***RESTFULL DESIGN PATTERNS***

***Introduction:***

***REST -REpresentation State Transfer***

Is introduced by **Roy Fielding** when he captured the basic architectural pattern that makes the web so successful. Observing how the web pages are organized and how they are linked to each other, REST is modeled around a large number of "Resources" which "link" among each other. Under the WS\* model, "Service" in the SOA is organized as large number of "Resources". Each resource will have a URI that make it globally identifiable.

A resource is represented by some format of "Representation" which is typically extracted by an idempotent HTTP GET. The representation may embed other URI which refers to other resources..

On the other hand, REST down play the "Procedure" aspect and define a small number of "action" based on existing HTTP Methods. HTTP GET is used to get a representation of the resource. To modify a resource, REST use HTTP PUT with the new representation embedded inside the HTTP Body. To delete a resource, REST use HTTP DELETE. At the first glance, it seems REST is quite limiting in terms of the number of procedures that it can supported. It turns out this is not the case, REST allows any "Procedure" (which has a side effect) to use HTTP POST. Effectively, REST categorize the operations by its nature and associate well-defined semantics with these categories (ie: GET for read-only, PUT for update, DELETE for remove, all above are idempotent) while provide an extension mechanism for application-specific operations (i.e.: POST for application procedures which may be non-idempotent).

***Why we use REST full design pattern in HTML?***

Since resource is usually mapped to some state in the system, analyzing its lifecycle is an important step when designing how a resource is created and how an URI should be structured.

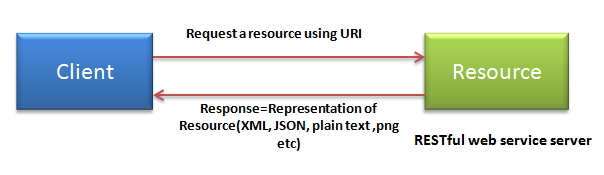
Typically there is some eternal, single Factory Resource which creates other resources. Factory resource typically represents the "type" of resources. Factory resources usually have a static, well-known URI, which is suffixed by a plural form of the resource type. Some examples are...

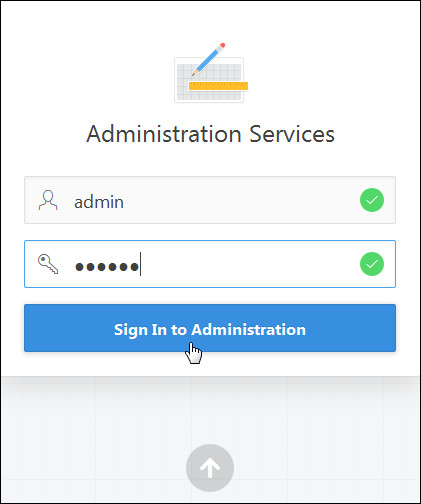
***http://xyz.com/book  
http://xyz.com/user***

"**Resource Instance**", which is created by the "Factory Resource" usually, represents an instance of that resource type. "Resource instances" typically have a limited life span. Their URI typically contains some unique identifier so that the corresponding instance of the resource can be located.

***BENEFITS OF REST FULL DESIGN PATTERNS:***

* You have set of resources that you want to manipulate.
* You want to support navigation between resources.
* Scalability.





***TYPES OF RESOURCES:***

1. Creating Resource.
2. Finding Resource.
3. Modify Resource.
4. Delete Resource.
5. Resource Reference.
6. Transaction Resource.

***SOLUTION:***

RESTful Web services (and Web services in general) pose the first serious test of the principles of REST, as identified by Fielding, the emergence of RESTful Web services, in response to WS– services can serve as an indication that REST is the correct architecture for the Web. This is no longer the case – both styles are being used in all domains. The new challenge is to use them correctly, and to be able to align them to solve the real problems of the enterprise. Can RESTful services scale up to the enterprise-size challenges? We believe so. Amazon, Google, Yahoo, Microsoft, and other big companies have been building large, scalable, extensible, and relatively secure systems on the Web. RESTful services have the same basic principles to follow. This concludes our whirlwind overview of how Web services relate to REST, in theory and in practice.