Instituto Politécnico de Tomar

Escola Superior Tecnologia de Tomar

Projecto de Sistemas de Informação

"Documentação de CouchDB e JSON"

2012/2013

Índice

CouchDB

Base de dados de código aberto que se foca na facilidade de uso e na filosofia de ser "uma base de dados" que abrange a web.

É uma base de dados não relacional que utiliza JSON para armazenar os dados e Javascript como linguagem de consulta.

Características:

- Facilidade de replicação;
- Replicação incremental;
- Tolerância a falhas.

Ao contrário das bases de dados relacionais, o CouchDB não armazena os dados e relacionamentos em tabela; cada base de dados é uma colecção de documentos independentes e cada documento mantém os seus próprios dados e esquemas.

JSON ("Javascript Object Notation")

É um formato "leve" para intercâmbio de dados computacionais.

É um subconjunto da notação de objecto Javascript mas o seu uso não requer Javascript exclusivamente.

Características:

- Facilidade de escrita de um analisador JSON (em relação ao XML). Mesmo em Javascript, JSON pode ser analisado trivialmente usando a função eval();
- Crescente suporte através de packages de terceiros. Linguagens suportadas: ActionScript, C/C++, C#, Java, Javascript, PHP, ASP.net, etc.

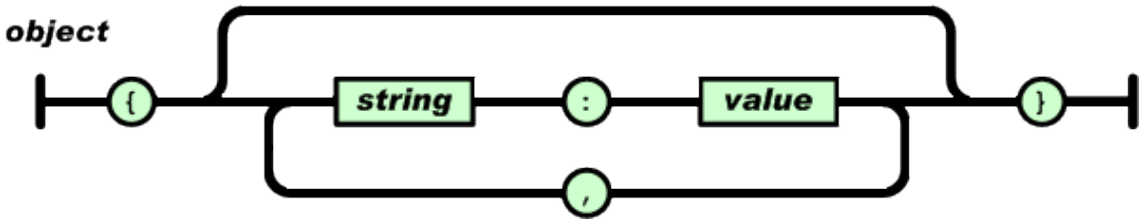
Node.js

O Node.js é um interpretador de Javascript do lado do servidor. Possibilita ao programador a criação de aplicações altamente escaláveis. Os programas são escritos em Javascript, do lado do servidor, e são orientados a eventos assíncronos para minimizar o *overhead* e aumentar a escalabilidade.

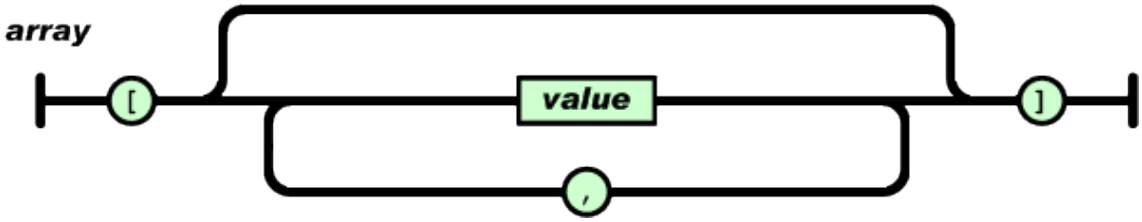
Introdução ao JSON

Em JSON os dados são apresentados da seguinte forma:

