REST and XQuery

Getting The Balance Right

Ron Hitchens - OverStory
@ronhitchens

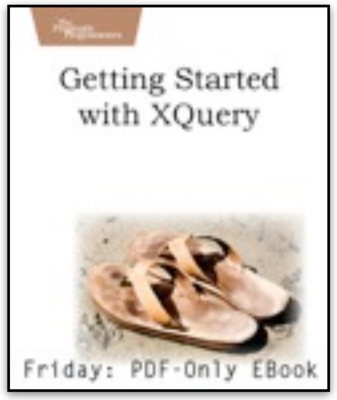
Ron Hitchens

Now: Tech Lead RESTful web services at Wiley

Soon: Founder/Chief Architect at OverStory

Java & XQuery author, Java Champion

Five years at MarkLogic, wrote XCC



Thanks to..

Norman Walsh

Lead Engineer at MarkLogic

Wrote the MarkLogic rest: library

2

REST: Representational State Transfer

URI: Uniform Resource Identifier

RESTful services transfer descriptions of things that are locatable by URIs

- "Send me an HTML representation of resource X"
- "Use this JSON rep. to replace your resource Y"
- "Make a new resource from this XML, return URI"

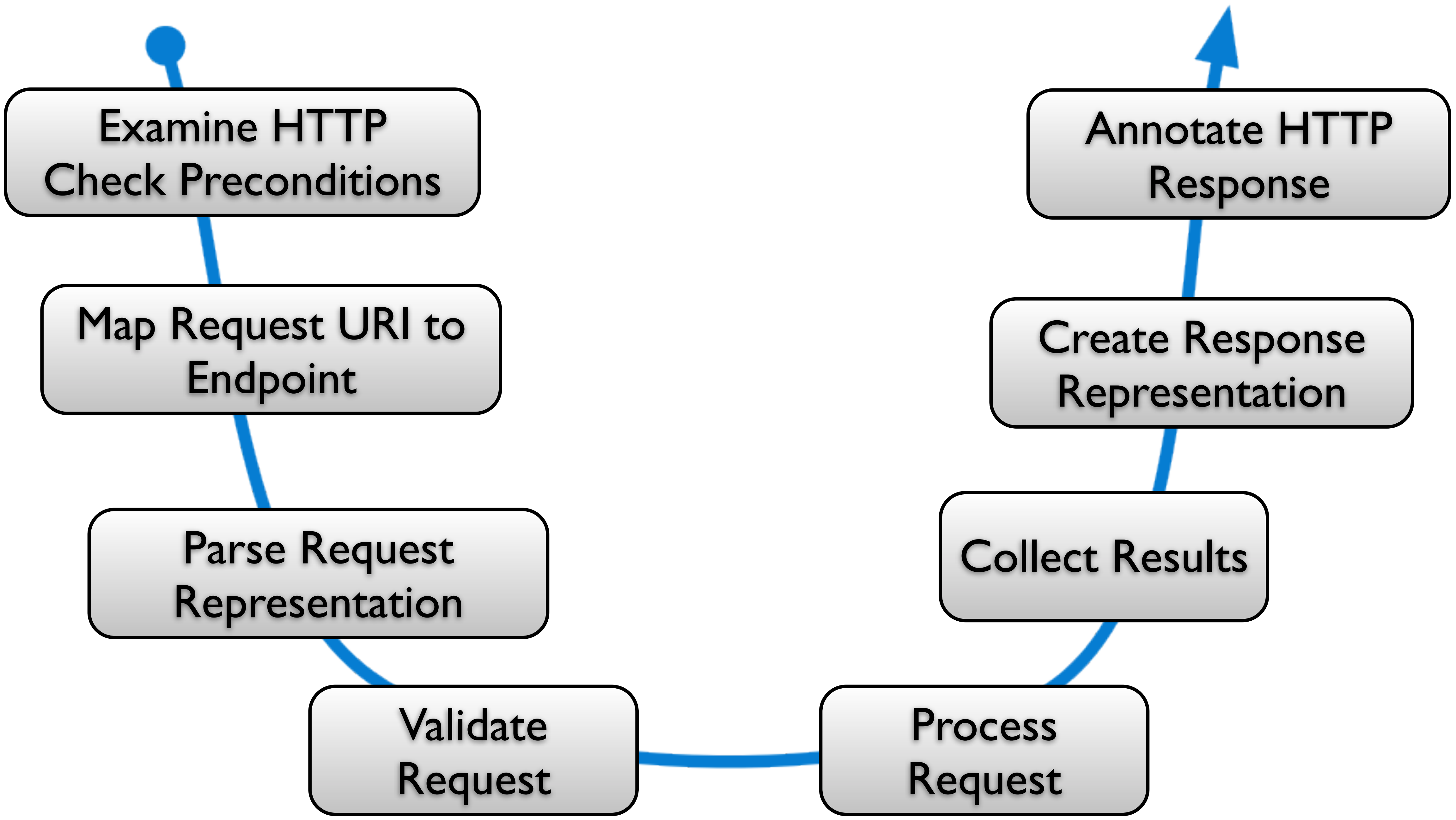
REST is descriptive, not imperative

REST in a multi-tier stack

It rocks because...

