特别专题 语言 & 开发

ARKit何以从同类技术中胜出？


本文作者从技术角度剖析了扩增现实技术的发展趋势和目前面临的难点与局限，并重点介绍了Apple的AR技术：ARKit为何可以看作是目前最适合，也最有潜力的AR解决方案。

浏览所有 **语言 & 开发**

- 架构 & 设计
  - 架构
  - 企业架构
  - 性能和可伸缩性
  - 设计
  - 案例分析
  - 设计模式
  - 安全

特别专题 架构 & 设计

ARKit何以从同类技术中胜出？


本文作者从技术角度剖析了扩增现实技术的发展趋势和目前面临的难点与局限，并重点介绍了Apple的AR技术：ARKit为何可以看作是目前最适合，也最有潜力的AR解决方案。

浏览所有 **架构 & 设计**

- 数据科学
  - 大数据
  - NoSQL
  - 数据库

特别专题 数据科学

Reladomo：自备全套功能的企业级开源Java ORM（二）

Goldman Sachs在投资银行业务领域广为人知，它同样也是一家领先的技术公司。Reladomo是Goldman Sachs主要采用的Java ORM，现在开源发布了。在本文中，Goldman Sachs的技术专家Mohammad Rezaei将为我们介绍Reladomo的一些高级特性，包括分片、缓存、双时态方法、性能以及测试。

浏览所有 **数据科学**

- 文化 & 方法
  - Agile
  - 领导能力
  - 团队协作
  - 测试
  - 用户体验
  - Scrum
  - 精益

特别专题 文化 & 方法

Reladomo：自备全套功能的企业级开源Java ORM（二）

Goldman Sachs在投资银行业务领域广为人知，它同样也是一家领先的技术公司。Reladomo是Goldman Sachs主要采用的Java ORM，现在开源发布了。在本文中，Goldman Sachs的技术专家Mohammad Rezaei将为我们介绍Reladomo的一些高级特性，包括分片、缓存、双时态方法、性能以及测试。

浏览所有 **文化 & 方法**

- DevOps
  - 持续交付

- 自动化操作
- 云计算

特别专题 DevOps

MyHeritage是如何实现发布到生产环境的

本文介绍了MyHeritage的持续交付之旅，它实现了从耗费时间、令人痛苦的手工发布过程，向一种完全自动部署流水线的转变。

浏览所有 **DevOps**

架构
移动
运维
云计算
AI
大数据
容器
前端
QCon
ArchSummit
CNUTCon
百度
极客搜索

全部话题

Maven实战（九）——打包的技巧

 /作者 许晓斌 发布于 2011年6月20日. 估计阅读时间: 2 分钟 /智能化运维、Serverless、DevOps.....2017年有哪些最新运维技术趋势？CNUTCon即将为你揭秘! 14 讨论

“打包“这个词听起来比较土，比较正式的说法应该是”构建项目软件包“，具体说就是将项目中的各种文件，比如源代码、编译生成的字节码、配置文件、文档，按照规范的格式生成归档，最常见的当然就是JAR包和WAR包了，复杂点的例子是Maven官方下载页面的分发包，它有自定义的格式，方便用户直接解压后就在命令行使用。作为一款”打包工具“，Maven自然有义务帮助用户创建各种各样的包，规范的JAR包和WAR包自然不再话下，略微复杂的自定义打包格式也必须支持，本文就介绍一些常用的打包案例以及相关的实现方式，除了前面提到的一些包以外，你还能看到如何生成源码包、Javadoc包、以及从命令行可直接运行的CLI包。

Packaging的含义

任何一个Maven项目都需要定义POM元素packaging（如果不写则默认值为jar）。顾名思义，该元素决定了项目的打包方式。实际的情形中，如果你不声明该元素，Maven会帮你生成一个JAR包；如果你定义该元素的值为war，那你会得到一个WAR包；如果定义其值为POM（比如是一个父模块），那什么包都不会生成。除此之外，Maven默认还支持一些其他的流行打包格式，例如ejb3和ear。你不需要了解具体的打包细节，你所需要做的就是告诉Maven，”我是个什么类型的项目“，这就是**约定优于配置**的力量。

为了更好的理解Maven的默认打包方式，我们不妨来看看简单的声明背后发生了什么，对一个jar项目执行mvn package操作，会看到如下的输出：

```
[INFO] --- maven-jar-plugin:2.3.1:jar (default-jar) @ git-demo ---
[INFO] Building jar: /home/juven/git_juven/git-demo/target/git-demo-1.2-SNAPSHOT.jar
```

