「S2Chronos」のご紹介とその使い方

じゅんいち☆かとう(id:j5ik2o)

アジェンダ

- 自己紹介
- 今回のねらい
- S2Chronosとは？
- 現状の課題と課題解決とは？
- S2Chronosのコンセプト
- S2Chronosでの実装とデモ
- 今後のロードマップ

自己紹介

- ID
  - http://d.hatena.ne.jp/j5ik2o
  - j5ik2o@gmail.com
- はじめてのコーディング
  - FM-New7上でBASICでお絵かきソフトを書いた
- はじめてのSeasar
  - S2JSF, S2Dao
- そして今では…
  - Teeda, S2Dao, SAStruts, Mayaa, S2JDBC, S2Chronos, S2Config, S2Mai, S2Buri…

自己紹介

- 最近の活動
  - S2Chronos
    - 今からプレゼンするネタです。
  - S2Config
    - 環境に依存する設定情報を外部ファイルで扱うためのフレームワークです。
  - Jiemamy Project
    - 都元ダイスケ氏のプロジェクトです。
  - S2Container
    - 勢いでS2JUnit4のpostBindFields/preUnbindFieldsの実装。

今回のねらい

- **S2Chronosを知ってもらう**
- **S2Chronosに興味、関心を持ってもらう**
- **S2Chronosを試してもらう**
- S2Chronosを実際に使ってもらう

S2Chronosとは？

S2Chronosとは？

- S2Chronos
  - http://s2chronos.sandbox.seasar.org/

- 「ある時間」に関連する「業務ロジック」を「簡単に」作って「実行する」ためのフレームワーク。
  - 要するに「バッチ処理」を実装するためのフレームワーク

現状の課題とその課題解決とは？

【現状の課題】バッチ処理の実装方法

- UNIX/Linux
  - cron
  - atコマンド
- Windows
  - タスクスケジューラ
  - atコマンド
- Java
  - quartz
  - EJB Timer

【現状の課題】 cron編

```
public class BatchMain {
    public static int main(String[] arg) {
        SingletonS2ContainerFactory.init();
        S2Container s2Container =
    SingletonS2ContainerFactory.getContianer();
        UserDao userDao =
    s2Container.getComponent(UserDao.class);
        List<User> userList = userDao.selectAll();
        exportCSV("/var/lib/hoge/userList",userList);
        SingletonS2ContainerFactory.destory();
    }
}
```

- crontab –e で月末にBatchMain#mainが起動するように設定する。

【現状の課題】 cron編

- メリット
  - CPUやメモリの消費を軽減できる。
- デメリット
  - ブートストラップコードを書くのは手間。
  - 起動時間の条件はOS依存なるため、動作確認が面倒。
  - 同一プログラムで複数のバッチ処理を管理する場合は、ロジックの切り分けが必要。
  - バッチ処理だけに別プロジェクトを用意するのが面倒。使いたいビジネスロジックはウェブアプリのプロジェクトにある。

【現状の課題】 quartz編

```
public class Batch {
    public static void main(String[] arg){
        SchedulerFactory factory = new
    StdSchedulerFactory();
        Scheduler scheduler = factory.getScheduler();
        scheduler.start();
        JobDetail jobDetail = new
    JobDetail("myJob", null,
    ActiveMemberJob.class);
        Trigger trigger = new
    CronTrigger(“myTrigger”, “group1”, “0 */1 * *
    * ?”);
        sched.scheduleJob(jobDetail, trigger);
    }
}
```

【現状の課題】 quartz編

```
public class ActiveMemberJob implements
  Job {
  public void execute(JobExecutionContext
  context) throws JobExecutionException {
      UserDao userDao =
  SingletonS2Container.getComponent(U
  serDao.class);
  List<User> userList =
  userDao.selectAll();
  exportCSV("/var/lib/hoge/userList",us
  erList);
  }
}
```

【現状の課題】 quartz編

- メリット
  - cronとは違い、スケジューラもJava実装。
  - バッチ処理をクラス単位で実装して、スケジューリング可能。
  - スケジューラとバッチ処理をウェブアプリと同一プロジェクトにすることが可能。
- デメリット
  - cronに比べるとリソースは消費する。
  - 登場するクラスが直感的でない。
  - S2とシームレスな連携ができない。
  - POJO指向ではない。

【課題解決】課題解決の方向性

評価軸	現状の課題		課題解決の方向性
	cron	quartz	
システムリソース消費	○	△	○
ブートスラップコードの最小化	△	△	○
スケジューラのJava実装	×	○	○
クラス単位のスケジューリング	△	○	○
シンプルなクラス構成	△	△	○
POJO対応	×	×	○
S2コンテナとの親和性	△	△	○
ウェブアプリケーションへの内包	×	○	○

S2Chronosのコンセプト

S2Chronosのコンセプト

コンセプト	特徴
ブートストラップコードの最小化/ウェブアプリケーションへの内包	・スケジューラは3行で起動可能 ・extensionを使えば、ウェブアプリケーションへの組み込みも簡単
スケジューラのJava実装	・Javaで実装
タスク単位のスケジューリング	・個別に起動条件を設定可能
シンプルなクラス構成	・基本的にはタスククラスとトリガークラスだけを使う
POJO対応	・タスククラスはPOJOに対応
S2コンテナとの親和性	・S2.4対応 ・SMART deployによるタスクの自動登録 ・HOT deployにも対応
その他	・例外処理 ・スレッドプール ・タスクフロー ・タスクグループ

S2Chronosでの実装とデモ

【実装】コンソールアプリ編

```
public class BatchMain{
    public static void main(String[] args) {
        SingletonS2ContainerFactory.init();

        SingletonS2Container.getComponent(Schedul
    er.class).process();
        SingletonS2ContainerFactory.destroy();
    }
}
```

【実装】コンソールアプリ編

```
@Task
@CronTrigger("0 */1 * * * ?")
public class ActiveMemberTask {
   public UserDao userDao;
   public void doExecute() {
       List<User> userList = userDao.selectAll();

       exportCSV(“/var/lib/hoge/userList”,userLi
   st);
   }
}
```

【実装】ウェブアプリ編

- s2chronos-extensionをビルドパスに通す。
- web.xmlに以下を追加して、先ほどのタスククラスを書くだけ。

```
<servlet>
    <servlet-name>chronosServlet</servlet-name>
     <servlet-class>
     org.seasar.chronos.extension.servlet.S2ChronosServlet
     </servlet-class>
     <load-on-startup>3</load-on-startup>
</servlet>
```

デモ

- デモ
  - シンプルなバッチ処理
  - 非同期なバッチ処理
  - タスクフロー
  - 例外処理

今後のロードマップ

今後のロードマップ

- 1.0
  - s2chronos-core
  - s2chronos-extension
- 1.0.1
  - s2chronos-core
    - イベント通知系の精査
  - s2chronos-extension
    - 永続化対応
    - 監視用I/Fの実装

ご静聴ありがとうございました