Solid RESTful Services with Apache CXF

APACHECON
DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

piksel

Using CXF to build JAX-RS Services

APACHECON
DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

piksel

About Me

- Lead Java Developer – Spain (with UK team)
- Working for **BSkyB** – Internet Streaming / Video on Demand
- Spent last 9 months designing and implemented JAX-RS interface that will serve video content to millions of consumer devices throughout the UK and territories
- Recently investigating how CXF might offer code-centric/annotation-driven configuration support for servers and clients

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

piksel

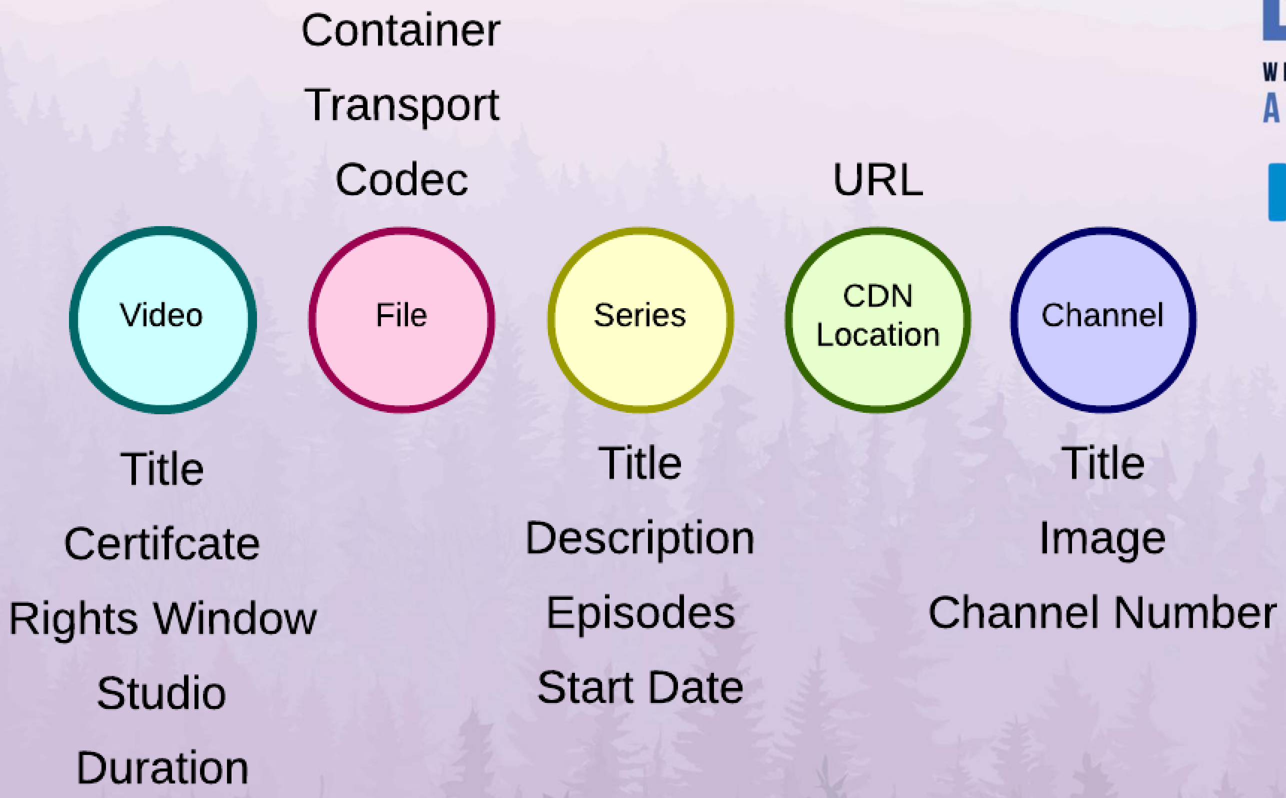
- **Video delivery solutions**, client devices, user experience (iOS, Android, Xbox, Set to box, etc)
- Cutting edge and rapid development / deployment technologies
- Globally for major broadcasters as well as for Sports, Faith, Education, Enterprise, Content Owners, Aggregators and Distributors

