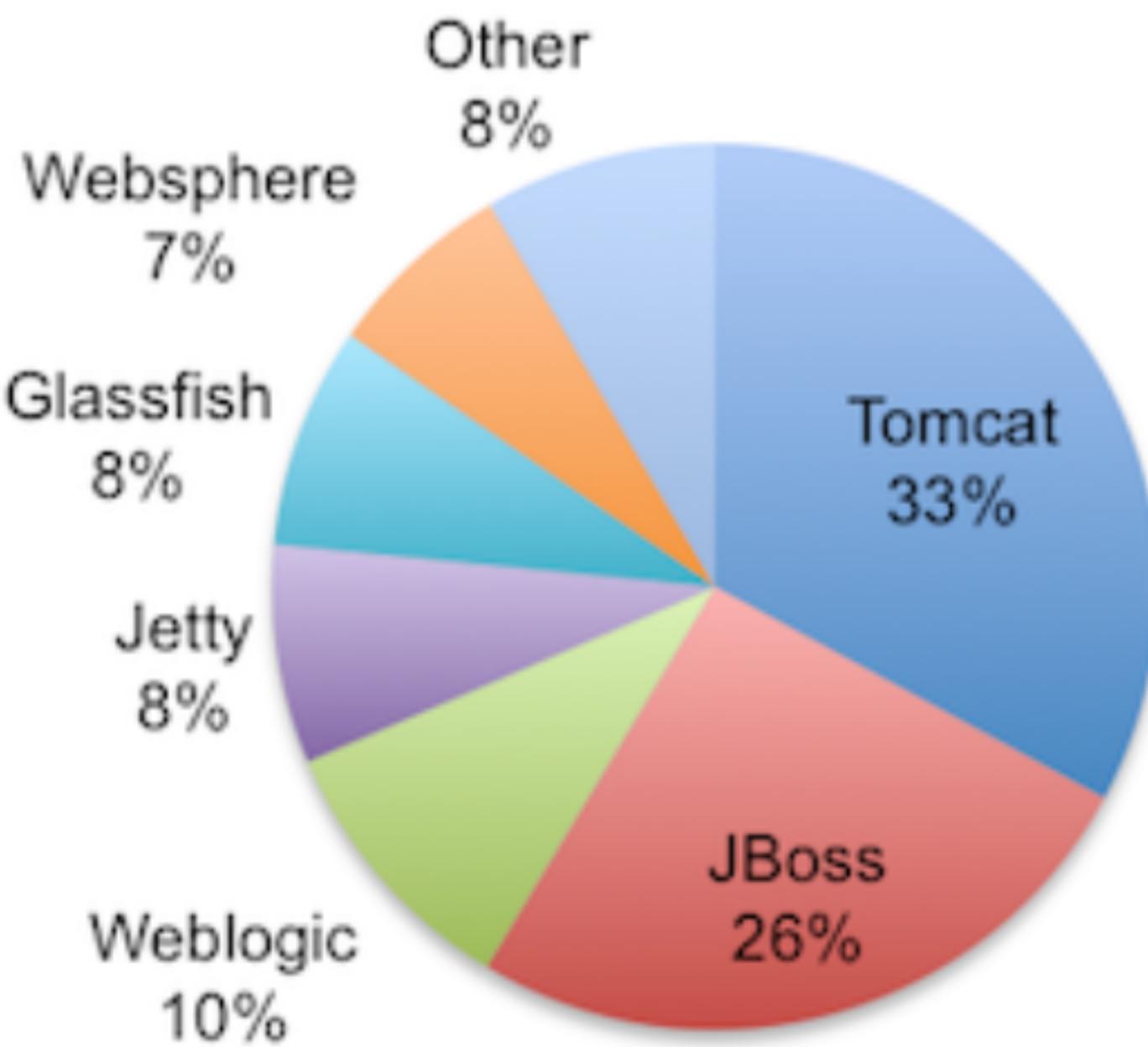


Figure 3. zeroturnaround 2011

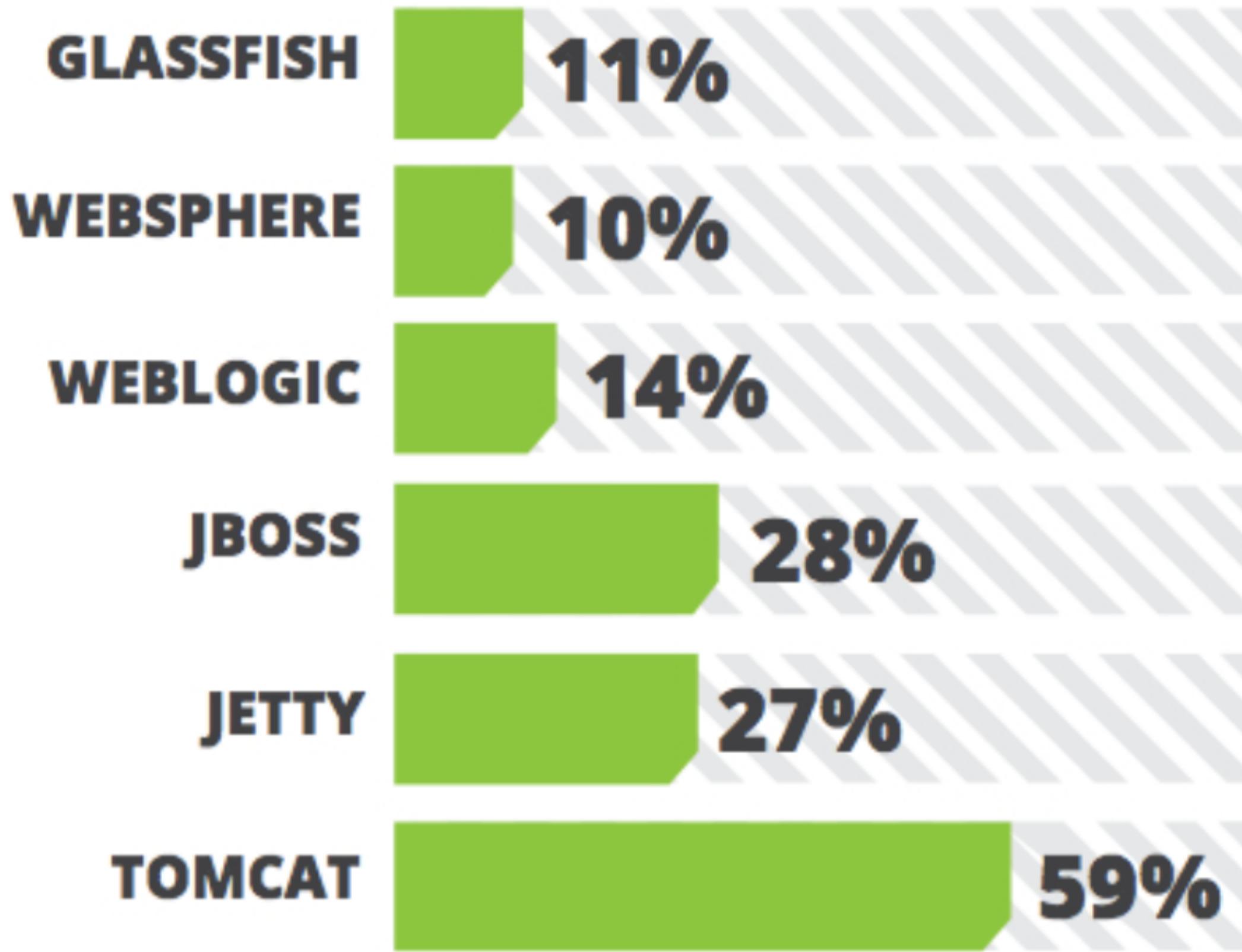


Figure 4. zeroturnaround 2012