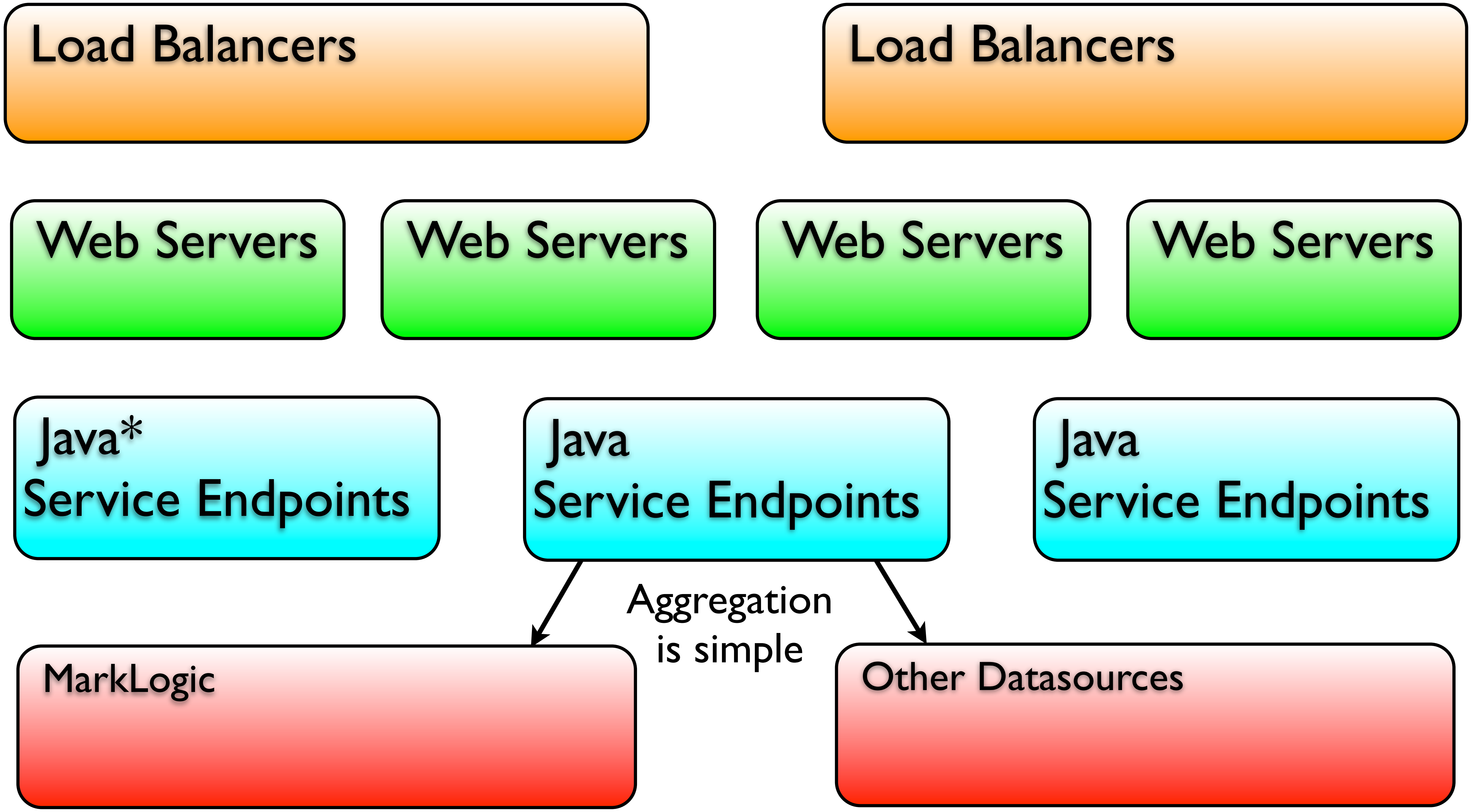
It sucks because...

Processing Steps in a REST Service



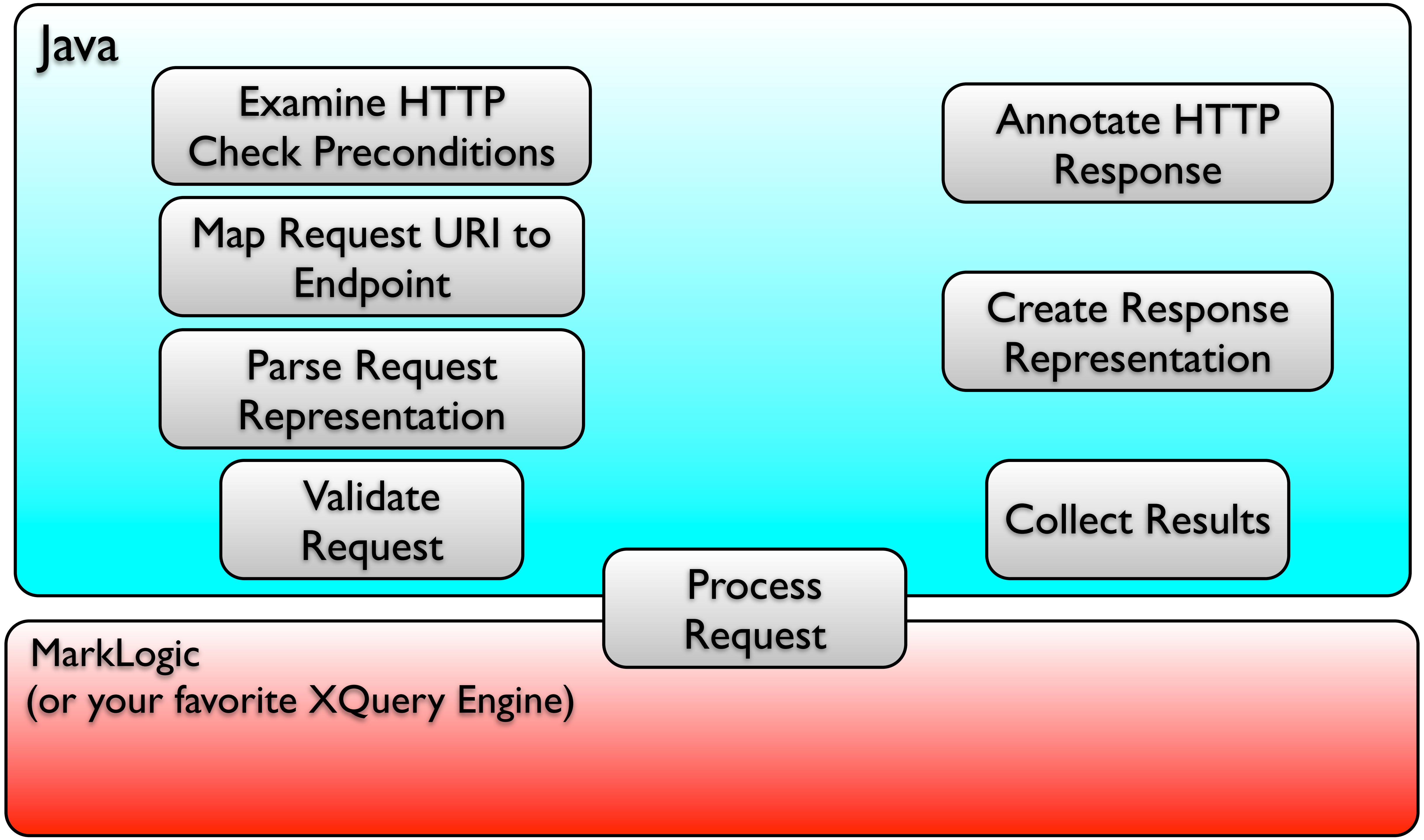
5

REST in a Multi-Tier Stack



* Or .NET, JRuby, Scala, Clojure, etc

Multi-Tier REST - Traditional Style


Java
Examine HTTP Check Preconditions
Map Request URI to Endpoint
Parse Request Representation
Validate Request
Process Request
Collect Results
Create Response Representation
Annotate HTTP Response
MarkLogic (or your favorite XQuery Engine)

This is good because...

Pick the best language for the job

Good separation of concerns

Leverage middleware services and libraries

Good tool support

MarkLogic is an expensive resource

Mundane "plumbing" done on commodity systems

This is bad because...

