怎样升级到Liferay数字化体验平台

LIFERAY

目录

简介

随着客户对全方位数字化体验的需求的增加，所以企业需要通过正确的技术武装自己，并且可以快速适应未来的数字技术的更新。Liferay数字化体验平台使数字化企业可以管理和提供跨渠道接触点一致的的客户体验，包括移动端、自助服务机、智能设备等。

升级到Liferay DXP 可以为呈现数字化体验奠定基础，通过服务与数字化用户建立日益紧密的联系，在最佳位置升级您的业务可以利用Liferay数字化业务的最新开发技术的优势包括呈现移动端体验，多渠道吸引和内容定位等

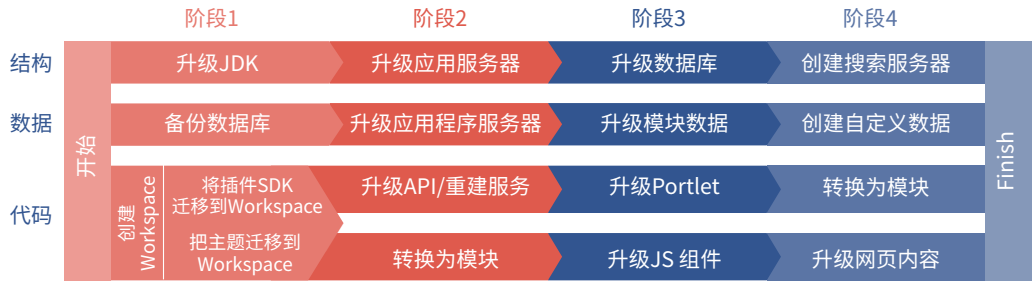
此白皮书旨在为您的业务建立Liferay推荐升级路径框架。主要的技术升级需要深入分析您的业务需求，进行周密的策划、测试和执行才能取得成功。在您计划和执行之前，Liferay全球服务团队，我们专业的咨询师团队，可以通过Liferay升级分析程序帮助您进行关键性需求分析。我们的全球服务团队在帮助客户升级到最新Liferay平台上有丰富的经验。这种对关键考虑因素进行综合分析的意识至关重要，您可以为企业做出最明智的决定并享受Liferay平台带来的优势。

设置合适的时间表

升级需要进行数据更改，代码修改和架构更改。升级过程本身就有很大的风险。考虑以下问题可以让您设置合适的时间表和期望，更好地管理风险。

1. **我现在是什么版本?**如果您的版本与现行Liferay版本相差几个版本，您需要牢记在最终升级到Liferay DXP之前，您的数据需要经过之前全部的升级路径。这意味着如果您是Liferay 6.1版本，那么您的数据需要先升级到Liferay 6.2版本，然后再升级到Liferay DXP。升级工具虽然可以帮助您处理，但是您可能需要更多手动方法。
2. **我有多少数据?**如果您的项目有大量的数据，有一个正确索引的数据库至关重要。如果您有一个更大的数据库，需要预留出更多的时间。
3. **我有多少网页内容、模板和结构需要升级?**一些人决定要在升级后重新写网页内容，而另一些人则决定重新使用之前的内容。
4. **我有多少Portlet需要升级?**不是所有的Portlet都需要升级到Liferay DXP，正确的分析会为您更好的预估时间和需要投入的工程师数量。
5. **我需要重写很多JSP吗?**JSP从6.1版本到6.2版本有很大改变，我们建议尽可能的移出已经重写的JSP文件。如果可以的话，大量JSP是支持插件扩展的，而不用覆写。
6. **我需要升级EXT吗?**好消息是通过Liferay DXP，我们已经创建了很多扩展点，所以不需要升级EXT。
7. **我需要转换成OSGi bundle吗?**这将花费更多时间，但是投资将是非常值得的。

LiferayDXP升级路径样本



架构改变

兼容性矩阵

既然您已经准备好升级，首先请确保您的环境匹配 LiferayDXP兼容矩阵。自从6.2版本发布后，许多环境生命周期已经结束；如果您升级自更古老的版本，将会更多环境结束生命周期。您的系统在一个受支持的环境中运行是很重要的，这样可以得到合适的支持。

