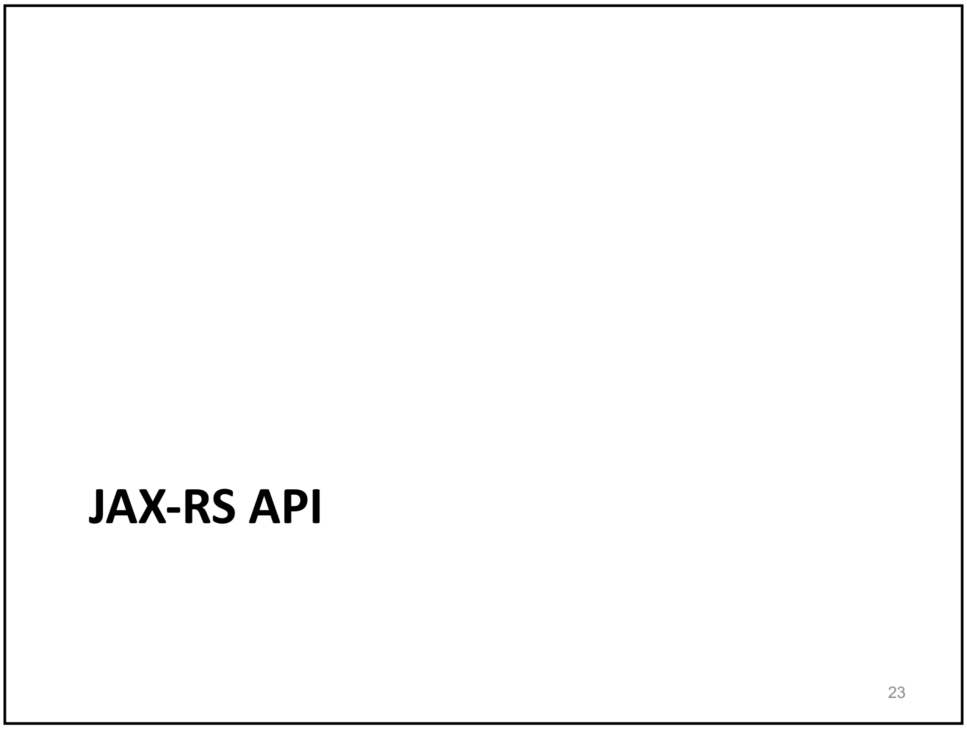
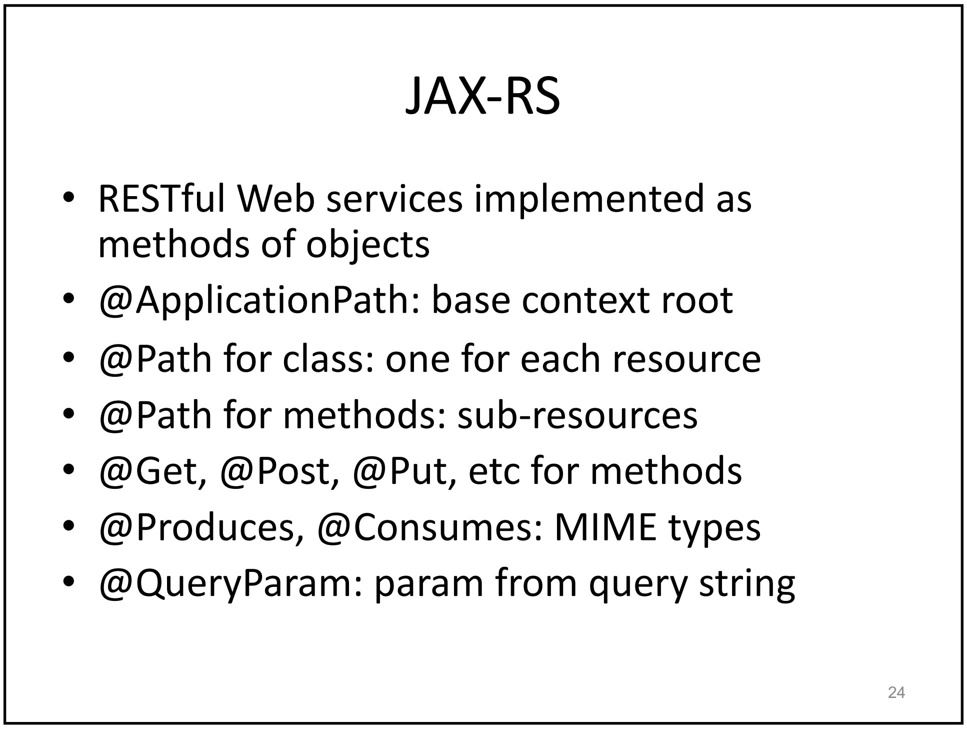
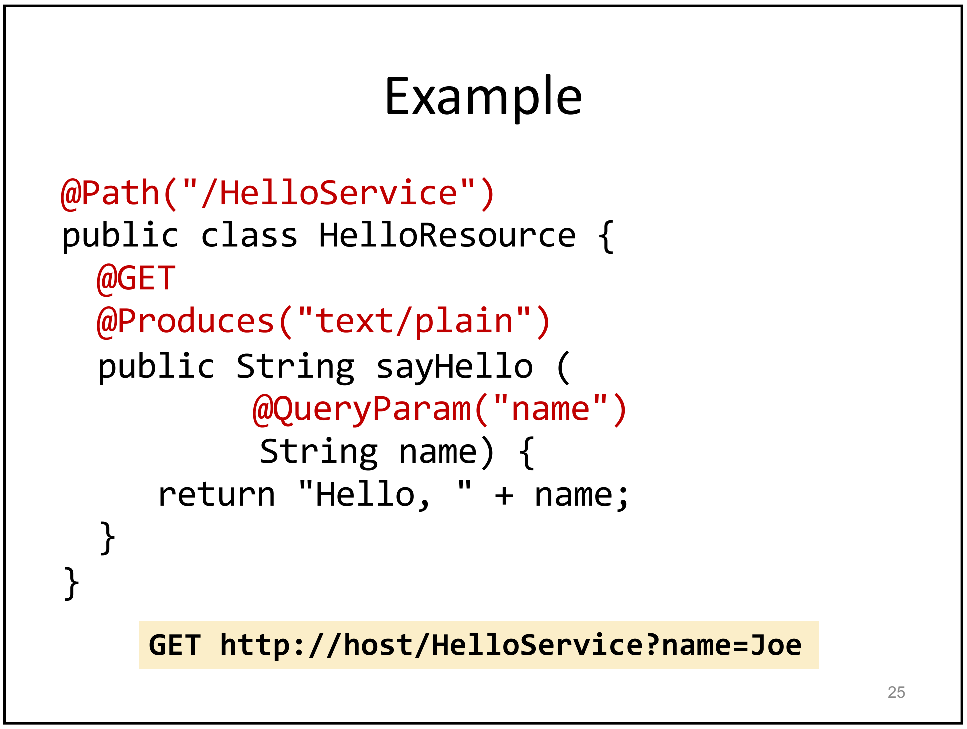