Several moving parts

XML in Java is awkward at best

 Horrendously inefficient at worst

If the middleware has to parse or modify XML, you're losing

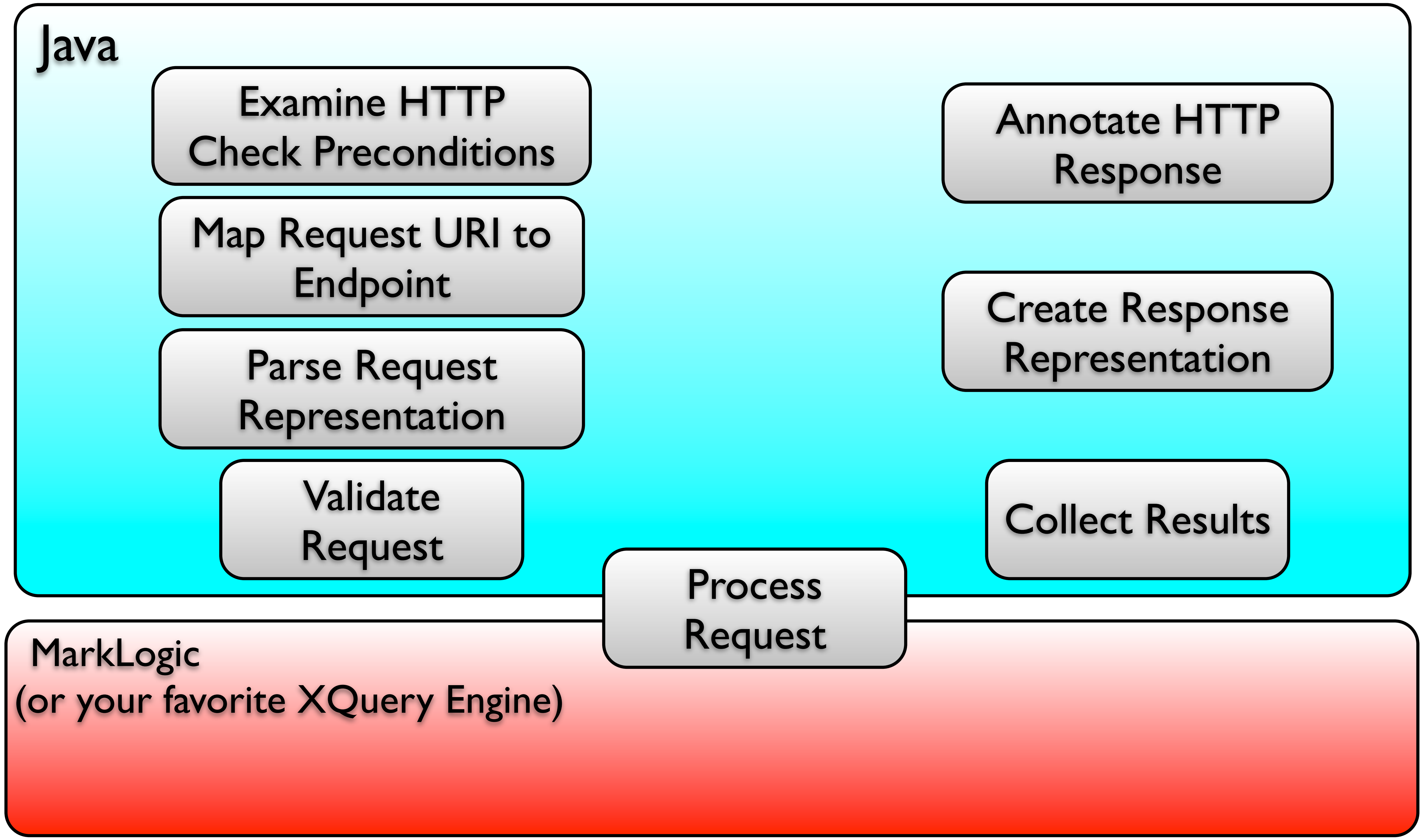
MarkLogic is your power tool, it should do the XML slicing and dicing

Let's split the difference...

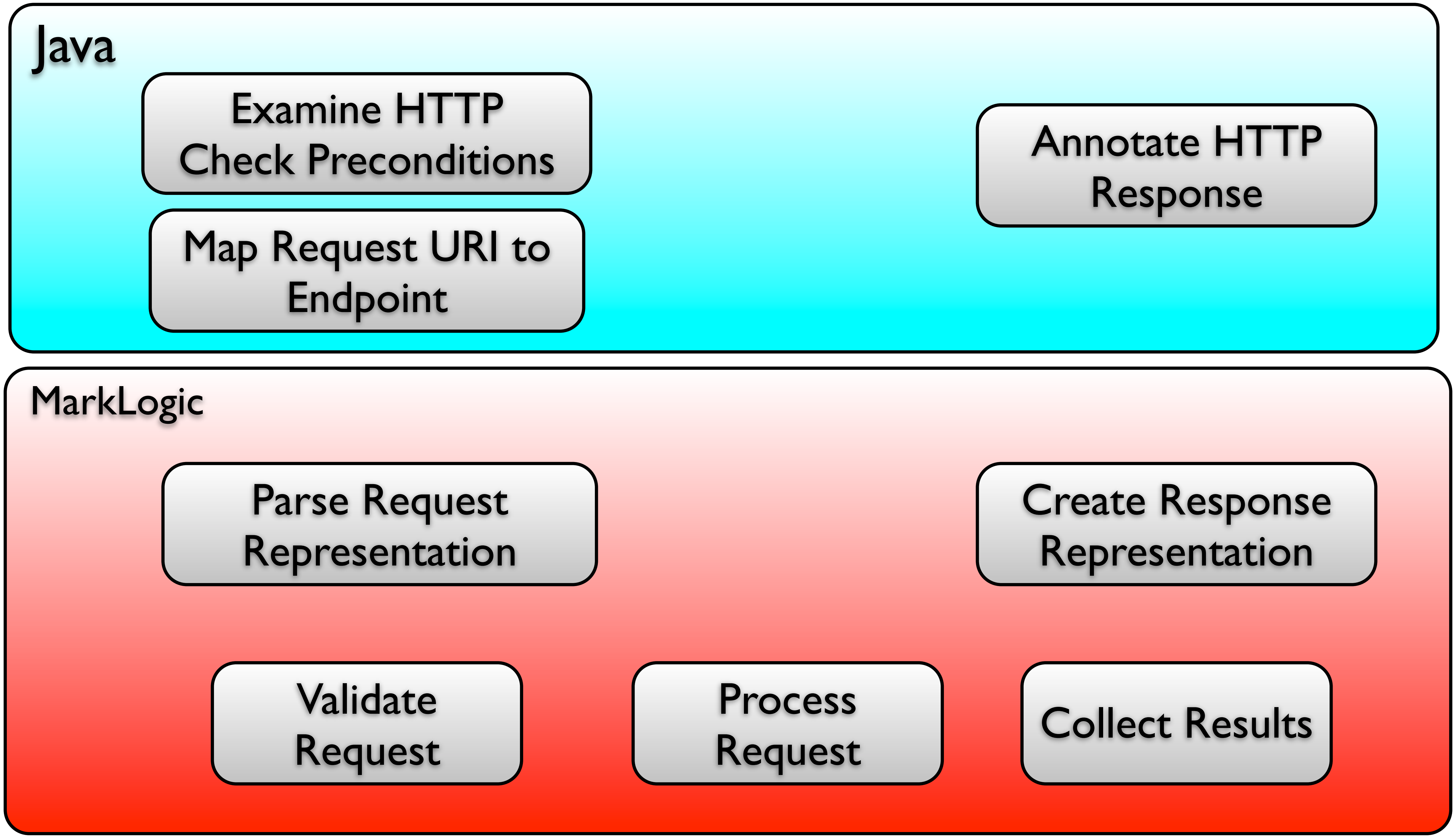
- Define your service in middleware
  - Container services, metrics, management, etc
- Delegate XML processing to MarkLogic
  - Collect data from other datasources, if needed, pass to XQuery as variables
  - MarkLogic produces final response payload
  - Middleware wraps it in HTTP and passes through
- XQuery does the XML: you're winning

