Reviving the Http Service
Felix Meschberger, Adobe Systems
Http Service 1.2

- More or less unmodified since OSGi R1 (May, 2000)
  - Resource Registration
  - Custom Authentication
- Based on Servlet API 2.1
  - No Filters
  - No Servlet API Listeners
  - No Error Pages
- Limited Servlet Registration (path prefix only)
- Complex registration:
  - Requires HttpServletResponse to register with init callback
  - Requires keeping state of successful registration
  - Requires HttpServletResponse to unregister
Revive and Update

- Servlet API Version
- Servlet API Elements
- Improved Registration
- Whiteboard Support
Requirements 1: Http Service Updates

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 supportServlet 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.
Requirements 2: Whiteboard Registration

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 HttpSession services used for Servlet and Filter registration.

HS-17 The solution MUST define how servlets, filters, and servlet listener services are matched with HttpSession 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.
Servlet API Version

- Which Servlet API to support/require?
  - Servlet API 3.0 for the latest and greatest?
  - Must not alienate Embedded Systems!

- Servlet API and the Platform
  - 2.4 requires Java 1.3
  - 2.5 requires Java 5
  - 3.0 requires Java 6

\[ \geq 2.4 \]
Servlet API Elements

- Servlet
- Error Pages
- Filter
- Event Listener
- EJB
- MIME mapping

Replacements
- EJB: Use Services
- MIME mapping: Use HttpContext
public interface HttpService {

    public void registerServlet(String alias,
        Servlet servlet,
        Dictionary initparams,
        HttpContext context)
        throws ServletException, NamespaceException;

    public void unregister(String alias);

}
public interface HttpService {

    public void registerServlet(String alias, Servlet servlet, Dictionary initparams, HttpContext context) throws ServletException, NamespaceException;

    public void unregister(String alias);

    }
Sample Traditional Servlet

```java
@Component(service = Servlet.class, property = "alias=/")
public class SampleServletOld extends HttpServlet {

    private HttpService httpService;
    private String servletAlias;
    private boolean servletRegistered;

    @Activate
    private void activate(Map<String, Object> config) throws ServletException, NamespaceException {
        this.servletAlias = (String) config.get("alias");
        this.httpService.registerServlet(this.servletAlias, this, null, null);
        this.servletRegistered = true;
    }

    @Deactivate
    private void deactivate() {
        if (this.servletRegistered) {
            this.httpService.unregister(servletAlias);
        }
    }

    @Reference(unbind = "unbindHttpService")
    private void bindHttpService(final HttpService httpService) {
        this.httpService = httpService;
    }

    private void unbindHttpService(final HttpService httpService) {
        if (this.httpService == httpService) {
            this.httpService = null;
        }
    }
}
```
Sample Traditional Servlet with new API

```java
@Component(service = Servlet.class, property = "alias=/")
public class SampleServletNew extends HttpServlet {

    private HttpService httpService;
    private boolean servletRegistered;

    @Activate
    private void activate(Map<String, Object> config) throws ServletException, NamespaceException {
        String servletAlias = (String) config.get("alias");
        this.httpService.registerServlet(servletAlias, this, null, null);
        this.servletRegistered = true;
    }

    @Deactivate
    private void deactivate() {
        if (this.servletRegistered) {
            this.httpService.unregisterServlet(this);
        }
    }

    @Reference(unbind = "unbindHttpService")
    private void bindHttpService(final HttpService httpService) {
        this.httpService = httpService;
    }

    private void unbindHttpService(final HttpService httpService) {
        if (this.httpService == httpService) {
            this.httpService = null;
        }
    }
}
```
Sample Whiteboard Servlet

```java
@Component(service = Servlet.class, property = "org.osgi.pattern=/")
public class SampleServletWhiteboard extends HttpServlet {
}

- No State Management
- No Http Service reference
```
## Alias vs. Pattern

<table>
<thead>
<tr>
<th>Http Service</th>
<th>Servlet API</th>
<th>Match Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>/alias</td>
<td>/prefix/*</td>
<td>Prefix Match</td>
</tr>
<tr>
<td></td>
<td>*.ext</td>
<td>Extension Match</td>
</tr>
<tr>
<td></td>
<td>&quot;&quot; (empty)</td>
<td>Context Root Match</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>Default Servlet</td>
</tr>
<tr>
<td></td>
<td>/exact/match</td>
<td>Exact Match</td>
</tr>
</tbody>
</table>
## Alias vs. Pattern

<table>
<thead>
<tr>
<th>Http Service</th>
<th>Servlet API</th>
<th>Match Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>/alias</td>
<td>/prefix/*</td>
<td>Prefix Match</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/exact/match</td>
<td></td>
<td>Exact Match</td>
</tr>
</tbody>
</table>

```java
public void registerServlet(String[] pattern,
        Servlet servlet,
        Dictionary<String, String> initparams,
        HttpContext context)
    throws ServletException, 
NamespaceException;
```

Default Servlet
Http Context and Servlet Context

Servlet Context 1

Http Context 1

Authentication

Attributes
Resources
MIME Types

Servlet Context n

Http Context n

Authentication

Attributes
Resources
MIME Types

Context Path
Http Sessions

Common Servlet Context
(e.g. from Servlet Container)
Event Listener

- Listeners are Services
  - Listener interface is Service Name
  - Supported Listeners depending on Servlet API Support
- Whiteboard Registration
- All Events from Http Service (not confined to ServletContext)
Error Pages

- Defined as mappings
  - Status Code -> Location
  - Exception Type -> Location

- Proposal: HttpService extension
  - registerErrorPage(int code, String location)
  - unregisterErrorPage(int code)
  - registerErrorPage(String exceptionType, String location)
  - unregisterErrorPage(String exceptionType)
Whiteboard Support

- Add-On to Http Service

- Gets Servlet, Filter, HttpContext services
- Registers with Http Service
- Matches Servlet to HttpContext with property

Diagram:
- HttpServlet
- Filter
- HttpServlet
- Servlet pattern
- Filter pattern
- HttpServlet
- HttpServlet context.name
## Http Service Whiteboard Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Servlet</th>
<th>HttpContext</th>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>osgi.http.context.id</td>
<td>Refers to HttpContext</td>
<td>Identifies</td>
<td>Refers to HttpContext</td>
</tr>
<tr>
<td>osgi.http.context.shared</td>
<td>n/a</td>
<td>Shared across Bundles ?</td>
<td>n/a</td>
</tr>
<tr>
<td>osgi.http.pattern</td>
<td>Registration Pattern</td>
<td>n/a</td>
<td>Registration Pattern</td>
</tr>
<tr>
<td>osgi.http.errorPage</td>
<td>Status Code Exception Type</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>osgi.http.init.*</td>
<td>Init Parameters</td>
<td>n/a</td>
<td>Init Parameters</td>
</tr>
</tbody>
</table>
Links

- RFP 150 Http Service Updates (https://www.osgi.org/bugzilla/show_bug.cgi?id=146)
- Discuss: osgi-dev@mail.osgi.org