Clojure Web Development

Philipp Schirmacher | Stefan Tilkov | innoQ

We'll take care of it. Personally.

© 2011 innoQ Deutschland GmbH

© 2011 innoQ Deutschland GmbH

Clojure

A practical Lisp variant for the JVM

Functional programming

Dynamic Typing

Full-featured macro system

Concurrent programming support

Bi-directional Java interop

Immutable persistent data structures

innoQ

© 2011 innoQ Deutschland GmbH

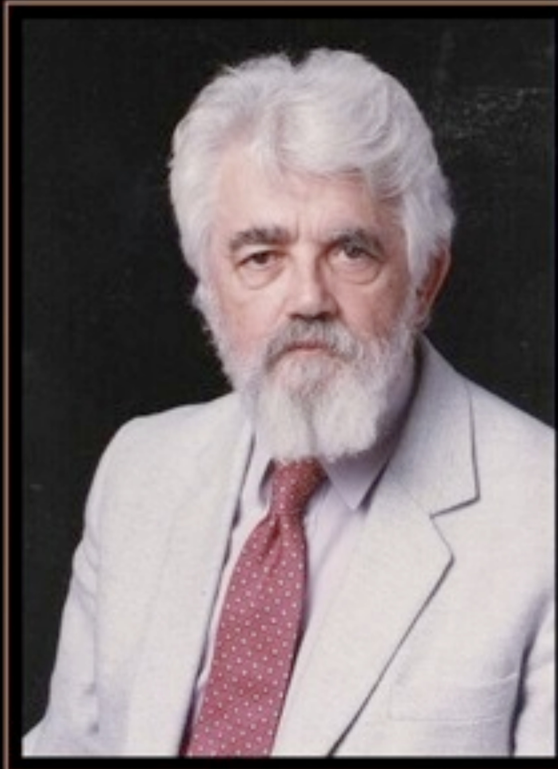
Lisp??

innoQ

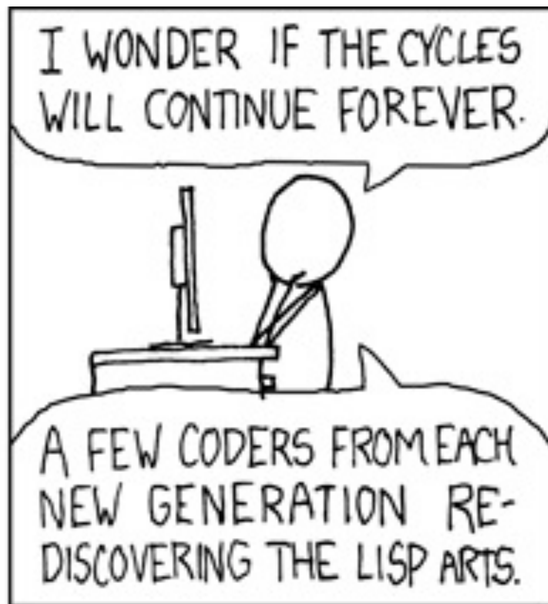
© 2011 innoQ Deutschland GmbH

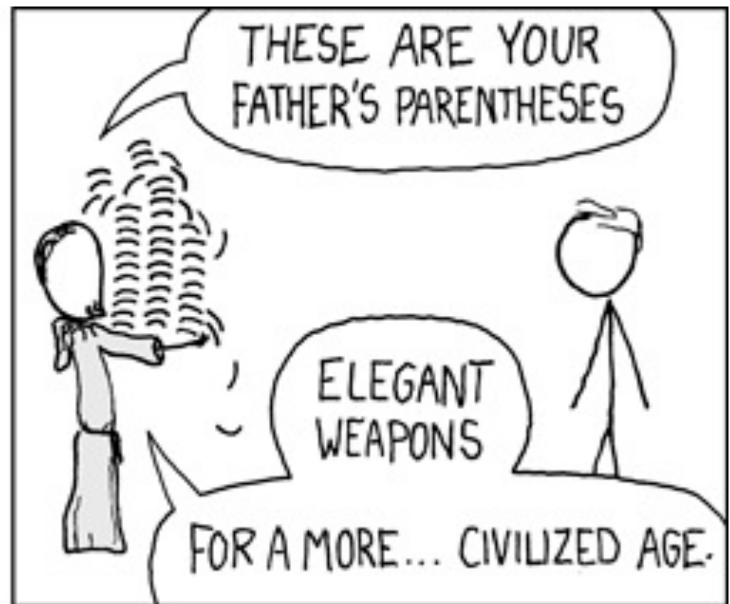
Lots of irritating silly parentheses?

