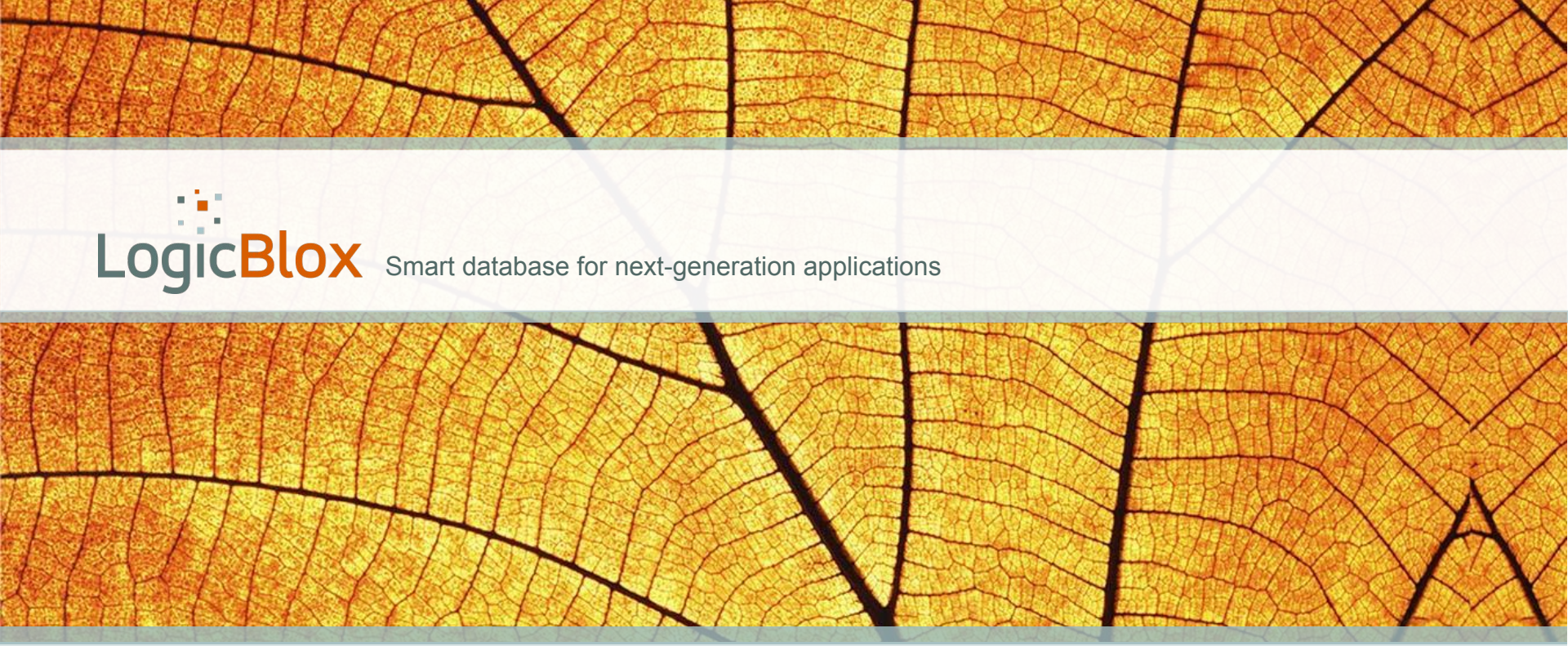


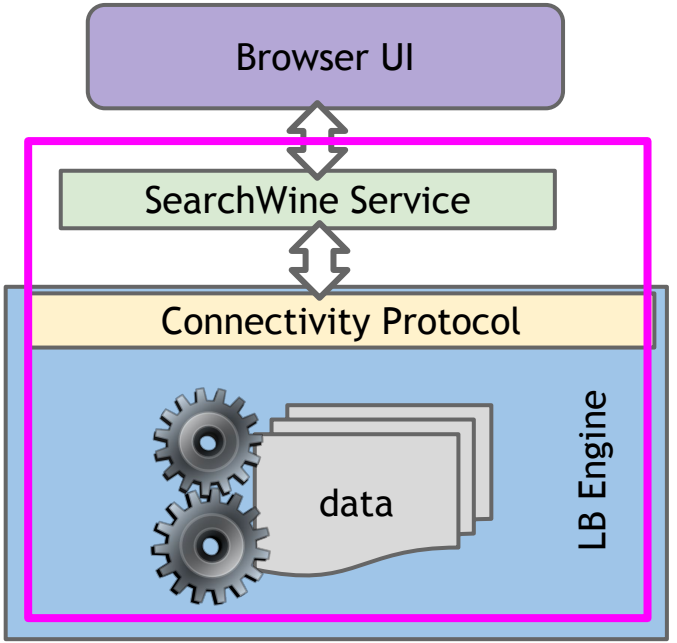
LogicBlox

Smart database for next-generation applications

LESSON 5 : BUILDING AND TESTING SERVICES

LOGICBLOX TRAINING

DISCOUNTWINES ARCHITECTURE



ANATOMY OF A SERVICE

- request/response protocol
- logic : given request, compute response
- other:
  - spin up a webserver
  - scalability to concurrent users
  - availability
  - proper logging
  - error handling and recovery
  - ...

ANATOMY OF A SERVICE

Essential Complexity

- request/response protocol
- logic : given request, compute response

Operational Complexity

- other:
  - spin up a webserver
  - scalability to concurrent users
  - availability
  - proper logging
  - error handling and recovery
  - ...

LOGICBLOX SERVICE FRAMEWORK

Essential Complexity

- request/response protocol
- logic : given request, compute response

- Lb web : a service framework
  - shared among all LogicBlox Services
  - only need to configure with
    - URI
    - protocol
    - encoding

A QUICK DEMO

```
% curl -iv \
    -X POST \
    -H 'Content-Type: application/json' \
    -H 'Accept: application/json' \
    -d '{}' \
    http://localhost:8080/dw/searchWines
```

```
POST /dw/searchWines HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Accept: application/json
Content-Length: 2

HTTP/1.1 200 OK

{"wine":
  [{"id": 1000,
    "description": "San Martino Riserva 2007",
    "year": 2007,
    "cost": 3.6,
    "retail": 6.99,"margin": 3.39}, ...]}
