

# WildFly 8 的新特性

---

作者 Arun Gupta · Red Hat · [@arungupta](https://twitter.com/arungupta)

**Arun Gupta**

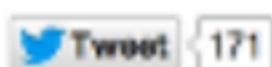
- Director, Developer Advocacy, Red Hat Inc.
- O'Reilly and McGraw Hill author
- Fitness freak

# 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!



**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

“

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

– zeroturnaround.com



The new and improved JBoss Application Server!

## 目标：

理解 WildFly 8 的新特性,以及 AS 7.X 的一些功能的改进

# WildFly 8 是什么?

# WildFly 8 是什么?

- 原先为“JBoss应用服务器(Application Server)”

# WildFly 8 是什么?

- 原先为“JBoss应用服务器(Application Server)”
- 是红帽JBoss企业应用平台(JBoss EAP)的上游项目

# WildFly 8 是什么?

- 原先为“JBoss应用服务器(Application Server)”
- 是红帽JBoss企业应用平台(JBoss EAP)的上游项目
- 快速，轻量，可管理的

# WildFly 8 是什么?

- 原先为“JBoss应用服务器(Application Server)”
- 是红帽JBoss企业应用平台(JBoss EAP)的上游项目
- 快速，轻量，可管理的
- 对开发人员友好

# WildFly 8 是什么?

- 原先为“JBoss 应用服务器(Application Server)”
- 是红帽JBoss企业应用平台(JBoss EAP)的上游项目
- 快速，轻量，可管理的
- 对开发人员友好
- 支持 Java EE 标准

# WildFly 8 是什么?

- 原先为“JBoss 应用服务器(Application Server)”,
- 是红帽JBoss企业应用平台(JBoss EAP)的上游项目
- 快速, 轻量, 可管理的
- 对开发人员友好
- 支持 Java EE 标准
- 开源

