JAX-RS 4.0 Brainstorming (Santiago)

General Injection

1. Remove `@Context` annotation using `@Inject` instead
    1. In parameter position where `Context` is not supported, just drop `@Context`
        1. See below new annotation `@Entity`
2. Consider dropping support for `@ContextResolver`
    1. These are akin to CDI producers
    2. With the exception of `@Produces` to filter media types
        1. Does not appear to be a popular feature in practice
3. Field and property injection as supported by CDI
    1. Add `@Qualifier` to all `*Param` annotations
        1. Make annotation members `@NonBinding`
        2. Implementations will likely require some processing in CDI extensions

---

Parameter Injection

1. Constructors: use `@Inject` at constructor level instead
    1. Constructors are managed by CDI, so use CDI rules here
    2. Should enable injection of JAX-RS types in table below
2. `@Context` can be used in parameter position, `@Inject` cannot
    1. Introduce `@Entity` to identify single entity param
3. Parameter injection in *resource/sub-resource methods* identified by an `@HttpMethod` annotation such as `@GET` , `@PUT` etc.
    1. Parameter injection in full control of the JAX-RS implementation -- no `@Inject` at method level
        1. Delegate to CDI to create the beans to inject
    2. Assuming `@Entity` annotation: every parameter without any annotation is injected as a normal CDI bean
        1. This includes JAX-RS types such as `SecurityContext` , `HttpHeaders` , etc. See table below and corresponding scopes.
            1. JAX-RS implementations will likely need to provide producers for all these types
4. Parameter injection in *sub-resource locators* identified by an `@Path` annotation and no `@HttpMethod` annotations
    1. Same injection support as for sub-resource methods above
    2. No `@Entity` parameter in this case
5. Remove redundant `@Suspended` since it is always use with `AsyncResponse`

Bean Discovery

1. `@Path`, `@ApplicationPath` and `@Provider` should be bean-defining annotations
    1. Resource classes in `@RequestScoped` by default
    2. Provider classes and application subclasses in `@ApplicationScoped` by default
    3. Any special processing for `ConstrainedTo`?
2. We should support synthetic applications when no `Application` subclass is defined

---

Types

1. Compile-time dependency with JAXB removed
    1. Removal of `Link.JaxbAdapter` and `Link.JaxbLink`

Constructors

1. For consistency and ease of implementation since constructors are called by CDI, we should forbid injection of `@*Param` in constructors
    1. This is not a big restriction since field and resource method param injection is always available

---

Application Subclasses

1. We should consider supporting zero and more than one `Application` subclasses
    1. Zero
        1. Use CDI to gather annotated resources and providers and create "synthetic" instance. See Bean Discovery above.
    2. More than one
        1. This can be useful to define different security configurations (implementation extensions)
        2. Application instance should be in request scope instead of application scope
            1. A problem if attempted to be injected without a request scope active

---

Types Injectable via @Context

Type	**Description**	Default Scope
SecurityContext	An injectable interface that provides access to security related information	Request
HttpHeaders	An injectable interface that provides access to HTTP header information	Request
UriInfo	An injectable interface that provides access to application and request URI information	Request
Configuration	A configuration state associated with a Configurable JAX-RS context	Application
Providers	An injectable interface providing runtime lookup of provider instances	Application
ResourceInfo	An injectable class to access the resource class and resource method matched by the current request	Request
ResourceContext	The resource context provides access to instances of resource classes	Application
Request	An injectable helper for request processing	Request
Application	Defines the components of a JAX-RS application and supplies additional meta-data	Application
Sse	Server-side entry point for creating OutboundSseEvent and SseBroadcaster.	Application
SseEventSink	The instance of SseEventSink can be only acquired by injection of a resource method parameter	Connection

Other Injection Annotations

Type	**Description**	**Notes**
PathParam	Binds the value of a URI template parameter or a path segment containing the template parameter to a	Can inject a PathSegmer

	resource method parameter, resource class field, or resource class bean property.	
BeanParam	The annotation that may be used to inject custom JAX-RS parameter aggregator value object into a resource class field, property or resource method parameter	
CookieParam	Binds the value of a HTTP cookie to a resource method parameter, resource class field, or resource class bean property.	
FormParam	Binds the value(s) of a form parameter contained within a request entity body to a resource method parameter. Values are URL decoded unless this is disabled using the Encoded annotation	
HeaderParam	Binds the value(s) of a HTTP header to a resource method parameter, resource class field, or resource class bean property.	
MatrixParam	Binds the value(s) of a URI matrix parameter to a resource method parameter, resource class field, or resource class bean property.	
QueryParam	Binds the value(s) of a HTTP query parameter to a resource method parameter, resource class field, or resource class bean property.	
Suspended	An injectable JAX-RS asynchronous response that provides means for asynchronous server side response processing	Injection of AsyncRespo

Providers

Can be limited to Client or Server using @ConstraintedTo annotation. Are automatically picked up by server runtime if annotated with @Provider.

Type	**Description**	Default Scope
ParamConverterProvider	Contract for a provider of ParamConverter instances	Application
Feature	A feature extension contract	Application

DynamicFeature	A meta-provider for dynamic registration of post-matching providers during a JAX-RS application setup at deployment time	Application
ExceptionMapper	Contract for a provider that maps Java exceptions to jakarta.ws.rs.core.Response	Application
MessageBodyReader	Contract for a provider that supports the conversion of a stream to a Java type.	Application
MessageBodyWriter	Contract for a provider that supports the conversion of a Java type to a stream.	Application
ReaderInterceptor	Interface for message body reader interceptors that wrap around calls to jakarta.ws.rs.ext.MessageBodyReader#readFrom.	Application
WriterInterceptor	Interface for message body writer interceptors that wrap around calls to jakarta.ws.rs.ext.MessageBodyWriter#writeTo.	Application