Representational state transfer (REST) is a style of software architecture. RESTful is generally used to refer to web services implementing that architecture. JAX-RS is the specification provided by Java to create RESTful web services according to REST architectural pattern.

There are many implementations of JAX-RS such as;

* RESREasy
* Jersey
* Restlet
* CFX

Important things to consider when implementing a REST web services.

* Adhere to best URL standards (Since REST is a concept not a standard)
  + Should not follow SOAP style URI patterns like createEmployee, deleteEmployee. Instead URLs like [www.sample.com/employee](http://www.sample.com/employee), or [www.sample.com/employee/123](http://www.sample.com/employee/123)
* Since there’s no contract between server and client as in SOAP WSDL, it is important to have an accurate, updated documentation.
* Carefully design media types
  + If the web servicer is going to use by various different consumers, it is important to define more than one media type. As an example XML and JSON both. Most of the new consumers will be able to JSON however old clients may not have the support consume JSON within the existing system or may be some consumers might like to stick to XML over JSON to keep the consistency. Therefor it’s better to produce both formats. As following code sample shows, we can expose the same method with more then one media type

@Produces({"application/xml","application/json"})

public String doGetAsXmlOrJson() {

// code

}

* Use appropriate verbs
  + Use HTTP GET method for information retrieval, POST for creating new resources, and PUT for updating existing resources.
* Version controlling mechanism
  + Important to maintain separate versions for new releases. Having a version number as part of URI this can be done.
* Caching
  + Need to analyze the appropriate caching mechanism for the web service. Caching can be done in client side, server side or in a proxy server.
* Error handling
  + Errors can be handled with HTTP status codes or having a proper error payloads.

Ex.

{

"errors": [

{

"userMessage": "Sorry, the requested resource does not exist",

"internalMessage": "No employee found in the database",

"code": 34,

"more info": "http://www.sample.com/api/v1/employee/errors/12345"

}

]

}

* Security/Authentication (HTTP Basic Authentication, OAuth or OAuth2)