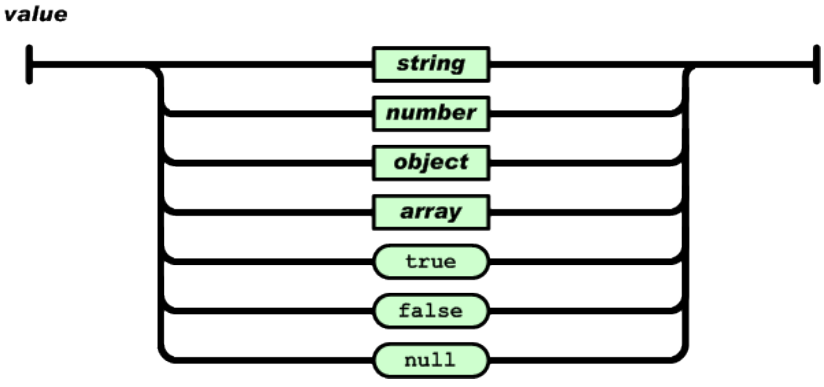
Um objecto é um conjunto desordenado de pares nome/valor. Um objecto começa com { (chave de abertura) e termina com } (chave de fecho). Cada nome é seguido por : (dois pontos) e os pares nome/valor são seguidos por , (vírgula).



Um array é uma colecção de valores ordenados. O array começa com [(colchete de abertura) e termina com] (colchete de fecho). Os valores são separados por , (vírgula).

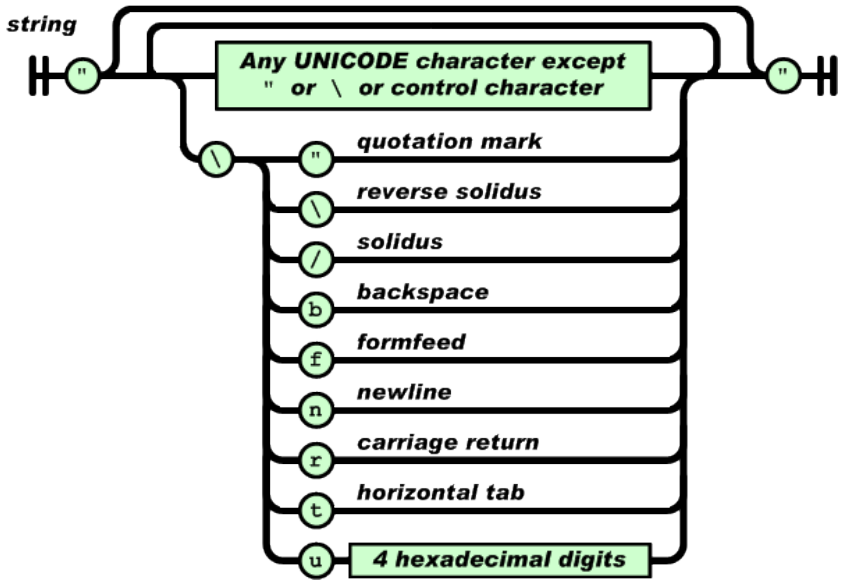


Um valor (value) pode ser uma cadeia de caracteres (string) ou um número ou um valor booleano (true ou false) ou null ou um objecto ou um array. Estas estruturas podem estar aninhadas:

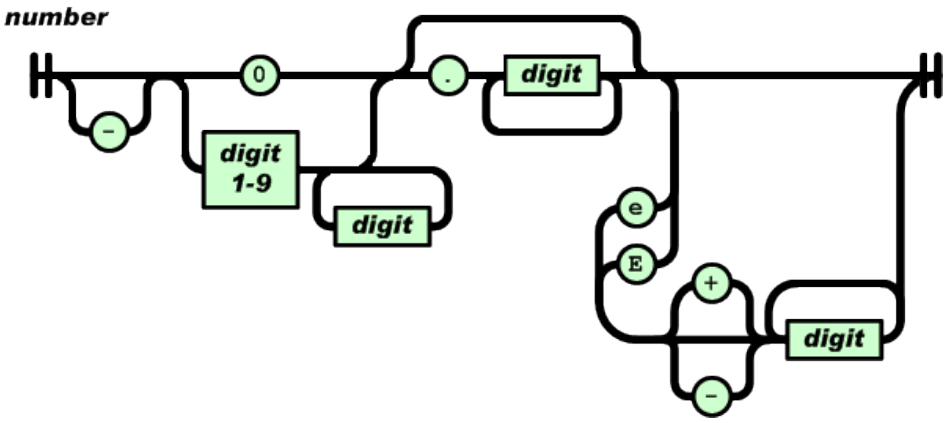


Uma string é uma colecção de um ou mais caracteres Unicode, envolvido entre aspas duplas usando barras invertidas como caracter de escape. Um caracter está

representado como um simples caracter de string. Uma cadeia de caracteres é parecida com uma cadeia de caracteres em C ou Java.