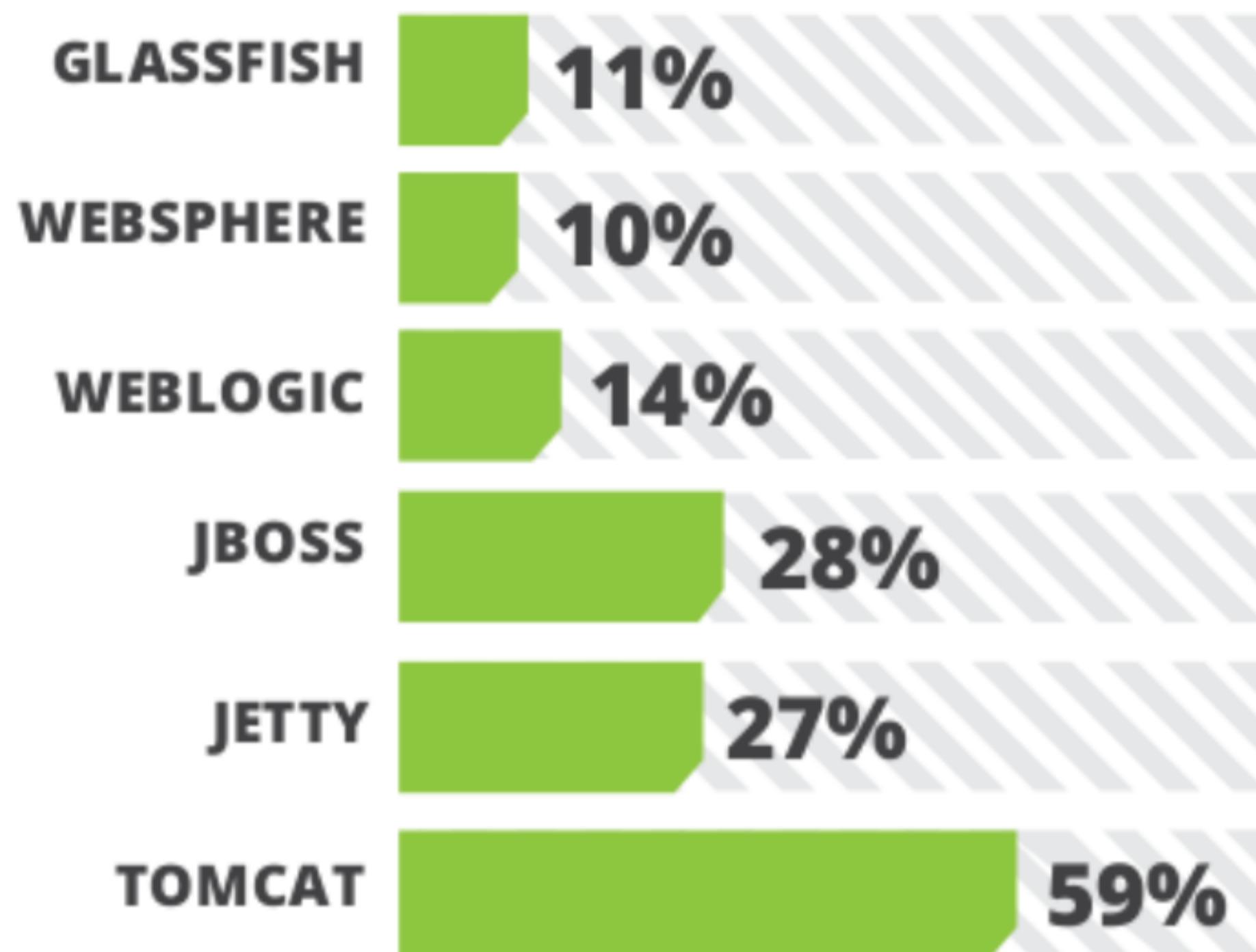


Figure 5. zeroturnaround 2013

## App Server most often used\*

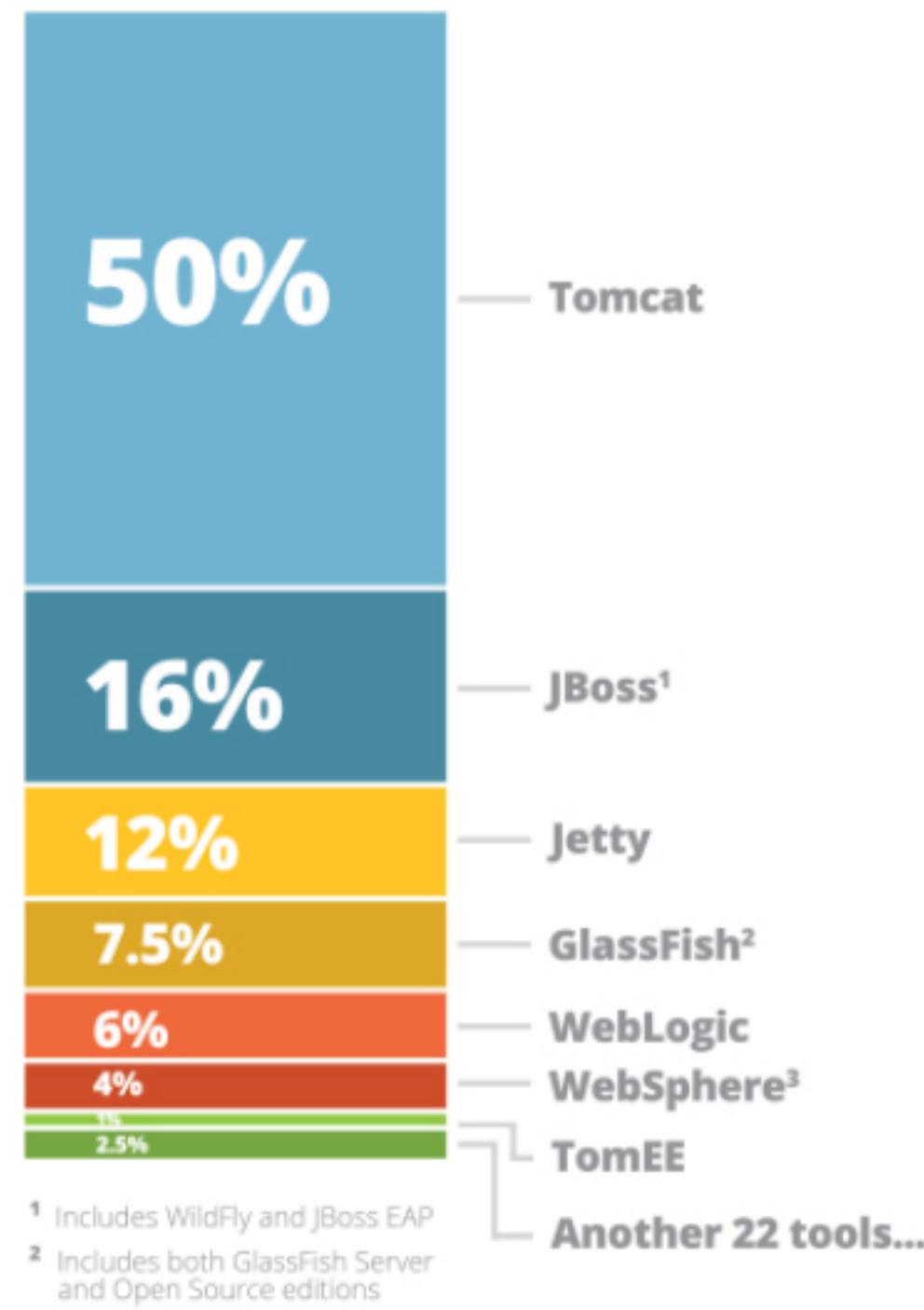