相比之下，对一个war项目执行mvn package操作，输出是这样的：

```
[INFO] --- maven-war-plugin:2.1:war (default-war) @ webapp-demo ---
[INFO] Packaging webapp
[INFO] Assembling webapp [webapp-demo] in [/home/juven/git_juven/webapp-demo/target/webapp-demo-1.0-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [/home/juven/git_juven/webapp-demo/src/main/webapp]
[INFO] Webapp assembled in [90 msecs]
[INFO] Building war: /home/juven/git_juven/webapp-demo/target/webapp-demo-1.0-SNAPSHOT.war
```

对应于同样的package生命周期阶段，Maven为jar项目调用了maven-jar-plugin，为war项目调用了maven-war-plugin，换言之，**packaging直接影响Maven的构建生命周期**。了解这一点非常重要，特别是当你需要自定义打包行为的时候，你就必须知道去配置哪个插件。一个常见的例子就是在打包war项目的时候排除某些web资源文件，这时就应该配置maven-war-plugin如下：

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-war-plugin</artifactId>
    <version>2.1.1</version>
    <configuration>
      <webResources>
        <resource>
          <directory>src/main/webapp</directory>
          <excludes>
            <exclude>**/*.jpg</exclude>
          </excludes>
        </resource>
      </webResources>
    </configuration>
  </plugin>
```

源码包和Javadoc包

本专栏的《坐标规划》一文中曾解释过，一个Maven项目只生成一个主构件，当需要生成其他附属构件的时候，就需要用上classifier。源码包和Javadoc包就是附属构件的极佳例子。它们有着广泛的用途，尤其是源码包，当你使用一个第三方依赖的时候，有时候会希望在IDE中直接进入该依赖的源码查看其实现的细节，如果该依赖将源码包发布到了Maven仓库，那么像Eclipse就能通过m2eclipse插件解析下载源码包并关联到你的项目中，十分方便。由于生成源码包是极其常见的需求，因此Maven官方提供了一个插件来帮助用户完成这个任务：

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-source-plugin</artifactId>
  <version>2.1.2</version>
  <executions>
    <execution>
      <id>attach-sources</id>
      <phase>verify</phase>
      <goals>
        <goal>jar-no-fork</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

类似的，生成Javadoc包只需要配置插件如下：

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>2.7</version>
  <executions>
    <execution>
      <id>attach-javadocs</id>
        <goals>
          <goal>jar</goal>
        </goals>
    </execution>
  </executions>
</plugin>
```

为了帮助所有Maven用户更方便的使用Maven中央库中海量的资源，中央仓库的维护者强制要求开源项目提交构件的时候同时提供源码包和Javadoc包。这是个很好的实践，读者也可以尝试在自己所处的公司内部实行，以促进不同项目之间的交流。

可执行CLI包

除了前面提到了常规JAR包、WAR包，源码包和Javadoc包，另一种常被用到的包是在命令行可直接运行的CLI（Command Line）包。默认Maven生成的JAR包只包含了编译生成的.class文件和项目资源文件，而要得到一个可以直接在命令行通过java命令运行的JAR文件，还要满足两个条件：

- JAR包中的/META-INF/MANIFEST.MF元数据文件必须包含Main-Class信息。
- 项目所有的依赖都必须在Classpath中。

Maven有好几个插件能帮助用户完成上述任务，不过用起来最方便的还是maven-shade-plugin，它可以让用户配置Main-Class的值，然后在打包的时候将值填入/META-INF/MANIFEST.MF文件。关于项目的依赖，它很聪明地将依赖JAR文件全部解压后，再将得到的.class文件连同当前项目的.class文件一起合并到最终的CLI包中，这样，在执行CLI JAR文件的时候，所有需要的类就都在Classpath中了。下面是一个配置样例：

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  <version>1.4</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>shade</goal>
      </goals>
      <configuration>
        <transformers>
```

