REGIONE AUTONOMA DELLA SARDEGNA

SISaR

ENGINEERING

Servizio di Transizione

Titolo: XMPI/JBF/JAUF
Codice: AREAS_AP_01_JBF
Profilo: Analista Programmatore

REGIONE AUTONOMA DELLA SARDEGNA

SISaR

ENGINEERING

ARGOMENTI DI QUESTE SLIDE

- Demoni e Agents

Funzionalità demone / La webapp demone

L'applicativo non è costituito solo da maschere. Molte delle procedure applicative agiscono in maniera "invisibile" all'utente. Sono attività che lavorano in background per fornire funzionalità come:

- Invio di email
- Produzione di stampe
- Avanzamento di processi di Workflow
- Comunicazioni con sistemi esterni

Essendo spesso operazioni molto onerose, queste attività devono necessariamente girare su una webapp dedicata e non condivisa con la normale webapp a cui accedono gli utenti.

Si parla così di **webapp demone**.

In base alle esigenze del cliente ed al carico di lavoro, si potranno avere più o meno webapp demoni. La loro creazione viene effettuata in fase d'installazione tramite il tool ABFInstaller.

Funzionalità demone / Tipologie di demoni

La webapp demone a disposizione sarà solo il contenitore in cui gireranno le nostre attività in background. La scelta esatta di queste attività (cosa far girare) dipende da noi con l'opportuna configurazione sull'applicativo di oggetti "**demoni**".

Un demone (anche detto "**demone server**") è un oggetto che si occupa di compiere costantemente e senza interruzioni una particolare attività. Le possibili tipologie a disposizione di queste attività sono generalmente stabilite dal laboratorio di sviluppo. Ogni modulo può quindi rilasciare un nuovo tipo di demone.

I moduli JBF/XMPI si occupano del rilascio di queste tipologie:

- Demone per Workflow (WORKFLOW)
- Agent Manager (SI_AGENTS)
- Stampe schedulate (SI_SCHED_REPORT)
- Demoni di integrazione (SI_HL7, SA_G4GLSAC, SA_G4GLXMP, SA_ORION)

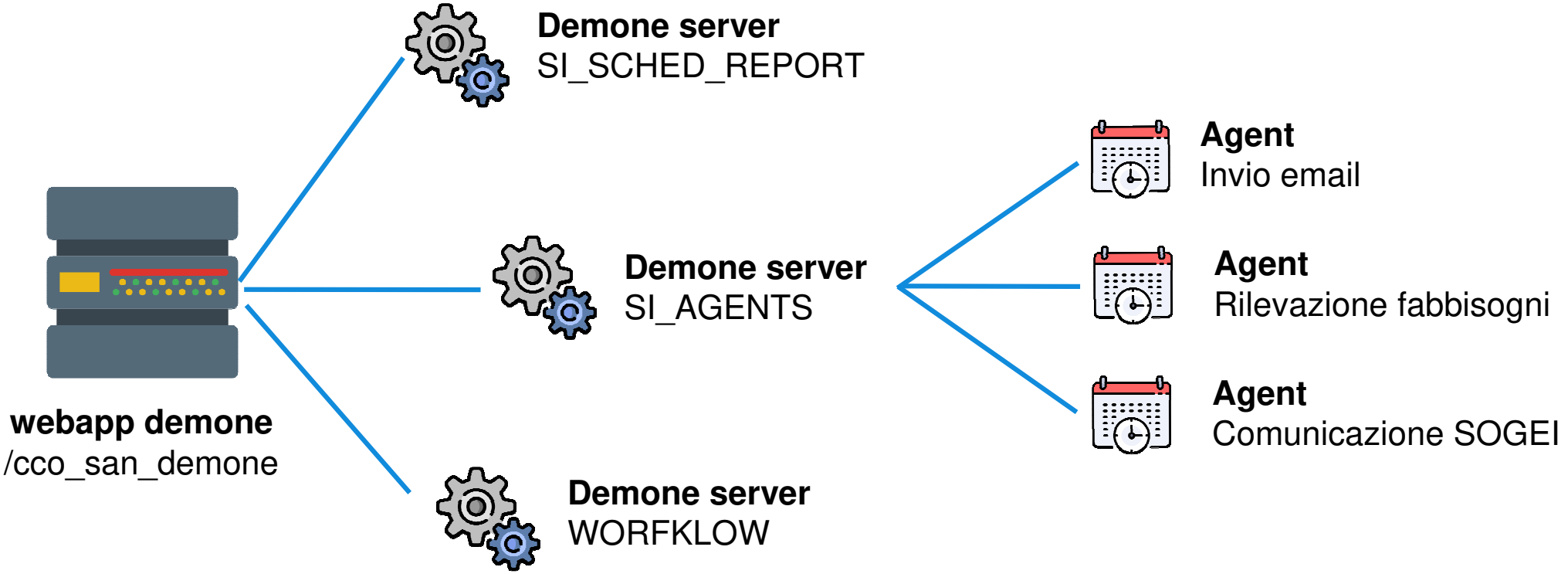
Funzionalità demone / Gli agent

Alcune attività in background non devono essere eseguite “costantemente”, ma devono invece lavorare ad intervalli regolari e dietro un’opportuna pianificazione. Un esempio può essere la necessità di inviare un’estrazione dati ad un sistema esterno ogni primo del mese.