Figure 6. zeroturnaround 2014

# WildFly



The new and improved JBoss Application Server!

## **Objective:**

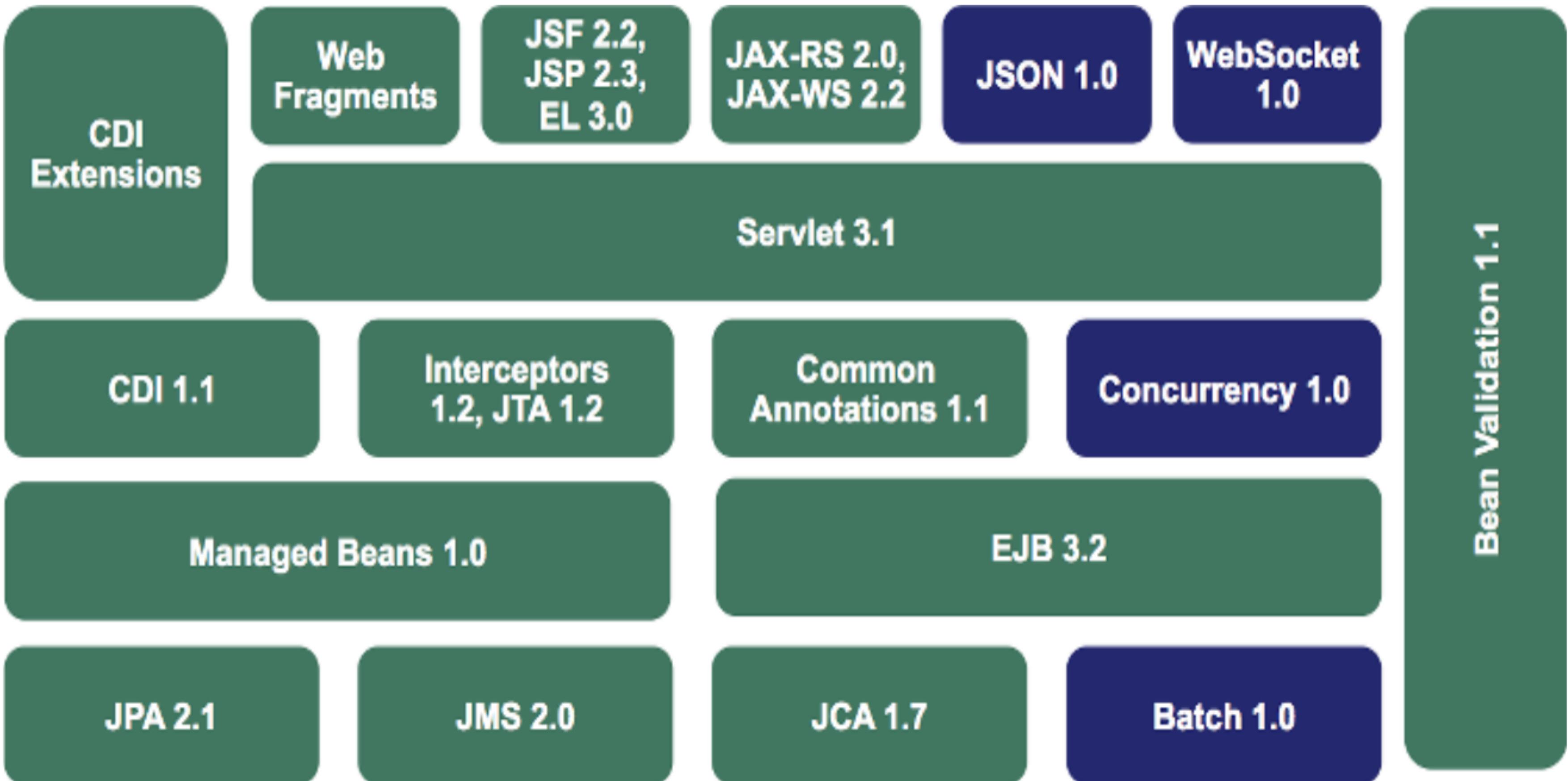
**Understand the new features of WildFly 8 and revise some of the features carry forward from AS 7.x.**

