RFP-193-Condition-Service

Final

7 Pages

Abstract

Declarative Services provide a nice and powerful method to give activation conditions for a Service or Component. The Condition Service attempts to extend this, in order to support more complex scenarios, that can’t be handled by the current mechanism.

0 Document Information

0.1 Table of Contents

0.2 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 6.1.

`Source code is shown in this typeface.`

0.3 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	*25.08.2018*	*Initial* *Jürgen Albert*
V01	*28.01.2019*	*Renamed to Condition Services* *Rework of the Requirements* *Jürgen Albert*
V02	*18.02.2019*	*Changes as discussed during the F2F in Berlin* *Jürgen Albert*

1 Introduction

Declarative Services are a comfortable and powerful tool with which managing the lifecycle of your services is easy and comfortable. The ability to define dependencies to other services together with target filters and cardinalities, make it unmatched by most other comparable mechanisms. Unfortunately the possibilities to set conditions, under which DS should activate or deactivate a Component are limited. Thus this RFP will explore an extension to DS allowing for more complex scenarios.

2 Application Domain

Components have a clearly defined Lifecycle. Unfortunately the conditions when a component should be activated and eventually registered as a service, can be more complex then the general availability of the directly used services. Thus a Mechanism would be useful to allow a more precise and complex definition of conditions surrounding the Lifecycle of a Component

2.1 Terminology + Abbreviations

2.1.1 Conditions

DS purely depends on References and there provided conditions to determine if a component can be activated. Conditions should set of rules that gives DS, CDI or Services a clue, when a component can be enabled or disabled that are independent of the references a service requires.

3 Problem Description

Services and Components are a great way, to represent processes in the real world or more artificial things like business processes.

DS already allows the developers to define a Cardinality to each service reference, that gives clues, when a service should be activated and deactivated. The Configuration Admin extends this capability, so target filters can be assigned at configuration time. Additionally the amount of services to inject, can be specified more precisely then in the component xml or with the Annotations.

This RFP will explore the Use cases that exceed the possibilities DS gives the developer. The mechanism is not necessarily in the direct domain of DS, but can be picked up by it.

All the Use cases that will follow, can be achieved somehow by using e.g. DS to activate a Component, that than uses Servicetrackers, to check if the special needs are fulfilled. This is often uncomfortable, and the principle is usually the same.

4 Use Cases

4.1 Service unavailable

Anton Develops a system, that is a on premise solution that a customer installs via Docker. Every installation specific configuration will be handled via a webfrontend after the first start. The system supports a couple of different login mechanisms, that utilize servlet filters. As long as no login filter is registered a default login filter must be available and only if no other login service is registered.

4.2 Conditions are only known at configuration but not at development time

Betty develops a Service that handles billing in a Webshop. The service consumes other services with the IBillingProvider Interface, that is available in a couple of implementations for e.g. Debit, VISA and PayPal. The Webshop is home to different tenents, which get the allowed billing services according to his individual payment plan. Thus Bettys Services will be configured at runtime according to the tenants account and must be activated only when e.g. the VISA and Debit is available.

4.3 Common indirect preconditions

Carl develops an RCP Application, where a lot of its components indirectly depend on the availability of the ExtensionRegistry and SWTApplication. He wants that his components only get enabled and activated, when this services are up and running.

4.4 ConfigurationPlugins for ConfigurationAdmin

Dorothy has a Configration Plugin for the ConfigurationAdmin, that sets certain passwords from a secure storage to internal configurations. In order to make her System work, she must make sure, that the ConfigurationAdmin waits to apply her configurations until her Plugin is registered.

5 Requirements

5.1.1 General

- G010 – It MUST provide a marker service that can be used as a mandatory injecte for any component.
- G020 – Everybody must be able to register its own Condition.
- G030 – There MUST always be a registered True Condition.
- G040 – Every registered Condition MUST have a persistent identifying property, that can be targeted by other components.
- G060 - The Condition must be supported in all injection models.
- G070 – The state of Condition and its changes MUST be observable.

5.1.2 Configuration of Conditions

- C010 – The Solution MUST provide a mechanism to create and program conditions via configuration.
- C020 – It MUST define a domain specific language to program Conditions. It must support boolean logic and cardinality.
- C030 – The configurable component MUST provide a DTO showing the state of its requirements.

5.1.3 Declarative Services

- DS010 – DS MUST support a mechanism to bind the enablement of a component to a Condition.
- DS020 - Every DS Component by default must depend on the True Condition of G030.
- DS030 – CDI MUST meet the Requrements DS010 and DS020 as well.

6 Document Support

6.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

6.2 Author's Address

Name	Jürgen Albert
Company	Data In Motion Consulting GmbH
Address	Kahlaische Str. 4, 07743 Jena, Germany
Voice	+49 1577 2521634
e-mail	j.albert@data-in-motion.biz

6.3 End of Document