```
          <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
            <mainClass>com.juvenxu.mavenbook.HelloWorldCli</mainClass>
          </transformer>
        </transformers>
      </configuration>
    </execution>
  </executions>
</plugin>
```

上述例子中的，我的Main-Class是com.juvenxu.mavenbook.HelloWorldCli，构建完成后，对应于一个常规的hello-world-1.0.jar文件，我还得到了一个hello-world-1.0-cli.jar文件。细心的读者可能已经注意到了，这里用的是cli这个classifier。最后，我可以通过**java -jar hello-world-1.0-cli.jar**命令运行程序。

自定义格式包

实际的软件项目常常会有更复杂的打包需求，例如我们可能需要为客户提供一份产品的分发包，这个包不仅仅包含项目的字节码文件，还得包含依赖以及相关脚本文件以方便客户解压后就能运行，此外分发包还得包含一些必要的文档。这时项目的源码目录结构大致是这样的：

```
pom.xml
src/main/java/
src/main/resources/
src/test/java/
src/test/resources/
src/main/scripts/
src/main/assembly/
README.txt
```

除了基本的pom.xml和一般Maven目录之外，这里还有一个src/main/scripts/目录，该目录会包含一些脚本文件如run.sh和run.bat，src/main/assembly/会包含一个assembly.xml，这是打包的描述文件，稍后介绍，最后的README.txt是份简单的文档。

我们希望最终生成一个zip格式的分发包，它包含如下的一个结构：

```
bin/
lib/
README.txt
```

其中bin/目录包含了可执行脚本run.sh和run.bat，lib/目录包含了项目JAR包和所有依赖JAR，README.txt就是前面提到的文档。

描述清楚需求后，我们就要搬出Maven最强大的打包插件：[maven-assembly-plugin]。它支持各种打包文件格式，包括zip、tar.gz、tar.bz2等等，通过一个打包描述文件（该例中是src/main/assembly.xml），它能够帮助用户选择具体打包哪些文件集合、依赖、模块、和甚至本地仓库文件，每个项的具体打包路径用户也能自由控制。如下就是对应上述需求的打包描述文件src/main/assembly.xml：

```
<assembly>
  <id>bin</id>
  <formats>
    <format>zip</format>
  </formats>
  <dependencySets>
    <dependencySet>
      <useProjectArtifact>true</useProjectArtifact>
      <outputDirectory>lib</outputDirectory>
    </dependencySet>
  </dependencySets>
  <fileSets>
    <fileSet>
      <outputDirectory>/</outputDirectory>
      <includes>
        <include>README.txt</include>
      </includes>
    </fileSet>
    <fileSet>
      <directory>src/main/scripts</directory>
      <outputDirectory>/bin</outputDirectory>
      <includes>
        <include>run.sh</include>
```

```
      <include>run.bat</include>
    </includes>
  </fileSet>
 </fileSets>
</assembly>
```

- 首先这个assembly.xml文件的id对应了其最终生成文件的classifier。
- 其次formats定义打包生成的文件格式，这里是zip。因此结合id我们会得到一个名为hello-world-1.0-bin.zip的文件。（假设artifactId为hello-world，version为1.0）
- dependencySets用来定义选择依赖并定义最终打包到什么目录，这里我们声明的一个depenencySet默认包含所有所有依赖，而useProjectArtifact表示将项目本身生成的构件也包含在内，最终打包至输出包内的lib路径下（由outputDirectory指定）。
- fileSets允许用户通过文件或目录的粒度来控制打包。这里的第一个fileSet打包README.txt文件至包的根目录下，第二个fileSet则将src/main/scripts下的run.sh和run.bat文件打包至输出包的bin目录下。

打包描述文件所支持的配置远超出本文所能覆盖的范围，为了避免读者被过多细节扰乱思维，这里不再展开，读者若有需要可以去参考这份文档。

最后，我们需要配置maven-assembly-plugin使用打包描述文件，并绑定生命周期阶段使其自动执行打包操作：

