Jersey Framework 맛보기

2011.8.5

김용환

knight76 at gmail.com

특징

- REST 표준 API 구현
  - JAX-RS: Java API for RESTful Web Services (JSR-311)
- Servlet 3.0 지원
- HTML 포스트 요청에도 사용가능
- JAXB / JSON 지원
- JRebel(kill 프로세스 없이 웹 어플 재시작 가능한 솔루션) 연동
- Test Framework이 잘되어 있음
- Spring/Juice과 연동 쉬움
- 용도 REST 방식의 OPEN API Framework.

정보

- 홈페이지 : http://jersey.java.net/
- 사용팀
  - Ehcache Server (Terracotta)
  - Apache Camel (The Apache Software Foundation)
  - Apache ActiveMQ (The Apache Software Foundation)
- Dependency
  - Grizzly servlet webserver (Tomcat에서 사용할 수 있음)
  - Jersey-server
  - JAXB

정보

- 최신 버전
  - Jersey 1.0.3.1 implements JAX-RS 1.0.
  - Jersey 1.8 implements JAX-RS 1.1.
- 문서화 : 잘되어 있음
  - http://jersey.java.net/nonav/documentation/latest/user-guide.html
  - http://wikis.sun.com/display/Jersey/Main
- 커뮤니티 활동성 : 무난함
  - 한 달에 user 메일은 200~400 통 사이
  - 버전업 : 1~4개월 사이에 꾸준한 버전업
    - 2010년 5월 1.2
      2011년 8월 1.9 개발 중 (조만간 출시 예정)

맛보기 Demo

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

 1    // The Java class will be hosted at the URI path "/helloworld"
 2    @Path("/helloworld")
 3    public class HelloWorldResource {
 4
 5        // The Java method will process HTTP GET requests
 6        @GET
 7        // The Java method will produce content identified by the MIME Media
 8        // type "text/plain"
 9        @Produces("text/plain")
10         public String getClichedMessage() {
11             // Return some cliched textual content
12             return "Hello World";
13         }
14     }
```

맛보기 Demo

```
 1 public class Main {
 2
 3     public static void main(String[] args) throws IOException {
 4
 5         final String baseUri = "http://localhost:9998/";
 6         final Map<String, String> initParams =
 7                                 new HashMap<String, String>();
 8
 9         initParams.put("com.sun.jersey.config.property.packages",
10                 "com.sun.jersey.samples.helloworld.resources");
11
12         System.out.println("Starting grizzly...");
13         SelectorThread threadSelector =
14                         GrizzlyWebContainerFactory.create(baseUri, initParams);
15         System.out.println(String.format(
16           "Jersey app started with WADL available at %sapplication.wadl\n" +
17           "Try out %shelloworld\nHit enter to stop it...", baseUri, baseUri));
18         System.in.read();
19         threadSelector.stopEndpoint();
20         System.exit(0);
21     }
22 }
```

맛보기 DEMO

서버 실행 (main)

```
Starting grizzly...
8월 05, 2011 11:01:55 오전
org.glassfish.grizzly.http.server.NetworkListener start
정보: Started listener bound to [localhost:9998]
8월 05, 2011 11:01:55 오전 org.glassfish.grizzly.http.server.HttpServer
start
정보: [HttpServer] Started.
Jersey app started with WADL available at
http://localhost:9998/application.wadl
Try out http://localhost:9998/helloworld
Hit enter to stop it...
```

```
8월 05, 2011 11:02:33 오전
com.sun.jersey.api.core.PackagesResourceConfig init
정보: Scanning for root resource and provider classes in the packages:
  com.sun.jersey.samples.helloworld.resources
8월 05, 2011 11:02:33 오전
com.sun.jersey.api.core.ScanningResourceConfig logClasses
정보: Root resource classes found:
  class com.sun.jersey.samples.helloworld.resources.HelloWorldResource
8월 05, 2011 11:02:33 오전
com.sun.jersey.api.core.ScanningResourceConfig init
정보: No provider classes found.
8월 05, 2011 11:02:33 오전
com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
정보: Initiating Jersey application, version 'Jersey: 1.8 06/24/2011 12:17
PM'
```

Client

```
$ curl http://localhost:9998/helloworld
```

(결과 출력) Hello World

맛보기 DEMO

- Description 정보

```
$ curl http://localhost:9998/application.wadl
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
    <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 1.8
06/24/2011 12:17 PM"/>
    <resources base="http://localhost:9998/">
        <resource path="/helloworld">
            <method id="getClichedMessage" name="GET">
                <response>
                    <representation mediaType="text/plain"/>
                </response>
            </method>
        </resource>
    </resources>
</application>
```

@PATH

- 상대적인 URI path를 의미
- 예)
  - @Path("/users/{username}")
    -> http://example.com/users/Galileo

```
1 @Path("/users/{username}")
2 public class UserResource {
3
4     @GET
5     @Produces("text/xml")
6     public String getUser(@PathParam("username") String userName) {
7         ...
8     }
9 }
```

@Path("users/{username: [a-zA-Z][a-zA-Z_0-9]}") 가능

Http Method

- @GET, @PUT, @POST, @DELETE, @HEAD

```
1 @PUT
2 public Response putContainer() {
3     System.out.println("PUT CONTAINER " + container);
4
5     URI uri =  uriInfo.getAbsolutePath();
6     Container c = new Container(container, uri.toString());
7
8     Response r;
9     if (!MemoryStore.MS.hasContainer(c)) {
10        r = Response.created(uri).build();
11    } else {
12        r = Response.noContent().build();
13    }
14
15    MemoryStore.MS.createContainer(c);
16    return r;
17 }
```

@Produce

- MIME media type 정보 알림