10

Doing It The Hard Way, Redux...

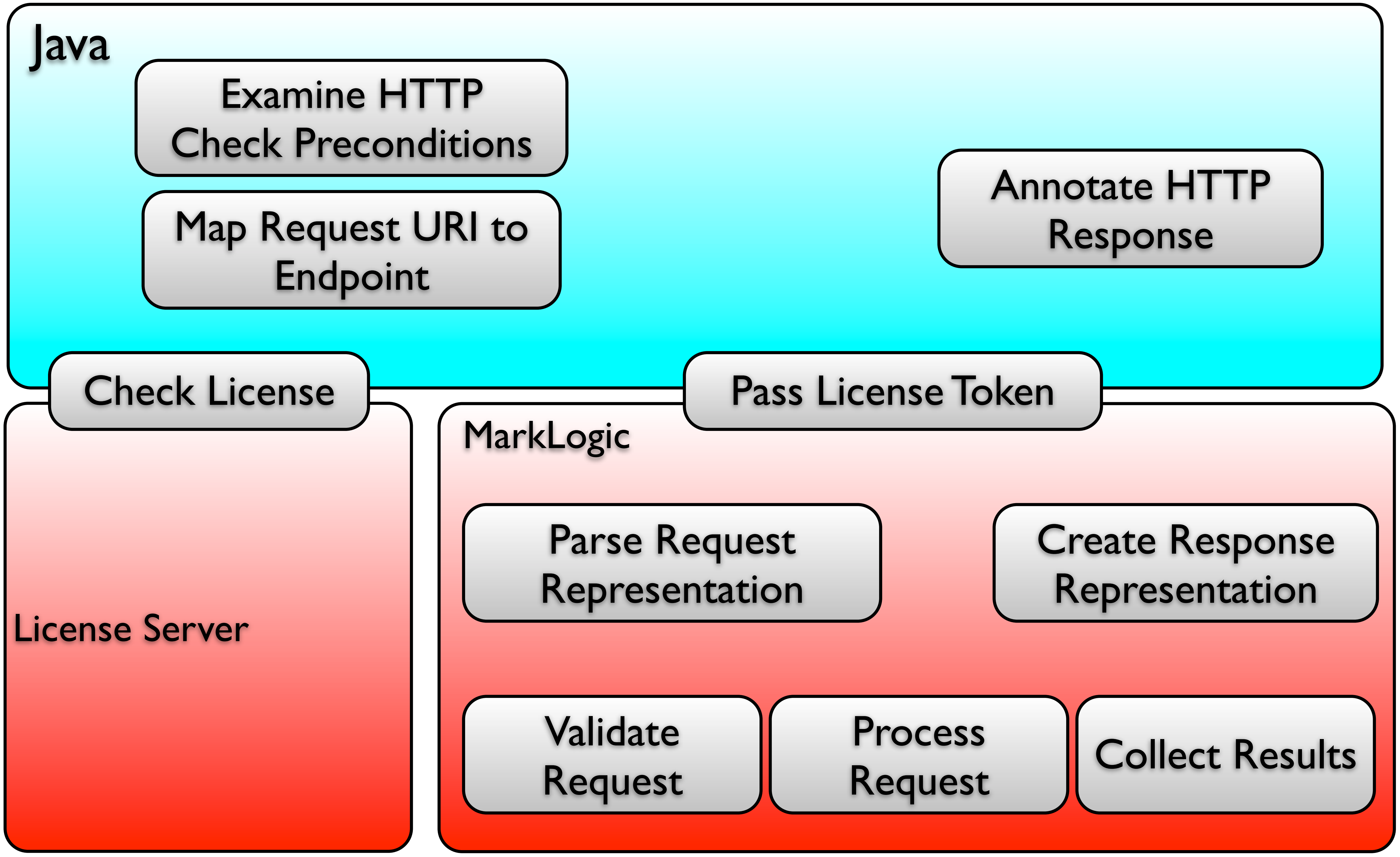

Java
Examine HTTP Check Preconditions
Map Request URI to Endpoint
Parse Request Representation
Validate Request
Process Request
Collect Results
Create Response Representation
Annotate HTTP Response
MarkLogic (or your favorite XQuery Engine)

A Better Balance


Java
Examine HTTP Check Preconditions
Map Request URI to Endpoint
Annotate HTTP Response
MarkLogic
Parse Request Representation
Create Response Representation
Validate Request
Process Request
Collect Results