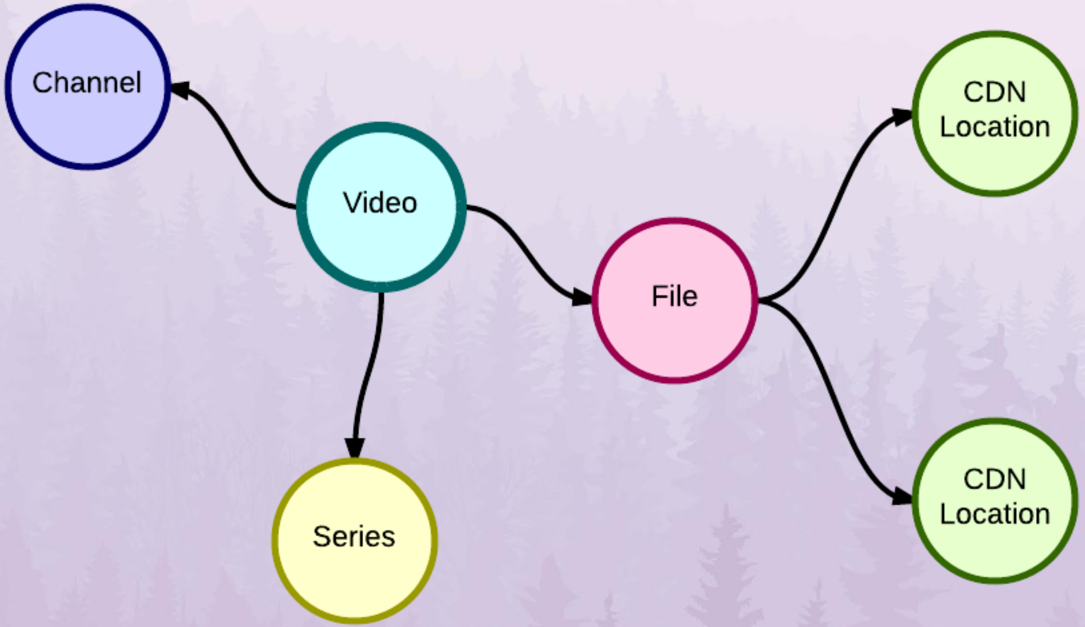
Agenda

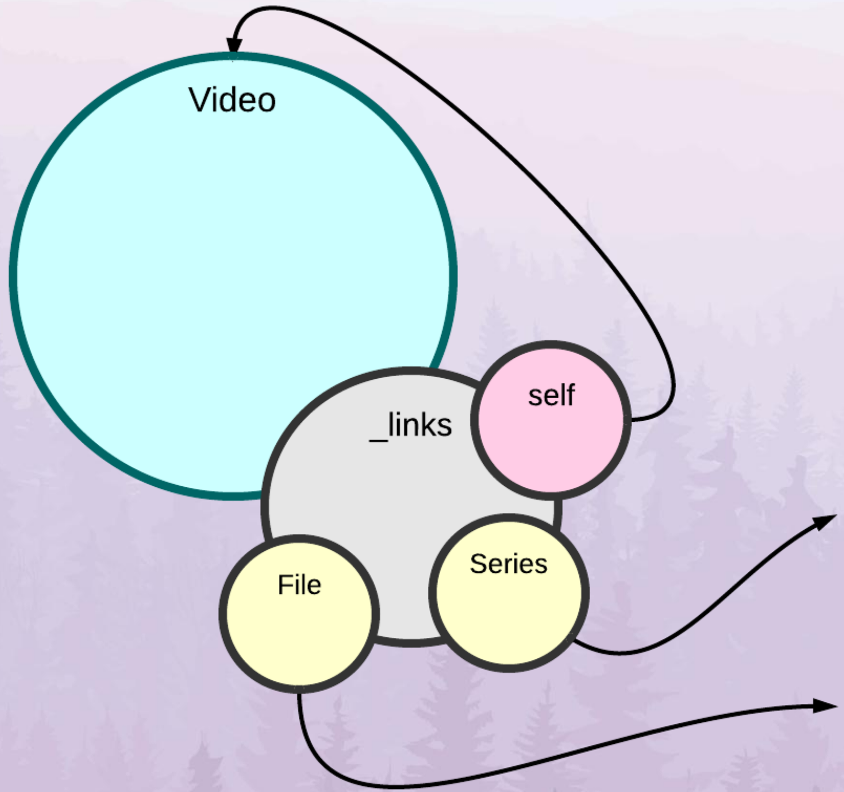
- Linking Resources
- HATEOAS with representation expansion and inclusion
- Integration testing approach
- Sundry Annotations Use
- Code-centric configuration
- Reflections on CXF and Community

