Jakarta EE - Hands-on Lab

HOL4852

Jakarta EE - Hands-on Lab

- **David Heffelfinger** - @ensode
  Principal Consultant, Ensode Technology, LLC
  http://www.ensode.com

- **Bob Larsen** - @direHerring
  Staff Software Engineer, Pivotal Labs
  https://pivotal.io

Jakarta EE 8 - Modernization & Simplification

- **JAX-RS 2.1** - Reactive Client API, Server-Sent Events, …
- **Servlet 4** - HTTP/2, Server Push, …
- **JSON-B** - Java <-> JSON binding
- **JSON-P 1.1** - Updates to JSON standards, JSON Collectors, …
- **CDI 2.0** - Async Event, Observers ordering, SE support, …
- **Bean Validation 2.0** - Embrace Java SE 8, new constraints, …
- **JSF 2.3** - Improved CDI, WebSocket, SE 8 integration, …
- **Security API** - Portable Identity Store, Authentication & Security Context

Agenda

- JSON-B 1.0
- Bean Validation 2.0
- Security API
- Servlet 4.0
  - Server Push
- JAX-RS 2.1
  - Reactive Client
  - Server Sent Events
- CDI 2.0
  - Asynchronous Events
  - Observer Ordering

Hands On Labs

- Short exercises to reinforce the material
- Requirements
  - A modern JDK 8, e.g. Java SE 8 update 144!
    - Not a JRE not Java 9… yet!
  - NetBeans 8.2
    - With Jakarta EE support!
  - GlassFish 5.0
  - Internet connectivity

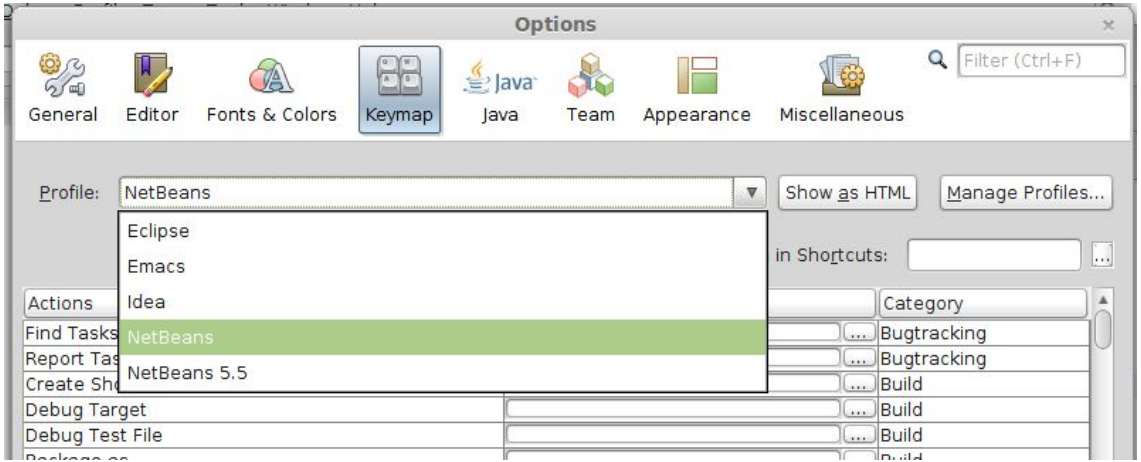
Why NetBeans?