```

REQUEST & RESPONSE PROTOCOL

- specification by Google protocol buffer

logiql/protocols/searchWines.proto

```
package protocols.searchWines;

message Request {
}


message Response {
   repeated Wine wine = 1;
}

message Wine {
   required int64 id = 1;
   required string description = 2;
   required int64 year = 3;

   required double cost = 6;
   required double retail = 7;
   required double margin = 8;
}
```

READ REQUEST, WRITE RESPONSE

logiql/protocols/searchWines.proto

```
package protocols.searchWines;

message Request {
}


message Response {
   repeated Wine wine = 1;
}

message Wine { ... }
```

```
% protoc  --
java_out=...
```

searchWines.java

```
class Request {
  ...
}
class Response {
  Wine getWineList() { ... }
  Response addWine(int i, Wine w) { ... }
}
class Wine {
  ...
}
```

READ REQUEST, WRITE RESPONSE

protocols/searchWines.proto

```
package protocols.searchWines;

message Request {
}


message Response {
   repeated Wine wine = 1;
}

message Wine { ... }
```

% proto2datalog

searchWines_proto.logic

```
protocols:searchWines:Request(x) -> .

protocols:searchWines:Response(x) -> .

protocols:searchWines:Response:wine[x,i] = y
  -> protocols:searchWines:Response(x),
     int(i),
     protocols:searchWines:Wine(y).

protocols:searchWines:Wine(x) -> .
```

% protoc
--java_out=...

searchWines.java

```
class Request {
  ...
}
class Response {
  Wine getWineList() { ... }
  Response addWine(int i, Wine w) { ... }
}
class Wine {
  ...
}
```