搜索

您将面对的最大变化是Liferay现在需要搜索引擎作为一个单独的JVM运行，在同一个或专用的物理/虚拟服务器上运行。

我们发现，搜索已经成为任何网站的一个必不可少的部分，搜索引擎的性能和可扩展性文件不同于Portal的搜索引擎，可以更主动地服务于页面体验。过去，Liferay Portal 不是支持嵌入式的Lucene的搜索引擎就是远程部署Solr搜索引擎。使用DXP，我们继续提供Solr作为支持的远程搜索引擎。不过，建议首选使用Elasticsearch，嵌入式开发过程和生产环境下的独立模式可以使用相同的搜索引擎。

如果您以前使用Solr，您将只需要确保您的Solr模块是最新的，并且Solr服务器升级到5.0版本。请参阅我们的搜索文档来配置您的搜索服务器。此外请注意，您可以在同一台服务器上运行您的搜索服务器，但是需要另外的JVM，如果您的资源有限，不推荐这个方法。

JDK

另一个变化是兼容性矩阵已经升级到JDK8，所有应用程序服务器已经与JDK8兼容，请确保您的应用程序服务器已正确配置。

部署计划

通过Liferay DXP中OSGi容器的介绍，您的部署计划需要更改，因为有不同的方法来部署您的插件。

- **WAR** 是被熟知的Liferay插件网页应用程序（如，*-theme.war, *-portlet.war, *-web.war）。在Liferay DXP中不推荐使用WAR，因为您不会从由OSGi提供的显性依赖和生命周期管理模式中获得收益。
- **Bundle/Module** 是已经转换成OSGi Bundles的插件。他们只是简单的装载OSGi元数据的Java JAR文件。Bundle可以只部署在OSGi容器中。Bundle不可以访问部署在WAR中的服务，除非服务已经在Liferay的核心服务中。这是我们所推荐的方法：所有新的开发都使用OSGi bundle。对于Liferay所有新的开发，也都会使用OSGi方法。
- **WAB** 是网页应用包。如果您部署一个WAR到OSGi容器，Liferay将把WAR转换成WAB。您不用进行转换就可以得到Bundle带来的所有优势。这是用于为Liferay老版本的部署旧版本遗留应用的推荐方法。

现在Liferay自动部署目录会部署到OSGi容器，部署到您网页应用程序的部署文件夹的唯一方法是通过您应用程序服务器的部署机制。复制到Liferay部署文件夹中的任何文件都将被自动拖入到Liferay OSGi容器中。

注意：您不能直接使用应用程序服务器的部署工具部署Liferay文件（WAR，模块，WAB）。

当在应用程序服务器中部署OSGi管理集群（例如，JBoss领域模型、WebLogic w/ Node Manager、WebSphere w/ Deployment Manager），您可以使用Liferay的集群部署助手。

此工具可以把特定部署文件包成一个特定的WAR。当应用程序部署并启动WAR时，启动机制会把OSGi模块拖入应用程序服务器上的Liferay数字化企业版OSGi部署文件夹中。

数据库升级

准备工作

既然您已经把架构升级到支持的版本，我们可以把注意力放在数据上。以防出现升级失败，首先我们应该进行适当的备份。

如果您是6.1版本，接下来请确保您正在运行权限算法6。

此外请记住，您所有的数据库索引需要正确引用。一个索引的缺失可能造成升级缓慢。如果您稍后运行到一个缓慢的升级，您可以回去添加额外的临时索引来加快升级。

我们还建议您在这个过程中禁用搜索索引。为了实现这一点，您可以通过以下内容添加com.liferay.portal.search.configuration.IndexStatusManagerConfiguration.cfg到您的 your osgi/configs/ folder中：

```
indexReadOnly=true
```

通过此法，在升级过程中您可以避免索引并节省时间。一旦您升级了Portal，确保设置此属性为false，然后您可以在控制面板中索引所有对象。

升级工具

在Liferay DXP中，数据库升级工具是独立的。

运行此工具：

```
java -jar com.liferay.portal.tools.db.upgrade.client.jar
```

在运行之前，升级需要配置三个文件：

1. app-server.properties,
2. portal-upgrade-database.properties,
3. portal-upgrade-ext.properties.