Sky Catalogue

- Video and Stream Metadata – Sky GO (BSkyB)
  - Videos/Movies/Series/CDNs/Channels/…
- Many client types with brandings
- HATEOAS – Hypermedia to control client state
- Hypermedia linking scheme (similar to HAL)
- Customisable representations of resources
- Uses Spring programming model


Container
Transport
Codec
URL
Video
File
Series
CDN Location
Channel
Title
Certifcate
Rights Window
Studio
Duration
Title
Description
Episodes
Start Date
Title
Image
Channel Number


Channel
Video
File
CDN Location
CDN Location
Series


Video
_links
self
File
Series

Linking Schema

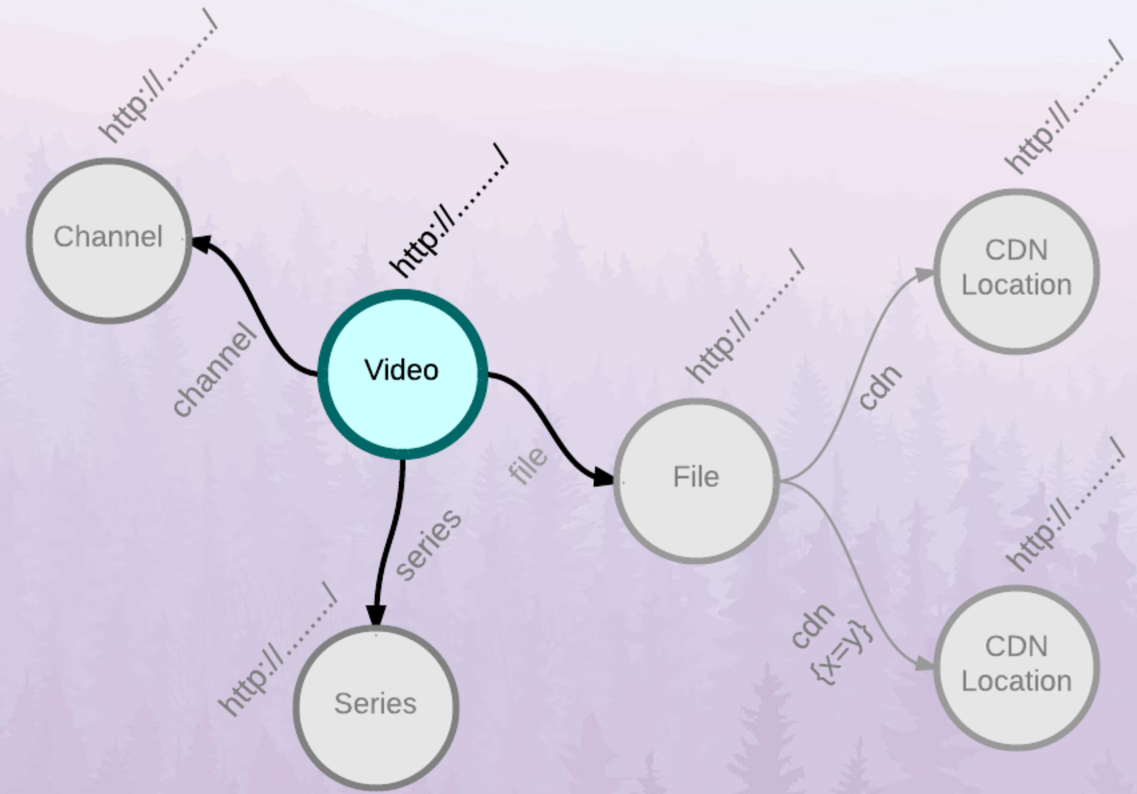
```
{
  “title”: “The Wire”,
   ….
   “_links”: [
     {
      “_href:”http://path/”,
      ”_rel”: “file”,
      “_attributes”: {“key”=“value”}
      },
  ]
}
```

