

---

# CouchDB application development

Developing and deploying CouchApps

Jakob Westhoff <jakob@php.net>

PHPBarcamp.at

May 3, 2010

# About Me

---

- Jakob Westhoff
  - PHP developer for more several years
  - Computer science student at the TU Dortmund
  - Co-Founder of the PHP Usergroup Dortmund
  - Active in different Open Source projects

# Asking the audience

---

- Who is actively using CouchDB?

# Asking the audience

---

- Who is actively using CouchDB?
- Who played with CouchDB just for fun?

# Asking the audience

---

- Who is actively using CouchDB?
- Who played with CouchDB just for fun?
- Who has heard of CouchDB, but never used it?

# Asking the audience

---

- Who is actively using CouchDB?
- Who played with CouchDB just for fun?
- Who has heard of CouchDB, but never used it?
- Who has already developed a CouchApp?

# Asking the audience

---

- Who is actively using CouchDB?
- Who played with CouchDB just for fun?
- Who has heard of CouchDB, but never used it?
- Who has already developed a CouchApp?
- Who does not know at all, what a CouchApp is?

# Goals of this session

---

- Understand what a CouchApp actually is

# Goals of this session

---

- Understand what a CouchApp actually is
- Learn what features of CouchDB can be used to realize a CouchApp

# Goals of this session

---

- Understand what a CouchApp actually is
- Learn what features of CouchDB can be used to realize a CouchApp
- Get a glimpse of third party tools and frameworks for CouchApp development

# What is CouchDB?

---

- Document based NoSQL database

# What is CouchDB?

---

- Document based NoSQL database
- Written in Erlang

# What is CouchDB?

---

- Document based NoSQL database
- Written in Erlang
- MapReduce based indexing

# What is CouchDB?

---

- Document based NoSQL database
- Written in Erlang
- MapReduce based indexing
- Javascript uses as embedded language

# What is CouchDB?

---

- Document based NoSQL database
- Written in Erlang
- MapReduce based indexing
- Javascript uses as embedded language
- RESTful JSON API

# What is CouchDB?

---

- Document based NoSQL database
- Written in Erlang
- MapReduce based indexing
- Javascript uses as embedded language
- RESTful JSON API
- Build in incremental bi-directional replication

# CouchDB document example

---

- Documents are stored as JSON

# CouchDB document example

- Documents are stored as JSON

```
{
  "_id": "recipe-some-recipe",
  "_rev": "1-859e8f9a06b864ebca3a929750c17537",
  "date": 1270067388,
  "title": "Some_recipe",
  "ingredients": [
    "Sugar",
    "Flour"
  ],
  "instructions": "Some_instructions",
  "user": "user-foobar",
}
```

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as **data store**

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as data store
  - **Stored inside** the CouchDB

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as data store
  - Stored inside the CouchDB
  - **Served** using the CouchDB Http-Server

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as data store
  - Stored inside the CouchDB
  - Served using the CouchDB Http-Server

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as data store
  - Stored inside the CouchDB
  - Served using the CouchDB Http-Server
- Most CouchApps are

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as data store
  - Stored inside the CouchDB
  - Served using the CouchDB Http-Server
- Most CouchApps are
  - Mostly written in Javascript

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as data store
  - Stored inside the CouchDB
  - Served using the CouchDB Http-Server
- Most CouchApps are
  - Mostly written in Javascript
  - Using some sort of Javascript library

# What is a CouchApp?

---

- Application build on top of CouchDB which fits the following criteria:
  - Using CouchDB as data store
  - Stored inside the CouchDB
  - Served using the CouchDB Http-Server
- Most CouchApps are
  - Mostly written in Javascript
  - Using some sort of Javascript library
- CouchDB replication can be used to deploy applications

# Excursion: Futon

---

- Futon is CouchDBs administration interface

# Excursion: Futon

---

- Futon is CouchDBs administration interface
- Automatically available on every CouchDB
  - [http://localhost:5984/\\_utils/](http://localhost:5984/_utils/)