JAX-RS API



JAX-RS

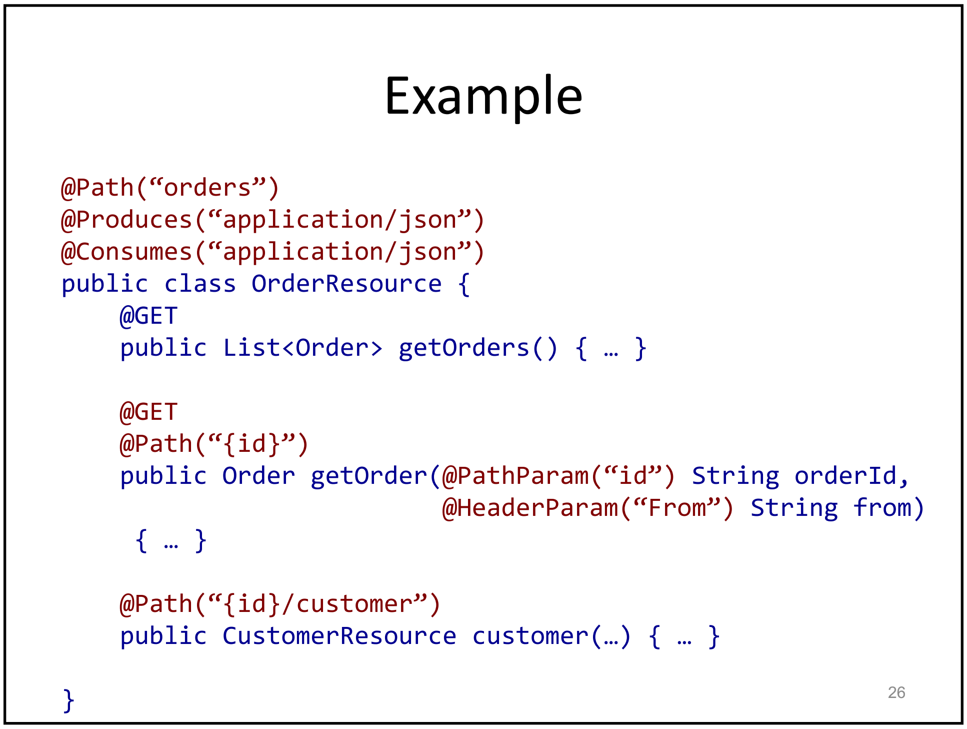
- RESTful Web services implemented as methods of objects
- @ApplicationPath: base context root
- @Path for class: one for each resource
- @Path for methods: sub-resources
- @Get, @Post, @Put, etc for methods
- @Produces, @Consumes: MIME types
- @QueryParam: param from query string



Example

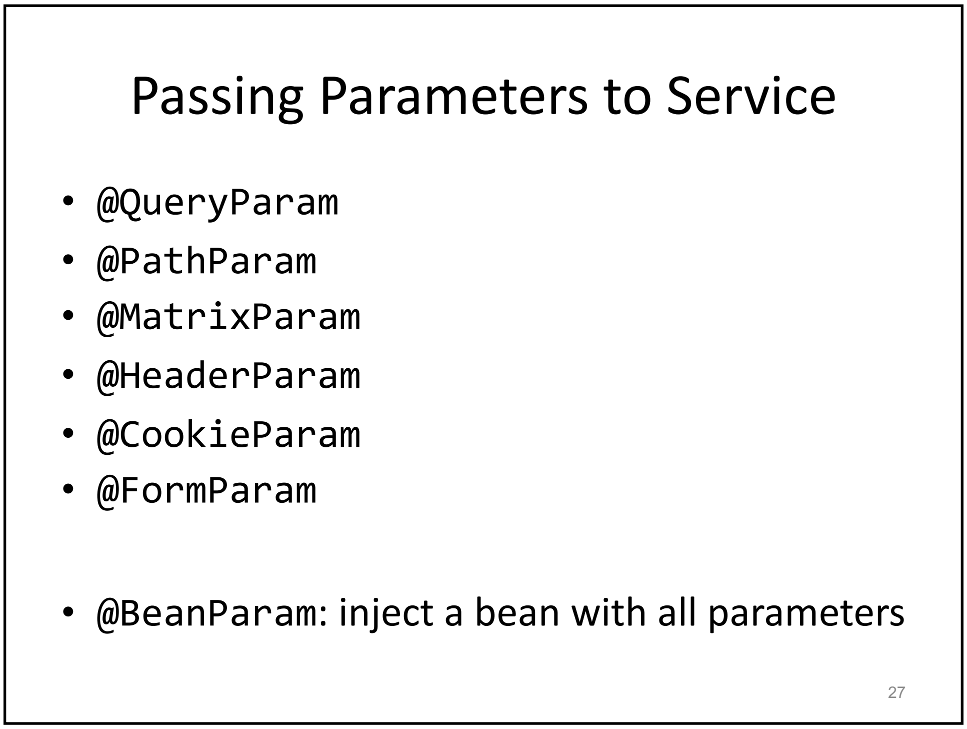
```
@Path("/HelloService")
public class HelloResource {
  @GET
  @Produces("text/plain")
  public String sayHello (
           @QueryParam("name")
           String name) {
     return "Hello, " + name;
  }
}
```

```
GET http://host/HelloService?name=Joe
```



Example

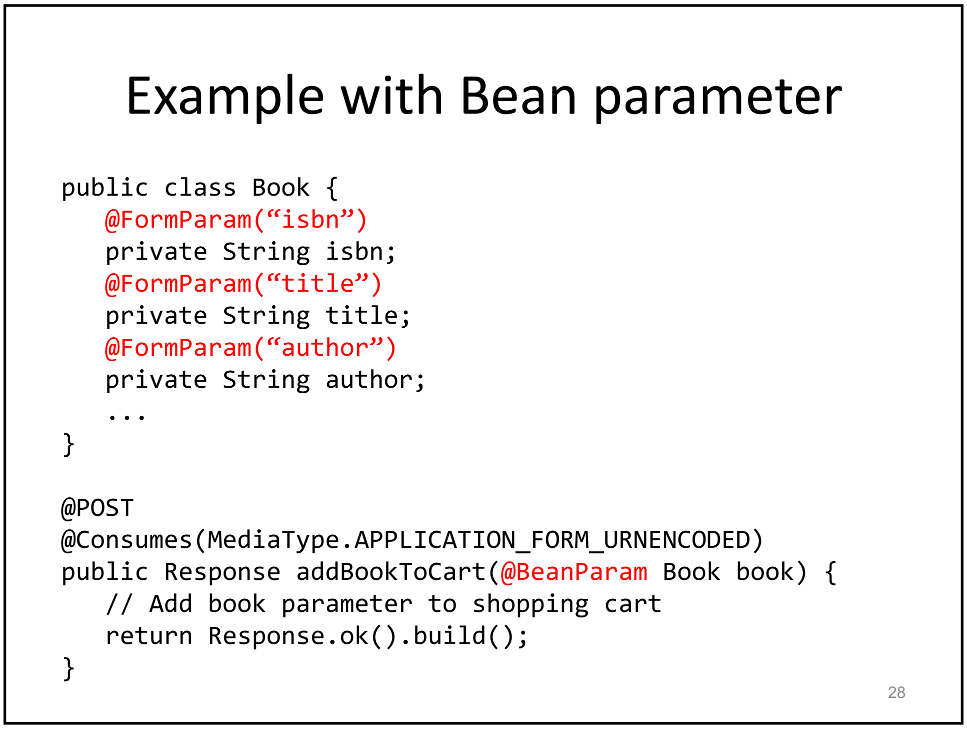
```
@Path(“orders”)
@Produces(“application/json”)
@Consumes(“application/json”)
public class OrderResource {
    @GET
    public List<Order> getOrders() { … }

    @GET
    @Path(“{id}”)
    public Order getOrder(@PathParam(“id”) String orderId,
                          @HeaderParam(“From”) String from)
     { … }

    @Path(“{id}/customer”)
    public CustomerResource customer(…) { … }

}
```



Passing Parameters to Service

- @QueryParam
- @PathParam
- @MatrixParam
- @HeaderParam
- @CookieParam
- @FormParam

- @BeanParam: inject a bean with all parameters



Example with Bean parameter

```
public class Book {
   @FormParam(“isbn”)
   private String isbn;
   @FormParam(“title”)
   private String title;
   @FormParam(“author”)
   private String author;
   ...
}

@POST
@Consumes(MediaType.APPLICATION_FORM_URNENCODED)
public Response addBookToCart(@BeanParam Book book) {
   // Add book parameter to shopping cart
   return Response.ok().build();
}
```

