Web Services

Dominic Duggan
Stevens Institute of Technology

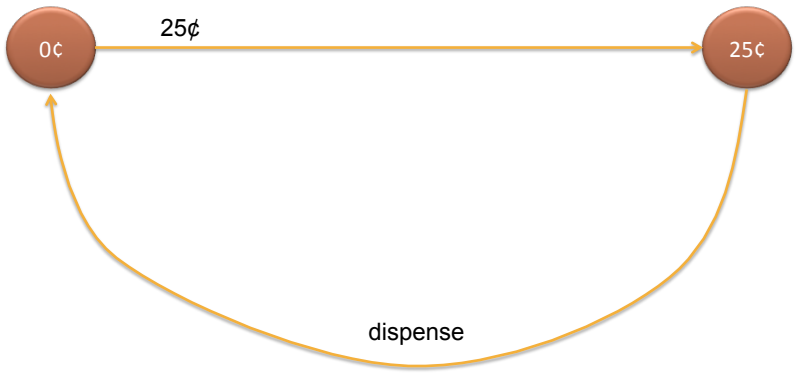
REPRESENTATIONAL STATE TRANSFER (REST)

Representational State Transfer

- Software architecture for the Web
- Web browsing as navigation of hypermedia network

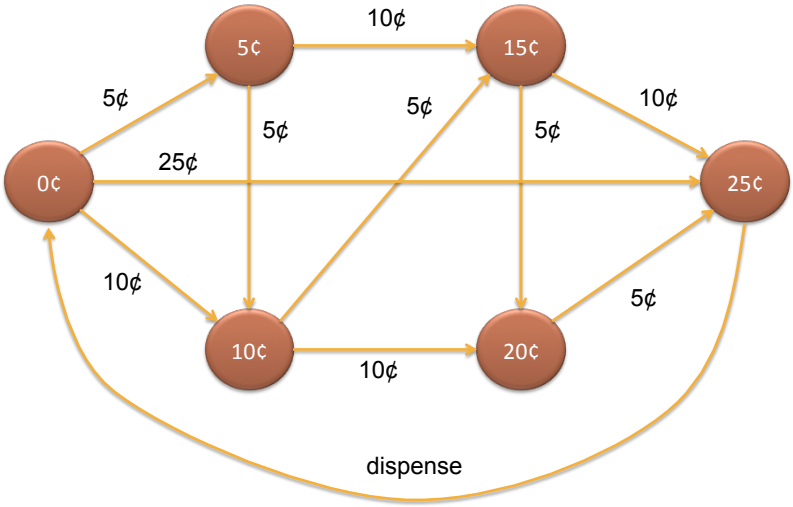
3

State Machine



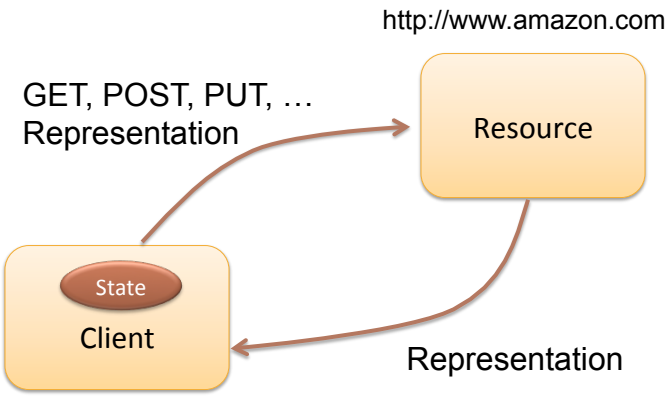
4

State Machine



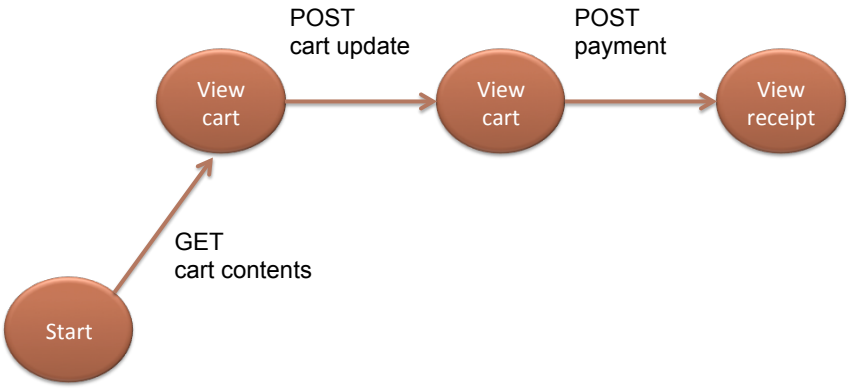
5

Representational State Transfer



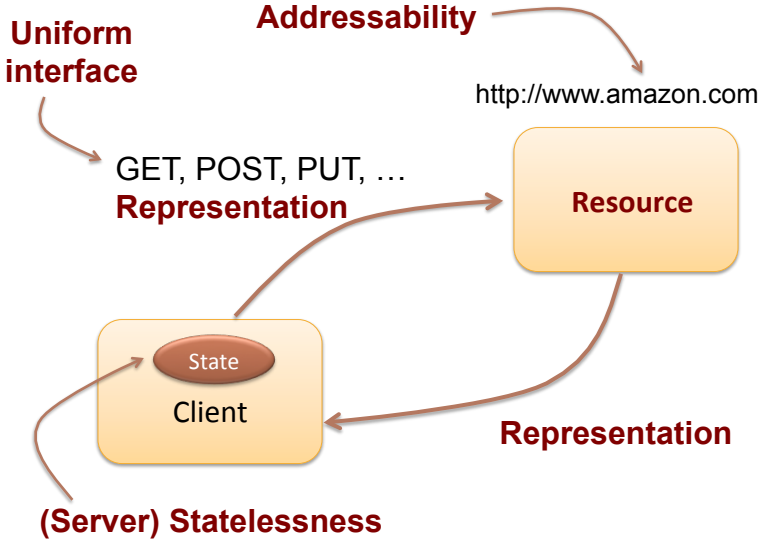
6

Client as State Machine



7

Representational State Transfer



8

Representational State Transfer

- Originally a software architecture for the Web
- Emerged as an alternative architecture for Web services
  - Resource-oriented architecture

SOAP / WSDL	REST
Service (operation) oriented	Resource oriented
One endpoint URL	URL for each individual resource
Application-defined verbs	Fixed set of HTTP verbs

9

REST Verbs

- Retrieve: HTTP GET
- Create:
  - HTTP PUT for new URI or
  - HTTP POST for existing URI (server decides result URI)
- Modify: HTTP PUT to existing URI
- Delete: HTTP DELETE

- Merge updates: HTTP PATCH
- Retrieve metadata only: HTTP HEAD
- Check which methods are supported: HTTP OPTIONS

- No other operations besides these

10

Example: Amazon Simple Storage Service (S3)

- S3 is based on two concepts