# Excursion: Futon

- Futon is CouchDBs administration interface
- Automatically available on every CouchDB
  - [http://localhost:5984/\\_utils/](http://localhost:5984/_utils/)

The screenshot displays the 'Overview' page of the CouchDB Futon interface. At the top left, there is a '+ Create Database ...' button. Below it is a table listing databases. The table has four columns: 'Name', 'Size', 'Number of Documents', and 'Update Seq'. The rows are as follows:

Name	Size	Number of Documents	Update Seq
livesearch	12.1 KB	3	3
_users	108.1 KB	6	29
[blurred]	52.1 KB	9	55
foo	1.0 MB	1	4
[blurred]	123.8 MB	21	115

Below the table, there are navigation controls: 'Showing 1-5 of 5 databases', '← Previous Page', 'Rows per page: 10', and 'Next Page →'. On the right side, there is a sidebar with the CouchDB logo (a red silhouette of a person falling) and the text 'CouchDB relax'. Below the logo is a 'Tools' menu with options: Overview, Configuration, Replicator, Status, and Test Suite. Underneath is a 'Recent Databases' section listing '\_users', 'livesearch', and another blurred entry. At the bottom of the sidebar, there is a 'Signup or Login' link and the text 'Futon on Apache CouchDB 0.11.0'.

# Which CouchDB Version to use?

---

- CouchDB 0.11 or greater

# Which CouchDB Version to use?

---

- CouchDB 0.11 or greater
  - User management

# Which CouchDB Version to use?

---

- CouchDB 0.11 or greater
  - User management
  - Authentication API: Cookie, OAuth, plain-HTTP

# Which CouchDB Version to use?

---

- CouchDB 0.11 or greater
  - User management
  - Authentication API: Cookie, OAuth, plain-HTTP
  - Per-db reader access lists

# Which CouchDB Version to use?

---

- CouchDB 0.11 or greater
  - User management
  - Authentication API: Cookie, OAuth, plain-HTTP
  - Per-db reader access lists
  - Validation functions

# Which CouchDB Version to use?

---

- CouchDB 0.11 or greater
  - User management
  - Authentication API: Cookie, OAuth, plain-HTTP
  - Per-db reader access lists
  - Validation functions
  - Url rewriting

# Which CouchDB Version to use?

---

- CouchDB 0.11 or greater
  - User management
  - Authentication API: Cookie, OAuth, plain-HTTP
  - Per-db reader access lists
  - Validation functions
  - Url rewriting
  - Vhost configuration

# CouchApp techniques

---

- Store all application files as **document attachments**

# CouchApp techniques

---

- Store all application files as **document attachments**
  - Attachments can be stored for any document inside CouchDB

# CouchApp techniques

---

- Store all application files as **document attachments**
  - Attachments can be stored for any document inside CouchDB
  - Can be directly accessed using the RESTful interface

# CouchApp techniques

---

- Store all application files as document attachments
  - Attachments can be stored for any document inside CouchDB
  - Can be directly accessed using the RESTful interface
- Use CouchDBs **authentication backend** for user management

# CouchApp techniques

---

- Store all application files as document attachments
  - Attachments can be stored for any document inside CouchDB
  - Can be directly accessed using the RESTful interface
- Use CouchDBs authentication backend for user management
- Use **validation** function for access restrictions

# CouchApp techniques

---

- Store all application files as document attachments
  - Attachments can be stored for any document inside CouchDB
  - Can be directly accessed using the RESTful interface
- Use CouchDBs authentication backend for user management
- Use validation function for access restrictions
- Use **show** and **list** functions to output HTML

# CouchApp techniques

---

- Store all application files as document attachments
  - Attachments can be stored for any document inside CouchDB
  - Can be directly accessed using the RESTful interface
- Use CouchDBs authentication backend for user management
- Use validation function for access restrictions
- Use show and list functions to output HTML
- Access CouchDB using **Javascript** and **XHR**