# What is WildFly 8?

- Previously called “JBoss Application Server”
- Upstream for Red Hat JBoss Enterprise Application Platform (JBoss EAP)
- Fast, lightweight, manageable
- Developer friendly
- Supports Java EE standards and beyond
- Open source

# **WildFly 8 main features**

- Java EE7 support
- High performance web server **Undertow**
- Reduced port usage
- Role based administration control & auditing
- Patching



*Figure 7. WildFly: Java EE 7*

# WildFly: Java EE 7 WebSocket (JSR 356)

## ChatServer.java

```
@ServerEndpoint("/chat") ①
public class ChatEndpoint {
    @OnMessage ②
    public void message(String message,
                        Session session) ③
        throws IOException, EncodeException {
        for (Session peer : session.getOpenSessions()) {
            peer.getBasicRemote().sendText(message);
        }
    }
}
```

- ① Creates a WebSocket endpoint, defines the listening URL
- ② Marks the method that receives incoming WebSocket message
- ③ Payload of the WebSocket message

# WildFly: Java EE 7 Batch (JSR 352)

## job.xml

```
<job id="myJob" xmlns="http://xmlns.jcp.org/xml/ns/javaee" version="1.0">
  <step id="myStep" >
    <chunk item-count="3"> ①
      <reader ref="myItemReader"/> ②
      <processor ref="myItemProcessor"/> ③
      <writer ref="myItemWriter"/> ④
    </chunk>
  </step>
</job>
```

- ① Item-oriented processing, number of items in chunk
- ② Item reader for chunk processing
- ③ Item processor for chunk processing
- ④ Item writer for chunk processing

# WildFly: Java EE 7 JSON (JSR 353)

## CreateJson.java

```
JsonObject jsonObject = Json.createObjectBuilder() ①
    .add("apple", "red") ②
    .add("banana", "yellow")
    .build(); ③
StringWriter w = new StringWriter();
JsonWriter writer = Json.createWriter(w); ④
writer.write(jsonObject);
```

- ① Creates a JSON object builder
- ② Adds a name/value pair to the JSON object
- ③ Returns the JSON object associated with this builder
- ④ Writes the JSON object to the writer

# WildFly: Java EE 7 JAX-RS (JSR 339)

## RunClient.java

```
Client client = ClientBuilder.newClient(); ①  
WebTarget target = client.target("..."); ②  
target.register(Person.class);  
Person p = target  
    .path("{id}") ③  
    .resolveTemplate("id", "1")  
    .request(MediaType.APPLICATION_XML) ④  
    .get(Person.class); ⑤
```

- ① `ClientBuilder` is the entry point
- ② Build a new web resource target, specifies the path
- ③ Sub resource URI
- ④ Define the accepted response media types
- ⑤ Call HTTPGET, specify the type of resource

# WildFly: Java EE 7 JMS (JSR 343)

## SendMessage.java

```
@JMSDestinationDefinition(name="myQueue", interfaceName="javax.jms.Queue") ①
@Resource(mappedName="myQueue")
Queue syncQueue;

@Inject
// @ConnectionFactory("java:comp/DefaultConnectionFactory") ②
private JMSContext context; ③

context.createProducer().send(syncQueue, "..."); ④
```

① Create destination resource during deployment

② Default JMS connection factory

③ Main interface of the simplified API

④ Fluent builder API, runtime exceptions

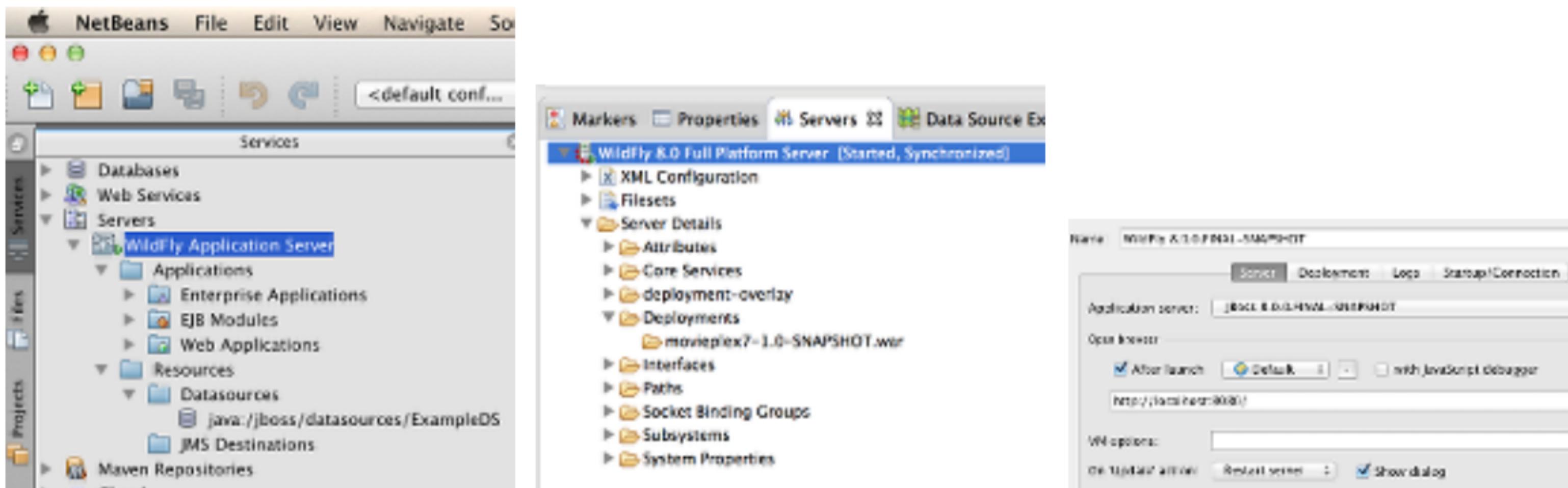
# Java Transaction API 1.2 (JSR 907)