您可以手动或使用工具完成配置。数据升级分成两部分，升级核心与过去方法类似，下一部分将升级OSGi 模块。在默认情况下，升级工具可以自动升级这两部分。

疑难解答

如果您成功运行升级，您可以跳过此部分，如果您在运行中遇到问题，这里有些提示可以帮助您。

如果您在升级核心数据和运行时遇到问题，记住开始之前我们的提示。大多数升级问题都来自于损坏的数据。修复此问题的唯一方法就是修复或移除已损坏的数据。不幸的是，这意味着在修复问题后，您不得不重新启动升级。这将使您的升级过程变得冗长，这里介绍的技巧可以帮助您避免重新启动，我们将为您展示哪些区域您可以安全的对您的数据库进行快照并设置新的重置点；仔细的为他们贴上标签，作为您的历史记录。

Liferay DXP的进步之一是把实体“核心”和一系列“模块”进行分离。这使我们能够采取更小的增量升级，帮助分流和解决数据相关问题。

独立的核心升级

注意：分离和升级门户核心和模块只应作为调试的一部分，创建数据清除脚本，并在非生产环境内进行测试。我们通常建议对生产环境数据库的副本进行测试升级，并希望在生产环境进行全面升级之前解决所有问题。

您可以配置门户只升级核心，而不升级模块，通过在osgi/configs/ folder 中添加名为com.liferay.portal.upgrade.internal.configuration.ReleaseManagerConfiguration.cfg的文件，内容为：

```
autoUpgrade=false
```

核心升级分版本，您可以使用portal-ext.properties升级运行。

```
upgrade.processes.master=\
    com.liferay.portal.upgrade.UpgradeProcess_6_0_12_to_6_1_0\,\
    com.liferay.portal.upgrade.UpgradeProcess_6_1_1\,\
    com.liferay.portal.upgrade.UpgradeProcess_6_2_0\,\
    com.liferay.portal.upgrade.UpgradeProcess_7_0_0\,\
    com.liferay.portal.upgrade.UpgradeProcess_7_0_1
```

您还应该禁用执行VerityProcess，通过设置门户属性：

```
verify.frequency=0
```

所有的升级都成功完成后，您可以重新激活VerifyProcess。如果您是6.2以前的版本，您可以尝试通过upgrade.process.master属性配置升级过程的增量测试。您可以将数据库还原到升级之前的版本，当然，这只能在非生产环境中测试您的升级。一旦您成功测试v和修复了所有关于数据完整性的问题，您应该执行前面讨论过的升级过程。

如果问题在模块升级过程中出现，您需要手动配置升级工具执行升级过程。完成核心升级并进行快照。如果您遇到问题，您可以从这里重新开始。

OSGi模块可以单独更新。如果成功执行的话，可以提供安全的快照。

为使用更深入的技术，您可以尝试使用调试器。它的好处在于，在某些情况下，您可以在内存里修复损坏的数据并保存在数据库中，修复您的数据而不需要重新启动。这需要对数据库表有较高水平的知识背景，只推荐经验丰富的Liferay开发人员。这项技术也开辟了在模块升级过程中的新选项，因为OSGi在过程中已经升级。您可以使用调试器在任意一步停止，并进行数据库快照。

升级模块

为了运行模块升级，您可以使用Gogo Shell。

1. 通过执行localhost 11311执行shell。
2. 在升级的命名空间使用可用的命令，例如：
    a. upgrade:list

b. upgrade:execute

c. upgrade:check

d. verify:list

e. verify:execute

输入upgrade:list，控制台将显示可以升级的模块，因为已涵盖所有升级依赖项。

如果您没有看到任何的模块，那是因为我们需要先升级依赖项。您可以输入命令scr:info {upgrade_qualified_class_name} 来检查哪些依赖项未处理。例如：