12

Aggregation - One example



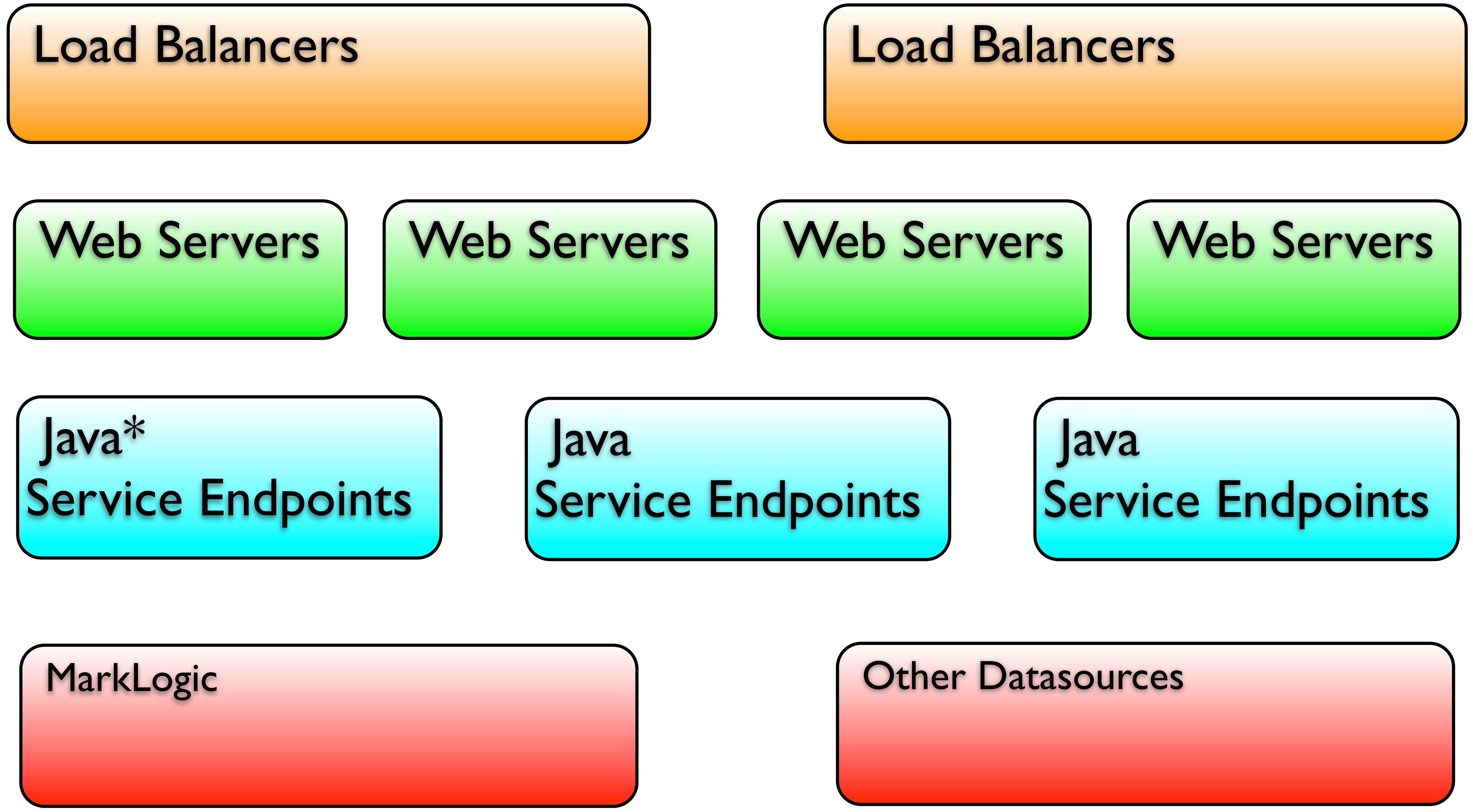
13

REST entirely in MarkLogic (or some other XQuery engine)

It rocks because...

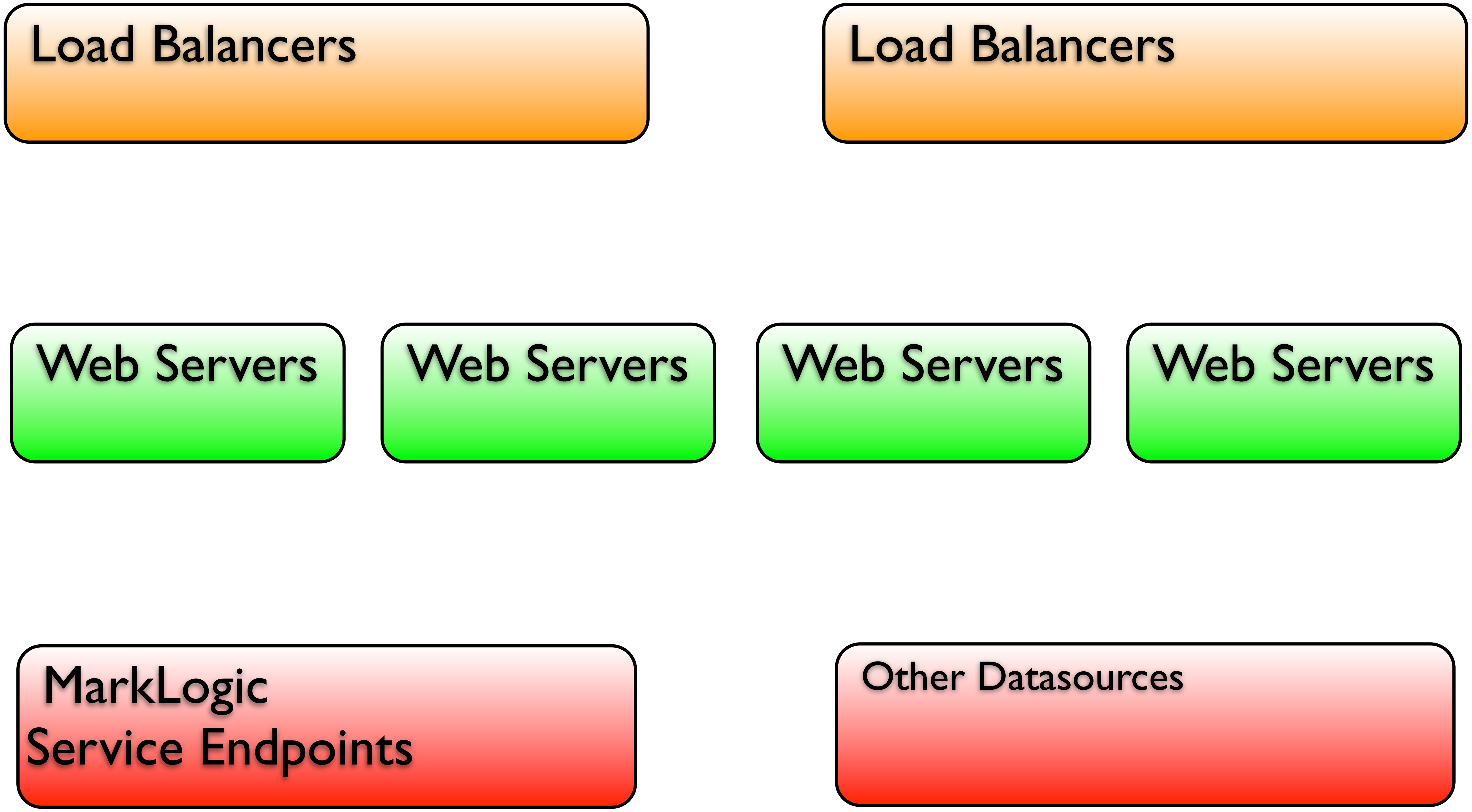
It sucks because...