© 2011 innoQ Deutschland GmbH



© 2011 innoQ Deutschland GmbH





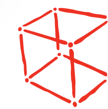
http://xkcd.com/297/

innoQ
© 2011 innoQ Deutschland GmbH

http://www.flickr.com/photos/nicolasrolland/3063007013/

Rich Hickey
http://www.tbray.org/ongoing/When/200x/2008/09/25/-big/R0010774.jpg.html

Clojure Environment



NetBeans IDE

IntelliJIDEA

Clojuresque (Gradle)

maven

Leiningen

innoQ

© 2011 innoQ Deutschland GmbH

Data structures

Numbers	`2 3 4 0.234` `3/5 -2398989892820093093090292321`
Strings	`"Hello" "World"`
Characters	`\a \b \c`
Keywords	`:first :last`
Symbols	`a b c`
Regexps	`#"Ch.*se"`
Lists	`(a b c)` `((:first :last "Str" 3) (a b))`
Vectors	`[2 4 6 9 23]` `[2 4 6 [8 9] [10 11] 9 23]`
Maps	`{:de "Deutschland", :fr "France"}`
Sets	`#{"Bread" "Cheese" "Wine"}`

innoQ

© 2011 innoQ Deutschland GmbH

Syntax

innoQ

© 2011 innoQ Deutschland GmbH

“You’ve just seen it”

– Rich Hickey

innoQ

© 2011 innoQ Deutschland GmbH

Generic Data Types

innoQ

© 2012 innoQ Deutschland GmbH

```clojure
{:name "Clojure"
 :features [:functional :jvm :parens]
 :creator "Rich Hickey"
 :stable-version {:number "1.4"
                  :release "2012/04/18"}}
```

innoQ

© 2012 innoQ Deutschland GmbH

Functions

innoQ

© 2012 innoQ Deutschland GmbH

```
(+ 1 2)
> 3
```

```
(:city {:name "innoQ"
        :city "Monheim"})

> "Monheim"
```

```
(map inc [1 2 3])

> (2 3 4)
```

innoQ

© 2012 innoQ Deutschland GmbH

```
(defn activity [weather]
  (if (nice? weather)
    :surfing
    :playstation))
```

© 2012 innoQ Deutschland GmbH

innoQ

```
(defn make-adder [x]
  (fn [y]
    (+ x y)))


(def add-two (make-adder 2))


(add-two 3)
> 5
```

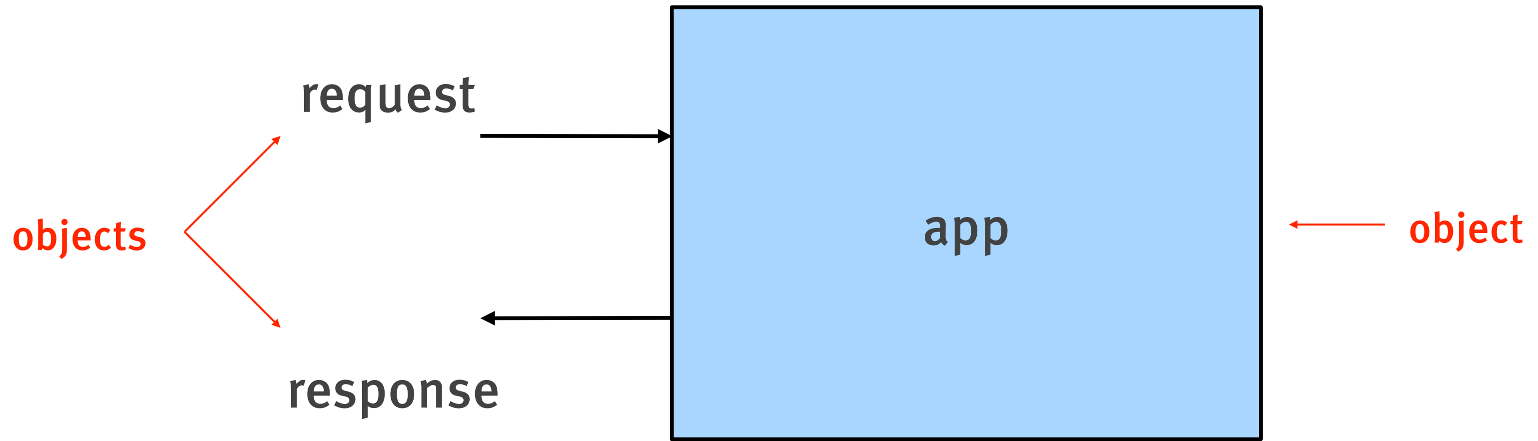
innoQ

© 2012 innoQ Deutschland GmbH

Web Development?

innoQ

© 2012 innoQ Deutschland GmbH



© 2012 innoQ Deutschland GmbH

```java
public interface Servlet  {

     void init(ServletConfig servletConfig)
          throws ServletException;

     ServletConfig getServletConfig();

     void service(ServletRequest servletRequest,
                       ServletResponse servletResponse)
          throws ServletException, IOException;

     String getServletInfo();

     void destroy();
}
```