# WildFly 8 主要特性

## WildFly 8 主要特性

- Java EE7 支持

## WildFly 8 主要特性

- Java EE7 支持
- 高性能的Web服务器 **Undertow**

## WildFly 8 主要特性

- Java EE7 支持
- 高性能的Web服务器 **Undertow**
- 减少端口使用

## WildFly 8 主要特性

- Java EE7 支持
- 高性能的Web服务器 **Undertow**
- 减少端口使用
- 基于角色的管理控制和审计能力

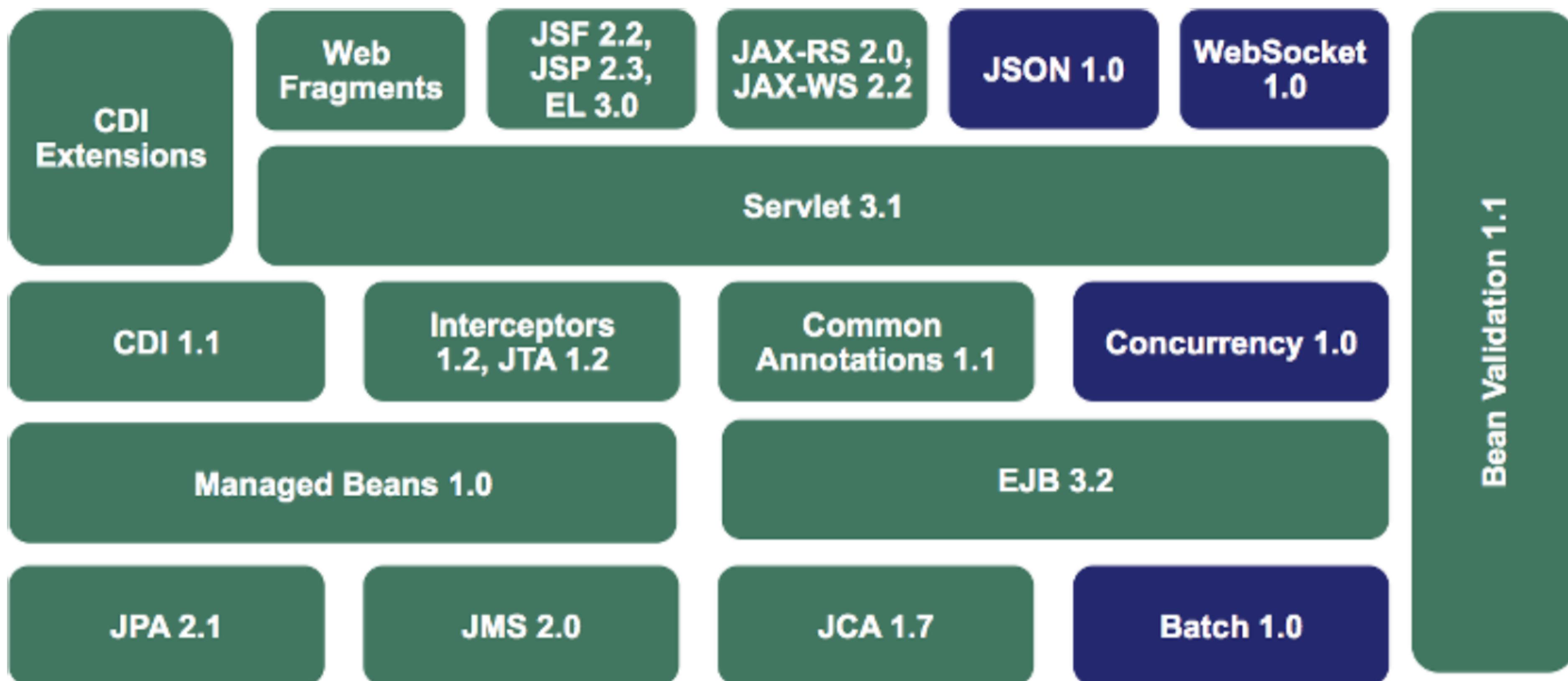
## WildFly 8 主要特性

- Java EE7 支持
- 高性能的Web服务器 **Undertow**
- 减少端口使用
- 基于角色的管理控制和审计能力
- 自动打补丁机制

## WildFly 8 主要特性

- Java EE7 支持
- 高性能的Web服务器 **Undertow**
- 减少端口使用
- 基于角色的管理控制和审计能力
- 自动打补丁机制
- 最小的 "core" 发布包

# 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 创建 WebSocket 端点, 定义 URL
- ② 设定方法来接收WebSocket消息
- ③ 处理WebSocket消息报文

## 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 对数据记录逐个进行处理，数据块的记录的个数
- ② 数据块处理，读出记录
- ③ 数据块处理，处理记录
- ④ 数据块处理，写入记录

# 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 创建JSON对象构建器
- ② 加入名称/值到JSON对象
- ③ 返回构建好的JSON对象
- ④ 通过JsonWriter写入

# WildFly: Java EE 7 Concurrency (JSR 236)

## RunMyTask.java

```
public class MyTask implements Runnable { ①

    @Override
    public void run() {
        . . .

    }

    @Resource(name = "DefaultManagedExecutorService") ②
    ManagedExecutorService defaultExecutor;

    executor.submit(new MyTask()); ③
}
```

① `Runnable` 或者 `Callable` 任务可以被提交

② `ManagedExecutor` 被注入, 提供缺省的服务资源

③ 提交要执行的任务

# 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 客户端入口点

② 创建一个新的资源目标点，给出URL路径

③ id定义

④ 定义可接收的媒体类型

⑤ 调用HTTP GET命令，并限定资源类型

# WildFly: Java EE 7 JMS (JSR 343)

## SendMessage.java

```
@JMSDestinationDefinition(name="myQueue", interfaceName="javax.jms.Queue") ①  
② @Resource(mappedName="myQueue")  
Queue syncQueue;  
  
③ @Inject  
// @JMSSConnectionFactory("java:comp/DefaultJMSSConnectionFactory")  
private JMSContext context; ④  
  
context.createProducer().send(syncQueue, "..."); ④
```