  - Buckets
    - Named container
  - Objects
    - Named piece of data, with metadata
    - Stored in buckets

11

S3 RPC Interface

- Object-oriented interface to S3
  - `CreateBucket`
  - `ListAllMyBuckets`
- Getter/setter methods on bucket and object "objects"
  - `S3Object.name()`
  - `S3Object.setValue()`
  - `S3Bucket.getObjects()`

12

S3 REST Interface

- Three types of resources
  - List of your buckets

    `https://s3.amazonaws.com`
  - A particular bucket (virtual host)

    `https://`*`name-of-bucket`*`.s3.amazonaws.com`
  - A particular s3 object inside a bucket

    `https://`*`name-of-bucket`*`.s3.amazonaws.com/`*`name-of-object`*

13

S3 REST Interface

- Example:
  - A particular bucket

    `https://jeddak.s3.amazonaws.com`
  - A particular s3 object inside a bucket
    - Object names:
      docs/manual.pdf, docs/security.pdf, talks/snt.pdf
    - Resource URIs:
      `https://jeddak.s3.amazonaws.com/docs/manual.pdf`
      `https://jeddak.s3.amazonaws.com/docs/security.pdf`
      `https://jeddak.s3.amazonaws.com/talks/snt.pdf`

14

S3 REST Interface

- Use HTTP methods as verbs

Verb	Bucket list	Bucket	Object
GET	List buckets	List bucket objects	Get value and metadata
HEAD			Get metadata
PUT		Create bucket	Set object value and metadata
DELETE		Delete bucket	Delete object