- *scr:info com.liferay.journal.upgrade.JournalServiceUpgrade*

输入upgrade:list {module_name}，控制台将显示升级您的模块所需要的步骤，为了理解是如何工作的，查看一下示例，可以了解它是如何工作的；如果您执行该命令的书签服务模块，您将会看到：

```
Registered upgrade processes for com.liferay.bookmarks.service
1.0.0
       {fromSchemaVersionString=0.0.1,
toSchemaVersionString=1.0.0-step-3, upgradeStep=com.liferay.
bookmarks.upgrade.v1_0_0.UpgradePortletId@497d1106}
       {fromSchemaVersionString=1.0.0-step-1,
toSchemaVersionString=1.0.0, upgradeStep=com.liferay.bookmarks.
upgrade.v1_0_0.UpgradePortletSettings@31e8c69b}
       {fromSchemaVersionString=1.0.0-step-2,
toSchemaVersionString=1.0.0-step-1, upgradeStep=com.liferay.
bookmarks.upgrade.v1_0_0.UpgradeLastPublishDate@294703b6}
       {fromSchemaVersionString=1.0.0-step-3,
toSchemaVersionString=1.0.0-step-2, upgradeStep=com.liferay.
bookmarks.upgrade.v1_0_0.UpgradeClassNames@7544b6e5}
```

也就是说从0.0.1版本的书签升级到1.0.0版本有可行的过程。完成这段升级，你需要进行四步操作。第一步在原初始版本开始，在目标版本结束（比如说最后一步），这个例子中的UpgradePortletId。最后一步在目标版本(1.0.0)开始和结束，(比如例子中的第一步)，这个例子中的UpgradePortletIdSettings。

输入upgrade:execute {module_name}，您将升级模块，考虑到如果在此过程中发生错误，则可以从最近执行的步骤重新开始，而不用再次执行整个过程。您可以通过执行upgrade:list {module_name} 来检查升级状态。

在Gogo shell 中输入upgrade:check，会显示哪个模块没有完成最终升级。因此您可以确定哪些模块的升级以失败告终。

为了理解命令行是怎么工作的，请参考这个例子：假设模块com.liferay.dynamic.data.mapping.service在步骤1.0.0-step-2 升级失败，在此时，如果您执行upgrade:check 的命令将得到：

```
Would upgrade com.liferay.dynamic.data.mapping.service from 1.0.0-
step-2 to 1.0.0 and its dependent modules
```

这意味着您需要修复此问题然后再次执行升级模块。请注意，一旦第一个升级完成，com.liferay.dynamic.data.mapping.service依赖项模块也需要升级。

另外，您可以执行verify:list执行验证过程。此项检查可以应用到所有验证程序中，执行verify:execute {verify_qualified_name} 并运行。

升级代码

最后，升级您的代码到Liferay DXP。如果您团队里有多名成员，这部分可以与数据升级同时开始。

重大变化

升级代码中最难的部分是了解新产品发生哪些变化。对于Liferay DXP来说，我们已经把变化记录在文档中，您可以查阅此链接。按照时间顺序列出现重大功能的变化，API或与第三方Liferay开发人员或用户签订的合同。在此文件中文档的类型更改包括：

- 被删除或替换的功能。
- 不兼容的API ——Java或JavaScript 公有API变更。
- 模板上下文中的可用变量的变更。
- Liferay主题和Portlet的可用CSS类的变更。
- 配置变更——在配置文件中的变更，如 portal.properties, system.properties等。
- 运行环境需求——Java 版本，J2EE 版本，浏览器版本等。
- 弃用或结束技术支持——例如，警告，在即将发布的版本中，将舍弃某项功能或API。
- 建议——例如，建议使用最新推出的API代替旧的API，尽管旧的API会保存在Liferay中并可以向上兼容。