# CouchApp techniques

---

- Store all application files as document attachments
  - Attachments can be stored for any document inside CouchDB
  - Can be directly accessed using the RESTful interface
- Use CouchDBs authentication backend for user management
- Use validation function for access restrictions
- Use show and list functions to output HTML
- Access CouchDB using Javascript and XHR
- Use CouchDB **vhosts** and **url rewriting** for nice urls

# Authentication backend: User Management

---

Managing users:

# Authentication backend: User Management

Managing users:

- Futon:



# Authentication backend: User Management

Managing users:

- Futon:



- RESTful API:

- Create a new document in the `_users` database
- [http://localhost:5984/\\_users/org.couchdb.user:foo](http://localhost:5984/_users/org.couchdb.user:foo)

# Authentication backend: Logging in (Cookie)

---

Logging in:

# Authentication backend: Logging in (Cookie)

---

Logging in:

- **POST** username and password to
  - `http://localhost:5984/_session`

# Authentication backend: Logging in (Cookie)

Logging in:

- **POST** username and password to
  - `http://localhost:5984/_session`
- Parameters **user** and **password**
- `application/x-www-form-urlencoded` encoded

# Authentication backend: Logging in (Cookie)

Logging in:

- **POST** username and password to
  - `http://localhost:5984/_session`
- Parameters **user** and **password**
- `application/x-www-form-urlencoded` encoded
  
- Post:  
`name=foo&password=bar`

# Authentication backend: Logging in (Cookie)

Logging in:

- **POST** username and password to
  - `http://localhost:5984/_session`
- Parameters **user** and **password**
- `application/x-www-form-urlencoded` encoded
  
- Post:  
`name=foo&password=bar`
  
- Response:  
`{"ok": true, "name": "foo", "roles": []}`

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`
- Multiple design documents allowed per database

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`
- Multiple design documents allowed per database
- Used to define:

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`
- Multiple design documents allowed per database
- Used to define:
  - View functions (Map&Reduce)

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`
- Multiple design documents allowed per database
  
- Used to define:
  - View functions (Map&Reduce)
  - List functions

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`
- Multiple design documents allowed per database
- Used to define:
  - View functions (Map&Reduce)
  - List functions
  - Show functions

# Excursion: Design documents

---

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`
- Multiple design documents allowed per database
- Used to define:
  - View functions (Map&Reduce)
  - List functions
  - Show functions
  - Validation functions

# Excursion: Design documents

- Design documents supply certain special functionalities per db
- Stored with special id `_design/some-id`
- Multiple design documents allowed per database
- Used to define:
  - View functions (Map&Reduce)
  - List functions
  - Show functions
  - Validation functions
  - ...

# Validation functions: Validate updates

---

- Validation of document creations or updates using Javascript functions

# Validation functions: Validate updates

---

- Validation of document creations or updates using Javascript functions
- Registration: Property `validate_doc_update` on any design document

# Validation functions: Validate updates

---

- Validation of document creations or updates using Javascript functions
- Registration: Property `validate_doc_update` on any design document
- Multiple validation functions on different design documents possible

# Validation functions: Validate updates

---

- Validation of document creations or updates using Javascript functions
- Registration: Property `validate_doc_update` on any design document
- Multiple validation functions on different design documents possible
  - Called one after another

# Validation functions: Validate updates

- Validation of document creations or updates using Javascript functions
- Registration: Property `validate_doc_update` on any design document
- Multiple validation functions on different design documents possible
  - Called one after another
  - No defined execution order: Functions need to be isolated

# Validation functions: Validate updates

- Validation of document creations or updates using Javascript functions
- Registration: Property `validate_doc_update` on any design document
- Multiple validation functions on different design documents possible
  - Called one after another
  - No defined execution order: Functions need to be isolated
  - One fails: validation fails

# Validation functions: Validate updates II

---

- Example

# Validation functions: Validate updates II

---

- Example
  - Ensure all documents have a `type` field
  - `added` is not allowed to be changed after document creation