Um número é similar a um número em C ou em Java, excepto quando não se usa números octais ou hexadecimais.



Podem ser inseridos espaços em branco em qualquer parte dos símbolos. Exceptuando pequenos detalhes de codificação, a linguagem é completamente descrita.

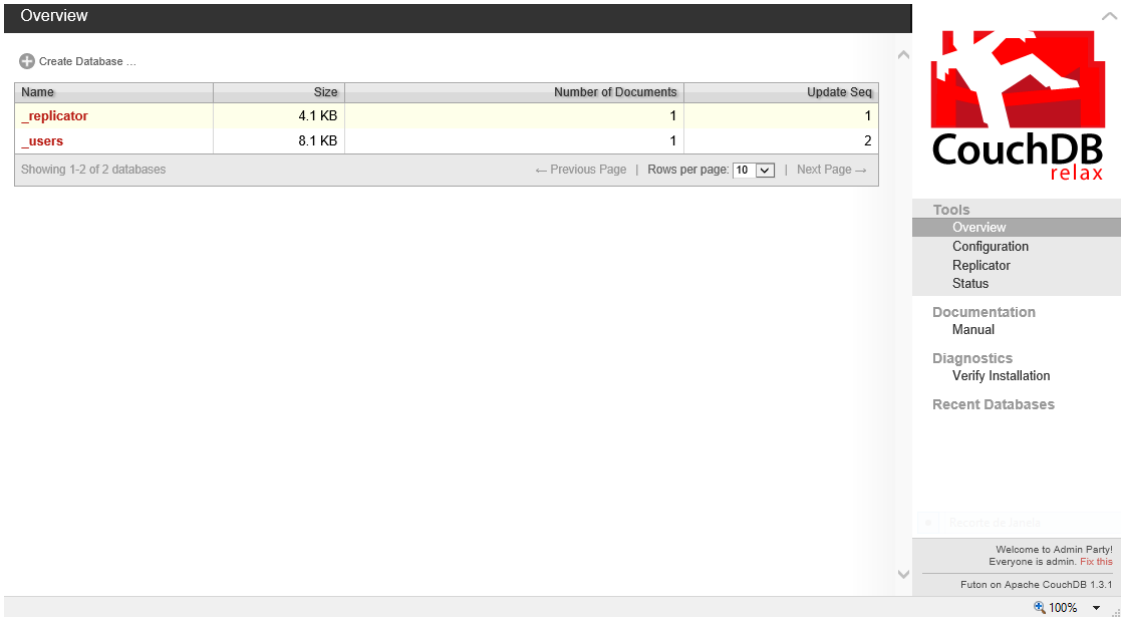
Instalação do CouchDB em S.O. Linux (CentOS)

A instalação do CouchDB é feita através dos seguintes comandos:

```
yum install curl -y
yum install couchdb –y
```

Depois do processo de instalação estar concluído, basta aceder a partir do seguinte endereço: http://<endereço IP do servidor>/_utils

Surge a interface de gestão da BD no ecrã.



