Felix HTTP

Paving the road to the future

Jan Willem Janssen and Marcel Offermans

SSID: Felix
Password: felixdemo

Browse to: http://10.61.0.161:8080/

Jan Willem Janssen

- Software architect at Luminis Technologies
- Currently working on PulseOn and Amdatu
- Committer and PMC member at Apache Felix and Apache ACE

Marcel Offermans

- Director at Luminis Technologies, Fellow at Luminis
- Currently working on Amdatu
- Apache Member, Committer and PMC member at Apache ACE and Apache Felix

Agenda

- Modular Web Applications
- Current State
- Available Extensions
- The New Specification
- New extensions
- Future Work
- Wrap Up

Modular Web Applications

Modular Architectures

- High Cohesion, Low Coupling
- Separation of Concerns
- Maintainable Code
- Reusable, Composable

Deployments

- Compose modules into different deployments
- Low bandwidth by just sending changed modules
- Fast deployments by being able to update running applications

Current State

Explicit Registration

```
public interface HttpService {
  void registerServlet(String alias, Servlet servlet,
      Dictionary initParams, HttpContext httpContext);

  void registerResources(String alias, String name,
      HttpContext httpContext);

  void unregister(String alias);

  HttpContext createDefaultHttpContext();
}
```

Servlets

```
httpService.registerServlet("/hello", new HttpServlet() {
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp) {
        resp.setContentType("text/plain");
        resp.getWriter().write("Hello ApacheCon world!");
    }
}, null /* initParams */, null /* httpContext */);
```

invoke the /hello servlet

```
...
```

Resources

```
httpService.registerResources("/site", "/resources", null /* httpContext */);
```

```
$ cat resources/hello
Hello ApacheCon world, I'm a static resource!
```

request a static resource called hello

```
...
```

HttpContext

- Provides secure access to servlets and resources.
- Layer of abstraction for resource loading.

```
public interface HttpContext {
    boolean handleSecurity(HttpServletRequest request,
            HttpServletResponse response);
    URL getResource(String name);
    String getMimeType(String name);
}
```

Web Application Specification

Part of the OSGi Enterprise Specification, chapter 128.

- Web Application Bundle (WAB) are extended Java EE WAR files.
- They have optional JSP support.
- Integration with OSGi `BundleContext` and service registry.

Available Extensions

Whiteboard

don't call us, we'll call you!

Register your **`Servlet`** in the service registry and add a property called **`alias`** containing its endpoint.

For more information:
http://www.osgi.org/wiki/uploads/Links/whiteboard.pdf

Filters

- Just like Servlets, these can be registered whiteboard style
- or use explicit registration:
  - the `ExtHttpService` service from Felix HTTP
  - the `WebContainer` service from PAX Web

Amdatu Web Resources

Part of the Amdatu.org open source project (Apache Licensed).

```
X-Web-Resource-Version: 1.1
X-Web-Resource: /amdatu-resources;resources
X-Web-Resource-Default-Page: index.html,/doc=javadoc.html
```

```
Include-Resource: resources=resources/basic
```

Demo

request a static resource called /amdatu-resources

```
...
```

Amdatu Web REST

Extensive support for REST endpoints based on industry standards.

- JAX-RS based annotation support.
- Includes support for Jackson mappings.
- Self-documenting endpoints with Swagger.

Demo

```
@Path("/rest")
@Description("Provides a demo REST endpoint")
public class DemoRestEndpoint {
  private String m_response = "Hello World!";

  @GET @Produces(MediaType.TEXT_PLAIN)
  @Description("Gives a plain text response")
  public String getPlainResponse() {
    return m_response;
  }

  @POST @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
  @Description("Allows one to set the response to return")
  public void setResponse(
      @FormParam("response") @DefaultValue("Default response") String newResponse) {
    m_response = newResponse;
  }
}
```

Self-documenting Endpoints

Swagger is a library that creates documentation for endpoints based on JAX-RS annotations plus some extras.

Go to Swagger documentation

The new specification

Whiteboard

- No longer an extension
- No explicit registration
- Specify HttpService to be used

Example

```
Dictionary props = new Hashtable();
props.put("osgi.http.whiteboard.servlet.pattern", "/slidemgr");
props.put("osgi.http.whiteboard.target", "(http.service=demo)");
context.registerService(Servlet.class.getName(), new SlideManager(), props);
```

(A)synchronous Servlets

- Servlet 3.0 API
- Support wildcard one or more patterns
- Process work outside the servlet lifecycle

Example code

```
class WorkerServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) {
        final AsyncContext asyncContext = req.startAsync(req, resp);
        asyncContext.start(new DeepThought(asyncContext));
    }
}
//
class DeepThought implements Runnable {
    private AsyncContext m_context = // ...
    public void run() {
        // ...do some hard work...
        TimeUnit.DAYS.sleep(356L * 7500000L);

        HttpServletResponse response = (HttpServletResponse) asyncContext.getResponse();
        response.setStatus(SC_OK);
        response.getWriter().printf("42");
        asyncContext.complete();
    }
}
```

Example registration

```
Dictionary props = new Hashtable();
props.put("osgi.http.whiteboard.servlet.pattern", "/worker/*");
props.put("osgi.http.whiteboard.servlet.asyncSupported", "true");
context.registerService(Servlet.class.getName(), new WorkerServlet(), props);
```