# Validation functions: Validate updates II

## ■ Example

- Ensure all documents have a **type** field
- **added** is not allowed to be changed after document creation

```
function(newDoc, oldDoc, userCtx) {  
  if (!newDoc.type) {  
    throw({ forbidden: "Mandatory field 'type' is missing" });  
  }  
  
  if (oldDoc && toJSON(oldDoc.added) !== toJSON(newDoc.added)) {  
    throw(...)  
  }  
}
```

# Validation functions: Access control

---

- Use `validation_doc_update` function as access control system

# Validation functions: Access control

---

- Use `validation_doc_update` function as access control system
  - Ensure the user is **logged in**
  - Ensure **username** and **author** field are identical

# Validation functions: Access control

- Use `validation_doc_update` function as access control system
  - Ensure the user is **logged in**
  - Ensure **username** and **author** field are identical

```
function(newDoc, oldDoc, userCtx) {
  if (!userCtx.name) {
    throw({ forbidden: "You need to be logged in to
      change documents." });
  }

  if ((oldDoc && userCtx.name !== oldDoc.author) ||
    userCtx.name !== newDoc.author) {
    throw (...)
  }
}
```

# Show and list functions

---

- User formatted data instead of plain JSON

# Show and list functions

---

- User formatted data instead of plain JSON
- **Show** functions format **documents**

# Show and list functions

---

- User formatted data instead of plain JSON
- **Show** functions format **documents**
- **List** functions format **views**

# Show and list functions

---

- User formatted data instead of plain JSON
- **Show** functions format **documents**
- **List** functions format **views**
- Multiple show and list functions are allowed **per-design-document**

# Show and list functions

- User formatted data instead of plain JSON
- **Show** functions format **documents**
- **List** functions format **views**
- Multiple show and list functions are allowed **per-design-document**
  
- Stored as `shows` and `lists` object

```
"shows" : {  
  "summary" : "function(doc, req) {...}" ,  
  "detail" : "function(doc, req) {...}"  
}
```

# Show function to output HTML

---

- Arguments

# Show function to output HTML

---

- Arguments
  - `doc` - Document to be returned

# Show function to output HTML

---

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object

# Show function to output HTML

---

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context

# Show function to output HTML

---

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context
    - GET parameter

# Show function to output HTML

---

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context
    - GET parameter
    - Accept header

# Show function to output HTML

---

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context
    - GET parameter
    - Accept header
    - ...

# Show function to output HTML

---

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context
    - GET parameter
    - Accept header
    - ...
- Return value is a response object

# Show function to output HTML

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context
    - GET parameter
    - Accept header
    - ...
- Return value is a response object
  - `body` - Content returned to the caller

# Show function to output HTML

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context
    - GET parameter
    - Accept header
    - ...
- Return value is a response object
  - `body` - Content returned to the caller
  - `headers` - HTML headers (Can be used to set Content-Type)

# Show function to output HTML

- Arguments
  - `doc` - Document to be returned
  - `req` - Request object
    - User context
    - GET parameter
    - Accept header
    - ...
- Return value is a response object
  - `body` - Content returned to the caller
  - `headers` - HTML headers (Can be used to set Content-Type)
  - `base64` - Can be used instead of `body` to send binary data

# Show function to output HTML II

---

- Example

# Show function to output HTML II

---

- Example
  - Output document `type` and `author` as HTML

# Show function to output HTML II

- Example
  - Output document `type` and `author` as HTML

```
function( doc, req ) {  
  
    return {  
        body: [  
            "<h1>The_",  
            doc.type,  
            "_document_has_been_created_by_",  
            doc.author,  
            "</h1>"  
        ].join(" ");  
    };  
  
}
```

# List functions: Streaming API

---

- Arguments

# List functions: Streaming API

---

- Arguments
  - `head` - Statistical data about the view

# List functions: Streaming API

---

- Arguments
  - `head` - Statistical data about the view
  - `req` - Request object