© 2012 innoQ Deutschland GmbH

```java
public interface HttpServletRequest extends ServletRequest {

    public String getAuthType();

    public Cookie[] getCookies();

    public Enumeration<String> getHeaders(String name);

    public Enumeration<String> getHeaderNames();

    public String getMethod();

    public String getQueryString();

    public String getRemoteUser();

    public HttpSession getSession(boolean create);

    public boolean authenticate(HttpServletResponse response)
        throws IOException,ServletException;

    public void login(String username, String password)
        throws ServletException;
...
```

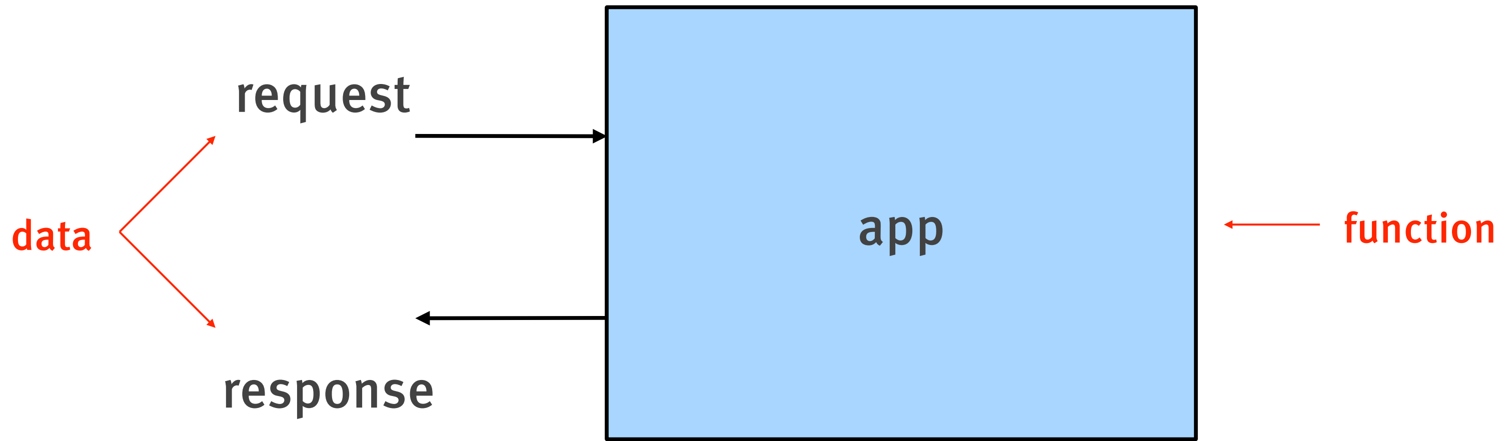
innoQ

© 2012 innoQ Deutschland GmbH

```java
public interface HttpServletResponse extends ServletResponse {

    public void addCookie(Cookie cookie);

    public boolean containsHeader(String name);

    public void sendError(int sc, String msg) throws IOException;

    public void sendRedirect(String location) throws IOException;

    public void setDateHeader(String name, long date);

    public void addDateHeader(String name, long date);

    public void setHeader(String name, String value);

    public void addHeader(String name, String value);

    public void setStatus(int sc);

    public int getStatus();
...
```

innoQ

© 2012 innoQ Deutschland GmbH



© 2012 innoQ Deutschland GmbH

Ring

© 2012 innoQ Deutschland GmbH

```clojure
(defn hello-world-app [req]
  {:status 200
   :headers {"Content-Type" "text/plain"}
   :body "Hello, World!"})


(hello-world-app {:uri "/foo"
                  :request-method :get})
> {...}


(run-jetty hello-world-app {:port 8080})
```

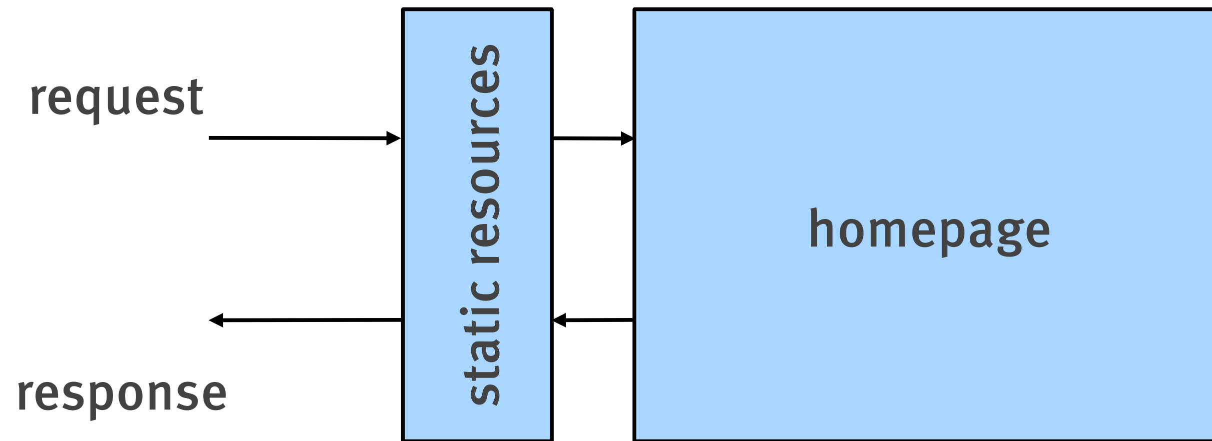
innoQ

© 2012 innoQ Deutschland GmbH

```clojure
(defn my-first-homepage [req]
  {:status 200
   :headers {"Content-Type" "text/html"}
   :body (str "<html><head>"
              "<link href=\"/pretty.css\" ...>"
              "</head><body>"
              "<h1>Welcome to my Homepage</h1>"
              (java.util.Date.)
              "</body></html>")})
```

