CRUOUx actual CRUD throughout the stack

Stack	Create	Read	Overwrite	Update	Delete
Front end	**P**ost	**G**et	**P**ut	**P**atch	**D**elete
Local store	**C**reate	**R**ead	**O**verwrite	**U**pdate	**D**elete
Web service	**P**ost	**G**et	**P**ut	**P**atch	**D**elete
Data store	**C**reate	**R**ead	**O**verwrite	**U**pdate	**U**pdate* mark as deleted (timestam ptz)

Front end

(HTML / JS, Cocoa[Touch,] win32, android, others)

1. **P**ost
2. **G**et
3. **P**ut
4. **P**atch
5. **D**elete

Data store local to the front end

(Redis, CoreData, sqlite, localStorage, elasticsearch(assisting index,) etc)

1. **C**reate
2. **R**ead
3. **O**verwrite
4. **U**pdate
5. **D**elete

Web services SOAP / REST API

(Swift w/kitura / perfect / vapor / others, Nginx w/upstream, NodeJS w/express / others, etc)

1. **P**ost
2. **G**et
3. **P**ut
4. **P**atch
5. **D**elete

Data store

(postgreSQL, couchdb, Joyent Manta, MySQL, elasticsearch(assisting index,) etc)

1. **C**reate
2. **R**ead
3. **O**verwrite
4. **U**pdate
5. **U**pdate* mark as deleted
   * Personal preference is a timestamp named _x
   in order not to loose history

Update* mark as deleted
Even though my * Personal preference is a timestamp named _x in order not to loose history

There may or may not be maintenance and performance benefits in producing a intermittent relational layer between your data and your UI to allow the user to delete the key to the data, yet with the data in a 3rd Data table where your actual data resides.

UI	Intermittent	Data
orderID	orderID	orderID
		orderDate
		orderName

Best of all worlds

Editorial tools for LAN / back office

(HTML / JS, Desktop dbs such as FileMaker Pro)

Pre publishing cycle

Deploying your product to your infrastructure

1. Integration receive data from supplier
2. Update digitally entered material to be of desired level
3. Approve editorial work
4. Publish editorial work

Publishing

Deploying your product to your infrastructure FileMaker XML export w/XSLT to

1. Pre rendered HTML
2. Pre rendered sitemap.xml
3. PostgreSQL
4. Elastic Search

Upload images

Example by use case CRUOUx or CRUD(simplified) calls using curl for Kitura, Restify, Express, Vapor, the likes

1. Create by using POST
2. Read by using GET
3. Update replace value at key for :id in payload by using PATCH
4. Overwrite replace :id by payload using PUT
5. Delete by using DELETE or mark for deletion by using PATCH

Create

$ curl -XPOST http[s]://localhost/

payload:

```
{
 "field1":"field1content",
 "field2":"field2content",
 "field3":"field3content"
}
```

Result a record is generated with 3 fields and their respective values as pr payload.

Read

$ curl -XGET http[s]://localhost/:id

No payload.

result:

```
{
 "field1":"field1content",
 "field2":"field2content",
 "field3":"field3content"
}
```

Update

$ curl -XPATCH http[s]://localhost/:id

payload:

```
{
 "field1":"field1contentNew",
 "field2":"field2contentNew"
}
```

Result: only field1 and field2 will have its content replaced for resource / record at key :id.

Result:


```
{
 "field1":"field1contentNew",
 "field2":"field2contentNew",
 "field3":"field3content"
}
```

Overwrite

$ curl -XPUT http[s]://localhost/:id

payload:

```
{
 "field1":"field1contentNew",
 "field2":"field2contentNew"
}
```

Result: All other fields than field1 and field2 will be emptied while field1 and field2 will have their content replaced for resource / record at key :id.

Result:


```
{
 "field1":"field1contentNew",
 "field2":"field2contentNew"
}
```


Note that field3 from POST is gone


```
{
 "field3":"field3content"
}
```

Delete

curl -XDELETE http[s]://localhost/:id

No payload

Result resource at :id will be gone

Gjermund G Thorsen
Keyboard Masseur

“On the day a computer is more to you than paper and pencil, that day you have become a fanatic.”
– Gjermund Gusland Thorsen

“I was evil, I did her harm, still she thought it to be good.
I had given her the greatest gift of them all, the gift of missing me.”
- Gjermund Gusland Thorsen

Made with Deckset