Instalação do Node.js em S.O. Linux (CentOS)

A instalação do Node.js pressupõe que sejam instaladas em primeiro lugar algumas dependências. Depois procede-se à instalação propriamente dita (a partir do código fonte):

Instalação das dependências:

```
yum install openssl-devel
yum install gcc-c++
```

Instalação do node.js a partir do código-fonte:

```
mkdir sources
cd sources
wget http://nodejs.org/dist/v0.10.5/node-v0.10.5.tar.gz
tar zxvf node-v0.10.5.tar.gz
cd node-v0.10.5
./configure
make
make install
mv /root/sources/node-v0.10.5/out/Release /opt/node-v0.10.5
ln -s /opt/node-v0.10.5/node /usr/bin/node
```

A interacção com o node.js é feita através de comandos:

```
node –v          # Devolve o nº. da versão do node.js instalada

node <nome do script.js>      # Executa um script
```

Sempre que forem necessárias bibliotecas no node.js estes podem ser procuradas no NPM Registry. Para instalar basta o seguinte comando:

```
npm install <package a instalar>

Exemplo: npm install express
```

Views

Para chamar uma view depois de guardada basta colocar o URL para mesma, por exemplo em localhost:

http://localhost:5984/mydb/_design/graph/_view/graph

mydb -> base de dados onde esta guardada a view

_design/graph -> indica que estamos a usar o documento de design/graph, diferente dos documentos json

_view/graph -> indica que vamos executar uma view com o nome grpah

http://localhost:5984/mydb/_design/graph/_view/graph?key="0001"

A adicao de ?key="0001" permite filtrar o resultado obtido, mostrando apenas os graphs do utilizador

O userId 0001 neste caso. De notar que a view é mesma mas a inclusão de um parametro de pesquisa permite filtrar os dados visualizados.

```
{
  "_id": "_design/graph",
  "_rev": "2-6661a83576a34492fb708137ca1c337a",
  "language": "javascript",
```

"views": {

"graph": {

"map": "function(doc) {\n\tif(doc.type == 'graph'){\n emit(doc.userId, {GraphID: doc._id, Graph:doc.graph});\n}\n \n}"

},

"users": {

"map": "function(doc) {\n\tif(doc.type == 'graph'){\n emit(doc.userId, {GraphID: doc._id, Graph:doc.graph});\n}\n \n}"

}

}

}

http://localhost:5984/mydb/_design/graph/_view/users

http://sitr.us/2009/06/30/database-queries-the-couchdb-way.html

"Map function" funcionam num documento e não podem aceder a dados de outro documento.

A vantagem disto é que as funções podem podem processar os dados em qualquer ordem independentemente do resto da informação.

CouchDB produz indexes estáticos através do output das "view map function" para que pesquisas (queries) a essas "views" sejam rápidas.

Quando um documento e alterado, o CouchD reconstruir os indexes de forma incremental apenas para esses documentos sem ter que reconstruir todos os indexes de inicio.

O CouchDB oferece grande velocidade de desempenho para grandes pedacos de dados.

Isto implica que não seja possível passar parâmetros dinamicamente para os nossos "map function" quando corremos uma pesquisa (query).

Não podemos pedir que seja mostrado apenas os dados de um utilizador com um determinado ultimo nome, a não ser que se queira uma "view" especifica para cada ultimo nome.

Na maioria dos casos e muito complicado criar "views" separadas para cada pesquisa (query) que possa vir a ser necessaria numa situacao pontual futura.

Por isso o que se pode fazer e fazer uma pesquisa "query" a uma "view" com um resultado genérico ao caso exemplificado acima e requisitar apenas os pares "key/value" que coincidam com a "key" em particular.

Referências bibliográficas

[1] CouchDB, in Wikipedia.org, url: https://pt.wikipedia.org/wiki/CouchDB, consultado a 4/6/2013

[2] JSON, in Wikipedia.org, url: https://pt.wikipedia.org/wiki/Json, consultado a 4/6/2013

[3] Introdução ao JSON, in json.org, url: http://www.json.org/json-pt.html, consultado a 4/6/2013

[4] Node.js, in Wikipedia, url: http://en.wikipedia.org/wiki/Nodejs, consultado a 10/06/2013

[5] MapReduce, url: http://www.kchodorow.com/blog/2010/03/15/mapreduce-the-fanfiction/, consultado a: 30/06/2013

[6] MapReduce, url: http://www.slideshare.net/Couchbase/couchconfberlinqueryingwithcouchbase, consultado a: 30/06/2013