From This...



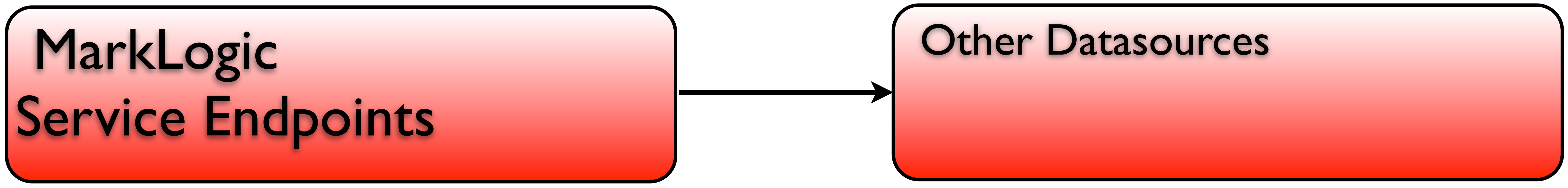
* Or .NET, JRuby, Scala, Closure, etc

To This. Or Even...



16

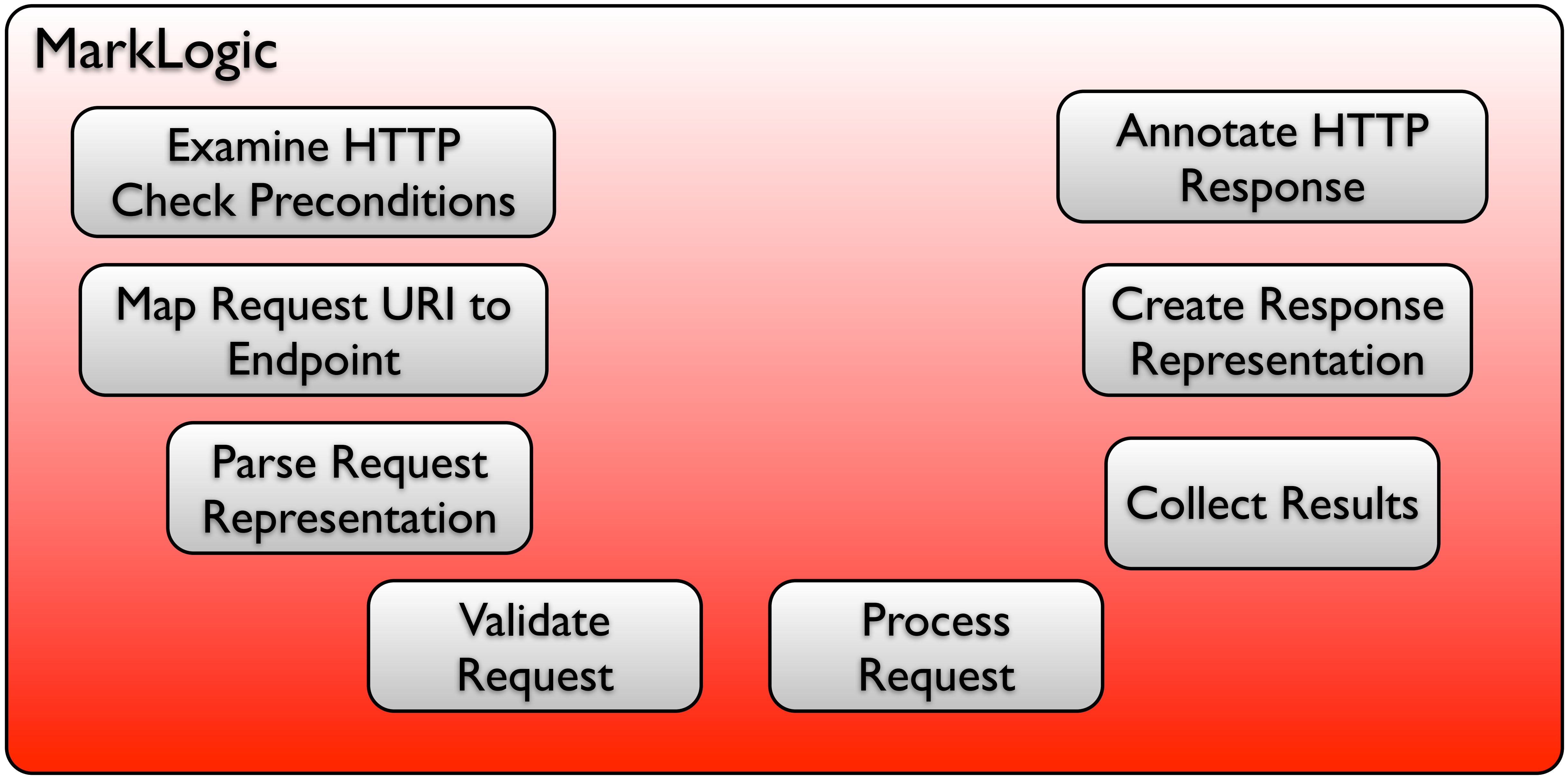
All MarkLogic All The Time



Aggregation
is a bit trickier

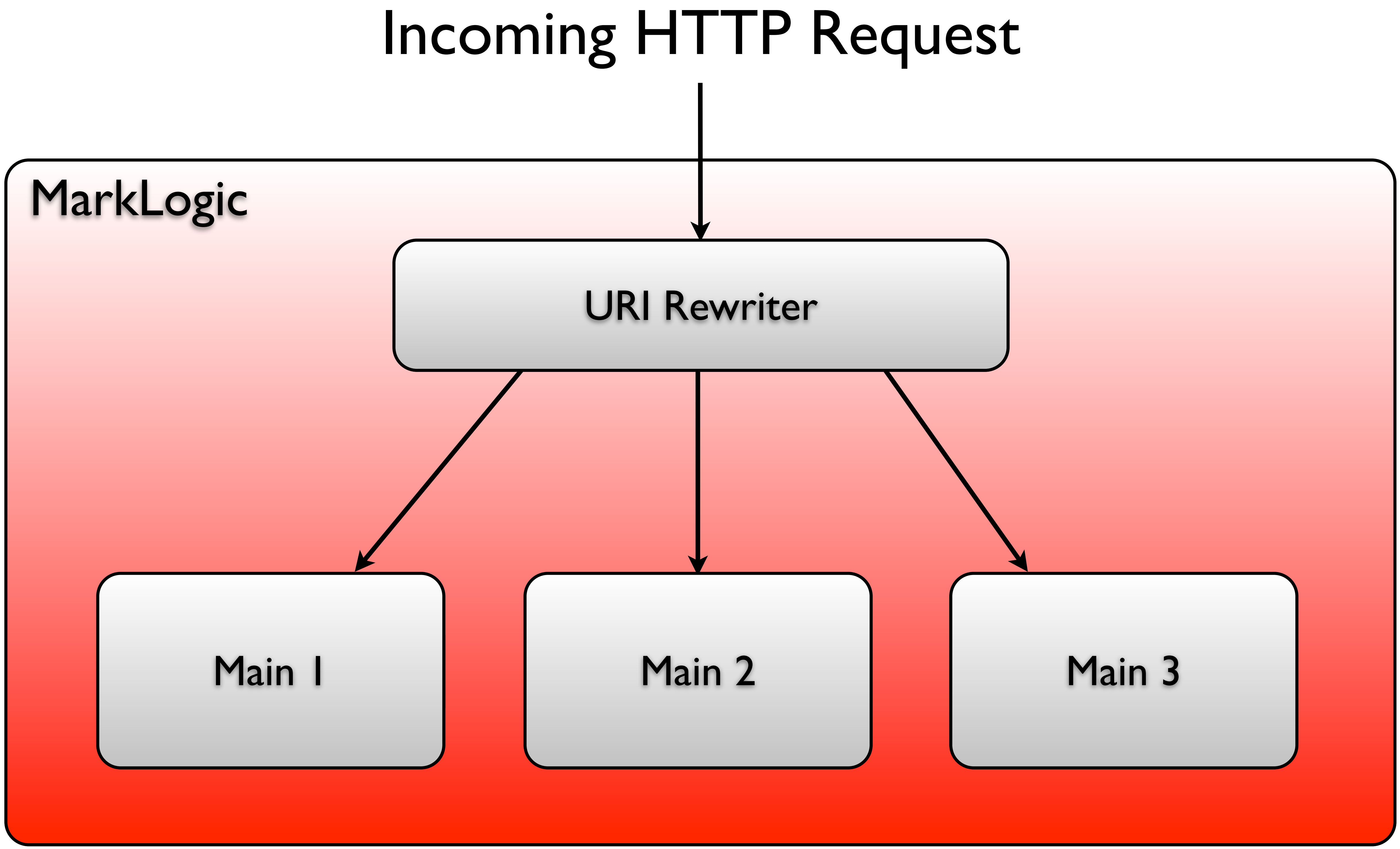
17

All-MarkLogic REST Processing



MarkLogic
Examine HTTP Check Preconditions
Map Request URI to Endpoint
Parse Request Representation
Validate Request
Process Request
Collect Results
Create Response Representation
Annotate HTTP Response

18


Incoming HTTP Request
MarkLogic
URI Rewriter
Main 1
Main 2
Main 3

This is good because...

It's even simpler

- Fewer moving parts

No impedance mismatch as data crosses layers of the stack

More opportunity to leverage your MarkLogic investment

It's all XQuery

This is bad because...

The XQuery ecosystem is less developed than the Java ecosystem

- You won't get the same tooling
- You may have to write more services

It's all XQuery

Simple endpoint definition in Java using Jersey with auto Response boxing and output filtering

```
@Path("/search")
public class SearchResource
{
    @Autowired
    private SearchProvider searchProvider;      // Injected by Spring, could be test a impl

    @GET
    @Produces({TEXT_HTML})
    @ResourceFilters(value = {SearchOutputFilter.class})
    public String searchUriToHtml (@Context UriInfo uriInfo)
    {
        SearchRequest searchRequest = new SearchParams (uriInfo).newSearchRequest();