innoQ

© 2012 innoQ Deutschland GmbH



© 2012 innoQ Deutschland GmbH

innoQ

```
(defn decorate [webapp]
  (fn [req]
    ...before webapp...
    (webapp req)
    ...after webapp...))
```

innoQ

© 2012 innoQ Deutschland GmbH

```
(defn decorate [webapp]
  (fn [req]
    (if (static-resource? req)
      (return-resource req)
      (webapp req))))
```

innoQ
© 2012 innoQ Deutschland GmbH

```
(defn wrap-resource [handler root-path]
  (fn [request]
    (if-not (= :get (:request-method request))
      (handler request)
      (let [path (extract-path request)]
        (or (resource-response path {:root root-path})
            (handler request))))))
```

innoQ

© 2012 innoQ Deutschland GmbH

```
(defn my-first-homepage [req] ...)

(def webapp
  (wrap-resource my-first-homepage "public"))

(run-jetty webapp {:port 8080})
```

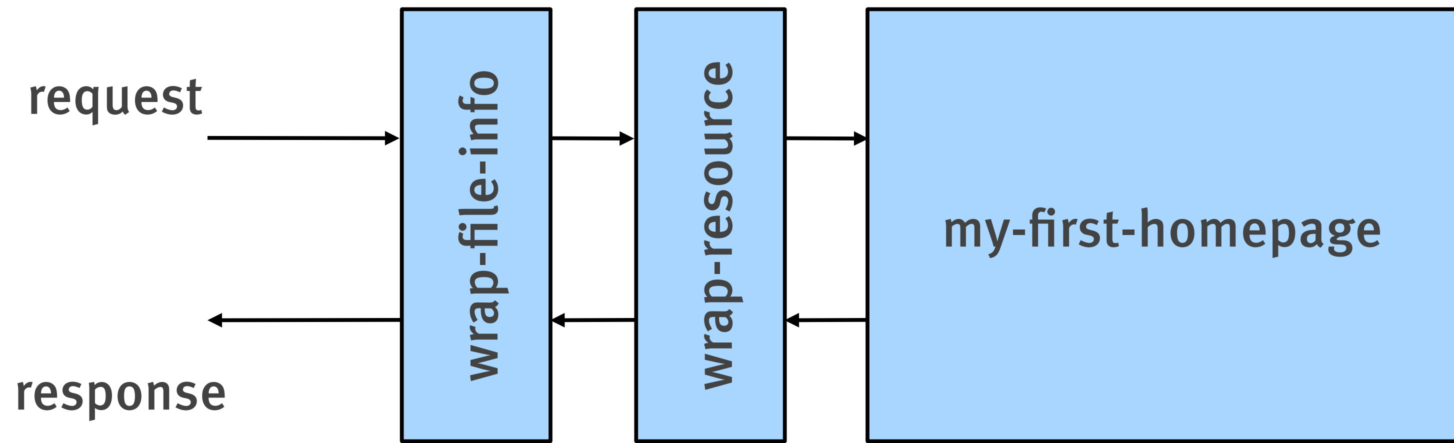
innoQ

© 2012 innoQ Deutschland GmbH

```
(webapp {:uri "/pretty.css"
         :request-method :get
         :headers {}})
> {:status 200
   :headers {}
   :body #<File ...resources/public/pretty.css>}
```

innoQ

© 2012 innoQ Deutschland GmbH



innoQ

© 2012 innoQ Deutschland GmbH

```clojure
(defn homepage [req] ...)

(def webapp
  (-> homepage
      (wrap-resource "public")
      wrap-file-info))

(run-jetty webapp {:port 8080})
```

```clojure
(wrap-file-info
 (wrap-resource
  homepage
  "public"))
```