15

HTTP

16

HTTP Request

```
GET /index.html HTTP/1.1
Host: www.example.org

...request headers...
```

17

HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 1 May 2011 21:38:14 GMT
Server: Apache/1.3.34 (Debian) mod ssl/2.8.25
OpenSSL/0.9.8c ...
Last-Modified: Wed, 25 Nov 2009 12:27:01 GMT
ETag: "7496a6-a0c-4b0d2295"
Accept-Ranges: bytes
Content-Length: 2572
Content-Type: text/html
Via: 1.1 www.example.org
Vary: Accept-Encoding
...
```

18

Request Headers

- Accept: for content negotiation
  - Content-Type: response header e.g.
    ATOM (application/atom+xml)
    RDF (application/rdf+xml)
    XHTML (application/xhtml+xml)
    Form-encoded key-value pairs (application/x-www-form-urlencoded)
- Authorization: app-defined auth info
  - WWW-Authenticate: response header with status code of 401 ("Unauthorized")

19

Request Headers

- Cookie: (non-standard)
  - Save-Cookie: to save cookie on client

20

Response Headers

- Last-modified: time of last modification
  - If-Last-Modified: request header for caching
- Etag: hash of metadata
  - If-None-Match: request header for caching
- Cache-Control: how long to cache
- Upgrade: upgrade protocol e.g. http to https
- Location: URI for newly created resource, redirection, …

21

Response Codes

- 1XX: for negotiation with Web server
  - E.g. 101 ("Switching protocols") with Upgrade: response header
- 2XX: to signal success
  - E.g. 200 ("Success"), 201 ("Created"), …
- 3XX: redirect clients
  - E.g. 303 ("See other"), 307 ("Temporary redirect")
- 4XX: client errors
  - E.g. 400 ("Bad request"), 404 ("Not found"), 401 ("Unauthorized"), 403 ("Forbidden")
- 5XX: server errors
  - E.g. 500 ("Internal server error")

22

JAX-RS API

23

JAX-RS

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

24

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
```

25

Passing Parameters to Service

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

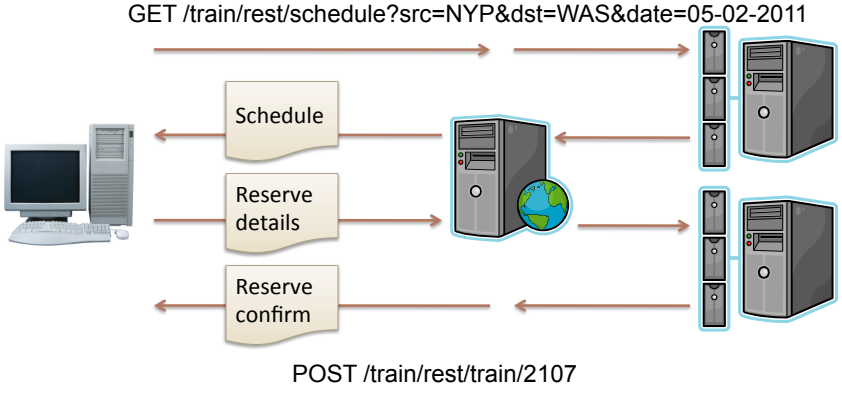
- @BeanParam: inject a bean with all parameters

26

EXAMPLE: TRAIN RESERVATION SERVICE

27

REST Web Service



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

29

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

30

Schedule Resource

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

   ...

  @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

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

  @Context UriInfo uriInfo;

  ... ISchedule scheduleService;

  @GET
  @Produces("application/vnd.trains+xml")
  public Response getSchedule (
                    @QueryParam("src") String start,
                    @QueryParam("dst") String destination,
                    @DefaultValue("today")
                    @QueryParam("date") String travelDate)
    {
      ScheduleRepresentation schedule =
              scheduleService.get(start, destination,
                                  travelDate, uriInfo);
      return Response.ok().entity(schedule).build();
    }
  }
}
```

32

REST Response