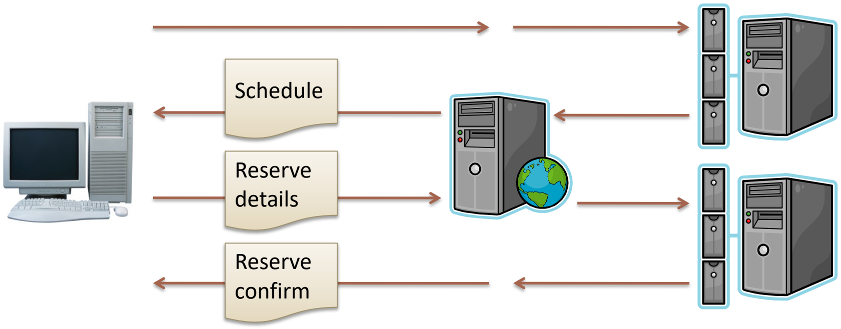
EXAMPLE: TRAIN RESERVATION SERVICE

29

29

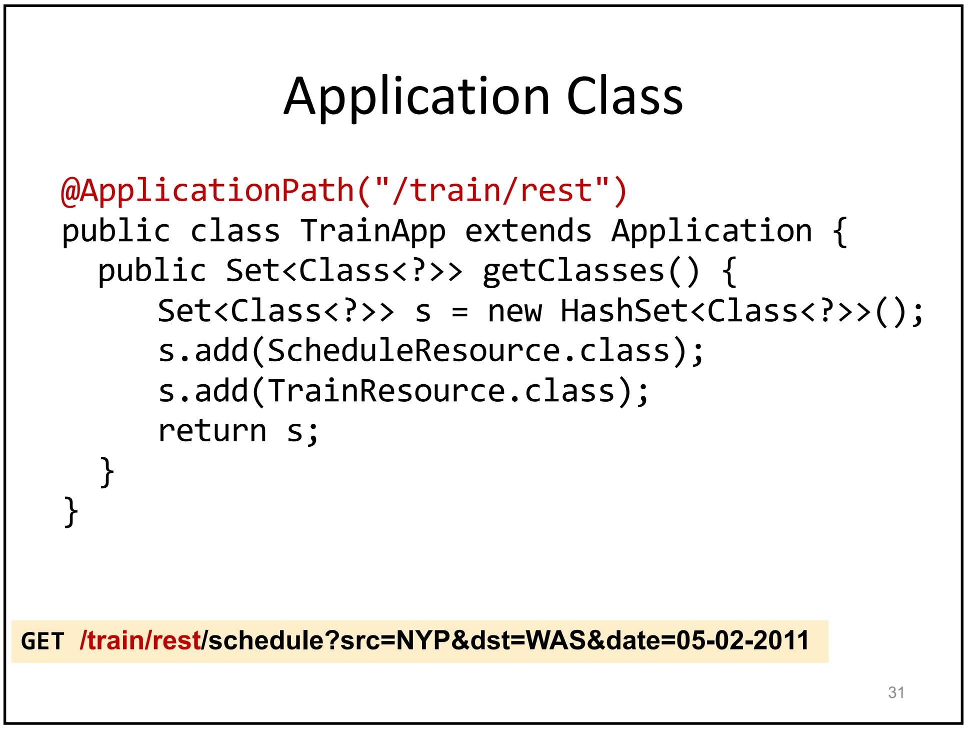
REST Web Service

GET /train/rest/schedule?src=NYP&dst=WAS&date=05-02-2011



POST /train/rest/train/2107

30

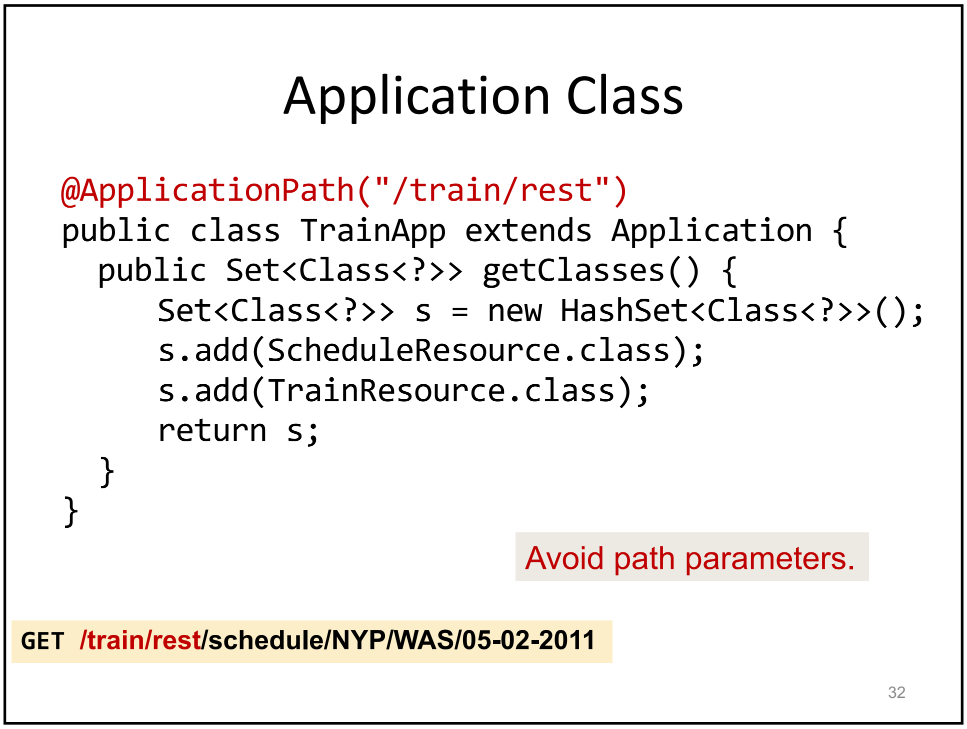


Application Class

```
@ApplicationPath("/train/rest")
public class TrainApp extends Application {
  public Set<Class<?>> getClasses() {
      Set<Class<?>> s = new HashSet<Class<?>>();
      s.add(ScheduleResource.class);
      s.add(TrainResource.class);
      return s;
  }
}
```

`GET /train/rest/schedule?src=NYP&dst=WAS&date=05-02-2011`

31



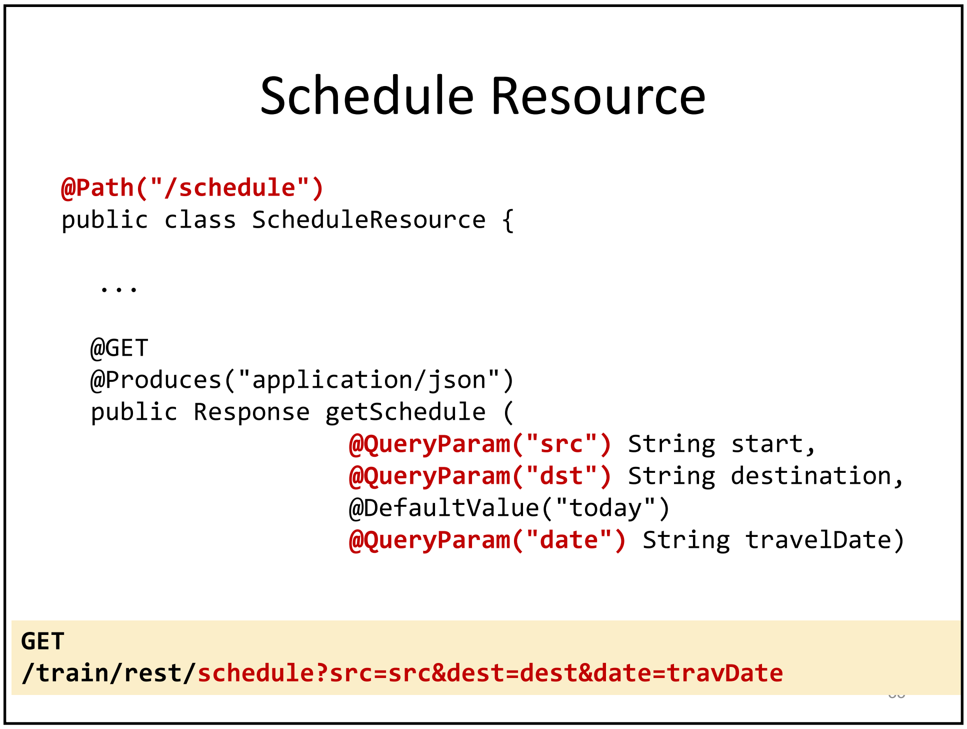
Application Class

```
@ApplicationPath("/train/rest")
public class TrainApp extends Application {
  public Set<Class<?>> getClasses() {
      Set<Class<?>> s = new HashSet<Class<?>>();
      s.add(ScheduleResource.class);
      s.add(TrainResource.class);
      return s;
  }
}
```

Avoid path parameters.