© 2012 innoQ Deutschland GmbH

```
(webapp {:uri "/pretty.css"
         :request-method :get
         :headers {}})
> {:status 200
   :headers {"Content-Length" "16"
             "Last-Modified" "Thu, 14 Jun ..."
             "Content-Type" "text/css"}
   :body #<File ...resources/public/pretty.css>}
```

innoQ

© 2012 innoQ Deutschland GmbH

wrap-resource

wrap-file

wrap-params

wrap-session

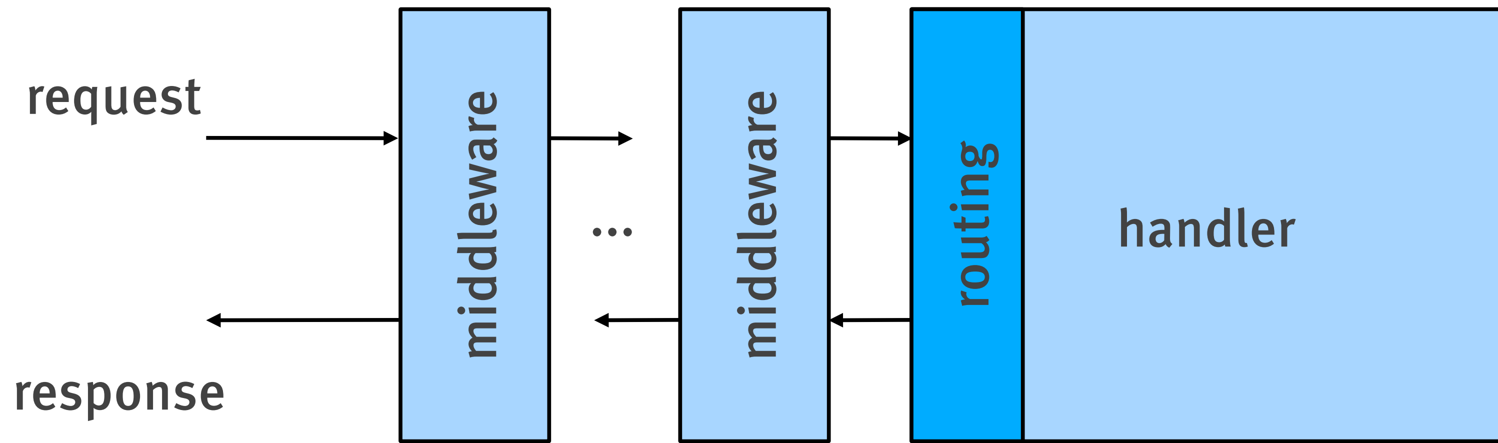
wrap-flash

wrap-etag

wrap-basic-authentication

innoQ

© 2012 innoQ Deutschland GmbH



© 2012 innoQ Deutschland GmbH

Compojure

innoQ

© 2012 innoQ Deutschland GmbH

```
(def get-handler
  (GET "/hello" []
       "Hello, World!"))


(get-handler {:request-method :get
              :uri "/hello"})
> {:body "Hello, World!" ...}


(get-handler {:request-method :post
              :uri "/hello"})
> nil
```

innoQ

© 2012 innoQ Deutschland GmbH

```
(def get-handler
  (GET "/hello/:name" [name]
       (str "Hello, " name "!")))


(def post-handler
  (POST "/names" [name]
        (remember name)
        (redirect (str "/hello/" name))))
```

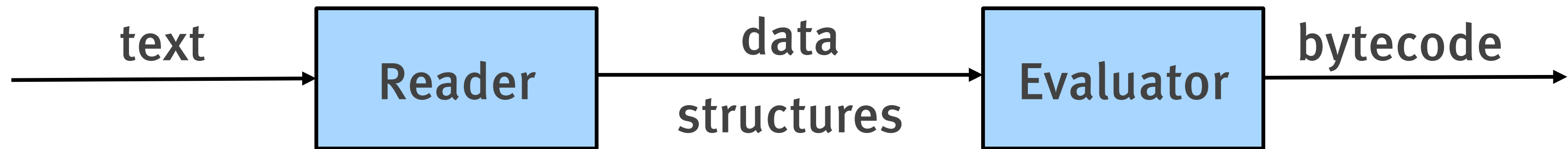
© 2012 innoQ Deutschland GmbH

Digression: Macros

innoQ

© 2012 innoQ Deutschland GmbH

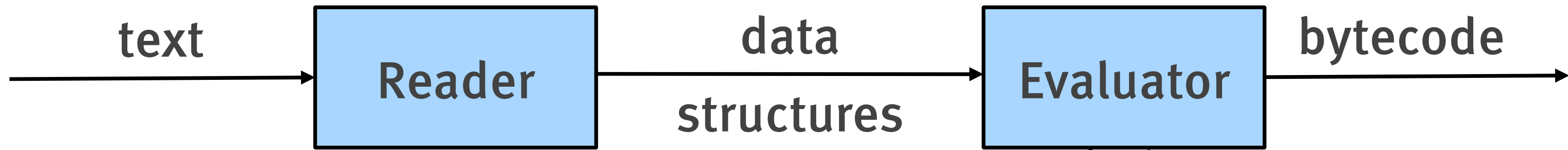
© 2012 innoQ Deutschland GmbH




```
“(if true
  (print "true"
  (print "false"))”
```

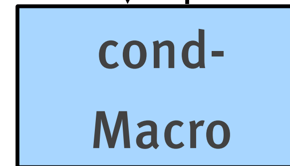
```
(if true
  (print "true"
  (print "false"))
```