```
<tr:Schedule xmlns:tr="http://
   www.example.org/schemas/train">
<tr:train>
  <tr:uri>
    http://www.example.org
    /train/rest/train/2103
  </tr:uri>
  <tr:time>0600</tr:time>
</tr:train>
<tr:train>
  <tr:uri>
    http://www.example.org
    /train/rest/train/2107
  </tr:uri>
  <tr:time>0700</tr:time>
</tr:train>
```

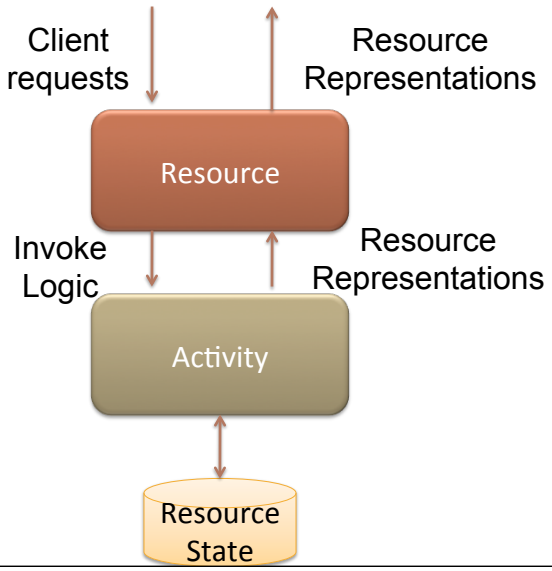
```
<tr:train>
  <tr:uri>
    http://www.example.org
    /train/rest/train/183
  </tr:uri>
  <tr:time>0717</tr:time>
</tr:train>
<tr:train>
  <tr:uri>
    http://www.example.org
    /train/rest/train/2109
  </tr:uri>
  <tr:time>0800</tr:time>
</tr:train>
...
</tr:Schedule>
```

33

ADDING BUSINESS LOGIC

34

Business Activities



35

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

  @Context UriInfo uriInfo;

  ISchedule scheduleService = new ScheduleActivity();

  @GET
  @Produces("application/vnd.trains+xml")
  public Response getSchedule (
                    @QueryParam("src") String start,
                    @QueryParam("dst") String destination,
                    @DefaultValue("today")
                    @QueryParam("date") String travelDate)
    {
      ScheduleRepresentation schedule =
              scheduleService.get(start, destination,
                                  travelDate, uriInfo);
      return Response.ok().entity(schedule).build();
    }
  }
}
```

36

Schedule Activity

```
public interface ISchedule {
  public ScheduleRepresentation get (
      String start, String destination, String travelDate,
      UriInfo ui );
}

public class ScheduleActivity implements ISchedule {
  public ScheduleRepresentation get (
      String start, String destination, String travelDate,
      UriInfo ui ) {
    List<Train> trains = ...;
    UriBuilder ub = ui.getBaseUriBuilder();
    ub = ub.path("train").path("{tid}");
    return new ScheduleRepresentation (trains, ub);
  }
}
```

37

Train Resource

```
@Path("/train")
public class TrainResource {
  @Context UriInfo uriInfo;
  ... ITrain trainService;
  @POST
  @Path("/{tid}")
  @Consumes("application/vnd.trains+xml")
  public Response makeReservation (
      @PathParam("tid") String tid,
      @DefaultValue("business")
      @QueryParam("class") String _class,
      String reserveInfo)
    URI reserveURI =
      trainService.reserve(tid, _class, uriInfo, reserveInfo);
    return Response.created(reserveURI).build();
  }
}
```

POST /train/rest/train/*tid*[?class=*class*]

38

Reservation Resource

```
@Path("/reservation")
public class ReservationResource {
  @Context UriInfo uriInfo;
  ... IReservation reservationService;
  @GET
  @PATH("/{rid}")
  @Produces("application/vnd.trains+xml")
  public ReservationRepresentation getReservation (
      @PathParam("rid") String rid )
  {
    ReservationRepresentation reservation =
      reservationService.get(rid, uriInfo);
    return reservation;
  }
```

GET /train/rest/reservation/*rid*

MANAGING STATE

Stateful Web Service

- `@PerSession` annotation (Jersey)
  - HTTP session shared between client & service
  - But violates REST
- Alternative: Give each shopping cart a URI
  - Now application must manage state

41

State Considerations

- Where to store state?
  - In a local file
    - Does not scale to a cluster of app servers
  - On a backend database
    - Need to identify state in the DB
- How to identify state?
  - Session identifier in client cookie
    - Violates REST
  - Path parameter in URI that identifies the state as a resource

42

Persistent Domain Object (PDO)

