

Обзор Cronjob

В SAP Hybris Commerce cronjobs играют важную роль в автоматизации различных задач и процессов в платформе электронной коммерции. Cronjobs используются для планирования и выполнения повторяющихся фоновых заданий, которые могут варьироваться от синхронизации продуктов и обработки заказов до импорта и экспорта данных. Вот обзор cronjobs в SAP Commerce:

Что такое Cronjob: Cronjob – это запланированная задача или фоновое задание, которое запускается через определенные интервалы или время. Название «cronjob» происходит от утилиты Unix/Linux cron, которая используется для планирования задач. В SAP Commerce cronjob управляются и планируются с помощью встроенного в платформу механизма cronjob.

Типы заданий cronjobs: SAP Commerce предоставляет множество предопределенных заданий cronjobs для общих задач электронной коммерции, и разработчики могут создавать собственные задания cronjobs для удовлетворения конкретных бизнес-требований. Некоторые распространенные типы заданий cronjobs включают:

- 1) Задания Cron для синхронизации продуктов: они используются для синхронизации данных о продуктах из внешних источников или систем.
- 2) Задания Cron по обработке заказов: они выполняют задачи, связанные с обработкой заказов, такие как проверка заказов и обновление статуса заказов.
- 3) Задания Cron по импорту/экспорту данных: используются для импорта и экспорта данных в систему электронной коммерции и из нее.
- 4) Задания Cron для отправки электронной почты: запланированные задачи для отправки уведомлений по электронной почте, информационных бюллетеней и маркетинговых кампаний.
- 5) Задания Cron по очистке и обслуживанию: отвечают за выполнение рутинных задач по обслуживанию, таких как очистка журналов или удаление устаревших данных.
- 6) Пользовательские задания CronJob: разработчики могут создавать пользовательские задания CronJob для автоматизации определенных бизнес-процессов.

Компоненты Cronjob

Cronjob – это важный компонент, используемый для автоматизации различных повторяющихся задач и процессов. Понимание важных компонентов cronjob имеет решающее значение для эффективного создания и управления этими запланированными заданиями. Вот основные компоненты cronjob в Hybris:

1) Модель Cronjob:

Назначение: Модель cronjob определяет метаданные для cronjob, такие как его имя, код и конфигурация.

Компоненты:

Код: Уникальный идентификатор cronjob.

Имя: Описательное имя cronjob.

Конфигурация задания: Информация о задаче, которая должна быть выполнена, включая код задачи, сведения о расписании и другие параметры.

Статус: Указывает, включено или отключено cronjob.

2) Конфигурация задания:

Назначение: Эта часть конфигурации cronjob определяет фактическую задачу, которая будет выполнена при запуске cronjob.

Компоненты:

Код задачи: Код, который идентифицирует задачу, которая будет выполнена. Этот код соответствует определенному классу Java.

Подробности задания: Дополнительная конфигурация, связанная с задачей, такая как входные параметры или настройки, необходимые для выполнения задачи.

Планирование: Определяет, когда и как часто должен запускаться cronjob. Сюда входит выражение cron, которое определяет расписание выполнения, и такие параметры, как однократное выполнение или повторяющееся выполнение.

3) Реализация задачи:

Назначение: Реализация задачи – это класс Java, отвечающий за выполнение фактической работы при запуске cronjob.

Компоненты:

Класс Java: Класс Java, который реализует интерфейс Performable<CronJobModel> или расширяет AbstractJobPerformable<CronJobModel>. Этот класс содержит логику для задачи.

Зависимости служб: Если задача зависит от служб или компонентов, они должны быть внедрены в реализацию задачи.

Этапы создания Cronjob

1) Определите задачу Cronjob: Определите конкретную задачу, которую будет выполнять ваш cronjob. Это может быть синхронизация данных, отправка электронной почты, операция очистки или любая другая повторяющаяся задача.

2) Создание пользовательской модели CronJob: Cronjobs определяются как модель в Hybris. Нам нужно добавить некоторые дополнительные параметры в готовую модель cronjob, затем мы создаем пользовательскую модель cronjob, как показано ниже:

```
1.   <типы элементов>
2.     <itemtype generate="true" code="CustomCronJob" extends="CronJob" autocreate="true">
3.       <атрибуты>
4.         <квалификатор атрибута="noofdaysold" тип="java.lang.Integer">
5.           <модификаторы необязательные="false"/>
6.           <тип сохранения="свойство" />
7.         </атрибут>
8.       </атрибуты>
9.     </тип_элемента>
10.   </itemtypes>
```

3) Определите класс службы для выполнения бизнес-задачи: напишите службу или задачу Java, которая выполняет работу, которую вы хотите, чтобы выполнял cronjob. Эта служба будет вызываться при запуске cronjob. Убедитесь, что эта служба следует соглашениям о службах Hybris.

```
публичный интерфейс MyCustomCronJobService {
void performTask();
}
```

4) Создайте компонент службы в XML: мы создадим класс компонента для указанного выше класса службы.

```
<bean id="myCustomJobService" class="de.hybris.cronjob.service.MyCustomCronJobService" />
```

5) Определим класс Job: Мы определим класс job , в который мы внедрим cronjobservice и выполним бизнес-логику. Мы реализуем класс abstractJobPerformable и переопределим метод perform.

```
1.   открытый класс MyCustomJob расширяет AbstractJobPerformable<CustomCronJob>
2.   {
3.     частный статический финальный Logger LOG = Logger.getLogger( MyCustomJob .class);
4.     @Resource (name=" myCustomJobService ")
5.     частный MyCustomCronJobService myCustomCronJobService;
6.     @Переопределить
7.     public PerformResult выполнить(final CustomCronJob cronJobModel)
8.     {
9.       myCustomCronJobService. performTask();
10.      вернуть новый PerformResult(CronJobResult.SUCCESS, CronJobStatus.FINISHED);
11.    }
12.  }
```

6) Определите Job в XML: Мы определим bean-компонент класса job, куда мы внедрим свойство cronjobservice и выполним бизнес-логику. Мы реализуем класс abstractJobPerformable и переопределим метод perform.

```
<bean id="myCustomJob" class="de.hybris.cronjob.job.MyCustomJob" parent="abstractJobPerformable">
  <имя_свойства=" myCustomJobService " ref=" myCustomJobService "/>
</боб>
```

7) Определим экземпляр Cronjob и триггер через импрекс: Мы создадим экземпляр модели customCronjob и свяжем его с соответствующим bean-компонентом задания. Мы создадим триггер customCronJob для планирования задания.

```
IN SERT_UPDATE CronJob; code[unique=true];job(код);singleExecutable; sessionLanguage(isocode)
;c ustomCronJob ; myCustomJob ;false;en
```

```
INSERT_UPDATE Триггер;cronjob(код)[уникальный=истина];cronExpression
;c ustomCronJob ; 0 0 0 * * ?
```

Об авторе

Пилюш Сингх — опытный энтузиаст технологий и преподаватель, страстно желающий сделать сложные концепции доступными для всех. Имея более чем 10-летний опыт работы в технологической отрасли в качестве консультанта, Пилюш специализируется на технологиях **Java** и **SAP Hybris** и обладает даром разбивать сложные темы на простые для понимания руководства.

Связаться с автором

[LinkedIn](#)

[Электронная почта](#)