```
1 @Path("/myResource")
2 @Produces("text/plain")
3 public class SomeResource {
4     @GET
5     public String doGetAsPlainText() {
6         ...
7     }
8
9     @GET
10     @Produces("text/html")
11     public String doGetAsHtml() {
12         ...
13     }
14 }
```

Accept: text/plain

```
1 @GET
2 @Produces({"application/xml",
"application/json"})
3 public String doGetAsXmlOrJson() {
4     ...
5 }
```

여러 개의 Acception 타입 지정가능

@Produce

- XML또는 JSON으로 전달 가능

```
@Singleton
@Path("/person")
public class PersonAction {
    @GET
    @Produces("text/xml")
    public Person getPersonMessage() {
```

```
@Singleton
@Path("/person")
public class PersonAction {
    @GET
    @Produces("application/json")
    public Person getPersonMessage() {
```

@QueryParam

- Query paramter를 받을 수 있음
- Default 값 지정 가능

```
1 @Path("smooth")
2 @GET
3 public Response smooth(
4         @DefaultValue("2") @QueryParam("step") int step,
5         @DefaultValue("true") @QueryParam("min-m") boolean hasMin,
6         @DefaultValue("true") @QueryParam("max-m") boolean hasMax,
7         @DefaultValue("true") @QueryParam("last-m") boolean hasLast,
8         @DefaultValue("blue") @QueryParam("min-color") ColorParam minColor,
9         @DefaultValue("green") @QueryParam("max-color") ColorParam maxColor,
10         @DefaultValue("red") @QueryParam("last-color") ColorParam lastColor
11         ) { ... }
```

Parameter 구하기

```
1 @GET
2 public String get(@Context UriInfo ui) {
3     MultivaluedMap<String, String> queryParams = ui.getQueryParameters();
4     MultivaluedMap<String, String> pathParams = ui.getPathParameters();
5 }
```

```
1 @GET
2 public String get(@Context HttpHeaders hh) {
3     MultivaluedMap<String, String> headerParams = hh.getRequestHeaders();
4     Map<String, Cookie> pathParams = hh.getCookies();
5 }
```

```
1 @POST
2 @Consumes("application/x-www-form-urlencoded")
3 public void post(MultivaluedMap<String, String> formParams) {
4     // Store the message
5 }
```

Sub Resource

```
 1 @Singleton
 2 @Path("/printers")
 3 public class PrintersResource {
 4
 5     @GET
 6     @Produces({"application/json", "application/xml"})
 7     public WebResourceList getMyResources() { ... }
 8
 9     @GET @Path("/list")
10     @Produces({"application/json", "application/xml"})
11     public WebResourceList getListOfPrinters() { ... }
12
13     @GET @Path("/jMakiTable")
14     @Produces("application/json")
15     public PrinterTableModel getTable() { ... }
.....
```

URI building

```
1 UriBuilder.fromUri("http://localhost/").
2     path("{a}").
3     queryParam("name", "{value}").
4     build("segment", "value");
```

"http://localhost/segment?name=value":

Exception 처리

```
1 @Path("items/{itemid}/")
2 public Item getItem(@PathParam("itemid") String itemid) {
3     Item i = getItems().get(itemid);
4     if (i == null)
5         throw new NotFoundException("Item, " + itemid + ", is not found");
6
7     return i;
8 }
```

```
1 public class NotFoundException extends WebApplicationException {
2     /**      * Create a HTTP 404 (Not Found) exception.       */
6     public NotFoundException() {
7         super(Responses.notFound().build());
8     }
9    public NotFoundException(String message) {
15        super(Response.status(Responses.NOT_FOUND).
16              entity(message).type("text/plain").build());
17    }
18 }
```

304 Response

```
1 public SparklinesResource(
 2        @QueryParam("d") IntegerList data,
 3        @DefaultValue("0,100") @QueryParam("limits") Interval limits,
 4        @Context Request request,
 5        @Context UriInfo ui) {
 6    if (data == null)
 7        throw new WebApplicationException(400);
 8
 9    this.data = data;
10
11    this.limits = limits;
12
13    if (!limits.contains(data))
14        throw new WebApplicationException(400);
15
16    this.tag = computeEntityTag(ui.getRequestUri());
17    if (request.getMethod().equals("GET")) {
18        Response.ResponseBuilder rb = request.evaluatePreconditions(tag);
19        if (rb != null)
20            throw new WebApplicationException(rb.build());
21    }
22 }
```

Request의
Etag 확인

Security

```
1 @Path("basket")
2 public ShoppingBasketResource get(@Context SecurityContext sc) {
3     if (sc.isUserInRole("PreferredCustomer") {
4         return new PreferredCustomerShoppingBaskestResource();
5     } else {
6         return new ShoppingBasketResource();
7     }
8 }
```

Rules of Injection

```
1 @Path("id: \d+")
 2 public class InjectedResource {
 3     // Injection onto field
 4     @DefaultValue("q") @QueryParam("p")
 5     private String p;
 6
 7     // Injection onto constructor parameter
 8     public InjectedResource(@PathParam("id") int id) { ... }
 9
10     // Injection onto resource method parameter
11     @GET
12     public String get(@Context UriInfo ui) { ... }
13
14     // Injection onto sub-resource resource method parameter
15     @Path("sub-id")
16     @GET
17     public String get(@PathParam("sub-id") String id) { ... }
18
19     // Injection onto sub-resource locator method parameter
20     @Path("sub-id")
21     public SubResource getSubResource(@PathParam("sub-id") String id) { ... }
22
23     // Injection using bean setter method
24     @HeaderParam("X-header")
25     public void setHeader(String header) { ... }
26 }
```

API

- Client/WebResource
- Building Request
- Receiving Response

End of Document