`GET /train/rest/schedule/NYP/WAS/05-02-2011`

32

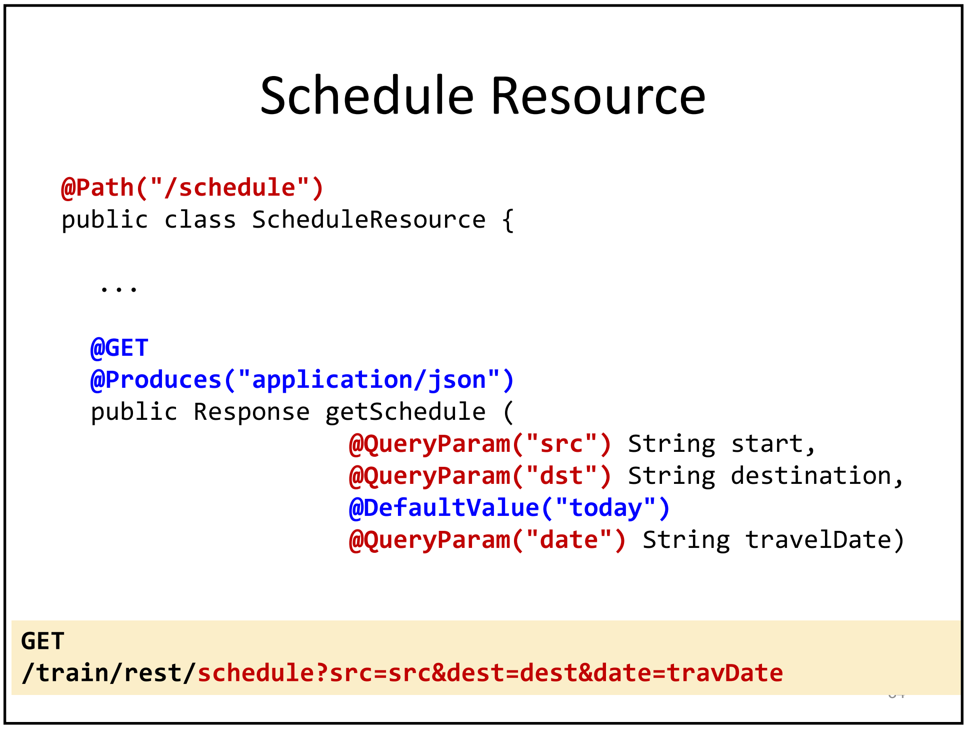


Schedule Resource

```
@Path("/schedule")
public class ScheduleResource {

   ...

  @GET
  @Produces("application/json")
  public Response getSchedule (
                    @QueryParam("src") String start,
                    @QueryParam("dst") String destination,
                    @DefaultValue("today")
                    @QueryParam("date") String travelDate)
```

GET /train/rest/schedule?src=src&dest=dest&date=travDate

33

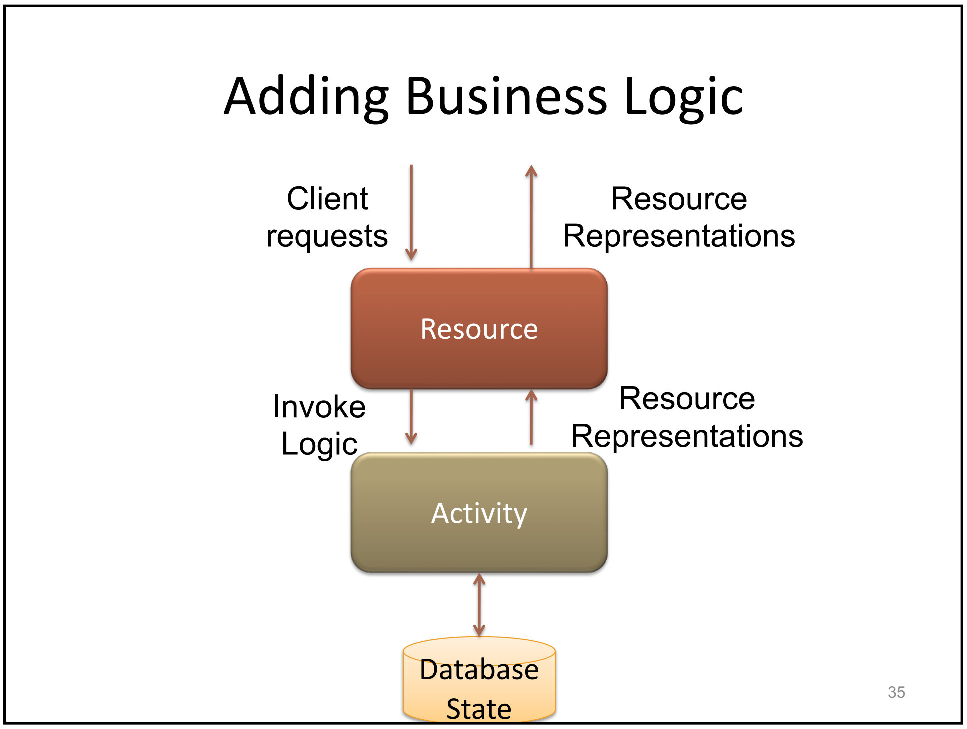


Schedule Resource

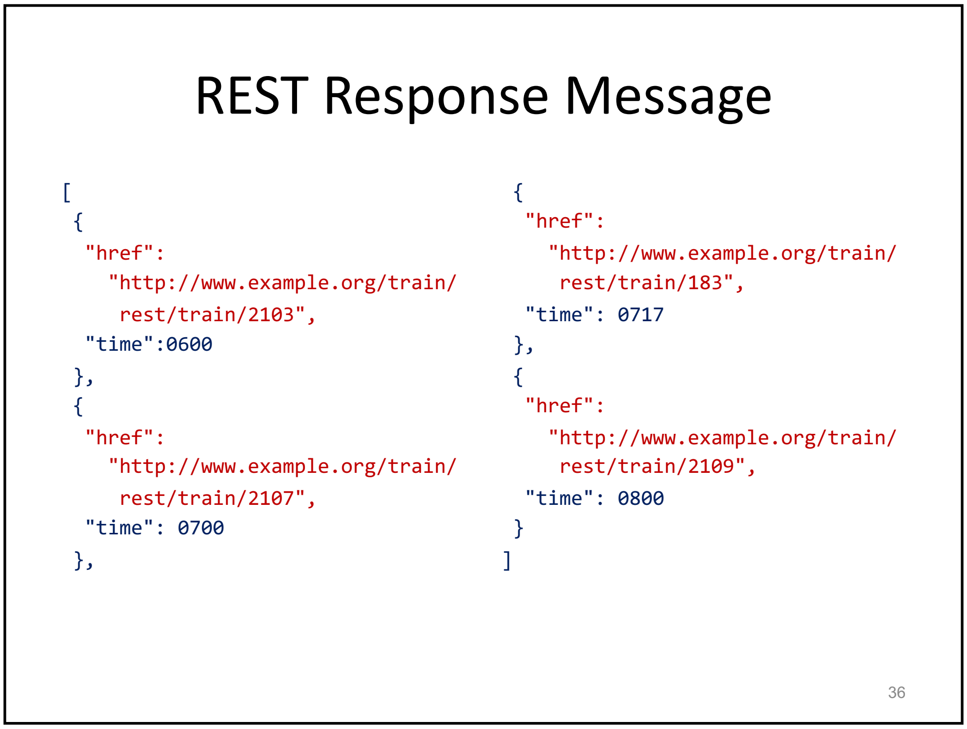
```
@Path("/schedule")
public class ScheduleResource {

   ...

  @GET
  @Produces("application/json")
  public Response getSchedule (
                    @QueryParam("src") String start,
                    @QueryParam("dst") String destination,
                    @DefaultValue("today")
                    @QueryParam("date") String travelDate)
```