Linking Schema

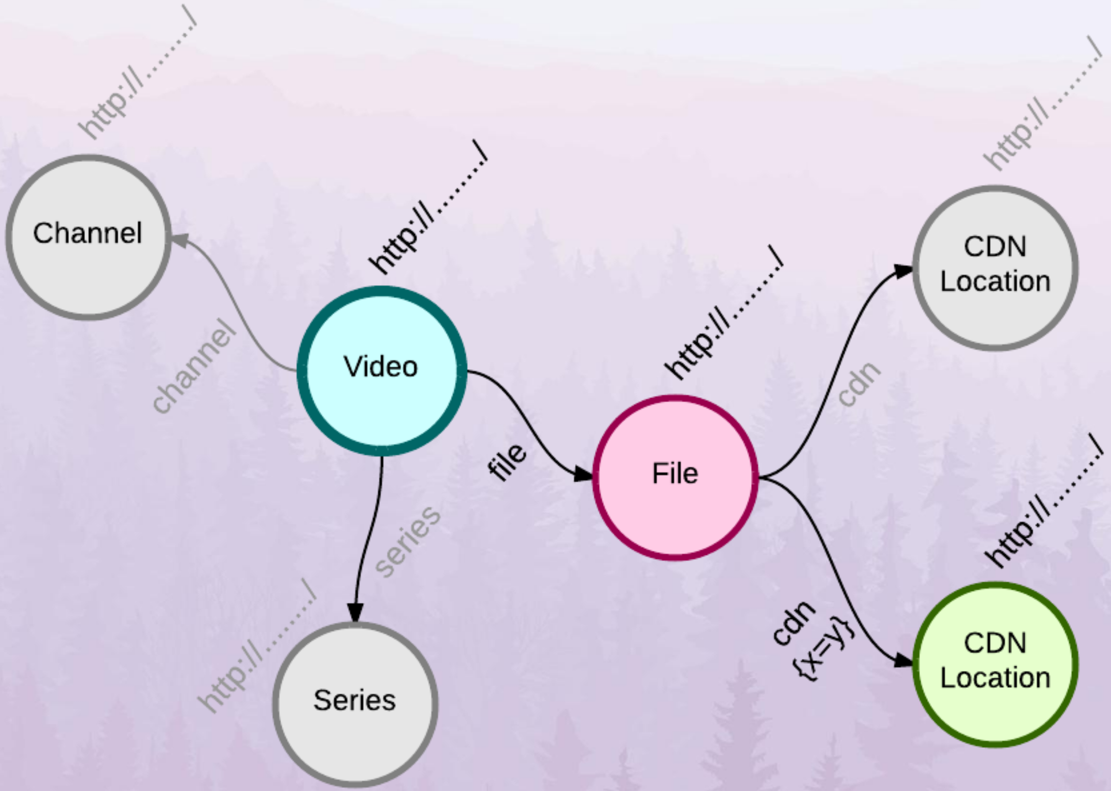
- Links objects are not part of the resource
  - Web Linking RFC (rfc5988) puts them in the header
- Resource structure is simple

```
interface Resource {
	Iterable<Link> getLinks();
}
```

- Self link is not the Identity – includes represent params



http://......../
Channel
channel
http://......../
Video
file
File
series
http://......../
Series
http://......../
cdn
http://......../
CDN Location
cdn {x=y}
http://......../
CDN Location



http://path/to/video?represent=(file(cdn;x=y))

Example

```
“title”: “The Wire”,
….
“_links”: [
    { “_href”: “…”, “_rel”: “channel”, “_attributes”: {},
      “channelId”: “1234”,
      “title”: “Atlantic Channel”,
      …
      “_links”: [ {…}, {…} ]
    }
 ]
```

Link Query Mini-Language

- **(rel)** – expansion
- **(rel(rel))** – nested expansion
- **(rela, relb, relc)** – 'OR' expansion
- **(rel;a=b)** – expansion with attribute criteria
- **(rel[fieldA, fieldB])** – expansion with field inclusion
- **(rel[fieldA, fieldB](rel))** – nested expansion with inclusion
- We actually used a **two part** rel: child/node or parent/node
- Can theoretically select large regions of data
  - Implemented a whitelist, but other expansion restrictions could be implemented

Link Implementation

```
class CatalogueLink implements Link {
     // …
    @JsonIgnore
    public Supplier<Resource> getLinkedResourceSupplier() {}

    public void setResource(Resource resource) {}

    @JsonUnwrapped
    public Resource getResource() {}
}
```

Link Building

- Get the target method
  message.getExchange()
  .get(OperationResourceInfo.class).getMethodToInvoke()
- JAX-RS 2.0 – **ResourceInfo –** Injectable interface