# List functions: Streaming API

---

- Arguments
  - `head` - Statistical data about the view
  - `req` - Request object
- Return value is not used

# List functions: Streaming API

---

- Arguments
  - `head` - Statistical data about the view
  - `req` - Request object
- Return value is not used
- Streaming API

# List functions: Streaming API

---

- Arguments
  - `head` - Statistical data about the view
  - `req` - Request object
- Return value is not used
- Streaming API
  - `getRow()` - Retrieve next row from the view, or null at the end

# List functions: Streaming API

- Arguments
  - `head` - Statistical data about the view
  - `req` - Request object
- Return value is not used
- Streaming API
  - `getRow()` - Retrieve next row from the view, or null at the end
  - `send(data)` - Send out the next piece of data to the caller

# List functions: Example

---

- Example

# List functions: Example

---

- Example
  - Output authors in list as unordered HTML list

# List functions: Example

- Example

- Output authors in list as unordered HTML list

```
function(head, req) {  
  var cur;  
  
  send('<ul>');  
  
  while(cur = getRow()) {  
    send('<li>' + cur.author + '</li>');  
  }  
  
  send('</ul>');  
}
```

# Query show and list functions

---

- Queried based on **design document** and **name**

# Query show and list functions

---

- Queried based on **design document** and **name**
- Show functions
- **GET** `/db/_design/mydesign/_show/show-name/doc-id`

# Query show and list functions

- Queried based on **design document** and **name**
- Show functions
- **GET** /db/\_design/mydesign/\_show/show-name/doc-id
- List functions
- **GET** /db/\_design/mydesign/\_list/list-name/view-name

# Show and list functions in real applications

---

- Use a templating system

# Show and list functions in real applications

---

- Use a templating system
  - Javascript micro template language based on John Resigs idea
  - <http://github.com/furf/jquery-template>

# Show and list functions in real applications

---

- Use a templating system
  - Javascript micro template language based on John Resigs idea
  - <http://github.com/furf/jquery-template>
- Use integrated `provides` and `registerType` functions to serve multiple formats based on the accept header

# Show and list functions in real applications

- Use a templating system
  - Javascript micro template language based on John Resigs idea
  - <http://github.com/furf/jquery-template>
- Use integrated `provides` and `registerType` functions to serve multiple formats based on the accept header
- Maybe do some sort of user authentication before showing content

# Url rewriting

---

- Urls like this one are neither readable nor usable

`/db/_design/mydesign/_list/list-name/view-name?limit=5`

# Url rewriting

---

- Urls like this one are neither readable nor usable

`/db/_design/mydesign/_list/list-name/view-name?limit=5`

- CouchDB supports internal url rewriting

# Url rewriting

---

- Urls like this one are neither readable nor usable

`/db/_design/mydesign/_list/list-name/view-name?limit=5`

- CouchDB supports internal url rewriting
  - Defined as **array** of **rewrites** on any design document

# Url rewriting

- Urls like this one are neither readable nor usable

```
/db/_design/mydesign/_list/list-name/view-name?limit=5
```

- CouchDB supports internal url rewriting
  - Defined as **array** of **rewrites** on any design document

```
"rewrites": [  
  {  
    "from": "/index",  
    "to": "_list/list-name/view-name?limit=5"  
  }  
]
```

# Advanced url rewriting

---

- Using named placeholders
  - Match each path segment

```
"from" : " /entry /:doc" ,  
"to" : " _show/show-name /:doc"
```

# Advanced url rewriting

- Using named placeholders
  - Match each path segment

```
"from": "/entry/:doc",  
"to": "_show/show-name/:doc"
```

- Using `match all` at the end of an url
  - Appends given query parameters

```
"from": "/entry/*",  
"to": "_show/show-name/*"
```

# Advanced url rewriting II

- Rewriting to query parameters

```
"from": "/page/:start",  
"to": "_list/list-name/pages",  
"query": {  
  "startkey": ":start"  
}
```