① JMSDestinationDefinition资源的创建和命名

② 缺省的 JMS connection factory

③ 简化的主要接口JMSContext

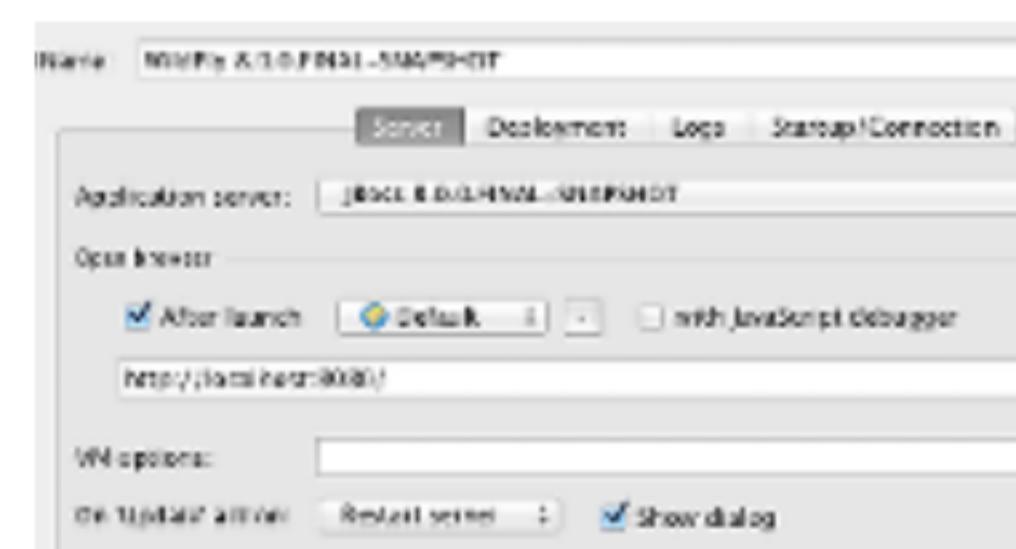
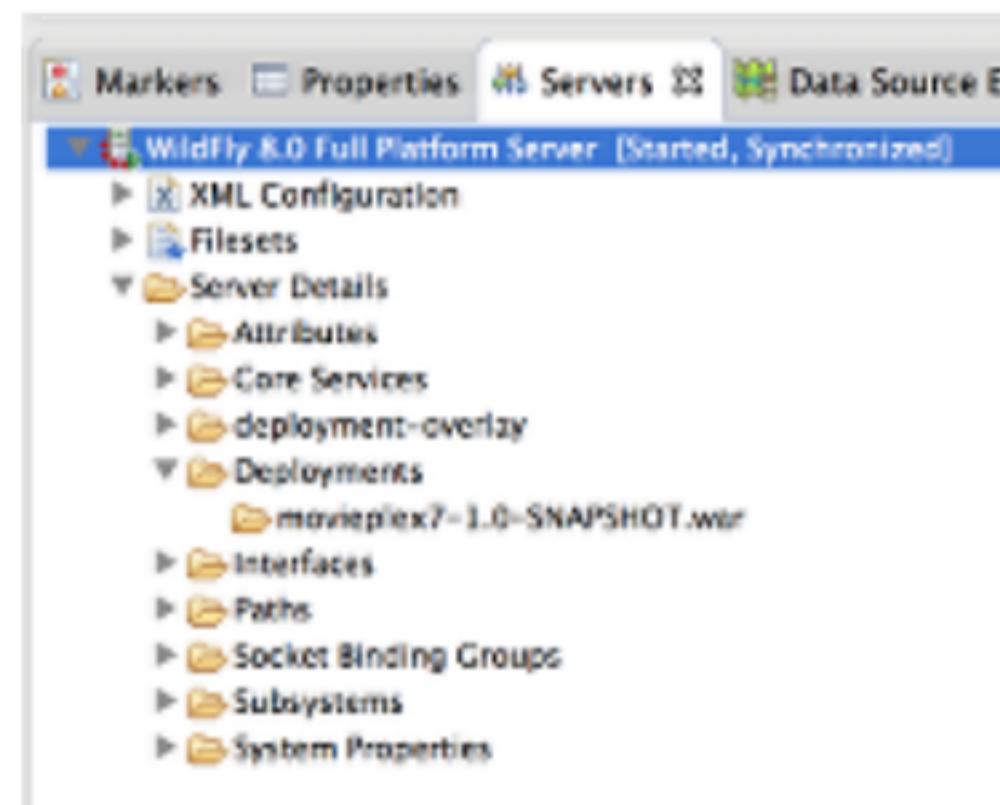
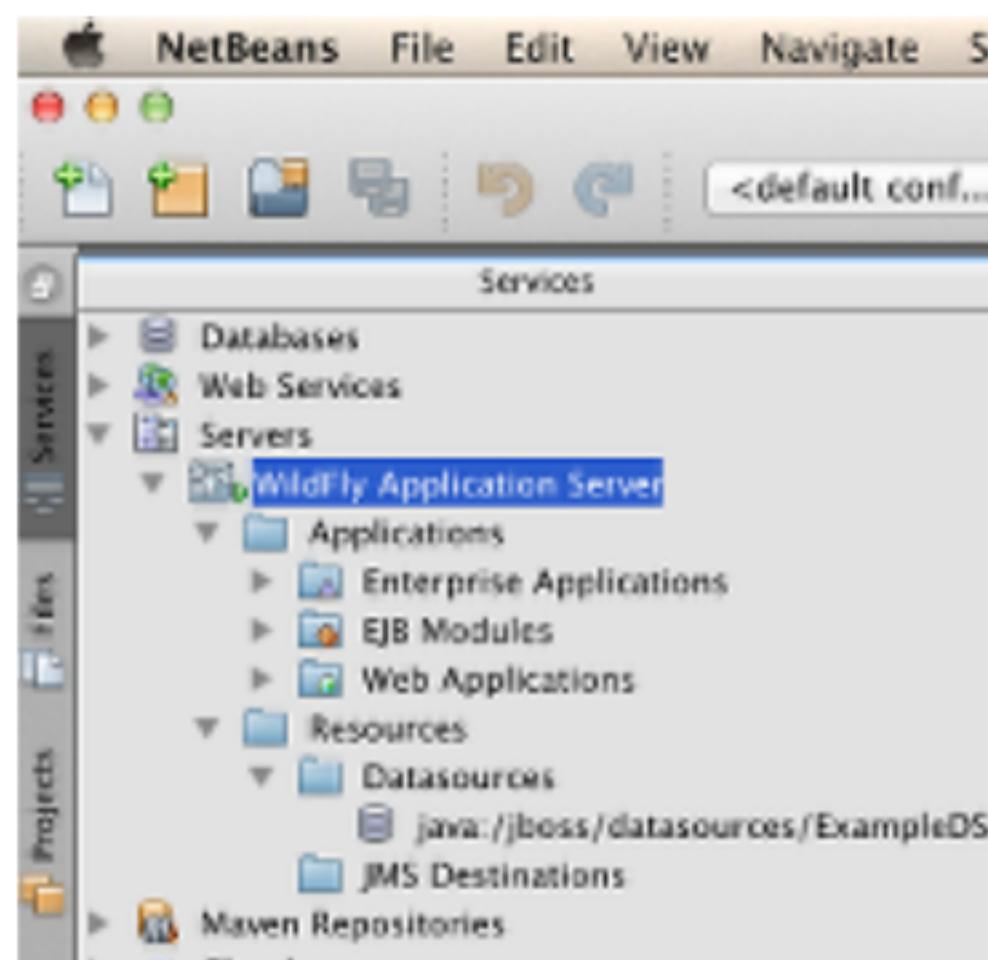
④ 流式构建API