© 2012 innoQ Deutschland GmbH




```
“(cond
  (< 4 3) (print "wrong")
  (> 4 3) (print "yep"))”
```

```
(cond
  (< 4 3) (print "wrong")
  (> 4 3) (print "yep"))
```




```
(if (< 4 3)
  (print "wrong")
  (if (> 4 3)
    (print "yep")
    nil))
```

innoQ

© 2012 innoQ Deutschland GmbH

```
(defmacro my-cond [c1 e1 c2 e2]
  (list 'if c1
        e1
        (list 'if c2
              e2
              nil)))


(my-cond
  false (println "won't see this")
  true (println "it works!"))
it works!
```

innoQ

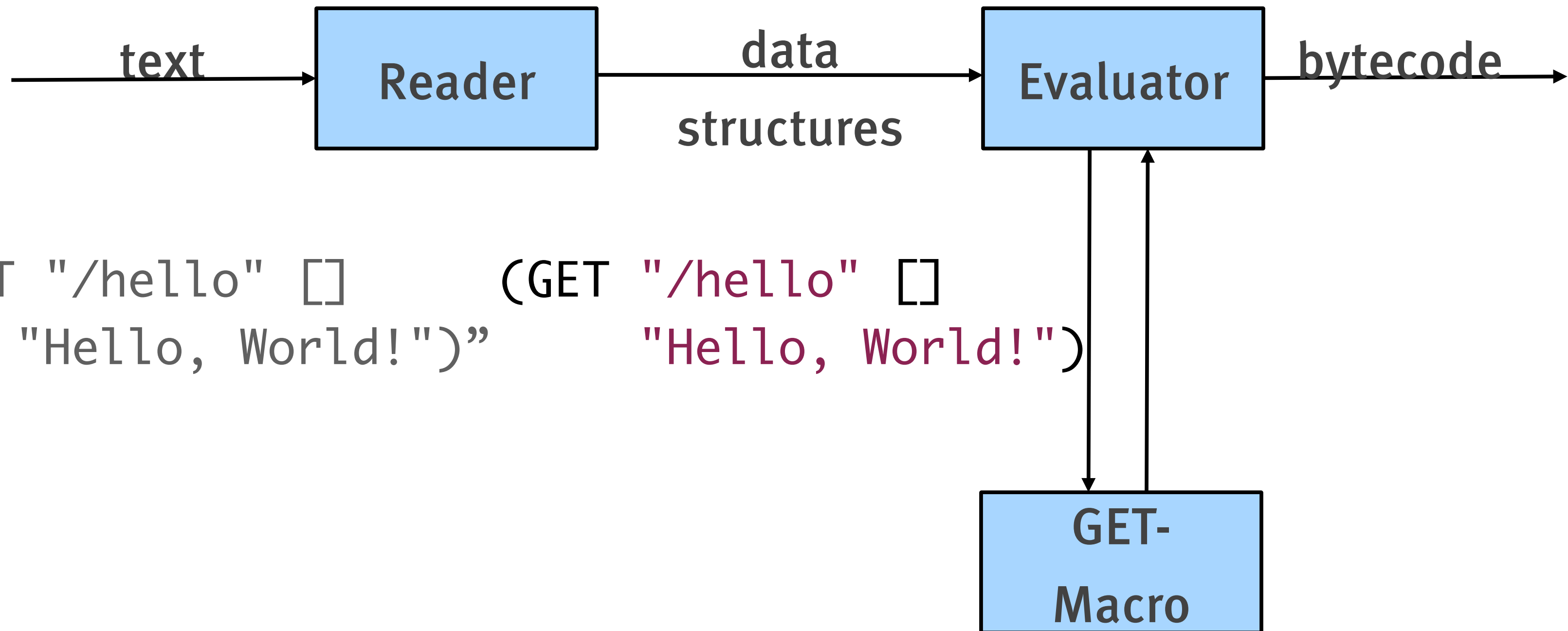
© 2012 innoQ Deutschland GmbH

```clojure
(defmacro my-cond [c1 e1 c2 e2]
  `(if ~c1
     ~e1
     (if ~c2
       ~e2
       nil)))
```

```clojure
(my-cond
  false (println "won't see this")
  true (println "it works!"))
it works!
```

innoQ

© 2012 innoQ Deutschland GmbH




```
(fn [req]
  (if (and (match (:uri req) "/hello")
           (= (:request-method req) :get))
    {:body "Hello, World!" …}
    nil))
```

innoQ

© 2012 innoQ Deutschland GmbH

Back to Compojure...

innoQ

© 2012 innoQ Deutschland GmbH

```
(def get-handler
  (GET "/hello/:name" [name]
       (str "Hello, " name "!")))



(def post-handler
  (POST "/names" [name]
        (remember name)
        (redirect (str "/hello/" name))))
```

innoQ

© 2012 innoQ Deutschland GmbH

```clojure
(defroutes todo-app
  (GET "/todos" []
       (render (load-all-todos)))
  (GET "/todos/:id" [id]
       (render (load-todo id)))
  (POST "/todos" {json-stream :body}
        (create-todo (read-json (slurp json-stream)))
        (redirect "/todos")))
```