GET /train/rest/schedule?src=src&dest=dest&date=travDate

34

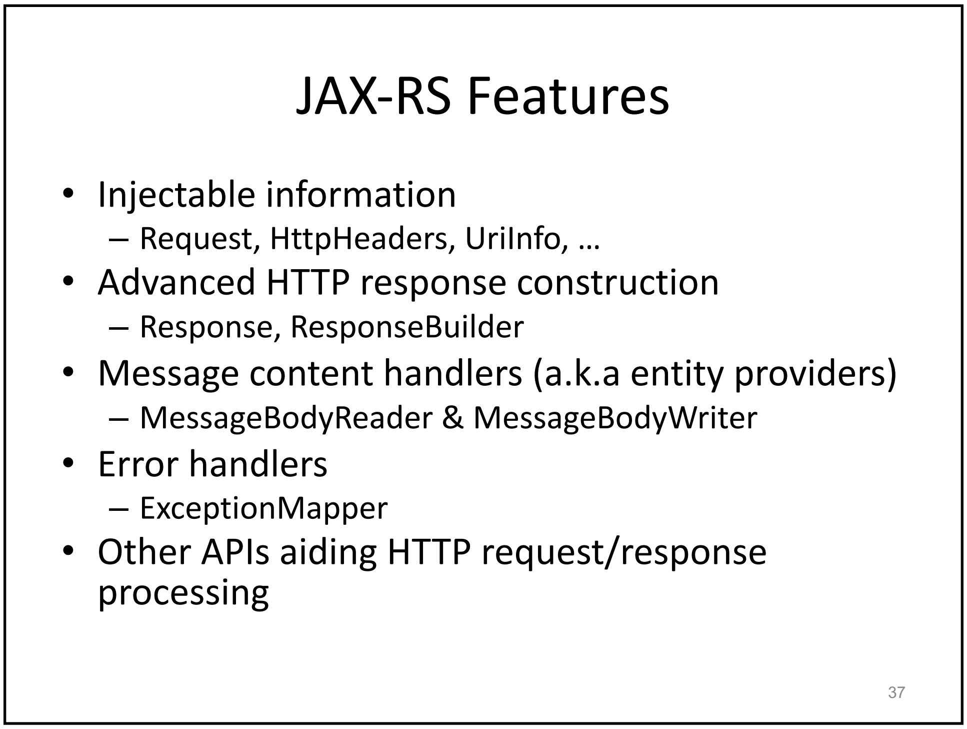


Adding Business Logic



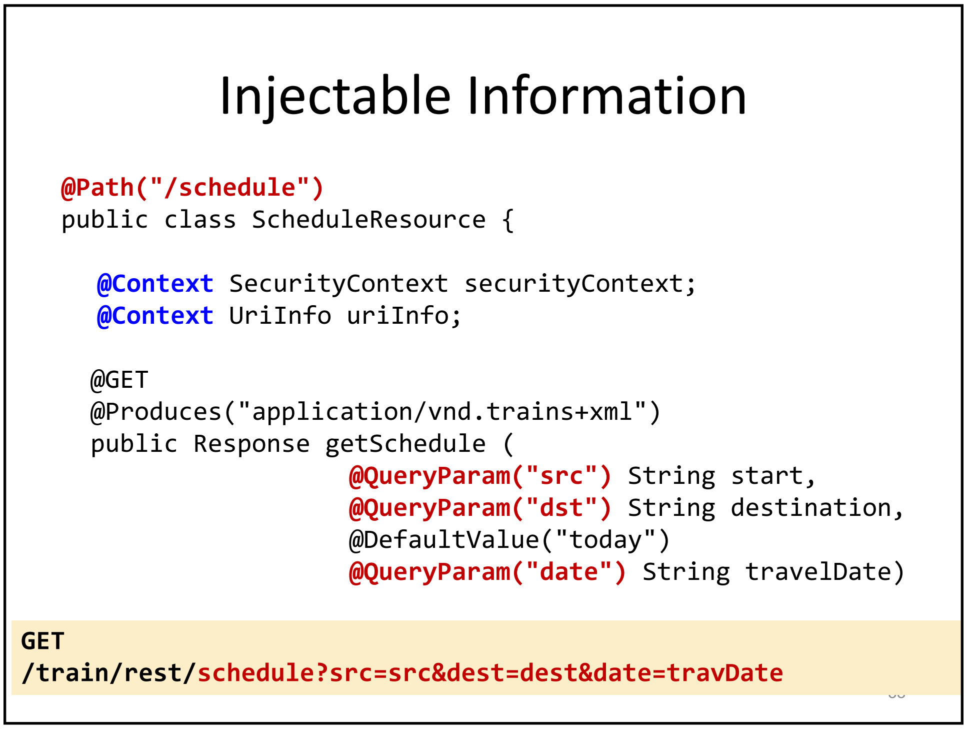
REST Response Message

```
[
 {
  "href":
    "http://www.example.org/train/
     rest/train/2103",
  "time":0600
 },
 {
  "href":
    "http://www.example.org/train/
     rest/train/2107",
  "time": 0700
 },
 {
  "href":
    "http://www.example.org/train/
     rest/train/183",
  "time": 0717
 },
 {
  "href":
    "http://www.example.org/train/
     rest/train/2109",
  "time": 0800
 }
]
```



JAX-RS Features

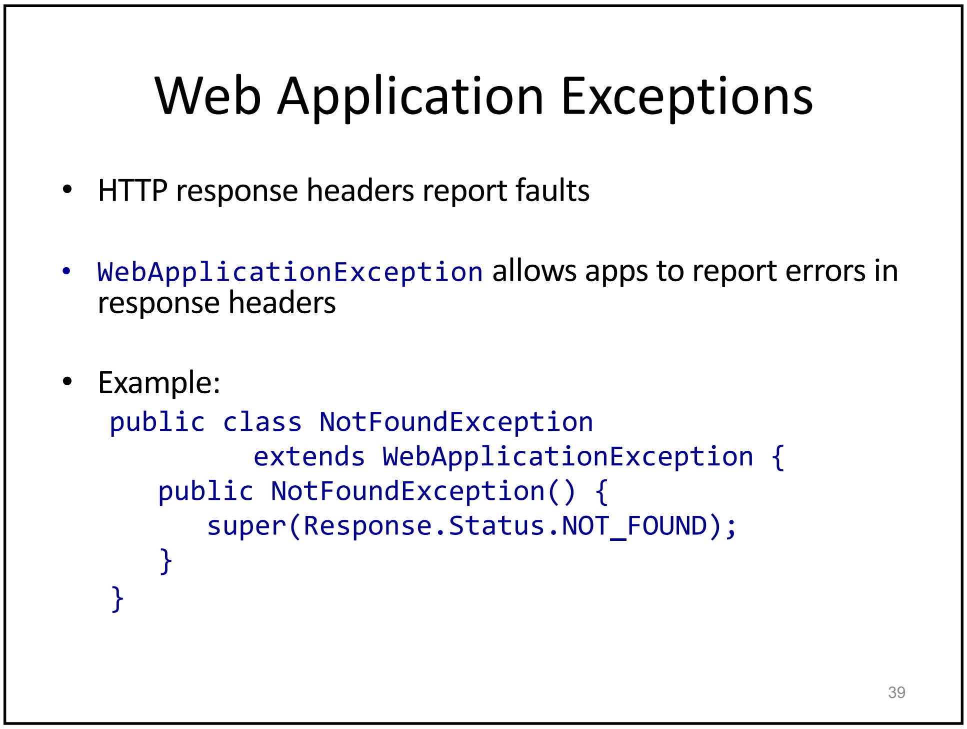
- Injectable information
  - Request, HttpHeaders, UriInfo, …
- Advanced HTTP response construction
  - Response, ResponseBuilder
- Message content handlers (a.k.a entity providers)
  - MessageBodyReader & MessageBodyWriter
- Error handlers
  - ExceptionMapper
- Other APIs aiding HTTP request/response processing