```
 <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>2.2.1</version>
  <configuration>
   <descriptors>
    <descriptor>src/main/assembly/assembly.xml</descriptor>
   </descriptors>
  </configuration>
  <executions>
   <execution>
    <id>make-assembly</id>
    <phase>package</phase>
    <goals>
     <goal>single</goal>
    </goals>
   </execution>
  </executions>
 </plugin>
```

运行**mvn clean package**之后，我们就能在target/目录下得到名为hello-world-1.0-bin.zip的分发包了。

小结

打包是项目构建最重要的组成部分之一，本文介绍了主流Maven打包技巧，包括默认打包方式的原理、如何制作源码包和Javadoc包、如何制作命令行可运行的CLI包、以及进一步的，如何基于个性化需求自定义打包格式。这其中涉及了很多的Maven插件，当然最重要，也是最为复杂和强大的打包插件就是maven-assembly-plugin。事实上Maven本身的分发包就是通过maven-assembly-plugin制作的，感兴趣的读者可以直接查看源码一窥究竟。

语言 & 开发 DevOps Maven 实战 持续集成 专栏 工具 Maven 敏捷技术 Java 敏捷 构建系统

相关主题:

相关内容

Microsoft开源JDBC驱动和Maven支持

在HubSpot是如何应对Fat JAR困境的

Reladomo：自备全套功能的企业级开源Java ORM（二）

Reladomo：自备全套功能的企业级开源Java ORM（一）

Java依旧排名第一，然而变化几何？

相关厂商内容

如何更好地构建微服务下的性能监控

游戏开发与运维实践经验谈

天猫DevOps转型实践

你对AIDevOps了解多少？

为什么DevOps带火了Spring？

相关赞助商

您好，朋友！

您需要 注册一个InfoQ账号 或者 登录 才能进行评论。在您完成注册后还需要进行一些设置。

获得来自InfoQ的更多体验。

告诉我们您的想法

请输入主题

信息

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

发送信息

社区评论 Watch Thread

主要是想研究一下war包中的/WEB-INF/lib包中不希望打包进其它的第三方包 2011年6月22日 11:56 by "杨 思勇"

这个如何配置？搞了好久没成功。

喜欢

- 回复
- 回到顶部

不错～ 2011年6月22日 11:39 by "Chen reniaL"

不错的入门指引～

如果一个项目想要制作在各种环境中（开发，测试，生产）使用的不同的包，就是用 maven-assembly-plugin 来实现的吧？

喜欢

- 回复
- 回到顶部

Re: 主要是想研究一下war包中的/WEB-INF/lib包中不希望打包进其它的第三方包 2011年6月23日 02:05 by "许 晓斌"

把你的<dependency>设置<scope>为provided即可</scope></dependency>

喜欢
- 回复
- 回到顶部

打包部署后的源代码包怎么用？ 2011年6月23日 01:26 by "zhe sui"

如果要在脚本中直接访问打包好的源代码包该怎么使用？编译好的jar包可以这样用：${junit:junit:jar}，源代码包不知道如何访问。

喜欢
- 回复
- 回到顶部

Re: 不错～ 2011年6月26日 07:48 by "Chen Casel"

从最佳实践的角度讲，war,ear,jar这种二进制文件应该是跨环境适用的，不同环境的配置应当是在配置文件里。maven-assembly-plugin功能强大，但一般项目可以不用它也能够实现打包与分发到不同环境。

喜欢
- 回复
- 回到顶部

Re: 主要是想研究一下war包中的/WEB-INF/lib包中不希望打包进其它的第三方包 2011年6月26日 07:50 by "Chen Casel"

你可以参考一下Creating Skinny WARs这一节maven.apache.org/plugins/maven-war-plugin/examp...

喜欢
- 回复
- 回到顶部

Re: 主要是想研究一下war包中的/WEB-INF/lib包中不希望打包进其它的第三方包 2011年9月2日 12:25 by "liu zongyao"

打依赖包老是遇到
java.net.MalformedURLException: no protocol: src/main/assembly/assembly.xml
错误， 不知道是什么原因？

喜欢
- 回复
- 回到顶部

打tar包的问题 2012年8月16日 04:11 by "wang charles"

Hi Juven;，
我有一个问题想请教你，比如当我用maven-assembly-plugin打一个tar分发包的时候，假设我的assembly.xml配置如下

<assembly>
<id>tarball</id>
<formats>
<format>tar</format>
</formats>

<fileSets>

<fileSet>