READ & WRITE PROTO PREDICATES

- state affected by external world : EDBs
  - logic for read/write needs to be delta rules

```
/****** On Request, create Response ******/
+ResponseConstructor[req]= response,
+protocols:searchWines:Response(response)
  <- +protocols:searchWines:Request(_).

/****** create Wine message **************/
 +WineConstructor[id]= result,
 +protocols:searchWines:Wine(result),
 +protocols:searchWines:Wine_id(result,id),
 +protocols:searchWines:Response_wine(response,id,result)
      <-
          +protocols:searchWines:Response(response),
          wine:wine_id(wine : id).
```

CLOSE LOOK : logiql/services/protobuf/searchWines.logic

```
block(`searchWines) {
  alias_all(`core),
  clauses(`{
    ResponseConstructor[req]= response ->
      protocols:searchWines:Request(req) ,
      protocols:searchWines:Response(response).
    lang:constructor(`ResponseConstructor).
    lang:pulse(`ResponseConstructor).

    WineConstructor[id]= wine -> int(id), protocols:searchWines:Wine(wine).
    lang:constructor(`WineConstructor).
    lang:pulse(`WineConstructor).

    /** on Request, create a Response **/
    +ResponseConstructor[req]= response,
    +protocols:searchWines:Response(response)
      <-
      +protocols:searchWines:Request(req).

    /** create Wine message **/
    +WineConstructor[id]= result,
    +protocols:searchWines:Wine(result),
    +protocols:searchWines:Wine_id(result,id),
    +protocols:searchWines:Wine_description(result,description),
    +protocols:searchWines:Wine_year(result,year_id),
    +protocols:searchWines:Wine_cost(result,cost),
    +protocols:searchWines:Wine_retail(result,retail),
    +protocols:searchWines:Wine_margin(result,margin),
    +protocols:searchWines:Response_wine(response,id,result)
      <-
      +protocols:searchWines:Response(response),
      wine:wine_id(wine : id),
      wine:hasDescription(wine,description),
      wine:ofYear(wine,year),
      attributes:year_id[year]=year_id,
      wine:cost(wine,cost),
      wine:retail(wine,retail),
      wine:margin(wine,margin).
  })
} <-- .
```

SERVICE CONFIGURATION

logiql/services/service_config.logic

```
default_protobuf_service(s),

/**** service URI *********/
service_by_prefix["/dw/searchWines"] = s,

/**** service protocol ****/
protobuf_protocol[s] = "protocols:searchWines",
protobuf_request_message[s] = "Request",
protobuf_response_message[s] = "Response",

/**** service encoding *****/
protobuf_encoding[s] = "JSON".
```

SNEAK PEEK : HIERARCHICAL SYNTAX

flat

```
default_protobuf_service(s),

/**** service URI *********/
service_by_prefix["/dw/searchWines"]
                    = s,

/**** service protocol ****/
protobuf_protocol[s] = "protocols:searchWines",
protobuf_request_message[s]
                    = "Request",
protobuf_response_message[s]
                    = "Response",

/**** service encoding *****/
protobuf_encoding[s] = "JSON".
```