Allo scopo vengono utilizzati gli **agent**. Gli agent sono delle “sub attività” che girano all’interno di un demone di tipo **SI_AGENTS** e che possono essere pianificate (o “schedulate”) per essere eseguite in determinati momenti della giornata, della settimana, del mese.

Il demone SI_AGENTS girerà costantemente ed avrà il compito di verificare quando, in accordo con la pianificazione, è il momento giusto per eseguire un agent.

Funzionalità demone / Struttura



Funzionalità demone / Esercitazione

La configurazione degli agent è sicuramente quella più diffusa nell'ambito dei demoni.

L'obiettivo è creare un nuovo demone di tipo SI_AGENTS a cui collegare dei nuovi agent da eseguire in intervalli diversi.

Sarà necessario:
1.Accedere alla gestione "Configurazione>>Sistema>>Demoni>>Demoni"
2.Censire un nuovo demone di tipo SI_AGENTS con nome "D_<cognome>". Prestare particolare attenzione all'utente e alla webapp che vengono associati
3.Accedere alla gestione "Configurazione>>Sistema>>Agent>>Attivazione"
4.Prendere in gestione il demone creato e selezionare il tab "Agents"
5.Attivare gli agent con il numero associato, provando una cadenza giornaliera, una settimanale ed una mensile.

Funzionalità demone / Start e stop di AREAS

A volte per completare una configurazione è necessario provvedere anche al riavvio di AREAS. Uno dei tanti esempi è proprio la configurazione di un nuovo demone. Per far si che la webapp demone "recepisca" il nuovo demone, è necessario il riavvio.

Se il demone era già configurato ed abbiamo semplicemente aggiunto un agent, non è necessario il riavvio.

Esistono due modalità possibili per "riavviare AREAS":

• Riavvio della webapp. È il sistema più semplice, in quanto si può fare tramite il pannello di amministrazione del tomcat.

• Riavviare l'intero application server. In questo caso occorrerà collegarsi tramite shell sulla macchina dove gira il tomcat e lanciare opportuni comandi.

Funzionalità demone / Start e stop della webapp

Per il riavvio della webapp occorre avere a disposizione due informazioni:

•URL del tomcat da riavviare

•Utente “**manager**” del tomcat

Prendiamo come esempio la webapp http://161.27.213.60:9109/corso/mainLogin.do

Il pannello di amministrazione del tomcat sarà allora accessibile digitando nel browser il seguente URL: http://161.27.213.60:9109/**manager/html**

Ci verranno richieste le credenziali dell’utente manager. Se non si è a conoscenza delle credenziali, occorre chiederle.

Il pannello mostrerà la lista delle webapp installate sul tomcat, individuiamo la webpp di nostro interesse e facciamo click su “**Stop**”. Attendiamo che il sistema risponda, e facciamo poi click su “**Start**”. Si sconsiglia l’utilizzo della funzionalità “Reload”.

Funzionalità demone / Start e stop del tomcat - 1

Le modalità di riavvio di un tomcat dipendono dal sistema operativo del server (Windows o Linux). Di seguito ci concentriamo sui sistemi Linux, largamente la maggioranza nel parco clienti di AREAS.

Per il riavvio del tomcat occorre avere:

- URL del tomcat da riavviare
- Credenziali SSH dell'utente che ha avviato il tomcat
- Un client SSH. Consigliamo di scaricare **putty** https://the.earth.li/~sgtatham/putty/latest/w32/putty.exe

Funzionalità demone / Uno sguardo al DB

Le tabelle dove abbiamo lavorato sono:

SI_DEMONESERVER – Tabella anagrafica dei demoni

```
SELECT * FROM SI_DEMONESERVER
```

SI_AGENTS – Tabella contenente la lista degli agent disponibili

```
---Query per conoscere gli agent associati ad un demone
SELECT * FROM SI_AGENTS WHERE SA_SERVER = 'codice_demone'
```

SI_WEBAPP – Tabella che mostra la lista di tutte le webapp disponibili

```
--Query per conoscere le webapp demoni disponibili
SELECT * FROM SI_WEBAPP WHERE WA_TIPO='D' AND WA_DISABLE IS NULL
```

Funzionalità demone / Esercitazione

Creiamo un nostro agent, abilitiamolo e vediamolo in esecuzione.

Dovremo innanzitutto creare una nuova classe che implementa l'interfaccia **PSGExt.agent.AgentInterface**. All'interno di questa classe, nel metodo .**run()**, andremo ad implementare la logica del nostro agent. La classe andrà poi censita all'interno della tabella **SI_AGENTS**.

Per monitorare un agent (SI_ESECUZIONI), occorre implementare l'interfaccia **PSGExt.agent.monitor.MonitorableAgentInterface**