- **@Transactional** Transaction boundaries on CDI managed beans
- **@TransactionScoped** CDI scope for bean instances scoped to the active JTA transaction

# CDI 1.1 (JSR 346)

- Automatic enablement for beans with scope annotation and EJBs
- “beans.xml” is optional
- Bean discovery mode
  - **all**: All types
  - **annotated**: Types with bean defining annotation
  - **none**: Disable CDI
- **@Vetoed** for programmatic disablement of classes

# WildFly: Java EE 7 IDEs



# WildFly: New web server (**Undertow**)

- Flexible and high-performance
- Blocking / non-blocking based on NIO
- Composition/handler based architecture
- Lightweight & fully embeddable
- Supports Servlet 3.1 & HTTP upgrade
- **mod\_cluster** supported

# WildFly new web server: Undertow

## NonBlockingHandler.java

```
Undertow.builder() ①
    .addListener(8080, "localhost")
    .setHandler(new HttpHandler() { ②
        @Override
        public void handleRequest(final HttpServerExchange exchange)
            throws Exception {
            exchange.getResponseHeaders()
                .put(Headers.CONTENT_TYPE, "text/plain");
            exchange.getResponseSender()
                .send("Hello World");
        }
    }).build().start(); ③
```

- ① Same API used for WildFly integration, fluent builder API
- ② Can create multiple handlers
- ③ Start the handler in JVM

# Undertow benchmarks

```
techempower@lg01:~$ wrk -d 30 -c 256 -t 40 http://10.0.3.2:8080/byte
Running 30s test @ http://10.0.3.2:8080/byte
  40 threads and 256 connections
Thread Stats      Avg      Stdev     Max   +/- Stdev
  Latency    247.05us    3.52ms  624.37ms  99.90%
  Req/Sec    27.89k     6.24k   50.22k  71.15%
  31173283 requests in 29.99s   3.83GB read
Socket errors: connect 0, read 0, write 0, timeout 9
Requests/sec: 1039305.27
Transfer/sec:    130.83MB
```

This is output from [Wrk](#) testing a single server running [Undertow](#) using conditions similar to Google's test (1-byte response body, no HTTP pipelining, no special request headers) **1.039 million requests per second.**

# WildFly: Port reduction

- Uses HTTP Upgrade
- Number of ports in default installation is two
  - 8080 for applications
  - 9990 for management
- Only overhead is the initial HTTP Upgrade request/response

# WildFly: HTTP Upgrade

## Client Request

```
GET / HTTP/1.1
Host: example.com
Upgrade: jboss-remoting
Connection: Upgrade
Sec-JbossRemoting-Key: dGh1IHNhbXBsZSSub25jZQ==
```

## Server Response

```
HTTP/1.1 101 Switching Protocols
Upgrade: jboss-remoting
Connection: Upgrade
Sec-JbossRemoting-Accept: s3pPLMBiTxaQ9kYGzhZBk+x0o=
```

# WildFly: Role based access control

- Pre-defined administrative and privileged Roles
  - Monitor, Operator, Maintainer, Deployer, Administrator, Auditor, Super User
- Roles is a set of Permissions
- Permissions specify which Actions (lookup, read, write) are allowed on resources
- Users or Groups are defined in Roles

# WildFly: Administrative audit logging

- Logging of connection/authentication events
- Logging of management operations
- Log message as JSON records
- Audit logging handlers
  - Local file
  - Syslog (UDP / TCP / TLS)

# WildFly: Patching

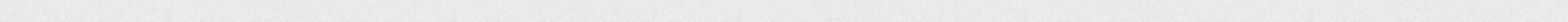
- Allows libraries and configuration updates in an installation
- Patches are zip bundles with updates and metadata
- Multiple one-off patches can be applied; invalidated by the next point/CP release
- Rollbacks are possible

# WildFly: Miscellaneous

- Improved JDK8 compatibility
- RESTEasy 3
- Hibernate search
- Per-deployment security permissions
- New public clustering API
- Pruned: CMP, JAX-RPC, JSR 88

