RFP 150 - Http Service Updates

Abstract

The current Http Service specification is based on Servlet API 2.1. As such it misses newer functionality such as Servlet Filters or event listeners. In addition use of the service does not support the recent whiteboard pattern approach. This RFP lists requirement to update the Http Service specification as well as possible create new specification for extended Web Applications in the context of OSGi.
0 Document Information

0.1 Table of Contents

0 Document Information................................................................................................................. 2
  0.1 Table of Contents.................................................................................................................. 2
  0.2 Terminology and Document Conventions............................................................................... 2
  0.3 Revision History..................................................................................................................... 2

1 Introduction................................................................................................................................. 3

2 Application Domain..................................................................................................................... 3
  2.1 Terminology + Abbreviations............................................................................................... 4

3 Problem Description................................................................................................................... 4

4 Use Cases................................................................................................................................... 4

5 Requirements.............................................................................................................................. 4

6 Document Support..................................................................................................................... 5
  6.1 References............................................................................................................................ 5
  6.2 Author’s Address................................................................................................................... 5
  6.3 End of Document.................................................................................................................... 5

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.
1 Introduction

The OSGi Specifications currently only contain limited specification support for creating Web Applications in an OSGi context:

- **Http Service Specification** based on Servlet API 2.1. Apart from being based on an old Servlet API version and being silent about how more recent versions are supported, the main problem with this specification is that a provider of servlets and resources has to grab the Http Service first before being able to register servlets and resources. There is no whiteboard pattern support.

- **Web Applications Specification** basically just defines how existing web applications may be enhanced with OSGi Manifest headers and deployed into the OSGi Framework as-is. This is fine for moving existing web applications with minimal changes into the OSGi framework.

Some thoughts are already listed on the OSGi Community Wiki at http://wiki.osgi.org/wiki/WebExperience.
2 Application Domain

Developers need to use the full extend of current Servlet API specifications (as of this writing Servlet API 3.0 is the most recent version). As such there is a need to register servlet filters and event listeners.

2.1 Terminology + Abbreviations

3 Problem Description

3.1 Support for dated Servlet API 2.1

Current support for web applications using the Http Service in traditional OSGi based applications is limited to servlets and resources. From the current Servlet API 3.0 specification the following functionality is missing:

- Servlet Filters
- Servlet Event Listeners
- Asynchronous Requests

At this moment some of this missing functionality is covered in a proprietary way. Examples are the Apache Felix Http Whiteboard support or the OPS4J Pax Web collection of bundles.

3.2 Dependency on the HttpService service

Currently the HttpService service (or one of them if multiple services exist in a framework) must be accessed to be able to register servlets and/or resources. In addition to register a servlet or resource an instance of the HttpContext interface is required.

This makes it very cumbersome to easily register servlets and resources. Particularly it is hard to come up with an HttpContext instance which for example uses an authentication mechanism available in the framework to implement the handleSecurity method.

To reduce (or simplify) this dependency it would be helpful to just register servlets as services and have them registered with a matching Http Service in a whiteboard pattern style. Likewise registration of static resources would be supported in an extender pattern style.

At this moment some of this missing functionality is covered in a proprietary way. Examples are the Apache Felix Http Whiteboard support or the OPS4J Pax Web collection of bundles.
3.3 Configuration
The Http Service specification currently declares a number of framework properties to configure the Http Service. This raises a number of issues:

- Unable to dynamically reconfigure the Http Service in an easy way
- Incomplete configuration. For example the local interface to bind to is not an official configuration property
- When the Http Service is implemented as bridge to a Servlet Container in which the OSGi framework is deployed (e.g. as part of a Web Application) these properties have no effect.

In addition the actual configuration of an Http Service instance cannot be easily be queried/introspected.

4 Use Cases

5 Requirements

5.1 Update to Http Service API
- **HS-1** The solution MUST define the relationship between the Http Service and Web Application specifications.
- **HS-2** The solution MUST update the Http Service specification to refer to the latest Servlet API specification and define to what extend the Http Service provides support.
- **HS-3** The solution MUST extend the HttpService service API to support Servlet registration with patterns as defined by the Servlet API specification (Section 12.2, Specification of Mappings, in the Servlet API 3.0 specification). This requirement aligns servlet registration to functionality provided by the Servlet API web application descriptor (web.xml).
- **HS-4** The solution MUST extend the HttpService service API to support registration of Servlet API filters with patterns as defined by the Servlet API specification (Section 12.2, Specification of Mappings, in the...
Servlet API 3.0 specification) or referring to servlets by their names. This requirement aligns mapping filters to requests to functionality provided by the Servlet API web application descriptor (web.xml).

HS-5  The solution MUST extend the HttpService service API to support registration of Servlet API listeners.

HS-6  The solution MUST add support for error page configuration.

HS-7  The solution MUST define how registered Servlets and Filters are named.

HS-8  The solution MUST clarify ServletContext implementation in the HttpService for both standalone and bridged Http Service implementations.

HS-9  The solution MUST clarify the ServletContext scope of Servlet API listeners registered through the HttpService.

HS-10 The solution MAY specify support for scripted request processing. For example supporting JSP with Tag Libraries.

HS-11 The solution MAY define how HttpService instances can be dynamically configured.

HS-12 The solution MUST define service registration properties for the HttpService to reflect configuration of the service.

5.2 Introduction of Whiteboard and Extender Pattern Support

The overall goal of Whiteboard and Extender Pattern Support is to register servlets, filters, resources, listeners and error pages in a Bundle without explicitly asking for the HttpService.

HS-13 The solution MUST define whiteboard registration of servlet services with the HttpService.

HS-14 The solution MUST define whiteboard registration of filter services with the HttpService.

HS-15 The solution MUST define whiteboard registration of servlet listener services with the HttpService.

HS-16 The solution MUST define registration of OSGi HttpContext services used for Servlet and Filter registration.

HS-17 The solution MUST define how servlets, filters, and servlet listener services are matched with HttpService services for registration.

HS-18 The solution MUST support registration of static resources according to the extender pattern.

HS-19 The solution MUST support registration of error pages according to the extender pattern.

HS-20 The solution MUST define a capability for the osgi.extender namespace. Bundles providing resources and/or error pages can then require this capability.

HS-21 The solution MUST define a capability for the whiteboard pattern registration in one of the standard namespaces (or a new namespace to be defined in the Chapter 135, Common Namespaces Specification). Bundles registering servlet, filter, and/or servlet listener services can then require this capability.
6 Document Support

6.1 References


6.2 Author's Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Felix Meschberger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>Adobe Systes, Inc.</td>
</tr>
<tr>
<td>Address</td>
<td>Barfüsserplatz 6, 4055 Basel, Switzerland</td>
</tr>
<tr>
<td>Voice</td>
<td>+41 61 226 55 49</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:fmeschbe@adobe.com">fmeschbe@adobe.com</a></td>
</tr>
</tbody>
</table>

6.3 End of Document