<directory>src/main/scripts</directory>
<outputDirectory>/scripts</outputDirectory>
</fileSet>
<fileSet>
<directory>src/main/standalone</directory>
<outputDirectory>/standalone</outputDirectory>
</fileSet>
...

这时候，我最后打出来的tarball.tar结构如下,如果你 tar -tvf tarball.tar时候
你会看到：tarball/standalone/
tarball/standalone/file1.txt
tarball/scripts/
tarball/scripts/1.sh

但是，我最终想要的结果tarball.tar，当tar -tvf tarball.tar时，应该有如下的结构：
standalone/
standalone/file1.txt
scripts/
scripts/1.sh

（ Note; 这种结构应该和 tar -cf tarball.tar scripts/ standalone/ 命令的输出结果一致）

不知道你清楚没有我的意思，我意思就是在用maven-assembly-plugin打包时候，不希望有一个我们的<assembly>的id包含在我们想包含的内容的外层，请问在assembly.xml，或者在pom.xml中应该如何设置？

</assembly></filesets></assembly>

喜欢

- 回复
- 回到顶部

Re: 打tar包的问题 2012年11月12日 11:12 by "lee drug"

> Hi Juven;，
> 我有一个问题想请教你，比如当我用maven-assembly-plugin打一个tar分发包的时候，假设我的assembly.xml配置如下
>
> <assembly>
> <id>tarball</id>
> <formats>
> <format>tar</format>
> </formats>

<fileSets>
<fileSet>
<directory>src/main/scripts</directory>
<outputDirectory>/scripts</outputDirectory>
</fileSet>
<fileSet>
<directory>src/main/standalone</directory>
<outputDirectory>/standalone</outputDirectory>
</fileSet>
...

这时候，我最后打出来的tarball.tar结构如下,如果你 tar -tvf tarball.tar时候
你会看到：tarball/standalone/
tarball/standalone/file1.txt
tarball/scripts/
tarball/scripts/1.sh

但是，我最终想要的结果tarball.tar，当tar -tvf tarball.tar时，应该有如下的结构：
standalone/
standalone/file1.txt
scripts/
scripts/1.sh

（Note; 这种结构应该和 tar -cf tarball.tar scripts/ standalone/ 命令的输出结果一致）

> 不知道你清楚没有我的意思，我意思就是在用maven-assembly-plugin打包时候，不希望有一个我们的<assembly>的id包含在我们想包含的内容的外层，请问在assembly.xml，或者在pom.xml中应该如何设置？
>
> </assembly></filesets></assembly>

试一下在assembly文件里面加入<includeBaseDirectory>false</includeBaseDirectory>

喜欢

- 回复
- 回到顶部

Re: 主要是想研究一下war包中的/WEB-INF/lib包中不希望打包进其它的第三方包 2012年11月22日 06:14 by "L koala"

把assembly.xml 放到 src/main/assembly/文件夹下就好了，原文"如下就是对应上述需求的打包描述文件src/main/assembly.xml"这里写错了，应该是"src/main/assembly/assembly.xml"。

喜欢

- 回复
- 回到顶部

添加了assembly后，如果只想生成简单的jar,不使用assembly的配置，可以实现吗? 2013年4月25日 05:00 by "zhong he"

添加了assembly后，如果只想生成简单的jar,不使用assembly的配置，可以用命令实现吗?
因为assembly只是可能只是在部署的时候会用，以后更新代码什么只需更新jar就可以了，可是
mvn clean package命令还是都全生成，有可选参数只生成jar都好了

喜欢
- 回复
- 回到顶部

关于配置几点疑惑 2014年3月19日 10:44 by "ng allen"

为什么不使用maven的推荐目录src/assembly/，
为什么不使用maven-assembly-plugin插件的推荐目录src/main/resources/assemblies/，
后者默认会被打包进WEB-INF，这不是希望的，那么前者呢？

喜欢
- 回复
- 回到顶部

bucuo 2014年4月22日 04:14 by "张童 鸥翔鱼游"

不错的入门指引~

喜欢
- 回复
- 回到顶部

Re: 主要是想研究一下war包中的/WEB-INF/lib包中不希望打包进其它的第三方包 2014年11月18日 11:31 by "徐 世超"