© 2012 innoQ Deutschland GmbH

innoQ

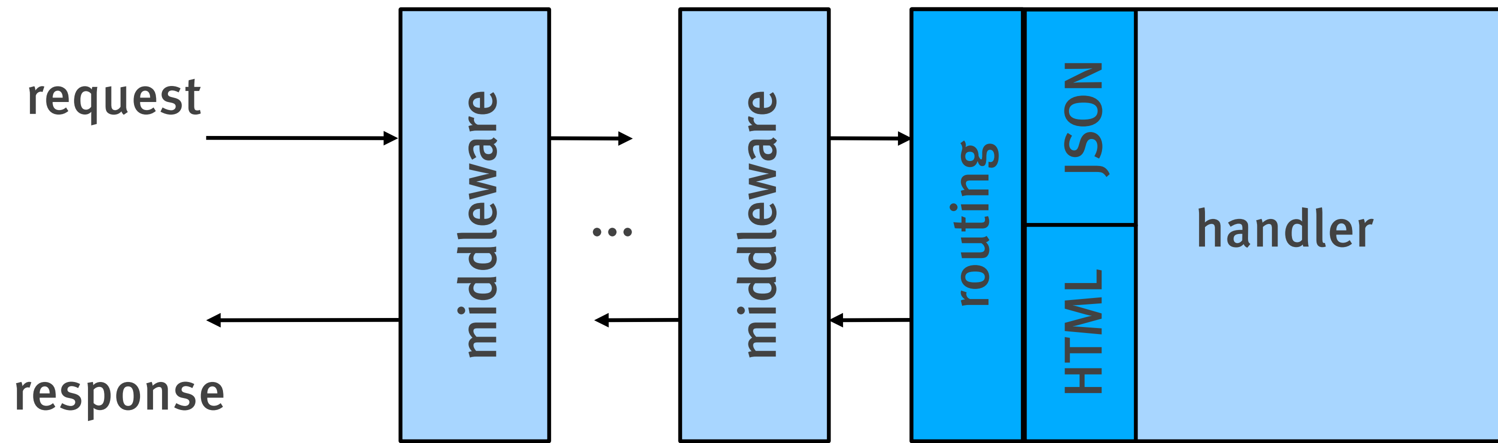
```
(defroutes more-routes
  (context "/todos/:id" [id]
    (DELETE "/" []
            (delete-todo id)
            (redirect "/todos"))
    (PUT "/" {json-stream :body}
         (update-todo (read-json (slurp json-stream)))
         (redirect (str "/todos/" id)))))
```

© 2012 innoQ Deutschland GmbH

```
(defroutes complete-app
  todo-app
  more-routes
  (not-found "Oops."))

(def secure-app
  (wrap-basic-authentication complete-app allowed?))

(run-jetty (api secure-app) {:port 8080})
```

innoQ

© 2012 innoQ Deutschland GmbH



© 2012 innoQ Deutschland GmbH

Hiccup

innoQ

© 2012 innoQ Deutschland GmbH

```
<element attribute="foo">
  <nested>bar</nested>
</element>
```

```
[:element]
```

innoQ

© 2012 innoQ Deutschland GmbH

```
<element attribute="foo">
  <nested>bar</nested>
</element>
```

```
[:element {:attribute "foo"}]
```

innoQ

© 2012 innoQ Deutschland GmbH

```
<element attribute="foo">
  <nested>bar</nested>
</element>
```

```
[:element {:attribute "foo"}
 [:nested]]
```

© 2012 innoQ Deutschland GmbH

```
<element attribute="foo">
  <nested>bar</nested>
</element>
```

```
[:element {:attribute "foo"}
 [:nested "bar"]]
```

innoQ

© 2012 innoQ Deutschland GmbH

```
<html>
  <head><title>Foo</title></head>
  <body><p>Bar</p></body>
</html>

(def hiccup-example
  [:html
   [:head [:title "Foo"]]
   [:body [:p "Bar"]]])

(html hiccup-example)
> "<html>...</html>"
```

innoQ

© 2012 innoQ Deutschland GmbH

```clojure
(def paul {:name "Paul" :age 45})

(defn render-person [person]
  [:dl
   [:dt "Name"] [:dd (:name person)]
   [:dt "Age"] [:dd (:age person)]])

(html (render-person paul))
> "<dl><dt>Name</dt><dd>Paul</dd>...</dl>"
```

© 2012 innoQ Deutschland GmbH

```clojure
(defn update [{:keys [id text author time]}]
  (list [:img.avatar {:src (avatar-uri author) :alt author}]
        [:div.content text]
        [:div.meta
         [:span.author (link-to (str "?author=" author) author)]
         [:span.time (link-to (str "/" id) time)]]))


(defn list-page [items next request]
  (common/layout
    [:div [:ul.updates (map (fn [item] [:li.post (update item)]) items)]]))
```