# Advanced url rewriting II

- Rewriting to query parameters

```
"from": "/page/:start",  
"to": "_list/list-name/pages",  
"query": {  
  "startkey": ":start"  
}
```

- Address a context outside of the design document

- Targets are always relative to the design document
- Relative navigation with .. is allowed

```
"from": "/raw/:doc",  
"to": "../../:doc"
```

# Advanced url rewriting III

- Rewriting based on HTTP methods

```
{
  "from" : "/entry/*" ,
  "to" : "_show/show-name/*" ,
  "method" : "GET"
},
{
  "from" : "/entry/*" ,
  "to" : "_update/update-name/*" ,
  "method" : "POST"
},
```

# Query rewritten urls

---

- Queried on design document using special `_rewrite` path

# Query rewritten urls

---

- Queried on design document using special `_rewrite` path
- `/db/_design/mydesign/_rewrite/some-url`

# Query rewritten urls

---

- Queried on design document using special `_rewrite` path
- `/db/_design/mydesign/_rewrite/some-url`
- Better, but far from being perfect → **Vhost configuration**

# Vhost configuration

---

- CouchDB supports url changing based on the provided Host

# Vhost configuration

---

- CouchDB supports url changing based on the provided Host
- Custom root path for every host/domain

# Vhost configuration

---

- CouchDB supports url changing based on the provided Host
- Custom root path for every host/domain
- Configured in `local.ini`

# Vhost configuration

- CouchDB supports url changing based on the provided Host
- Custom root path for every host/domain
- Configured in `local.ini`
- Direct all requests to the `_rewrite` handler

`[vhosts]`

`example.com = /mydb/_design/mydesign/_rewrite`

# Using CouchApp for deployment

---

- CouchApp does not only categorize these application type

# Using CouchApp for deployment

---

- CouchApp does not only categorize these application type
- It is a utility written in Python
  - <http://github.com/couchapp/couchapp>

# Using CouchApp for deployment

---

- CouchApp does not only categorize these application type
- It is a utility written in Python
  - <http://github.com/couchapp/couchapp>
- Deployment system for CouchApps to CouchDBs

# Using CouchApp for deployment

---

- CouchApp does not only categorize these application type
- It is a utility written in Python
  - <http://github.com/couchapp/couchapp>
- Deployment system for CouchApps to CouchDBs
- Providing some common widgets like login or profile

# CouchApp: Folder structure

- Organize your applications into a special folder structure

```
.
|-- _attachments
|-- _id
|-- lists
|-- rewrites
|-- shows
|   '-- myshow.js
|-- updates
|-- validate_doc_update.js
'-- views
    '-- myview
        |-- map.js
        '-- reduce.js
```

# CouchApp: Javascript preprocessing

---

- CouchApps preprocesses Javascript before deploying it

# CouchApp: Javascript preprocessing

---

- CouchApps preprocesses Javascript before deploying it
- Provides two useful macros: `!code` and `!json`

# CouchApp: Javascript preprocessing

---

- CouchApps preprocesses Javascript before deploying it
- Provides two useful macros: `!code` and `!json`
- `// !code lib/helper/validation_utilities.js`

# CouchApp: Javascript preprocessing

- CouchApps preprocesses Javascript before deploying it
- Provides two useful macros: `!code` and `!json`
- `// !code lib/helper/validation_utilities.js`
  - Include Javascript code from any other file
  - Reuse code throughout your application functions

# CouchApp: Javascript preprocessing

- CouchApps preprocesses Javascript before deploying it
- Provides two useful macros: `!code` and `!json`
- `// !code lib/helper/validation_utilities.js`
  - Include Javascript code from any other file
  - Reuse code throughout your application functions
- `// !json lib.templates.mytemplate`

# CouchApp: Javascript preprocessing

- CouchApps preprocesses Javascript before deploying it
- Provides two useful macros: `!code` and `!json`
- `// !code lib/helper/validation_utilities.js`
  - Include Javascript code from any other file
  - Reuse code throughout your application functions
- `// !json lib.templates.mytemplate`
  - Dot notation of path to `.json` file
  - Available using the full object path:  
`lib.templates.mytemplate`