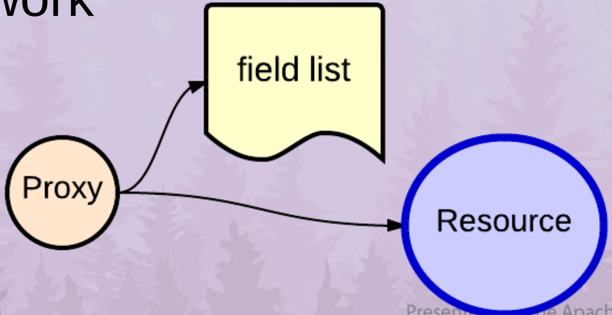
- **@Path("/videos/{videoId}")**
  @Resource("video") – richer link building
  - link.to(Video.class).withId(videoId).withAttribute("a","b")
    .withType("child").withSupplier(new VideoSupplier ….);

Some Observations

- Expansion - Increasing popularity
  - Spoken about at conferences
  - Used in the wild (Netflix, Stormpath)
  - Documented in various books
  - rfc5988 web linking (headers) is **insufficient** for expansion
- To what extent could a framework support such a pattern?
  - Resource interface with Links
  - Supplier for linked resources
  - Expansion/Inclusion mini language
  - Engine

Field Inclusion

- Minimise bandwidth
- Performance improvements
- Implemented using a CGLIB wrapper
- Forward to getters if in field list, otherwise **null**
- Needs to respect serialisation annotations/directives
- Works irrespective of serialisation framework



APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

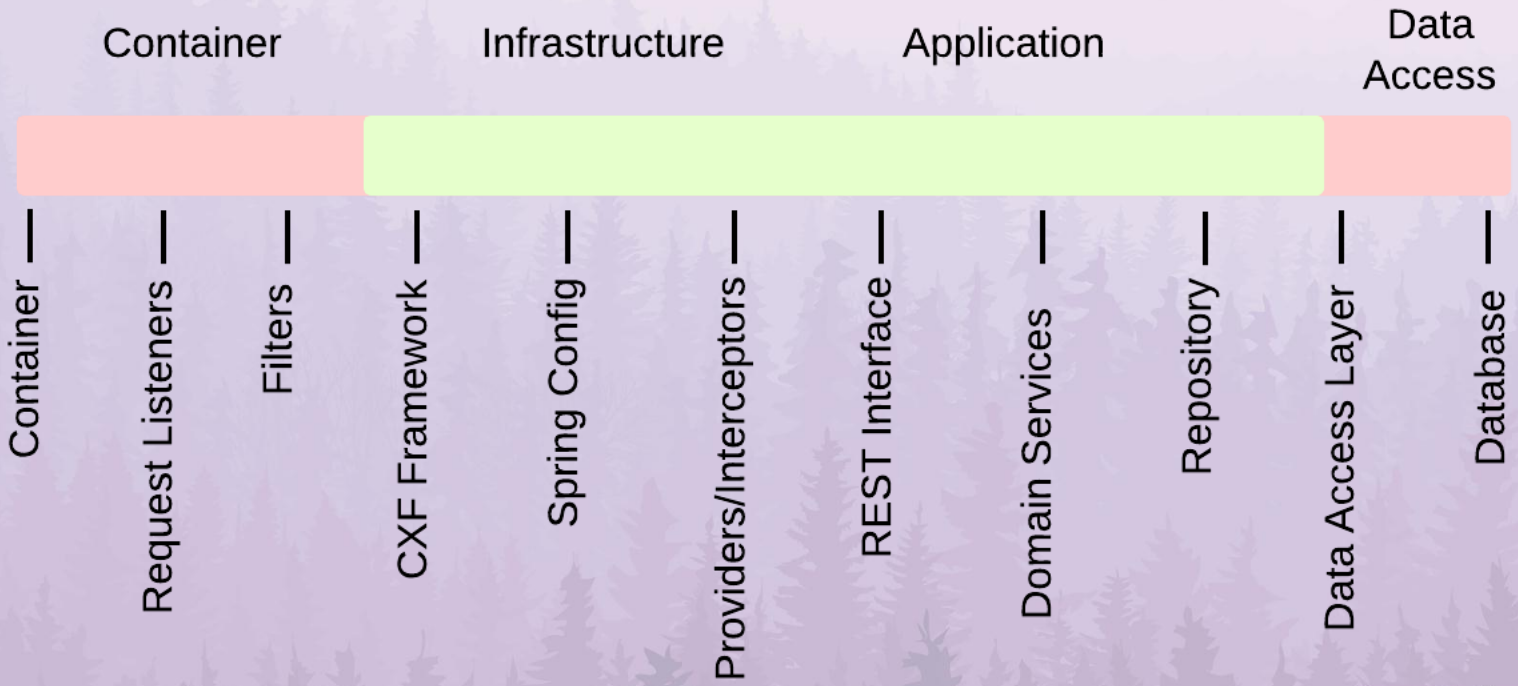
piksel

Integration Testing

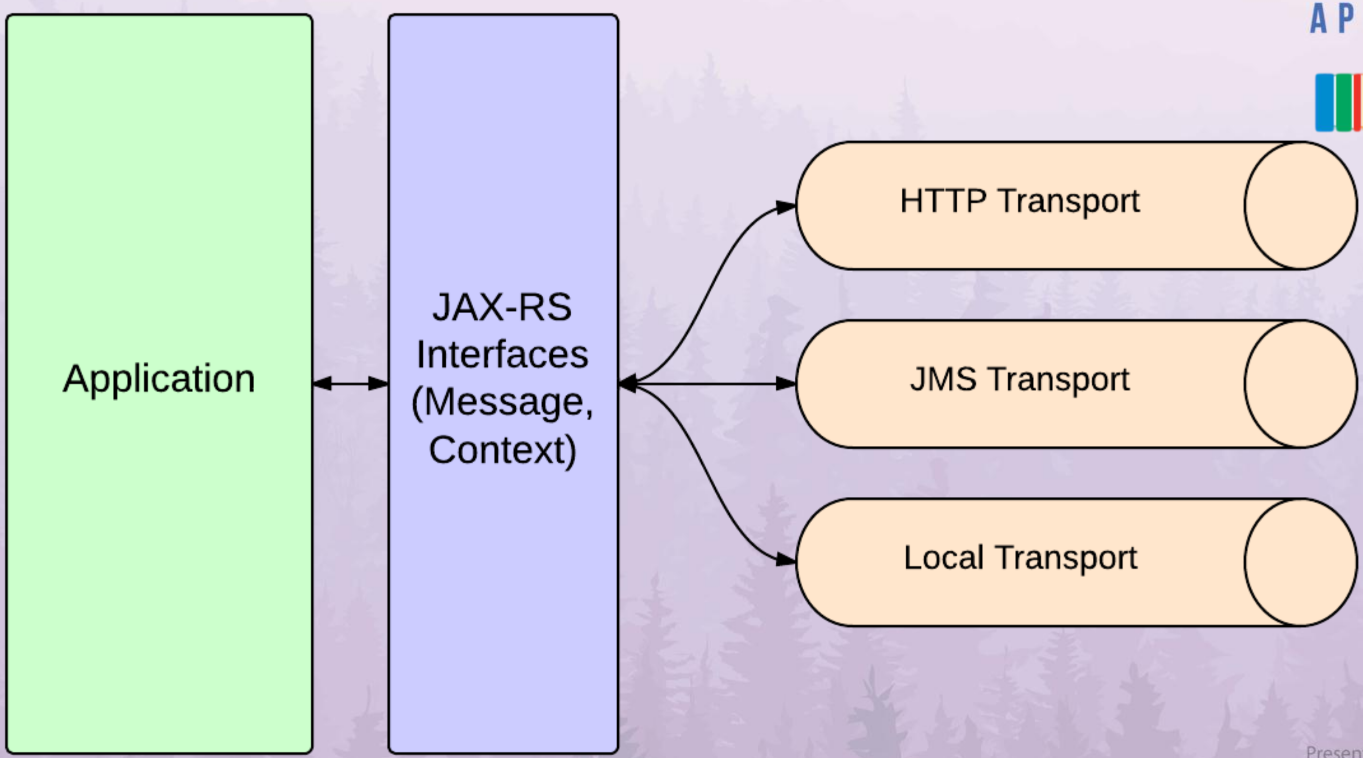
Goals

- Fast – immediate feedback
- Not a functional/acceptance test
    - No container or database
- Includes application stack
    - CXF Framework
    - Interceptors and Providers
    - Application Code

Scope


Container
Infrastructure
Application
Data Access
Container
Request Listeners
Filters
CXF Framework
Spring Config
Providers/Interceptors
REST Interface
Domain Services
Repository
Data Access Layer
Database