- **@Transactional** CDI managed beans 上定义事务边界
- **@TransactionScoped** CDI 的范围，定义 JTA 事务期的 bean 实例

## Contexts and Dependency Injection 1.1 (JSR 346)

- 对于定义了范围的bean和EJB，CDI自动生效
- “beans.xml” 成为可选项配置
- Bean 发现模式
  - **all**: 所有的类型
  - **annotated**: 定义了annotation的bean类型
  - **none**: 禁止CDI
- **@Vetoed** 在classes上标记不被识别
- 拦截器和装饰器上定义顺序和优先级的annotation

# WildFly: Java EE 7 IDEs



## WildFly: 新的 web 服务器 (**Undertow**)

## WildFly: 新的 web 服务器 (**Undertow**)

- 可扩展，高性能

## WildFly: 新的 web 服务器 (**Undertow**)

- 可扩展，高性能
- 基于阻塞/非阻塞的**NIO API**

## WildFly: 新的 web 服务器 (**Undertow**)

- 可扩展，高性能
- 基于阻塞/非阻塞的**NIO API**
- 基于**Composition/handler**架构

## WildFly: 新的 web 服务器 (**Undertow**)

- 可扩展，高性能
- 基于阻塞/非阻塞的**NIO API**
- 基于**Composition/handler**架构
- 轻量的，可以被嵌入使用

## WildFly: 新的 web 服务器 (**Undertow**)

- 可扩展，高性能
- 基于阻塞/非阻塞的**NIO API**
- 基于**Composition/handler**架构
- 轻量的，可以被嵌入使用
- 支持 **Servlet 3.1** 和 **HTTP upgrade** 协议