# CouchApp: Deployment

---

- Specifying all parameters on the commandline

```
couchapp push http://user:pw@localhost:5984/mydb
```

# CouchApp: Deployment

- Specifying all parameters on the commandline

```
couchapp push http://user:pw@localhost:5984/mydb
```

- Use `.couchapprc` file

```
{
  "env": {
    "default": {
      "db": "http://user:pw@localhost:5984/mydb-dev"
    },
    "production": {
      "db": "http://user:pw@example.com/mydb"
    }
  }
}
```

# Further reading

---

- Things we didn't have time to discuss

# Further reading

---

- Things we didn't have time to discuss
  - Update functions
    - [http://wiki.apache.org/couchdb/Document\\_Update\\_Handlers](http://wiki.apache.org/couchdb/Document_Update_Handlers)

# Further reading

---

- Things we didn't have time to discuss
  - Update functions
    - [http://wiki.apache.org/couchdb/Document\\_Update\\_Handlers](http://wiki.apache.org/couchdb/Document_Update_Handlers)
  - View functions
    - [http://wiki.apache.org/couchdb/Introduction\\_to\\_CouchDB\\_views](http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views)

# Further reading

---

- Things we didn't have time to discuss
  - Update functions
    - [http://wiki.apache.org/couchdb/Document\\_Update\\_Handlers](http://wiki.apache.org/couchdb/Document_Update_Handlers)
  - View functions
    - [http://wiki.apache.org/couchdb/Introduction\\_to\\_CouchDB\\_views](http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views)
  - `_changes` stream
    - <http://books.couchdb.org/relax/reference/change-notifications>

# Further reading

- Things we didn't have time to discuss
  - Update functions
    - [http://wiki.apache.org/couchdb/Document\\_Update\\_Handlers](http://wiki.apache.org/couchdb/Document_Update_Handlers)
  - View functions
    - [http://wiki.apache.org/couchdb/Introduction\\_to\\_CouchDB\\_views](http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views)
  - `_changes` stream
    - <http://books.couchdb.org/relax/reference/change-notifications>
  - Evently javascript library
    - <http://vimeo.com/9847214>
- Further documentation

# Further reading

- Things we didn't have time to discuss
  - Update functions
    - [http://wiki.apache.org/couchdb/Document\\_Update\\_Handlers](http://wiki.apache.org/couchdb/Document_Update_Handlers)
  - View functions
    - [http://wiki.apache.org/couchdb/Introduction\\_to\\_CouchDB\\_views](http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views)
  - `_changes` stream
    - <http://books.couchdb.org/relax/reference/change-notifications>
  - Evently javascript library
    - <http://vimeo.com/9847214>
- Further documentation
  - CouchDB: The definitive guide
    - <http://books.couchdb.org/relax/>

# Further reading

- Things we didn't have time to discuss
  - Update functions
    - [http://wiki.apache.org/couchdb/Document\\_Update\\_Handlers](http://wiki.apache.org/couchdb/Document_Update_Handlers)
  - View functions
    - [http://wiki.apache.org/couchdb/Introduction\\_to\\_CouchDB\\_views](http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views)
  - `_changes` stream
    - <http://books.couchdb.org/relax/reference/change-notifications>
  - Evently javascript library
    - <http://vimeo.com/9847214>
- Further documentation
  - CouchDB: The definitive guide
    - <http://books.couchdb.org/relax/>
  - The CouchDB wiki
    - <http://wiki.apache.org/couchdb/>

Questions, comments or annotations?

Slides: <http://westhoffswelt.de/portfolio.htm>

Contact: Jakob Westhoff <[jakob@php.net](mailto:jakob@php.net)>

Twitter: @jakobwesthoff

Please leave comments and vote at: <http://joind.in/1612>