# **Carry forward from AS 7.x**

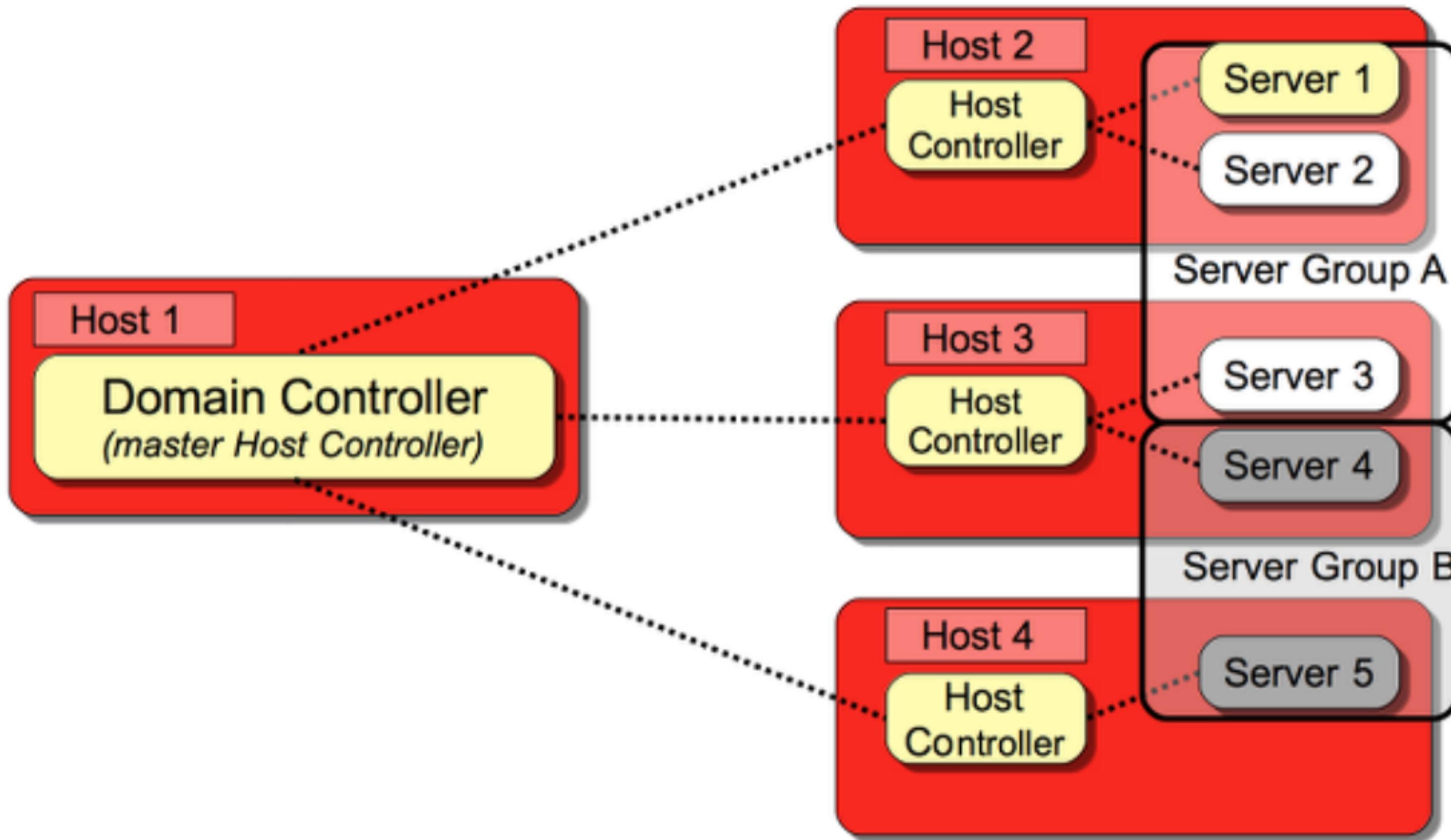
- **Standalone and Managed Domain**
- **Centralized Administration**
  - **Command Line Interface (jboss-cli)**
  - **Admin Console**
  - **Configuration files**



# Standalone and Managed Domain

- **Standalone:** single independent instance
- **Managed domain:** manage multiple WildFly instances from a single control point
  - HA from fail-over: removes single point of failure
  - HA from load-balancing: timely response to client, even with high-volume requests