- Domain entity persisted to DB
  - Object = row
  - Instance variable = column
- Entity class annotated with @Entity

```
@Entity
@Table(name = "SHOPPING_CART")
public class CartEntity implements Serializable
{
   ...
}
```

43

Entity Manager

```
@PersistenceContext(unitName = "ShoppingCartPU")
EntityManager em;

void beginTransaction() {
  em.getTransaction().begin();
}

CartEntity cart = em.find(CartEntity.class, ...);

void endTransaction() {
  em.getTransaction().commit();
}
```

Dependency Injection

44

Transactions

- All updates on a persistence context occur in the context of a transaction
  - Application-managed: explicit
  - Container-managed: lifespan of business objects (default)
- Either commit all changes at the end…
- …or abort, roll back all changes on DB

45

RESTFUL SHOPPING CART

46

RESTful Shopping Cart

```
@Path("/rest/shoppingcart/{cartId}")
public class ShoppingCartResource {
   @PathParam("cartId") String cartId;
   ... EntityManager em;
   @PUT
   public Response newCart () {









   }
}
```

47

RESTful Shopping Cart

```
@Path("/rest/shoppingcart/{cartId}")
public class ShoppingCartResource {
   @PathParam("cartId") String cartId;
   ... EntityManager em;
   @PUT
   public Response newCart () {
        long cartFk = Long.parseLong(cartId);
        CartEntity cart = em.find(CartEntity.class, cartId);
        if (cart != null)
           throw new WebApplicationException
                     (Response.Status.FORBIDDEN);
        cart = new CartEntity(cartId);
        em.persist(cart);
        return Response.status(Response.Status.CREATED);
   }
}
```

48

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);
   }
}
```

49

RESTful Shopping Cart

```
@GET
@Produces("application/json")
public CartRepresentation getCart () {
   CartEntity cart = em.find(CartEntity.class, cartId);
   if (cart == null) throw new NotFoundException();
   return new CartRepresentation(cart);
}
@DELETE
public Response deleteCart () {
   CartEntity cart = em.find(CartEntity.class, cartId);
   if (cart == null) throw new NotFoundException();
   em.remove(cart);
   return Response.ok();
}
```

50

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();
  }
}
```

51

RESTful Shopping Cart

```
@GET
@Produces("application/json”)
public CartRepresentation getCart () {
   CartEntity cart = em.find(CartEntity.class, cartId);
   if (cart == null) throw new MyNotFoundException();
   return new CartRepresentation(cart);
}
@DELETE
public Response deleteCart () {
   CartEntity cart = em.find(CartEntity.class, cartId);
   if (cart == null) throw new MyNotFoundException();
   em.remove(cart);
   return Response.ok();
}
```

52

REST CLIENT API

53

Jersey Client API

- Create a shopping cart:

```
Client client = ClientBuilder.newClient();

UriBuilder uriBase =
  UriBuilder.fromURI
  ("http://www.jeddak.org/rest/shoppingcart/joe");
URI resourceUri = uriBase.build();
WebTarget cart = client.target(resourceUri);

cart.request().put(Entity.text(“”));
```

54

Jersey Client API

- Add a film to the shopping cart:

```
Form f1 = new Form();
f1.add("product","Lawrence of Arabia");
f1.add("amount","24.95");

URI addUri = uriBase.path("add").build();
WebTarget cartAdd = c.target(addUri);

Entity<Form> f1Entity = Entity.entity(f1,
   MediaType.APPLICATION_FORM_URLENCODED_TYPE);
cartAdd.request().post(f1Entity);
```

55

Jersey Client API

- Add a book and review the shopping cart:

```
Form f2 = new Form();
f2.add("product","A Fistful of Dollars");
f2.add("amount","19.95");

Entity<Form> f2Entity = Entity.entity(f2,
   MediaType.APPLICATION_FORM_URLENCODED_TYPE);
cartAdd.request().post(f2Entity);

cart.request(MediaType.APPLICATION_JSON_TYPE);
    .get(CartEntity.class);
```

56

Catching Errors

```
Response response = cartAdd.request().post();

if (response.getStatusInfo() ==
    Response.Status.NO_CONTENT) {

  response = cart.get();
  if (response.getStatusInfo() == Response.Status.OK) {

      CartEntity cartEntity =
            response.getEntity(CartEntity.class);
      ...
  }
}
```

57