        SearchResult searchResult = searchProvider.performSearch (searchRequest);

        return searchResult.asString();
    }
}
```

Endpoints for GET and POST with explicit Response building

```
@Path("/search")
public class SearchResource
{
    @Autowired
    private SearchProvider searchProvider;      // Injected by Spring, could be test a impl
    private static final CacheControl cachePolicy = CacheControl.valueOf ("max-age=600");

    @GET
    @Produces({APPLICATION_VND_WILEY_WS_XML, APPLICATION_ATOM_XML, APPLICATION_XML, TEXT_XML}
    public Response searchUriToXml (@Context UriInfo uriInfo)
    {
        SearchRequest searchRequest = new SearchParams (uriInfo).newSearchRequest();
        SearchResult searchResult = searchProvider.performSearch (searchRequest);

        return Response.ok().entity (searchResult.asString())
            .type (APPLICATION_VND_WILEY_WS_XML_TYPE)
            .cacheControl (cachePolicy).build();
    }

    @POST
    @Produces({APPLICATION_VND_WILEY_WS_XML, APPLICATION_ATOM_XML, APPLICATION_XML, TEXT_XML}
    public Response searchXmlToXml (String searchReqXml)
    {
        SearchRequest searchRequest = new SearchXml (searchReqXml).newSearchRequest();
        SearchResult searchResult = searchProvider.performSearch (searchRequest);

        return Response.ok().entity (searchResult.asString())
            .type (APPLICATION_VND_WILEY_WS_XML_TYPE)
            .cacheControl (cachePolicy).build();
    }
```