## WildFly: 新的 web 服务器 (**Undertow**)

- 可扩展，高性能
- 基于阻塞/非阻塞的**NIO API**
- 基于**Composition/handler**架构
- 轻量的，可以被嵌入使用
- 支持 **Servlet 3.1** 和 **HTTP upgrade** 协议
- **mod\_cluster** 支持

# WildFly 新的 web 服务器: 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 流式API来构建

② 可以创建使用多个handler

③ 启动handler

## Undertow 性能测试数据

```
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.**



<http://www.techempower.com/blog/2014/03/04/one-million-http-rps-without-load-balancing-is-easy/>

# WildFly: 端口减少

WildFly: 端口减少

- 使用 HTTP Upgrade

## WildFly: 端口减少

- 使用 HTTP Upgrade
- 缺省安装后打开的端口只有两个
  - 8080 Web应用程序
  - 9990 管理界面

## WildFly: 端口减少

- 使用 HTTP Upgrade
- 缺省安装后打开的端口只有两个
  - 8080 Web应用程序
  - 9990 管理界面
- 通过HTTP Upgrade来处理各种请求

# WildFly: 基于角色的访问控制Role based access control

**WildFly: 基于角色的访问控制Role based access control**

- 预先定义的管理和特权用户角色(Roles)

- Monitor, Operator, Maintainer, Deployer, Administrator, Auditor, Super User

## WildFly: 基于角色的访问控制Role based access control

- 预先定义的管理和特权用户角色(Roles)
  - Monitor, Operator, Maintainer, Deployer, Administrator, Auditor, Super User
- 角色赋予权限(Permissions)

## WildFly: 基于角色的访问控制Role based access control

- 预先定义的管理和特权用户角色(Roles)
  - Monitor, Operator, Maintainer, Deployer, Administrator, Auditor, Super User
- 角色赋予权限(Permissions)
- 权限限定了可以对资源的各种 操作(Actions)(lookup, read, write)