- It comes bundled with everything we need
- Jakarta EE compliant application server (GlassFish)
- RDBMS (JavaDB), including a sample database
- It’s free and it rocks :)

Apache
NetBeans IDE

Why NetBeans? (cont'd)

- But I'm used to Eclipse/IntelliJ IDEA!
- Tools | Options | Keymap



JSON-B 1.0

- Augments JSON-P
- Automatically populate Java objects from JSON documents
- Automatically generate JSON documents from Java objects

Populating Java objects from JSON

```
public class Customer {
    private String salutation;
    private String firstName;
    private String middleName;
    private String lastName;
    private LocalDate dateOfBirth;

    //getters and setters omitted

}
```

Populating Java objects from JSON (cont’d)

```
{
   "salutation" :"Dr",
   "firstName" : "Oracle",
   "middleName" : "Code",
   "lastName" : "One",
   "dateOfBirth" : "1995-05-23"
}
```

Populating Java objects from JSON (cont'd)

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
public Response addCustomer(String customerJson) {
    ...
    Jsonb jsonb = JsonbBuilder.create();
    Customer customer = jsonb.fromJson(customerJson, Customer.class);
    ...
}
```

Generating JSON from Java Objects

```
Jsonb jsonb = JsonbBuilder.create();
String result = jsonb.toJson(customer);
```

Bean Validation 2.0

- API to validate date using annotations
- Can be used in Jakarta EE and Java SE
- New built-in constraints
  - @Email
  - @NotEmpty
  - @NotBlank
  - @Positive
  - @PositiveOrZero
  - @Negative
  - @NegativeOrZero
  - @PastOrPresent
  - @FutureOrPresent

Bean Validation 2.0 (cont’d)

- Map keys and values can be validated
  Map<**@Valid** Key, **@Valid** Value> validatedMap;
- Support for Java SE 8 @Optional annotation
- Support for Java SE 8 Date/Time API
- Support for validating container types (Collections, Maps, etc) by annotating type arguments of parameterized types
  List<**@NotEmpty** String> nonEmptyStringList;

Demo

- JSON-B 1.0 and Bean Validation 2.0

Exercise 1

- Go to https://github.com/dheffelfinger/j1-hol

Security API

- Simplify and Modernize security capabilities of Jakarta EE
- Identity Store
- Portable Authentication Mechanism
- Security Context

Security API - Identity Store

- Provide a storage system where caller credentials and data are stored
  - LDAP, DataBase, ...
- Perform caller validation and details retrieval
  - In - Valid caller name & password
  - Out - (Possibly different) caller name and/or associated group(s)
- Does not interact with the caller!
- Built-in
  - @LdapIdentityStoreDefinition
  - @DatabaseIdentityStoreDefinition

Security API - Identity Store

```
@DatabaseIdentityStoreDefinition(
    dataSourceLookup = "${'java:global/MyDS'}",
    callerQuery = "#{'select password from caller where name = ?'}",
    groupsQuery = "select group_name from caller_groups where caller_name = ?",
    hashAlgorithm = Pbkdf2PasswordHash.class,
    priorityExpression = "#{100}",
    hashAlgorithmParameters = {
        "Pbkdf2PasswordHash.Iterations=3072",
        "${applicationConfig.dyna}"
    }
)
```

Security API - Custom Identity Store

- New **IdentityStore** interface
  - validate(Credential)
  - getCallerGroups(CredentialValidationResult)

Security API - Authentication Mechanism

- CDI enabled version of **ServerAuthModule** that complies to the JASPIC Servlet Container Profile
- Encouraged to use an **IdentityStore**
  - Caller credential validation
  - Caller details retrieval
- Built-in
  - @BasicAuthenticationMechanismDefinition
  - @FormAuthenticationMechanismDefinition
  - @CustomFormAuthenticationMechanismDefinition

Security API - Custom Authentication Mechanism

- New **HttpAuthenticationMechanism** interface
  - void **cleanSubject**(HttpServletRequest, HttpServletResponse, HttpMessageContext)
  - AuthenticationStatus **validateRequest**(Req ,Resp, MCtx) throws AuthenticationException;
  - AuthenticationStatus **secureResponse**(Req ,Resp, MCtx) throws AuthenticationException;

Exercise 2

- Go to https://github.com/dheffelfinger/j1-hol

HTTP/2

- Reduce Web Applications Latency
- Preserve the HTTP Semantics
- Binary Framed Protocol
  - Multiplexing
  - Header Compression
  - ...
  - Server Push
- Supported in Servlet 4.0

Servlet 4 - HTTP/2 Server Push Support

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = resp.getWriter()) {

        PushBuilder pusher = req.newPushBuilder();
        if (pusher != null) {
             pusher.addHeader("Pushed", "from_GF5");
             pusher.path("payload.jpg")
                   .push();
        }

        out.println("<!DOCTYPE html>...");
    }
...
```