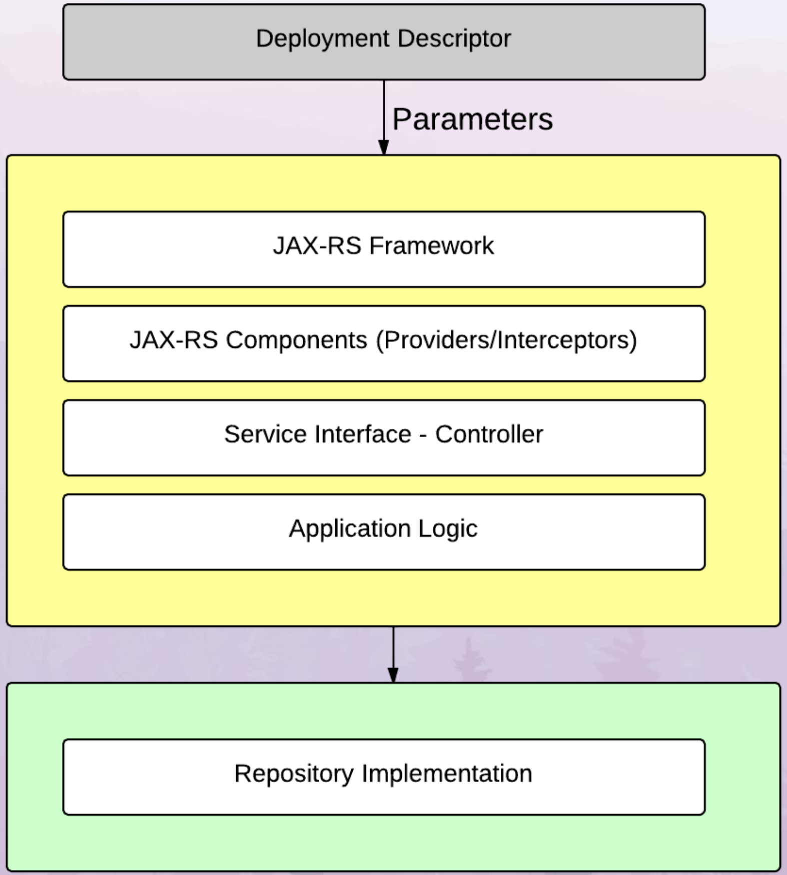
CXF Transports



Local Transport

- Messages do **not** leave the JVM
- Messages passed very quickly
- Ideal for integration testing
- Use the **local://** URL scheme
- Use **DIRECT_DISPATCH = true** on the client configuration

```
<jaxrs:server … address="local://local-address"
                        transport="http://cxf.apache.org/transports/local"
```


Deployment Descriptor
Parameters
JAX-RS Framework
JAX-RS Components (Providers/Interceptors)
Service Interface - Controller
Application Logic
Repository Implementation

Externalise Transport Cfg

cxf-servlet.xml

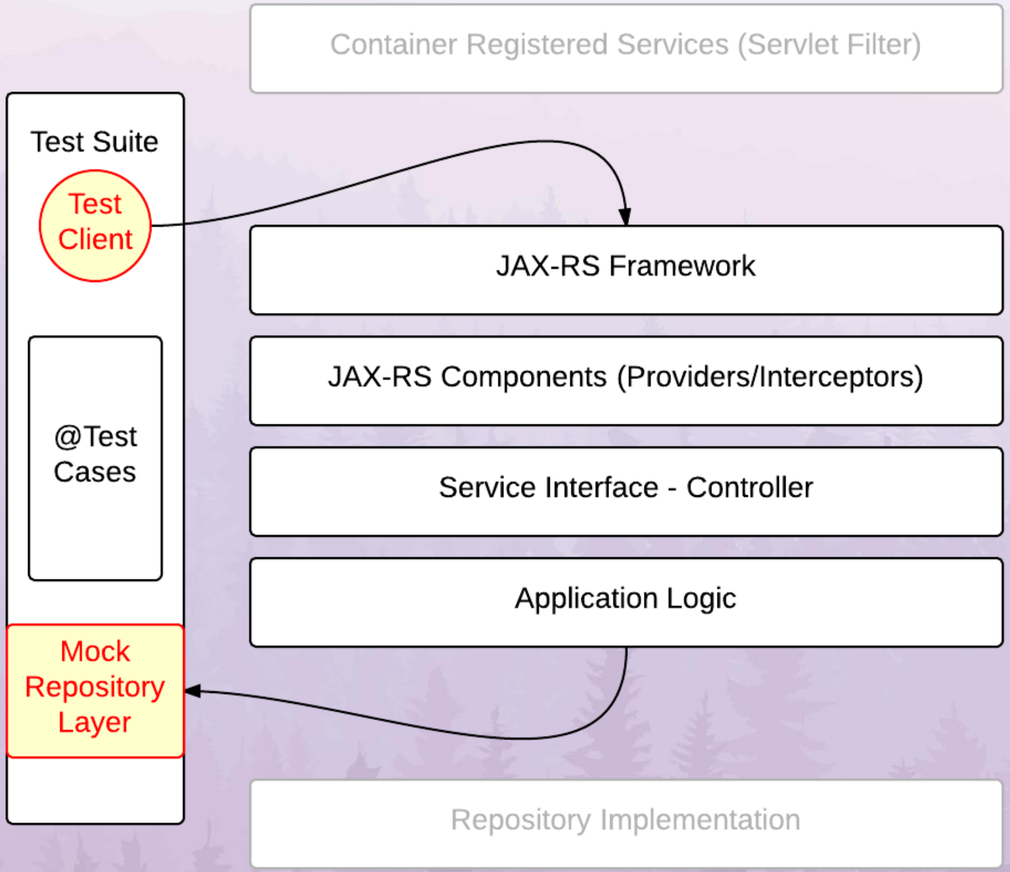
```
<jaxrs:server … address="${rest.service.address}"
                    transport="${rest.service.transport}"
```