## WildFly: 基于角色的访问控制Role based access control

- 预先定义的管理和特权用户角色(Roles)
  - Monitor, Operator, Maintainer, Deployer, Administrator, Auditor, Super User
- 角色赋予权限(Permissions)
- 权限限定了可以对资源的各种 操作(Actions)(lookup, read, write)
- 用户/Users)和组(Groups)在角色中定义

# WildFly: 管理审计日志

## WildFly: 管理审计日志

- 记录所有连接认证事件

## WildFly: 管理审计日志

- 记录所有连接认证事件
- 记录所有管理操作

## WildFly: 管理审计日志

- 记录所有连接认证事件
- 记录所有管理操作
- 记录信息是**JSON**格式

## WildFly: 管理审计日志

- 记录所有连接认证事件
- 记录所有管理操作
- 记录信息是**JSON**格式
- 通过不同**handlers**记录到
  - 本地文件
  - **Syslog (UDP / TCP / TLS)**

# WildFly: 自动补丁机制

## WildFly: 自动补丁机制

- 允许在安装时做库文件和配置的升级更新

## WildFly: 自动补丁机制

- 允许在安装时做库文件和配置的升级更新
- 补丁包含升级用到的文件和元数据信息，打成zip文件

## WildFly: 自动补丁机制

- 允许在安装时做库文件和配置的升级更新
- 补丁包含升级用到的文件和元数据信息，打包成zip文件
- 多次补丁包可以合并

## WildFly: 自动补丁机制

- 允许在安装时做库文件和配置的升级更新
- 补丁包含升级用到的文件和元数据信息，打包成zip文件
- 多次补丁包可以合并
- 可以回滚来撤销升级

# WildFly: 最小的 "core" 发布包

**WildFly: 最小的 "core" 发布包**

- **15 MB**

## WildFly: 最小的 "core" 发布包

- 15 MB
- 管理能力

## WildFly: 最小的 "core" 发布包

- 15 MB
- 管理能力
- 完整的并发服务控制框架

## WildFly: 最小的 "core" 发布包

- 15 MB
- 管理能力
- 完整的并发服务控制框架
- 模块化的类加载，可以用于多租户应用

## WildFly: 最小的 "core" 发布包

- 15 MB
- 管理能力
- 完整的并发服务控制框架
- 模块化的类加载，可以用于多租户应用
- 可插拔的热部署层

## WildFly: 最小的 "core" 发布包

- 15 MB
- 管理能力
- 完整的并发服务控制框架
- 模块化的类加载，可以用于多租户应用
- 可插拔的热部署层
- 内置轻量的**web**服务器

## WildFly: 其他內容

## WildFly: 其他內容

- 提高 JDK8 兼容性

## WildFly: 其他內容

- 提高 JDK8 兼容性