Injectable Information

```
@Path("/schedule")
public class ScheduleResource {

  @Context SecurityContext securityContext;
  @Context UriInfo uriInfo;

  @GET
  @Produces("application/vnd.trains+xml")
  public Response getSchedule (
                    @QueryParam("src") String start,
                    @QueryParam("dst") String destination,
                    @DefaultValue("today")
                    @QueryParam("date") String travelDate)
```

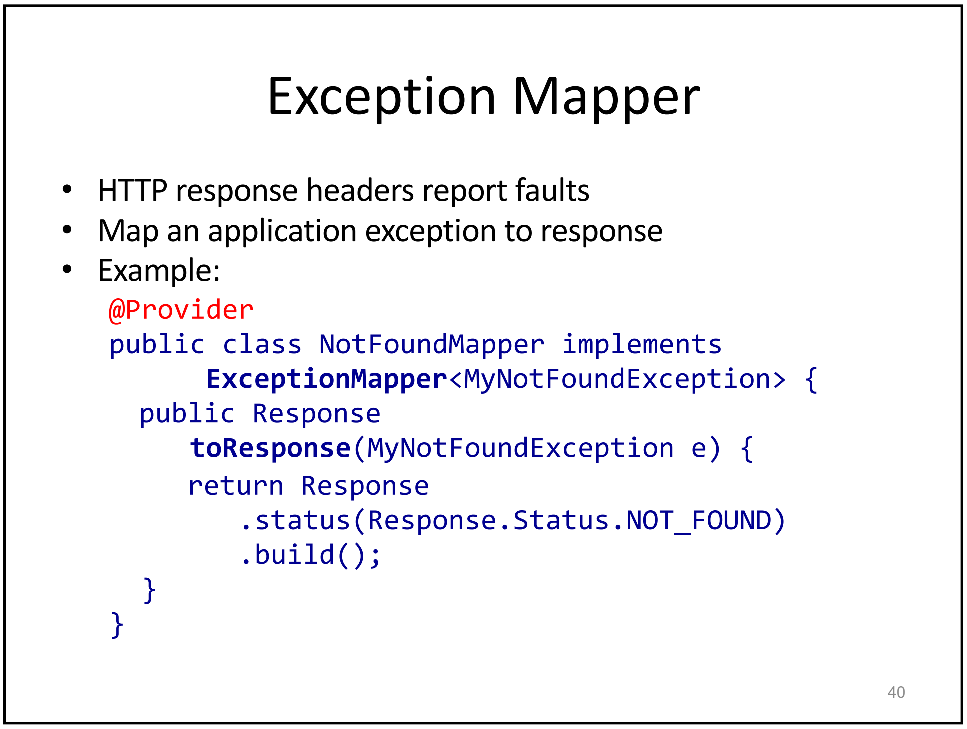
GET /train/rest/schedule?src=src&dest=dest&date=travDate



Web Application Exceptions

- HTTP response headers report faults
- `WebApplicationException` allows apps to report errors in response headers
- Example:

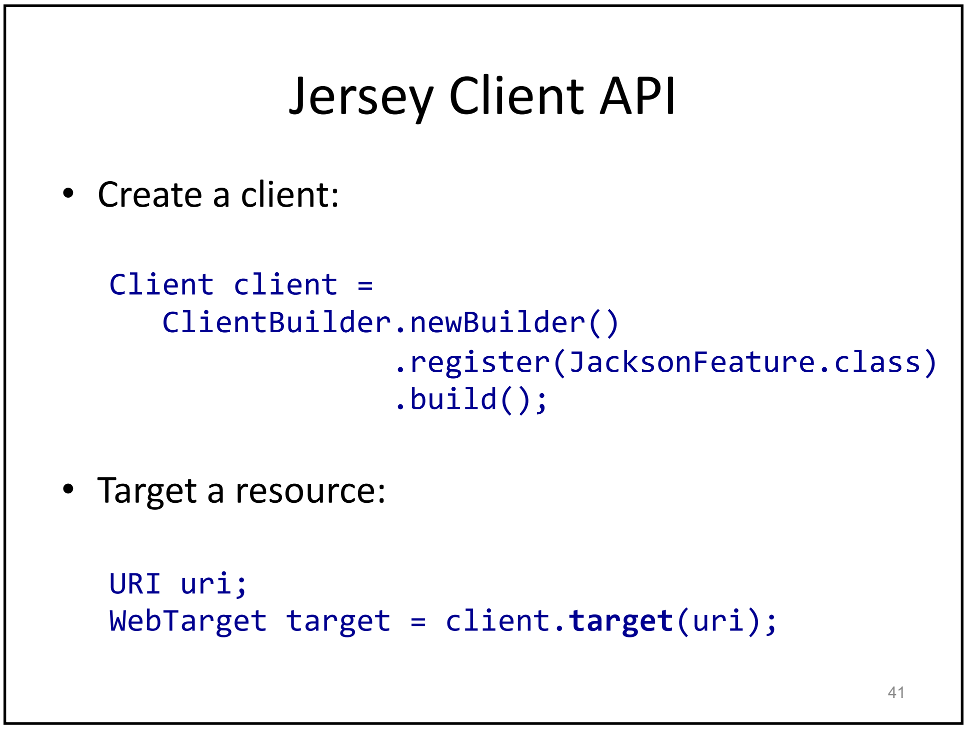
```
public class NotFoundException
        extends WebApplicationException {
   public NotFoundException() {
      super(Response.Status.NOT_FOUND);
   }
}
```



Exception Mapper

- HTTP response headers report faults
- Map an application exception to response
- Example:

```
@Provider
public class NotFoundMapper implements
      ExceptionMapper<MyNotFoundException> {
  public Response
     toResponse(MyNotFoundException e) {
     return Response
        .status(Response.Status.NOT_FOUND)
        .build();
  }
}
```