熟悉自己的文档并理解您的代码库的更改很重要。我们推荐您阅读文档来了解Liferay DXP在未来版本中将如何持续发展。

查阅Liferay DXP的重大改变，请参阅此清单。

Liferay命令行工具

作为DXP的一部分，Liferay投入巨资改善开发人员开发人员体验。改善之一就是把Gradle作为主要的创建架构。

Liferay Workspace

在Liferay DXP中，我们推荐您迁移到我们新的项目架构中，叫做Liferay Workspace。它是基于Gradle的，并支持基础的Plugins SDK项目。它还利用建立在Liferay DXP中所有新的主题工具，并与Liferay Developer Studio整合。

您的新项目将会像以下这样：

- project-folder
  - modules
  - plugins-sdk
  - themes
- 如果您的项目使用Plugin SDK，您可以移动您的Plugin SDK作为Liferay Workspace中的文件夹之一。请牢记，Plugin SDK只支持创建WAR，而不能创建OSGi包。也不推荐在Plugin SDK中使用主题支持。

Liferay Blade

在Workspace中，我们有一个新的命令行工具叫做Liferay Blade。使用它，您可以创建应用程序，扩展等。就像您使用SDK插件一样，但是它有更多的功能。当您做出更改时，Blade可以启动您的Liferay服务器或实时自动部署您的项目。它是创建新的Liferay Workspace的方式。

安装

OS X 或 LINUX

```
curl https://raw.githubusercontent.com/liferay/liferay-blade-cli/
master/installers/global | sudo sh
```

WINDOWS

首先安装 JPM访问 JPM4J [Windows installation] 设置指南：
https://www.jpm4j.org/#!/md/windows.

```
jpm install com.liferay.blade.cli.jar
```

用法

```
blade init folderName
cd folderName
```

解压新的Liferay7.0插件到您新的Wrokspace中并命名为 plugins-sdk，您可以通过现有的插件复制到新的sdk中。

迁移6.2 WAR 到Liferay DXP支持的WAR

首先必须要将6.2 Portlet升级到Liferay DXP Portlet。即使您打算将您的Portlet转换成OSGi模块，首先我们推荐您升级旧版WAR到由Liferay DXP支持的。如果您直接从旧版6.2 WAR 升级到DXP模块，会很难调试出哪些问题与API相关，哪些问题与升级迁移有关。

当您复制Portlet到Liferay 7.0 Plugin SDK 实例后，使用升级助手查找在Portlet中的的重大改变，并相应地更新它们。请确保更新将依赖更新为Liferay DXP的依赖。（例如ivy.xml, liferay-plugin-package.properties, 等）。

当完成升级流程并且有Liferay DXP支持的WAR时，您可以决定是否把您的Portlet转换成OSGi模块，我们列出以下情景，可以帮助您做出决定。

把Portlet 转换到OSGi 模块

既然我们已经创建了Workspace和升级了您的Portlet到Liferay DXP 支持的WAR，所以我们考虑到如果您想要把Portlet转换成OSGi模式。

以下情况您应该进行转换：

- 您有一个大型的应用程序并有很多行代码。例如，有很多开发人员在此应用程序上同时进行协作并频繁进行更改，将代码分离成模块可以提高效率，并使发布更加敏捷和频繁。
- 您的插件里有其他地方可以复用的部件。例如，假如有业务逻辑需要复用在不同项目中，需要做的不是将代码复制到几个不同的WAR中并部署这些WAR 到不同的地方，您可以将应用程序转换成模块，并在其他模块中调用这些模块提供的服务。

以下情况您不能进行转换：

- 您有一个与 JSR-168/286兼容的Portlet，并且您仍然想要把它部署到其他Portlet Container中，如果您想保留兼容性，推荐您保留传统WAR模式。
- 您正在使用一种复杂的旧版Web框架，严重依赖于Java EE的编程模式，必须通过OSGi进行工作的工作量将比您期望的要多。
- 您的插件与其他的JEE应用程序服务器进行交互，例如，EJB，消息驱动Bean等等，基于模块的应用程序与应用程序服务器进行交互时是不方便的。
- 您旧版的应用程序有有限的生命周期。

如果您决定转换Portlet到OSGi模块，我们将为您提供方法。

Portlet：

```
blade create [APPLICATION_NAME]
```

使用Service Builder的Portlet

```
blade create -t servicebuilder -p [ROOT_PACKAGE] [APPLICATION_
NAME]
```