web.xml

```
  <param-name>rest.service.transport</param-name>
  <param-value>http://…/transports/http</param-value>
```

Test

```
 context.getEnvironment().getPropertySources()
      .addFirst(new MapPropertySource("servletConfig", localConfig));
```


Container Registered Services (Servlet Filter)
Test Suite
Test Client
JAX-RS Framework
JAX-RS Components (Providers/Interceptors)
@Test Cases
Service Interface - Controller
Application Logic
Mock Repository Layer
Repository Implementation

Stubbing Persistence Layer

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

piksel

Demo

Supplying DAO Mocks

- @Bean methods return EasyMocks
- @Autowired mocked dependencies
- Supply expectations within the test
- **TestExecutionListener to control mocks**
  - Locate all mocks in context
  - Replay and Verify
- No XML
- Test set up visible directly within the test itself

Experiences

- Quick execution and large coverage
- Make effort to mirror container configuration in integration tests, including inheritance patterns
- DRY applies
  - Take the time to centralise integration test set up
  - Take the time to centralise mocking
- Does not cover container registered components
  - JAX-RS 2 Container Request Abstraction can help

Annotations

Error Mapping and Docs

APACHECON DENVER WESTIN DENVER DOWNTOWN APRIL 7-9, 2014

piksel

```
@ErrorCodes(
   @ErrorCode(
            ex=UserNotFound.class,
            status=404,
            message="User ${id} was not found in the system"),
   @ErrorCode(….)
)
@Path("/user/{userId}")
public UserResource getUser(@PathParam("userId") String id);
```

Stereotype Meta-Annotations

```
@UserManagement
@Path(“/user/{userId}”)
public UserResource getUser(@PathParam(“userId”) String id);
```

```
@ErrorCodes(
    @ErrorCode(ex=…, status=…,  message=“…”),
    @ErrorCode(….)
)
public @interface UserManagement {}
```

Security

Custom HMAC authentication implementation:

@Secure(HEADERS, BODY, ..)
@Path(“/user/{userId}”)
public UserResource getUser(@PathParam(“userId”) String id);

Annotation Configuration

- Found that annotations work well in test configuration
- Particularly good for defining test setup within the test
- Many other benefits of code-based configuration
- First class support available in many Spring-focussed projects, and growing
- Proposal for @Configuration style approach for configuring servers and proxies

https://github.com/paulalexwilson/jaxrs-cxf-spring-annotation

Examples

```
<jaxrs:server id="myRestService">
      <jaxrs:serviceBeans>…
      <jaxrs:features>…
      <jaxrs:providers>…
      <jaxrs:inInterceptors>…
      <jaxrs:outInterceptors>…
      <jaxrs:outFaultInterceptors>…
</jaxrs:server>
```

```
@JaxRsServer
class MyServer {
  @JaxRsService
  public Object myService() {…}
  @JaxRsFeature
  public Feature logging() {…}
  ….
}
```

Apache CXF Community

- #CXF IRC Channel
- Active and helpful
- 20+ bugs responded to
- Opportunity to feed into development and future of CXF

APACHECON DENVER
WESTIN DENVER DOWNTOWN
APRIL 7-9, 2014

paul.wilson@piksel.com

piksel

We're hiring!
http://www.piksel.com/company/careers/