链滴

Solo 跨版本升级

作者：88250

原文链接：https://ld246.com/article/1560169064267

来源网站：链滴

本文是《Solo 从设计到实现》的一个章节，该系列文章将介绍 Solo 这款 Java 博客系统是如何从无有的，希望大家能通过它对 Solo 从设计到实现有个直观地了解、能为想参与贡献的人介绍清楚项目也希望能为给~~重复发明~~重新定义博客系统的人做个参考
eart

自 Solo v3.0.0 起开始支持跨版本升级，用户直接部署最新版就可以从某个老版本进行升级，降低用的升级成本。同时，跨版本升级机制也让大大降低了实现自动升级的难度，通过简单的脚本和 crontab 定时任务就可以实现自动升级。

升级检查

Solo 在每次启动时都会进行升级检查，调用入口在 SoloServletListener#contextInitialized 方法，现为 UpgradeService#upgrade 方法：

```
/**
 * Upgrades if need.
 */
public void upgrade() {
    try {
        final JSONObject preference = optionQueryService.getPreference();
        if (null == preference) {
            return;
        }

        final String currentVer = preference.getString(Option.ID_C_VERSION); // 数据库中的版本
        if (SoloServletListener.VERSION.equals(currentVer)) {
            // 如果数据库中的版本和运行时版本一致则说明已经是最新版
            return;
        }

        // 如果版本较老，则调用对应的升级程序进行升级，并贯穿升级下去直到最新版
        switch (currentVer) {
            case "2.9.9":
                V299_300.perform();
            case "3.0.0":
                V300_310.perform();
            case "3.1.0":
                V310_320.perform();
            case "3.2.0":
                V320_330.perform();
            case "3.3.0":
                V330_340.perform();
            case "3.4.0":
                V340_350.perform();
            case "3.5.0":
                V350_360.perform();
            case "3.6.0":
                V360_361.perform();
            case "3.6.1":
                V361_362.perform();

                break;
            default:
```

```
                LOGGER.log(Level.ERROR, "Please upgrade to v3.0.0 first");
                System.exit(-1);
            }
        } catch (final Exception e) {
            LOGGER.log(Level.ERROR, "Upgrade failed, please contact the Solo developers or reports
this "
                    + "issue: https://github.com/b3log/solo/issues/new", e);
            System.exit(-1);
        }
    }
```

升级程序必须在业务逻辑开始前执行，因为 MySQL 为了保证数据一致性，有一个叫做“元数据锁”机制，不允许表结构变更（DDL）和其他数据事务操作（DML）一起进行。

升级实现

所有版本的升级程序均放置在包 org.b3log.solo.upgrade 下，每个升级程序实现要点如下：

- 如果需要变更表结构则进行变更
- 如果需要迁移数据则进行迁移
- 更新 option 中的 version 为最新版本

最简单的情况是只需升级 option 的 version，比如 v3.6.0 到 v3.6.1 的升级程序实现：

```
/**
 * Performs upgrade from v3.6.0 to v3.6.1.
 *
 * @throws Exception upgrade fails
 */
public static void perform() throws Exception {
    final String fromVer = "3.6.0";
    final String toVer = "3.6.1";

    LOGGER.log(Level.INFO, "Upgrading from version [" + fromVer + "] to version [" + toVer +
"]....");

    final BeanManager beanManager = BeanManager.getInstance();
    final OptionRepository optionRepository = beanManager.getReference(OptionRepository.c
ass);

    try {
        final Transaction transaction = optionRepository.beginTransaction();

        final JSONObject versionOpt = optionRepository.get(Option.ID_C_VERSION);
        versionOpt.put(Option.OPTION_VALUE, toVer);
        optionRepository.update(Option.ID_C_VERSION, versionOpt);

        transaction.commit();

        LOGGER.log(Level.INFO, "Upgraded from version [" + fromVer + "] to version [" + toVer
 "] successfully");
    } catch (final Exception e) {
        LOGGER.log(Level.ERROR, "Upgrade failed!", e);
```

```
        throw new Exception("Upgrade failed from version [" + fromVer + "] to version [" + toVer
+ "]");
    }
}
```

自动升级

使用 Docker 部署的话可以非常方便地实现自动升级，大致的脚本逻辑如下：

1. docker pull b3log/solo 拉取最新镜像
2. docker run 重启容器

我们每次发布版本都会构建好最新的 docker 镜像，用户可以通过 crontab 将升级脚本配置为定时执来实现自动更新。升级脚本可以参考这里进行编写。

如果你使用 war 包部署要实现自动升级就稍微有点麻烦了，大致的逻辑如下：

1. 通过社区接口检查是否有新版本 https://rhythm.b3log.org/version/solo/latest
2. 如果有新版本则下载 war https://github.com/b3log/solo/releases
3. 部署 war，配置文件可用脚本替换，更为简单的做法是通过环境变量 $LATKE_PROPS 和 $LATK_LOCAL_PROPS 来指定配置文件路径，这样就不用替换了

总之，推荐使用 Docker 进行部署，运维起来非常方便。