- RESTEasy 3

## WildFly: 其他內容

- 提高 JDK8 兼容性
- RESTEasy 3
- Hibernate search

## WildFly: 其他內容

- 提高 JDK8 兼容性
- RESTEasy 3
- Hibernate search
- 預先定义的安全权限

## WildFly: 其他內容

- 提高 JDK8 兼容性
- RESTEasy 3
- Hibernate search
- 预先定义的安全权限
- 新的集群API

## WildFly: 其他內容

- 提高 JDK8 兼容性
- RESTEasy 3
- Hibernate search
- 預先定义的安全权限
- 新的集群API
- 取消的內容: **CMP, JAX-RPC, JSR 88**

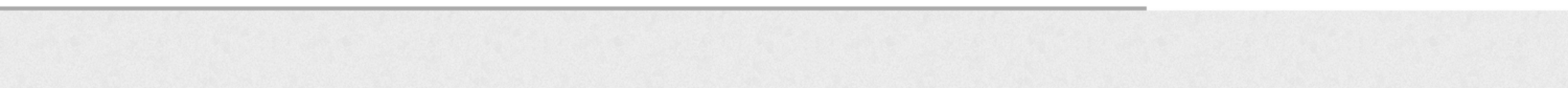
**WildFly: 在云端**



## AS 7.x 延续的特性

- 独立服务器(**Standalone**)和管理域(**Managed Domain**)
- 管理配置中心
  - 命令行 (**jboss-cli**)
  - 管理控制台 (**admin console**)
  - 配置文件

# Standalone 和 Managed Domain



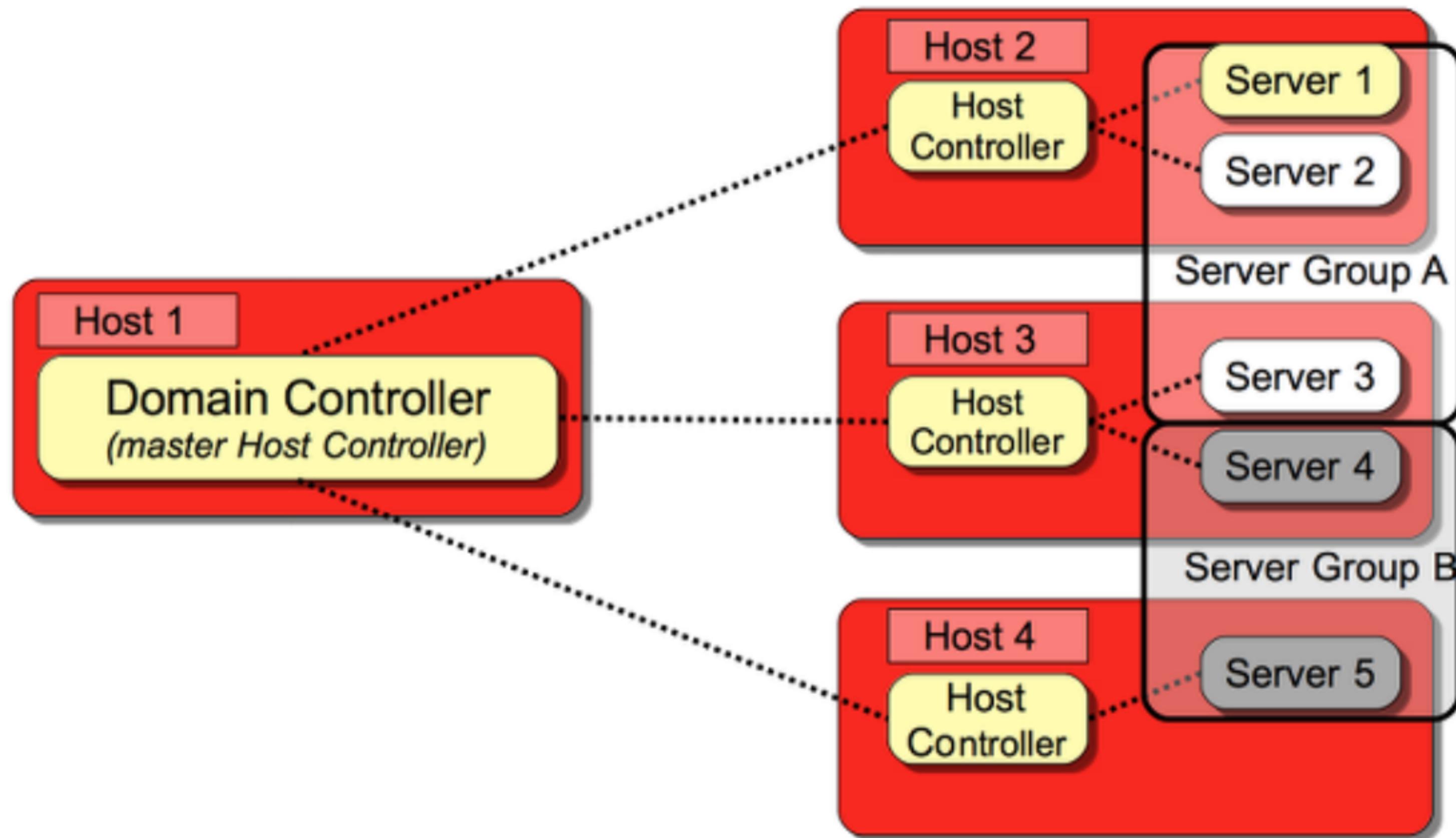
## Standalone 和 Managed Domain

- **Standalone:** 单独的实例

## Standalone 和 Managed Domain

- **Standalone:** 单独的实例
- **Managed domain:** 多个实例，被配置中心管理
  - HA from fail-over: 去除单点失败
  - HA from load-balancing: 客户端定时得到响应，即使存在大量请求

# Managed Domain



- 使用UDP来通信，可以配置成unicast TCP
- **mod\_cluster**, **mod\_jk**, **mod\_proxy**, ISAPI 或者 NSAPI 连接器
- 使用**mod\_cluster**的好处
  - 应用服务器做负载均衡因子计算
  - 服务器端的负载均衡参数通知给httpd
  - 动态配置HTTPD代理
  - 可以根据上下文信息，动态路由访问请求到节点

命令行



# 命令行

- **jboss-cli.sh bat**

命令行

- `jboss-cli.sh` `bat`
- 连接 `standalone` 或者 `Domain controller`

## 命令行

- `jboss-cli.sh` | `bat`
- 连接 `standalone` 或者 `Domain controller`
- 交互模式: `*nix shell` 风格
  - 上下文相关命令, `tab` 补齐功能

## 命令行

- `jboss-cli.sh` | `bat`