Exceptions are caught by Jersey and mapped

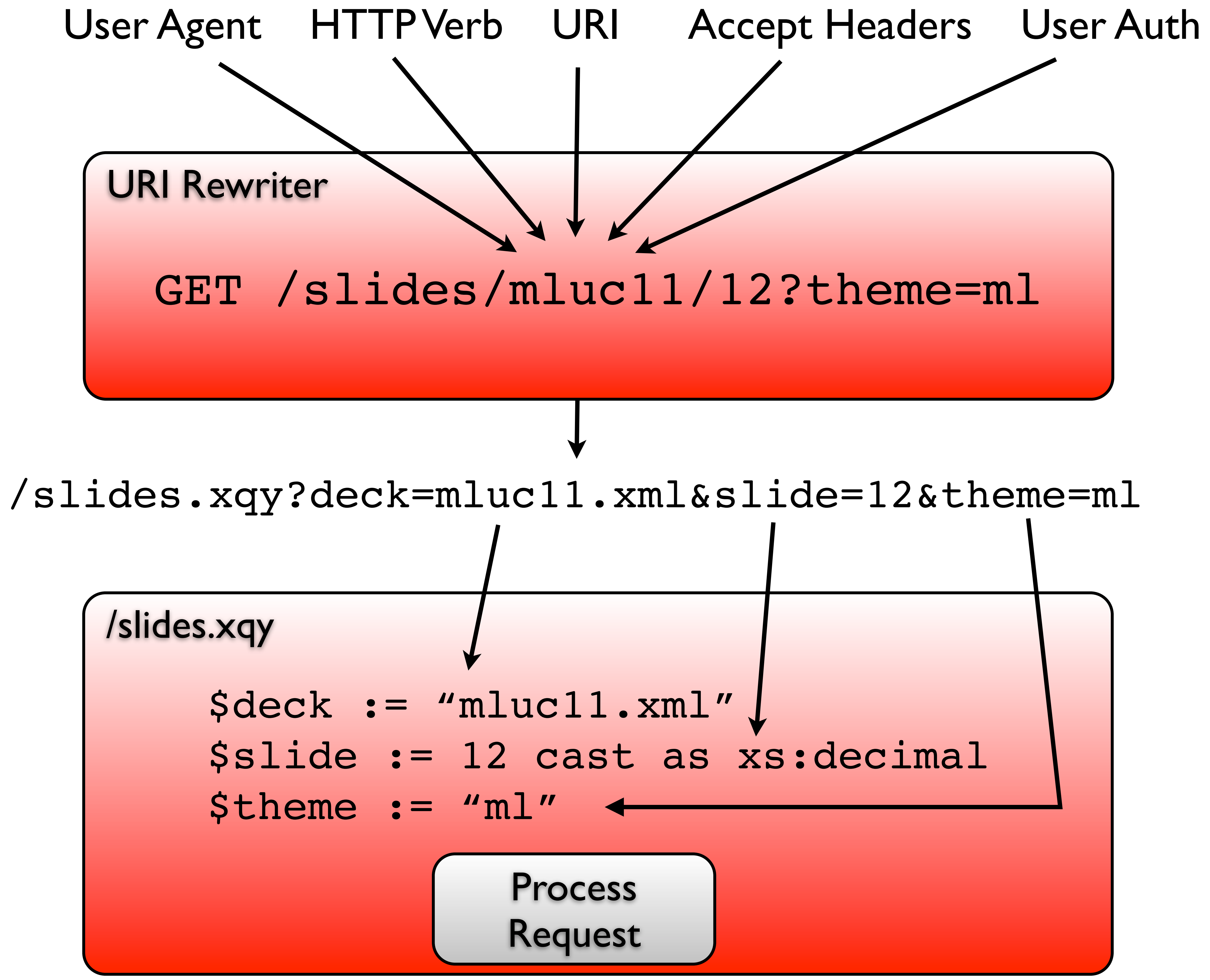
REST natively in MarkLogic

Setting up a single-tier REST service in MarkLogic

- The rest: library

It rocks because...

It sucks because...



https://github.com/marklogic/ml-rest-lib

Can’t we all just get along?

Java REST endpoints can call REST endpoints defined in MarkLogic

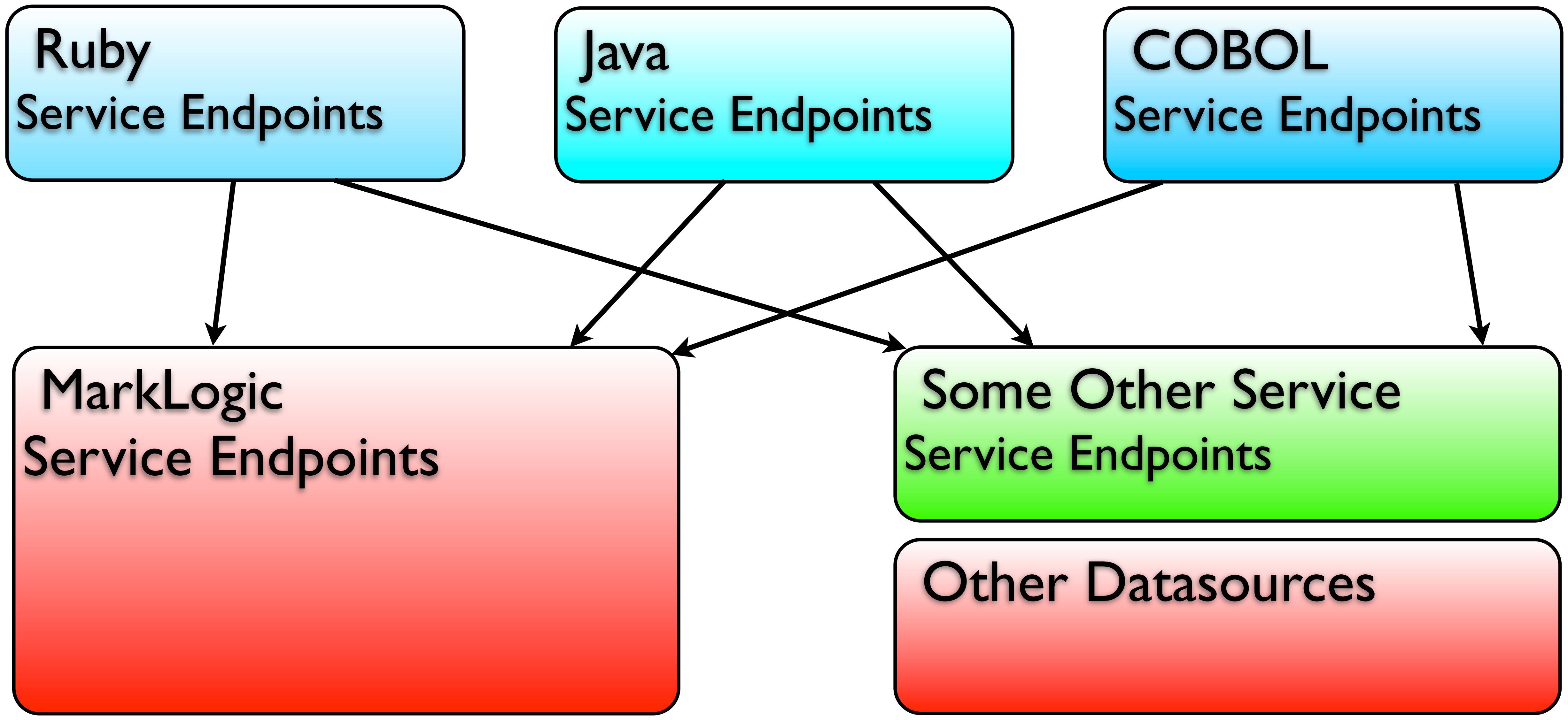
- Breaks dependency on Java/.NET
- Makes them callable from any language
- Better hides MarkLogic implementation details
- Allows services to evolve behind their interfaces

It’s turtles all the way down*

- Services on top of services on top of...

* http://en.wikipedia.org/wiki/Turtles_all_the_way_down

Tiered Service Architecture



In Summary

REST can be implemented in many ways

- Deep, heterogeneous software stack
- Leaner stack with better balance of concerns
- Single tier, all in MarkLogic

Pros and cons to each approach

The best choice for you depends on your situation

REST and XQuery

Getting The Balance Right

Ron Hitchens

ron@ronsoft.com

ron@overstory.co.uk

@ronhitchens