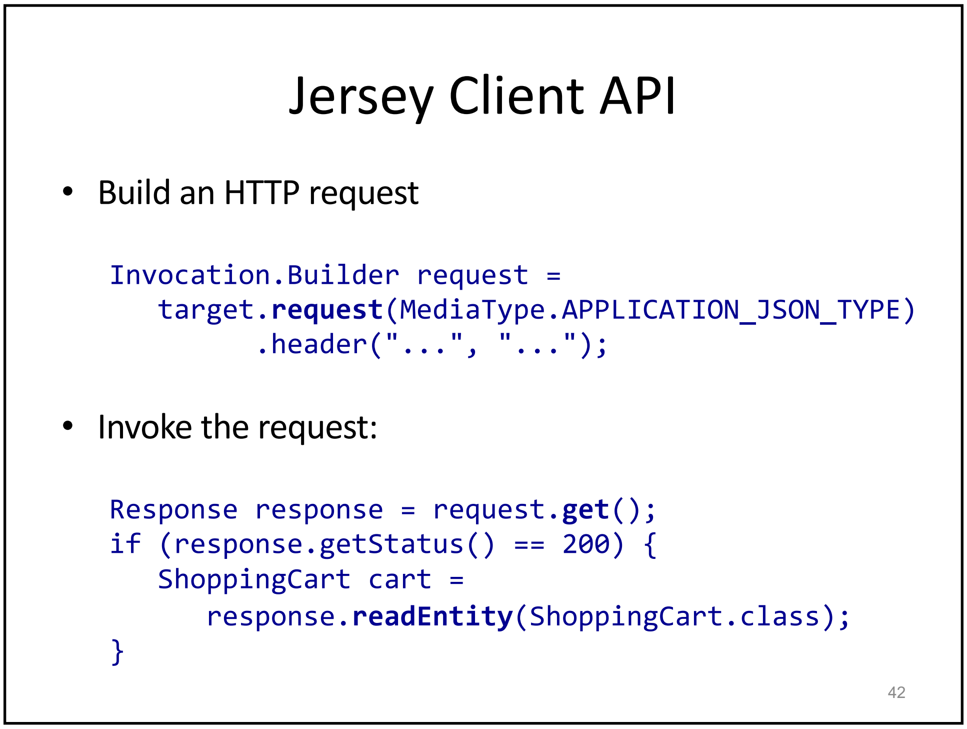
Jersey Client API

- Create a client:

```
Client client =
   ClientBuilder.newBuilder()
                 .register(JacksonFeature.class)
                 .build();
```

- Target a resource:

```
URI uri;
WebTarget target = client.target(uri);
```



Jersey Client API

- Build an HTTP request

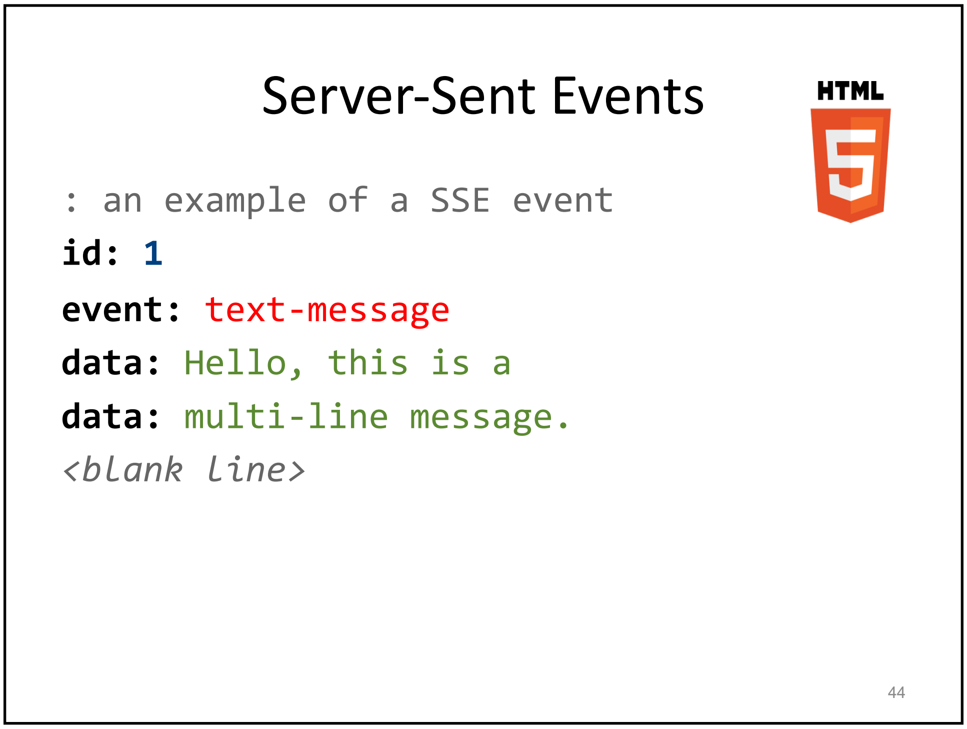
```
Invocation.Builder request =
   target.request(MediaType.APPLICATION_JSON_TYPE)
         .header("...", "...");
```

- Invoke the request:

```
Response response = request.get();
if (response.getStatus() == 200) {
   ShoppingCart cart =
      response.readEntity(ShoppingCart.class);
}
```