SNEAK PEEK : HIERARCHICAL SYNTAX

flat

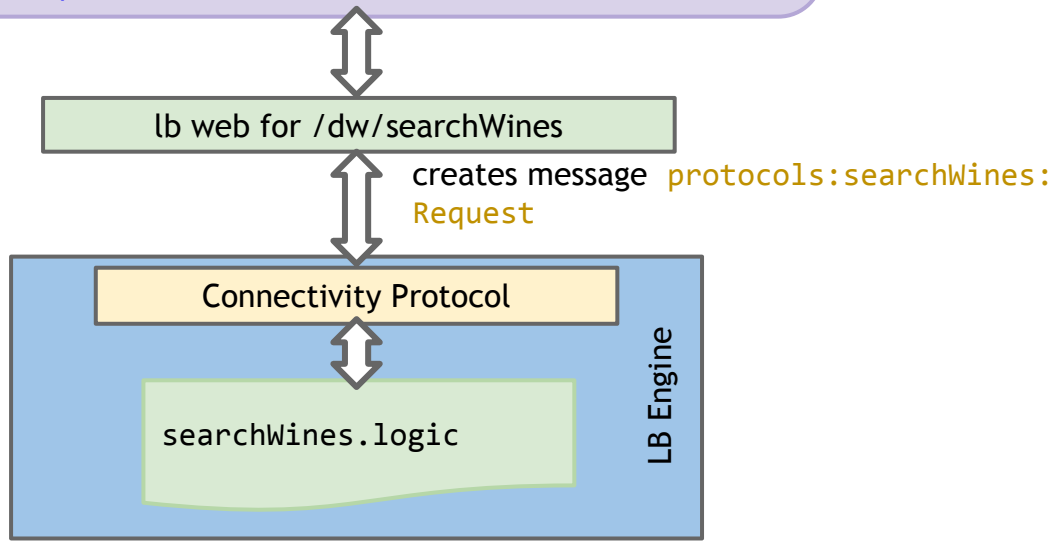
```
default_protobuf_service(s),

/**** service URI *********/
service_by_prefix["/dw/searchWines"]
                    = s,

/**** service protocol ****/
protobuf_protocol[s] = "searchWines",
protobuf_request_message[s]
                    = "Request",
protobuf_response_message[s]
                    = "Response",

/**** service encoding *****/
protobuf_encoding[s] = "JSON".
```

hierarchical

```
default_protobuf_service(s) {

  /**** service URI *********/
  service_by_prefix("/dw/searchWines"),


  /**** service protocol ****/
  protobuf_protocol("protocols:searchWines"),
  protobuf_request_message("Request"),

  protobuf_response_message("Response"),


  /**** service encoding *****/
  protobuf_encoding[] = "JSON"

}.
```

THE ARCHITECTURAL VIEW

```
% curl -iv \
    -X POST \
    -H 'Content-Type: application/json' \
    -H 'Accept: application/json' \
    -d '{}' \
    http://localhost:8080/dw/searchWines
```



LOGGING REQUEST & RESPONSE

```
% lb web-server log messages

% tail -f $LB_DEPLOYMENT_HOME/logs/current/lb-web-server.log
<6>2016-04-08 11:28:55,00000+00:00  INFO ProtoBuf          - {}
<6>2016-04-08 11:28:55,00000+00:00  INFO ProtoBuf          - \------------------------ json request (id:6, service:/dw/searchWines, ws:
discountwines) ------------------------/
<6>2016-04-08 11:28:55,00000+00:00  INFO ProtoBuf          - /------------------------- request (id:6, service:/dw/searchWines, ws:
discountwines) -------------------------\
<6>2016-04-08 11:28:55,00000+00:00  INFO ProtoBuf          - \------------------------- request (id:6, service:/dw/searchWines, ws:
discountwines) -------------------------/
<6>2016-04-08 11:28:55,16300+00:00  INFO ProtoBuf          - /------------------------- response (id:6, service:/dw/searchWines, ws:
discountwines) -------------------------\
<6>2016-04-08 11:28:55,16300+00:00  INFO ProtoBuf          - wine {
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   id: 1000
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   description: "San Martino Riserva 2007"
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   year: 2007
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   cost: 3.6
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   retail: 6.99
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   margin_percent: 94.0
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   origin: "Italy"
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   type: "red"
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          - }
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          - wine {
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   id: 1001
<6>2016-04-08 11:28:55,16400+00:00  INFO ProtoBuf          -   description: "Santa Ana Chardonnay Viognier 2010"
<6>2016-04-08 11:28:55,16500+00:00  INFO ProtoBuf          -   year: 2010
<6>2016-04-08 11:28:55,16500+00:00  INFO ProtoBuf          -   cost: 3.6
                ...
<6>2016-04-08 11:28:55,17100+00:00  INFO ProtoBuf          - \------------------------- response (id:6, service:/dw/searchWines, ws:
discountwines) -------------------------/
```