Filters

- Full support

Example

```
Dictionary props = new Hashtable();
props.put("osgi.http.whiteboard.filter.pattern", "/*");
props.put("osgi.http.whiteboard.filter.dispatcher",
        new String[] {"REQUEST", "INCLUDE", "FORWARD"});
context.registerService(Filter.class.getName(), new SecurityFilter(), props);
```

Listeners

- Full support
- All events

Example

```
final CountDownLatch latch = new CountDownLatch(1);
ServletContextListener contextListener = new ServletContextListener() {
    public void contextDestroyed(ServletContextEvent event) {}
    public void contextInitialized(ServletContextEvent event) {
        latch.countDown();
    }
};
Dictionary props = new Hashtable();
props.put("osgi.http.whiteboard.context.select", "DEFAULT");
context.registerService(ServletContextListener.class.getName(), contextListener, props);

assertTrue("HttpService not ready in time?!", latch.await(5, TimeUnit.SECONDS));
// continue with your itest...
```

Custom Error Pages

- By error code
- By exception

Example

```
Dictionary props = new Hashtable();
props.put("osgi.http.whiteboard.servlet.errorPage",
    new String[] {"500", "java.io.IOException"});
context.registerService(Servlet.class.getName(), new MyErrorHandlingServlet(), props);
```

New extensions

WebSockets

- RFC 6455
- “Real-time”
- Binary or text-based
- Two-way communication

Example

```
// Client-side
var wsConn = new WebSocket("ws://" + window.location.host + "/servlet", [ "my-protocol" ]);
wsConn.onmessage = function(event) {
    var data = event.data;
    // do something with data
}
```

```
// Server-side, registered at "/servlet"
class MyWebSocketServlet extends WebSocketServlet {
    public WebSocket doWebSocketConnect(HttpServletRequest request, String protocol) {
        if ("my-protocol".equals(protocol)) {
            return new WebSocket.OnTextMessage() {
                public void onOpen(Connection conn) {}
                public void onClose(int code, String reason) {}
                public void onMessage(String data) {}
            };
        }
        return null;
    }
}
```

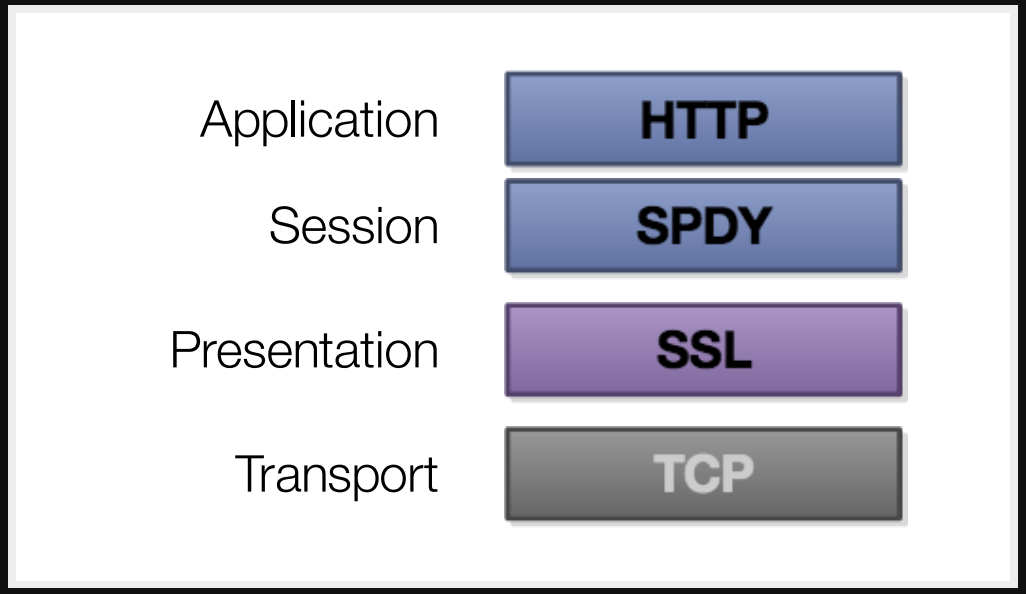
Demo

Multi-user Etch A Sketch

SPDY

Let's make the web faster

- Session layer on top of SSL
- Reduce bandwidth & lower latency
- Push multiple resources in a request
- Basis of HTTP 2.0



How SPDY fits in the OSI layer model.

SPDY vs WebSockets

	SPDY	WebSockets
Goal	optimize HTTP	2-way communication
Upgradeability	transparent	needs works
Secure?	✔ (mandatory)	✔ (if needed)
Two-way?	✔ / ✘	✔
Multiplexed	✔	✘
Prioritized	✔	✘

Demo

HTTP vs SPDY

Future Work

- Finalize support for new HttpService specification
- Upgrade to Jetty 9
- Allow “new style” WebSockets (JSR 356) to be used
- Improved support for SPDY

Wrap Up

- New/updated specifications
- New features, functionality & improvements
- Available extensions
- Build modular web applications

Questions?

Links

- felix.apache.org
- luminis-technologies.com
- bndtools.org
- amdatu.org
- bitbucket.org/marrs/apachecon2014-felix-http