SERVER-SENT EVENTS

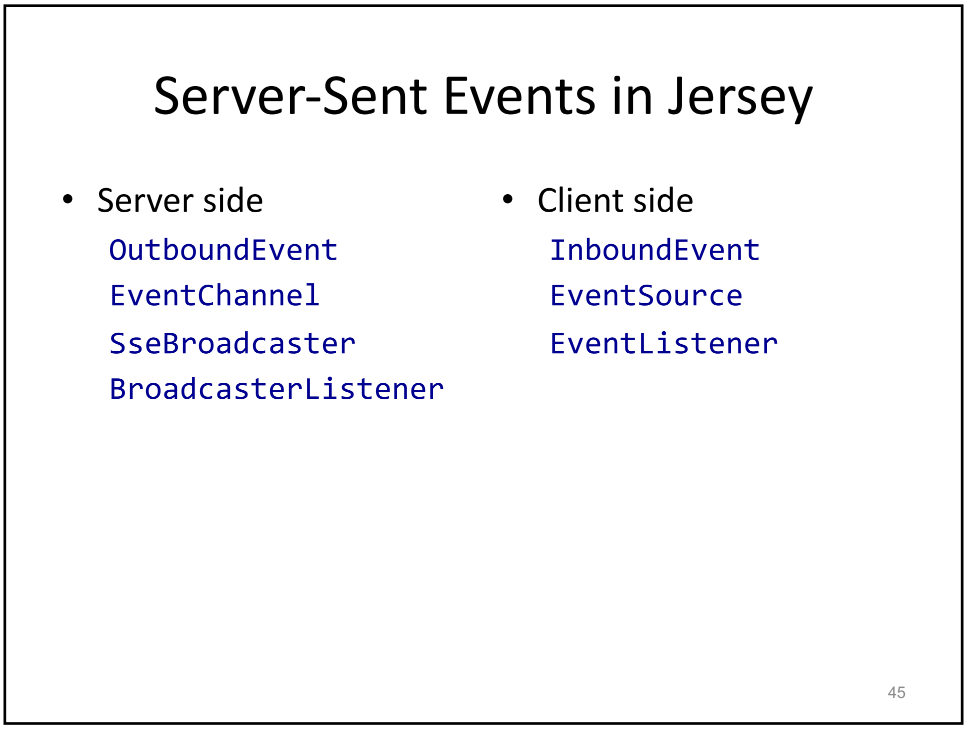
43



Server-Sent Events

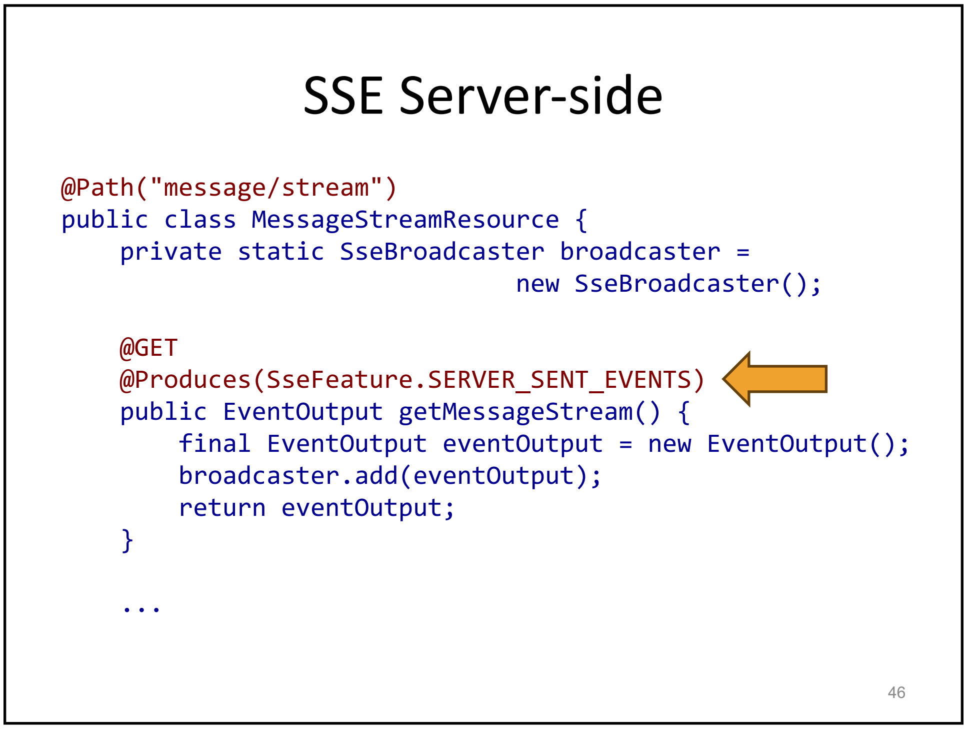
```
: an example of a SSE event
id: 1
event: text-message
data: Hello, this is a
data: multi-line message.
<blank line>
```

44



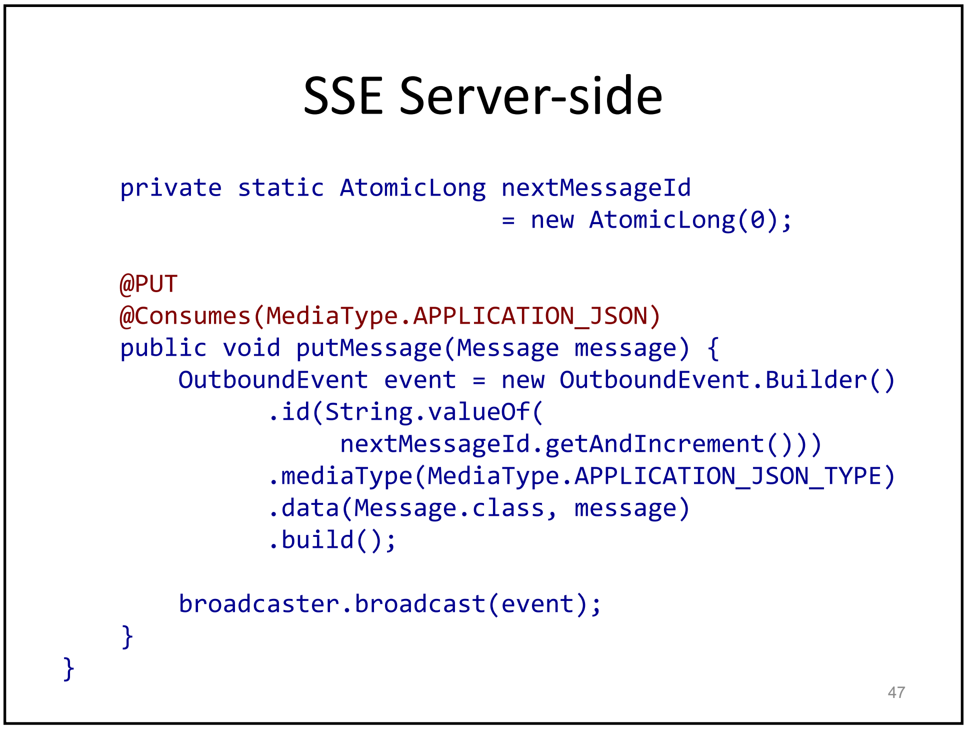
Server-Sent Events in Jersey

- Server side
  `OutboundEvent`
  `EventChannel`
  `SseBroadcaster`
  `BroadcasterListener`
- Client side
  `InboundEvent`
  `EventSource`
  `EventListener`



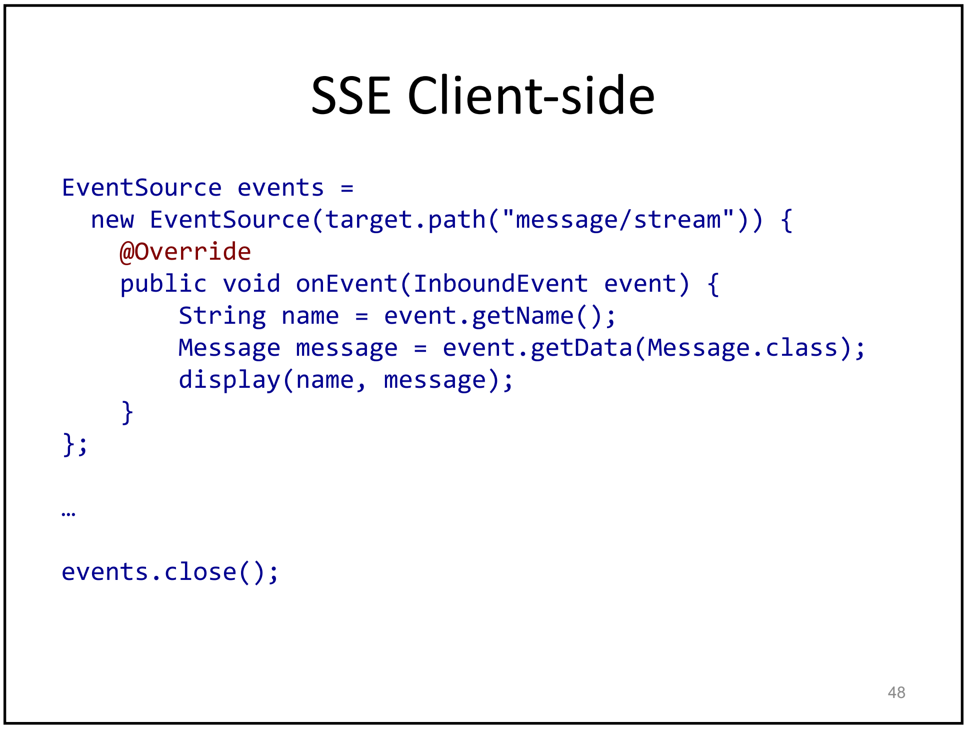
SSE Server-side

```
@Path("message/stream")
public class MessageStreamResource {
    private static SseBroadcaster broadcaster =
                                 new SseBroadcaster();

    @GET
    @Produces(SseFeature.SERVER_SENT_EVENTS)
    public EventOutput getMessageStream() {
        final EventOutput eventOutput = new EventOutput();
        broadcaster.add(eventOutput);
        return eventOutput;
    }

    ...
```



SSE Server-side

```
    private static AtomicLong nextMessageId
                              = new AtomicLong(0);

    @PUT
    @Consumes(MediaType.APPLICATION_JSON)
    public void putMessage(Message message) {
        OutboundEvent event = new OutboundEvent.Builder()
              .id(String.valueOf(
                   nextMessageId.getAndIncrement()))
              .mediaType(MediaType.APPLICATION_JSON_TYPE)
              .data(Message.class, message)
              .build();

        broadcaster.broadcast(event);
    }
}
```



SSE Client-side

```
EventSource events =
  new EventSource(target.path("message/stream")) {
    @Override
    public void onEvent(InboundEvent event) {
        String name = event.getName();
        Message message = event.getData(Message.class);
        display(name, message);
    }
};

…

events.close();
```

Jakarta RESTful Web Services

- Specification (JSR 311, 339, 370): https://jakarta.ee/specifications/restful-ws/
- Reference Implementation: https://eclipse-ee4j.github.io/jersey/