首先您会注意到的就是现在使用标准Maven项目结构。

- plugin-name
  - src

- main
  - java
  - resources
    - content
    - META-INF
      - resources
        - css
          - *.scss
        - *.jsp
- bnd.bnd
- build.gradle

在workspace中，bnd.bnd文件是非常重要的，因为它会自动在Gradle项目中应用liferay-gradle-plugin。liferay-gradle-plugin将应用 Gradle Java插件与其他非常有用的Liferay插件像 css-builder，source-formatter和lang-builder。

您将注意到您不需要 portlet.xml/liferay-portlet.xml 文件。该文件的内容应移入 Portlet Java 类中。

```
@Component(
  immediate = true,
  property = {
    “com.liferay.portlet.display-category=category.sample”,
    “com.liferay.portlet.icon=/icon.png”,
    “javax.portlet.name=1”,
    “javax.portlet.display-name=Tasks Portlet”,
    “javax.portlet.security-role-ref=administrator,guest,power-
user”,
    “javax.portlet.init-param.clear-request-parameters=true”,
    “javax.portlet.init-param.view-template=/view.jsp”,
    “javax.portlet.expiration-cache=0”,
    “javax.portlet.supports.mime-type=text/html”,
    “javax.portlet.resource-bundle=content.Language”,
    “javax.portlet.info.title=Tasks Portlet”,
    “javax.portlet.info.short-title=Tasks”,
    “javax.portlet.info.keywords=Tasks”,
  },
  service = Portlet.class
)
public class TasksPortlet extends MVCPortlet {
```

如果您创建一个服务生成器模板，您将看到生成三个不同的模板。

plugin-name-api - 这是服务接口将要连接的地方。
plugin-name-service - 这是服务实现的地方。
plugin-name-web - 这是您的Portlet本身。

这也将鼓励您进行模块化插件的开发。

主题升级

在Liferay DXP中，我们已经创建了一套新的主题工具，前端开发人员应该非常熟悉，使用Node js，yo和gulp。如果在您的Plugin SDK中现有主题，您可以将其迁移到新的Liferay Workspace。

升级主题：

```
blade migrateTheme [THEME_NAME]
```

迁移所有主题使用：

```
blade migrateTheme -a
```

您的主题将被迁移到一个新的主题文件夹，接下来我们需要转换CSS样式从Bootstrap 2转换到Bootstrap 3。我们需要利用Liferay创建另一个新的公共管理包。

接下来您要做的是重命名您所有的SASS文件，把*.css改成*.sass。在Liferay DXP中，SASS文件使用正确的扩展名，SASS编译器将只编译.scss文件扩展名的文件。

通过运行安装它：

```
npm install -g convert-bootstrap-2-to-3
```

然后您可以通过运行文件进行升级：

```
bs3 path/to/file
```

您可以在运行HTML文件、CSS文件和SASS 文件之前运行它，此工具也许不能解决所有问题，但是它会给您的业务带来很好的起点。其余的就取决于实施。

附加资源

升级到Liferay Portal 6.2 EE

升级到Liferay DXP

创建Liferay Workspace

主题任务文件

Bootstrap 转换文档

总结

对于那些选择继续探索的公司来说，升级到最新版本的Liferay DXP平台可以带来全方位的好处。但每个企业在权衡成本、风险和时间、劳动力和涉及的商业利益后需自行决定是否将升级作为业务方向。

了解更多

我们Liferay全球服务团队可以帮您制定个性化的升级计划并为您企业提供Liferay内部成功经验，了解更多关于Liferay数字化体验平台和咨询服务，请联系 sales-cn@liferay.com。

Liferay的产品帮助企业在网页端、移动端和连接设备上创建数字化体验。Liferay平台是开源的，因此它是更加可靠、创新和安全的。客户包括家乐福、Coach、达能、富士通、汉莎航空培训、西门子、法国兴业银行、VMware和联合国都在使用Liferay。访问我们请点击liferay.com。

171024