A CODE ORGANIZATION VIEW

- request/response protocol

```
logiql/protocols/searchWines.proto
```

- service logic & configuration

```
logiql/
  services/
     service_config.logic
     protobuf/
        searchWines.logic
  protocols/
```

CORE PROJECT : A CLOSER LOOK

core.project

```
core, projectname

lb_web, library

services/protocols/searchWines.proto, proto ,
descName=protocols:searchWines lifetime=transaction

core, module
services, module
```

TESTING YOUR SERVICE

```
class TestBasicService(bloxweb.testcase.PrototypeWorkspaceTestCase):

  prototype = "discountwines"

  def setUp(self):
    super(TestBasicService, self).setUp()
    self.client = lb.web.service.Client("localhost", 8080, "/dw/searchWines")

  def test_getWines(self):
    req = '{}'
    response = self.client.call_json(req)
    r = json.loads(response)
    print json.dumps(r, sort_keys = True, indent=4, separators=(',', ':'))
    self.assert_("wine" in r)
```

TESTING YOUR SERVICE

```
class TestBasicService(bloxweb.testcase.PrototypeWorkspaceTestCase):

  prototype = "discountwines"

  def setUp(self):
    super(TestBasicService, self).setUp()
    self.client = lb.web.service.Client("localhost", 8080, "/dw/searchWines")

  def test_getWines(self):
    req = '{}'
    response = self.client.call_json(req)
    r = json.loads(response)
    print json.dumps(r, sort_keys = True, indent=4, separators=(',', ':'))
    self.assert_("wine" in r)
```

INSPECTING JSON IN TESTS

```
...  print json.dumps(r, sort_keys = True, indent=4, separators=(',', ':'))  ...
```

INSPECTING JSON IN TESTS

```
...  print json.dumps(r, sort_keys = True, indent=4, separators=(',', ':'))  ...
```

```
{
  "wine" : [
    {
      "cost" : 3.6,
      "description" : "San Martino ...",
      "id" : 1000,
      "margin" : 3.39,
      "retail" : 6.99,
      "year" : 2007
    },
    {
      "cost" : 3.6,
        ...
    },
        ...
  ]
}
```

INSPECTING JSON IN TESTS

```
...  print json.dumps(r, sort_keys = True, indent=4, separators=(',', ':'))  ...
```

```
{
  "wine" : [
    {
      "cost" : 3.6,
      "description" : "San Martino ...",
      "id" : 1000,
      "margin" : 3.39,
      "retail" : 6.99,
      "year" : 2007
    },
    {
      "cost" : 3.6,
        ...
    },
        ...
  ]
}
```

```
self.assert_("wine" in r)
```

INSPECTING JSON IN TESTS

```
...  print json.dumps(r, sort_keys = True, indent=4, separators=(',', ':'))  ...
```

```
{
  "wine" : [
    {
      "cost" : 3.6,
      "description" : "San Martino ...",
      "id" : 1000,
      "margin" : 3.39,
      "retail" : 6.99,
      "year" : 2007
    },
    {
      "cost" : 3.6,
        ...
    },
        ...
  ]
}
```

```
self.assert_("wine" in r)

for wine in r["wine"] :

  self.assert_("cost" in wine)
```

INSPECTING JSON IN TESTS

```
...  print json.dumps(r, indent=4, separators=(',', ':')) ...
```

```
{
  "wine" : [
    {
      "cost" : 3.6,
      "description" : "San Martino ...",
      "id" : 1000,
      "margin" : 3.39,
      "retail" : 6.99,
      "year" : 2007
    },
    {
      "cost" : 3.6,
        ...
    },
        ...
  ]
}
```

```
self.assert_("wine" in r)

for wine in r["wine"] :

  self.assert_("cost" in wine)
  self.assert_(wine["cost"] > 0)
```

SUMMARY OF CONCEPTS

- building web services using lb-web
  - protocol : Google protocol buffers
  - request/response logic: delta rules
  - configuration : IDB rules
- testing web services (python framework)
- monitoring services : lb-web-server.log messages

LESSON 5 LAB EXERCISES

THANK YOU.