JAX-RS 2.1

- New Reactive Client API
- Server-Sent Events support
- JSON-B, JSON-P support
- ...

JAX-RS Reactive Client API

- Allows to compose pipeline of asynchronous services invocations
- Relies on CompletionStage API
- New Asynchronous invoker

JAX-RS Reactive Client API (cont'd)

```
CompletionStage<JsonObject> cfIp = client.target("http://api.ipify.org/")
                .queryParam("format", "json").request()
                .rx()
                .get(JsonObject.class);

Function<JsonObject, CompletionStage<JsonObject>> function = ip
                -> client.target("https://ipvigilante.com")
                        .path("json").path(ip.getString("ip")).request()
                        .rx()
                        .get(JsonObject.class);

cfIp.thenCompose(function)
    .thenAccept(consumer);
```

JAX-RS Server Sent Events (SSE)

- Allows server to push data to the client
- Supported by all “real” browsers (all but IE/Edge)
- Designed to support auto-reconnection, retrieval of lost messages and with a better suited protocol than the alternatives (COMET / Long polling / Websockets) for its purpose
- Not bi-directional though
- Server and client API available as of JAX-RS 2.1

JAX-RS SSE Server API

- SseEventSink - the SSE equivalent of an “OutputStream”; a channel to send events to the client
- SseBroadcaster - allows one to send the same events to several SseEventSinks
- OutboundSseEvent - a single event, from the server’s point of view (id, name, mediaType, data, comment)
- OutboundSseEvent.Builder - fluent builder for OutboundSseEvent
- Sse - allows you to create SseBroadcasters and OutboundSseEvent.Builders

JAX-RS SSE Server API

```
@Context
private Sse sse;
private volatile SseBroadcaster broadcaster;

@PostConstruct
public void init() {
    this.broadcaster = sse.newBroadcaster();
}
```

JAX-RS SSE Server API

```
@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public void register(@Context SseEventSink eventSink) {
    broadcaster.register(eventSink);
}
//Somewhere in your code...
    broadcaster.broadcast(sse.newEventBuilder()
            .id(String.valueOf(messageId))
            .mediaType(MediaType.APPLICATION_JSON_TYPE)
            .data(MyCustomEvent.class, info)
            .build());
```

JAX-RS SSE Client API

- SseEventSource - the SSE equivalent of an “InputStream”; a channel to collect events sent by the server
- InboundSseEvent - a single event, from the server’s point of view (id, name, mediaType, data, comment)

JAX-RS SSE Client API

```
SseEventSource source = SseEventSource.target(target).build();
source.register(event -> handle(event));
source.open();

private void handle(InboundSseEvent event) {
    MyCustomEvent mce = event.readData(MyCustomEvent.class,
            MediaType.APPLICATION_JSON_TYPE);
    // Do something useful
}
```

Demo

- JAX-RS Server-Sent Events

CDI 2.0

- Java SE support
- Asynchronous events
- Observer ordering
- Java SE 8 alignment
- ...

CDI 2.0 - Asynchronous events

```
@Inject
private Event<MyCustomEvent> event;

//Somewhere in your code…
MyCustomEvent mce = ...
event.fireAsync(mce);
```

CDI 2.0 - Observer ordering

```
    void on(@Observes @Priority(Interceptor.Priority.APPLICATION)
MyCustomEvent event) {
    //…
    }
```

Exercise 3

- Go to https://github.com/dheffelfinger/j1-hol