# Managed Domain



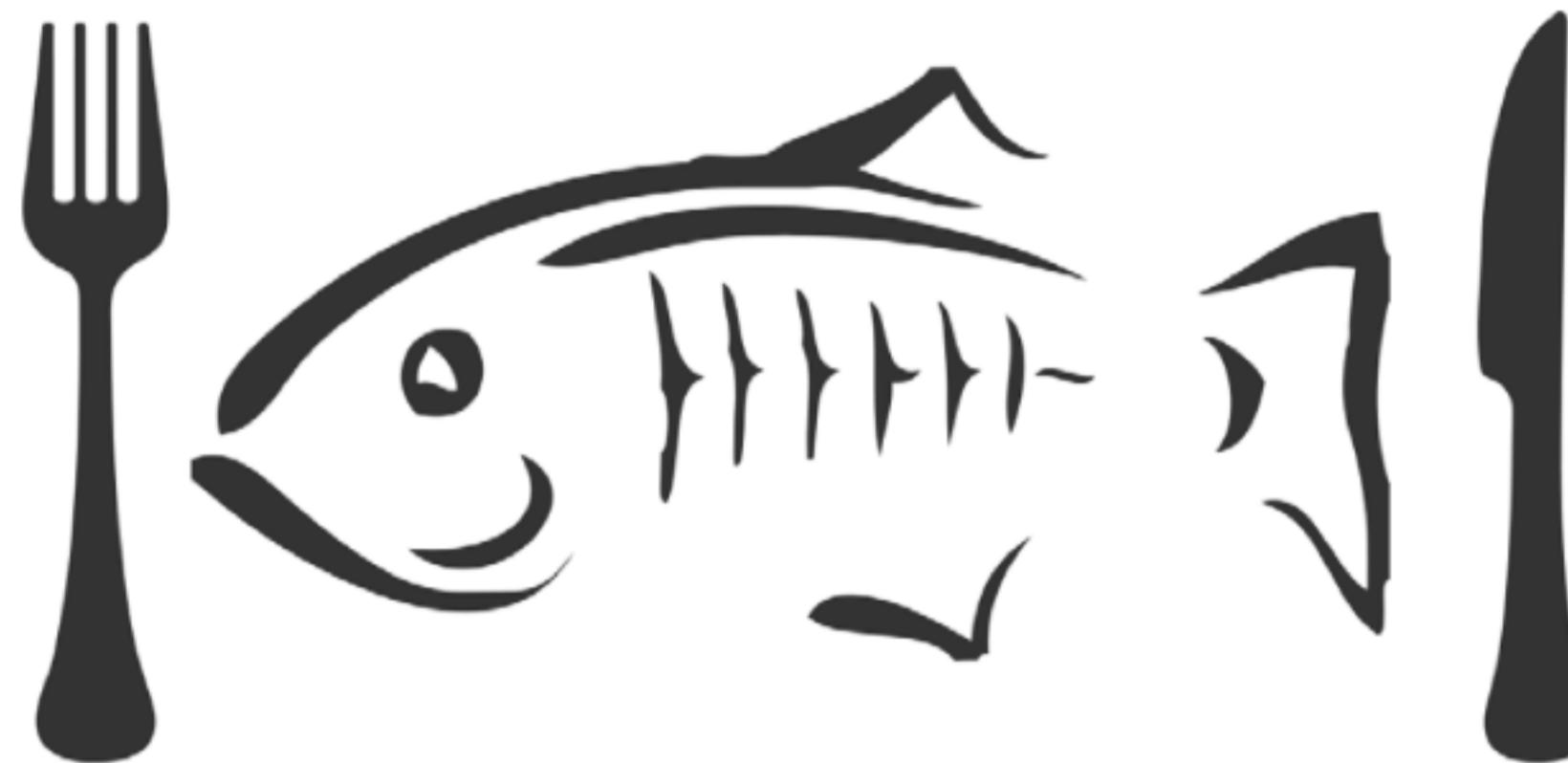
# Command Line Interface

- **jboss-cli.sh|bat**
- **Connects to standalone instance or Domain controller**
- **Interactive mode: \*nix-style shell**
  - **Contextual command and resource-tab completion**
- **Non-interactive mode: commands in files**
- **High-level compound operations**
- **Persistent changes**

# Admin Console

- **Simple**
  - **Fast**
  - **Lightweight**
  - **Avoids XML configuration**
  - **Single instance and domains**
  - **Mostly configuration, basic monitoring**
-

# Migrate to WildFly



- <http://wildfly.org/news/2014/02/06/GlassFish-to-WildFly-migration/>
  
- <http://zeroturnaround.com/rebellabs/abandon-fish-migrating-from-glassfish-to-jboss-or-tomee/>

“*the most logical decision when  
migrating from GlassFish is  
opting for an equivalent open  
source alternative*

– zeroturnaround.com

# **JBoss/WildFly or TomEE chosen because ...**

- **Vibrant developer community**
- **Amount of documentation on the web**
- **Sponsorship and support by market leaders**

- Java EE 7 compliant
- lightweight
- manageable
- highly scalable
- open source

**Now available!**



# References

- WildFly - <http://wildfly.org>, <http://github.com/wildfly>,  
@WildFlyAS
  - JBoss EAP 6.2 - <http://www.jboss.org/products/eap>
  - Java EE 7 samples - <https://github.com/javaee-samples/javaee7-samples>
  - Slides generated with Asciidoctor and DZSlides backend
  - Original slide template - Dan Allen & Sarah White
-

# Harald Pehl



@haraldpehl