- 连接 `standalone` 或者 `Domain controller`
- 交互模式: `*nix shell` 风格
  - 上下文相关命令, `tab` 补齐功能
- 非交互模式: 文件中包含要执行的操作

## 命令行

- **jboss-cli.sh|bat**
- 连接 standalone 或者 Domain controller
- 交互模式: \*nix shell风格
  - 上下文相关命令, tab补齐功能
- 非交互模式: 文件中包含要执行的操作
- 多个命令组合成一个操作

## 命令行

- **jboss-cli.sh|bat**
- 连接 standalone 或者 Domain controller
- 交互模式: \*nix shell风格
  - 上下文相关命令, tab补齐功能
- 非交互模式: 文件中包含要执行的操作
- 多个命令组合成一个操作
- 每个修改都会被记录

# 控制台



# 控制台

- 简单

## 控制台

- 简单
- 快速

## 控制台

- 简单
- 快速
- 轻量

## 控制台

- 简单
- 快速
- 轻量
- 不需要 XML 配置文件

## 控制台

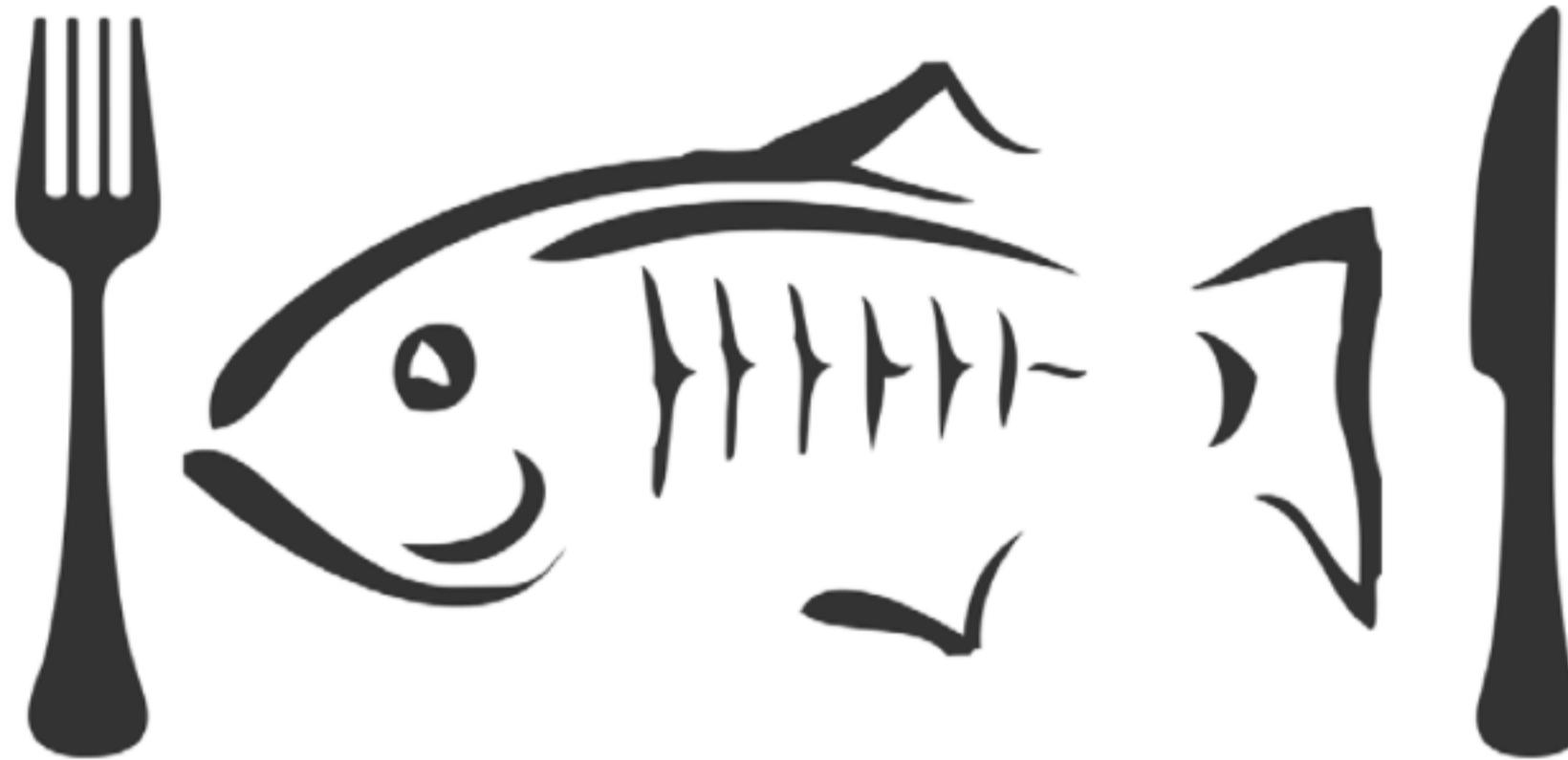
- 简单
- 快速
- 轻量
- 不需要 XML 配置文件
- 可以管理实例和域

## 控制台

- 简单
- 快速
- 轻量
- 不需要 XML 配置文件
- 可以管理实例和域
- 完成大多数配置管理功能，以及基本的监视功能
  - 注意，不能替代JON(Red Hat JBoss Operations Network)产品



## 迁移到WildFly



- i** <http://wildfly.org/news/2014/02/06/GlassFish-to-WildFly-migration/>
- i** <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 或者 TomEE 选型考虑 ...

- 活跃的开发者社区
- web上众多的文档
- 市场领导者的赞助和支持

- Java EE 7 兼容
- 轻量的
- 可管理的
- 高度可扩展
- 开源
- 应用服务器

马上使用!



# 开源技术大会



**DEVNATION**

April 13-16, 2014  
San Francisco, California • USA

**By and for  
developers  
across the globe.**

**Bringing together:**  
JUDCon CamelOne REACH CONNECT Developer Exchange

Learn more at [devnation.org](http://devnation.org)

## 参考

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

# Arun Gupta

 @arungupta