个人感觉 好像设成runtime的就行了吧

喜欢
- 回复
- 回到顶部

关闭

by

发布于

- 查看
- 回复
- 回到顶部

关闭

主题 您的回复

引用原消息

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

发送信息 取消

关闭

主题 您的回复

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

当有人回复此评论时请E-mail通知我

取消

关闭

OK

赞助商链接

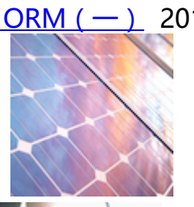
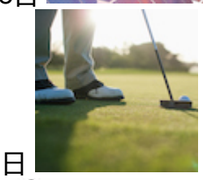
相关内容

- 架构师特刊：编程语言 2017年8月17日 
- Java依旧排名第一，然而变化几何？ 2017年8月17日
- 使用Spring Cloud Function框架进行面向函数的编程 2017年8月16日
- JetBrains当选JCP执行委员会委员 2017年8月9日
- 架构师（2017年7月） 2017年7月8日 

RELADOMO

- Reladomo：自备全套功能的企业级开源Java ORM（二） 2017年8月24日

RELADOMO

- Reladomo：自备全套功能的企业级开源Java ORM（一） 2017年8月21日 
- 案例学习：Jigsaw模块化迁移 2017年8月8日 
- 深入探索JVM自动资源管理 2017年7月11日 
- Java 老矣，尚能饭否？ 2017年6月29日
- 想知道垃圾回收暂停的过程中发生了什么吗？查查垃圾回收日志就知道了！ 2017年5月17日 

赞助商内容

- 

与国际大牛一道参与大师赛争夺战

  大数据竞赛平台DataCastle推出国际大师赛瞄准复杂网络的关键节点挖掘，面向全球精英极客寻求结合数学、物理、计算机的最前沿的方案。

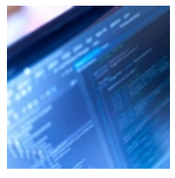
- 

Apache CarbonData+Spark应用实践之道

  来自美国Databricks、华为、上海等行业顶尖专家，将通过Spark SQL、Spark 2.2 CBO、CarbonData应用实践及2.0新技术规划等热点的分享，为你展示Apache CarbonData技术全貌，限额报名

来自美国Databricks、华为、上汽等行业顶尖专家，将通过Spark SQL、Spark 2.2 CBO、CarbonData应用实践及2.0新技术规划等热点的分享，为你展示Apache CarbonData技术全貌，限额报名~

相关内容

- JVM上的确定性执行机制 2017年5月12日 
- Invokedynamic：Java的秘密武器 2017年3月24日 
- 使用模板将Web服务的结果转换为标记语言 2017年3月23日 
- 多形态MVC式Web架构：完成实时响应 2017年3月22日 


- 测试RxJava2 2017年3月15日 
- 拥抱可变性 2017年3月8日 
- 阿里巴巴Java开发手册 2017年3月7日 
- InfoQ观点：Java EE的未来 2017年3月6日 
- RxJava2实例解析 2017年2月28日 
- 如何解决特征工程，克服工业界应用 AI 的巨大难关 2017年8月2日 
- DevOps是MindSet：工具也好文化也罢，人员才是关键 2017年5月19日 

相关内容

- Puppet 4 性能提升超2倍，升级前应该你知悉的变化 2017年3月23日 
- 滴滴是如何高效率处理线上故障的？ 2017年8月24日
- 2017敏捷大会主题演讲：全面打造领导力以及参与文化 2017年8月24日
- LLVM构建了PDB文档，并提供了PDB与YAML的转换工具 2017年8月24日
- .NET Core 2 发布，支持.NET 标准版 2.0 2017年8月24日
- 关于Oracle欲放手Java EE，Red Hat有话要说 2017年8月24日
- Swift 5确立了目标，定义了新的演进过程 2017年8月24日
- 英国温德姆度假租赁公司采用敏捷和DevOps 2017年8月24日
- Entity Framework Core 2.0的槽点 2017年8月24日
- 2017年敏捷沙滩大会：技术卓越、为持续交付优化的组织、容器安全 2017年8月24日
- ARKit何以从同类技术中胜出？ 2017年8月24日 

语言 & 开发

关于Oracle欲放手Java EE，Red Hat有话要说

2017年敏捷沙滩大会：技术卓越、为持续交付优化的组织、容器安全

.NET Core 2 发布，支持.NET 标准版 2.0

架构 & 设计

滴滴是如何高效率处理线上故障的？

关于Oracle欲放手Java EE，Red Hat有话要说

ARKit何以从同类技术中胜出？

文化 & 方法

2017年敏捷沙滩大会：技术卓越、为持续交付优化的组织、容器安全

英国温德姆度假租赁公司采用敏捷和DevOps

2017敏捷大会主题演讲：全面打造领导力以及参与文化

数据科学

Reladomo：自备全套功能的企业级开源Java ORM（二）

Facebook转向神经网络机器翻译（NMT）

Reladomo：自备全套功能的企业级开源Java ORM（一）

DevOps

滴滴是如何高效率处理线上故障的？

英国温德姆度假租赁公司采用敏捷和DevOps

InfoQ每周精要

订阅InfoQ每周精要，加入拥有25万多名资深开发者的庞大技术社区。

您的邮箱 订阅