© 2012 innoQ Deutschland GmbH

```
(link-to "http://www.innoq.com" "click here")
> [:a {:href "http://www.innoq.com"} "click here"]
```

```
(form-to [:post "/login"]
         (text-field "Username")
         (password-field "Password")
         (submit-button "Login"))
> [:form {:action "POST" ...} [:input ...] ...]
```

innoQ

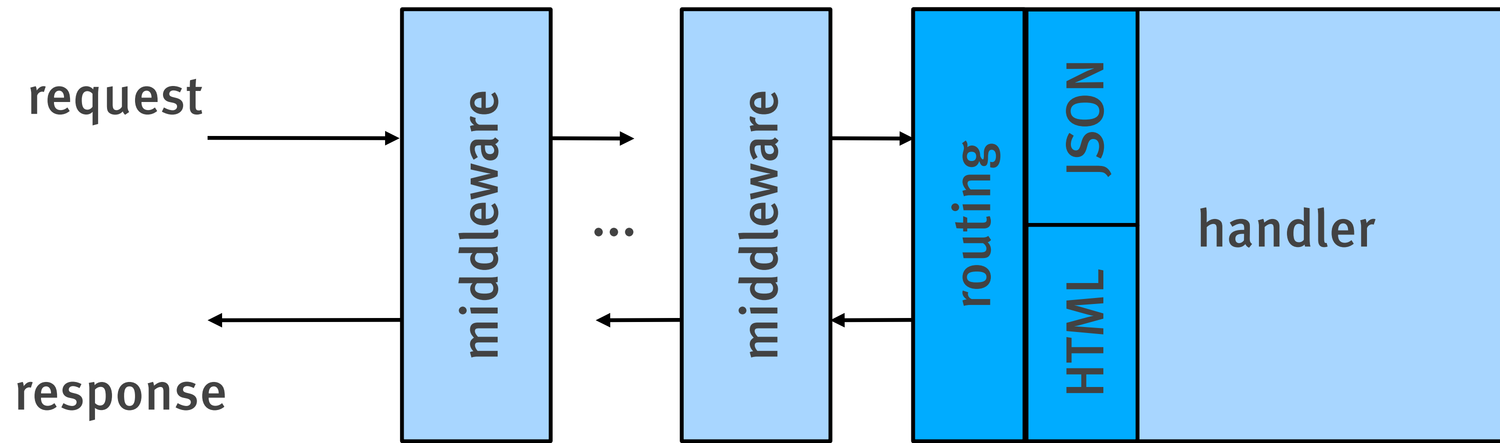
© 2012 innoQ Deutschland GmbH

```
<div id="my-id" class="class1 class2">
  foo
</div>
```

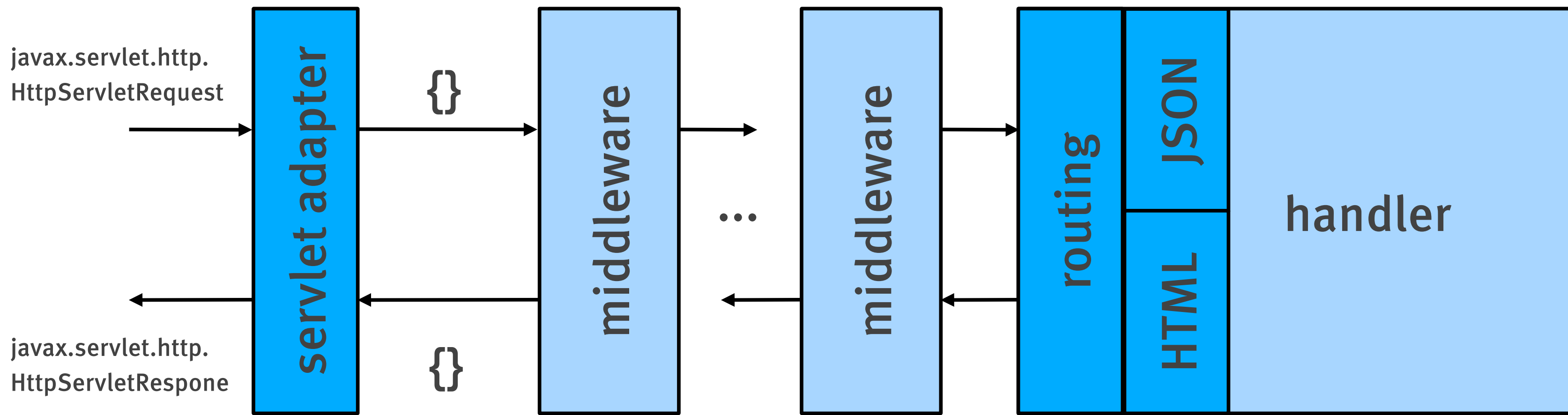
```
[:div#my-id.class1.class2 "foo"]
```

© 2012 innoQ Deutschland GmbH

innoQ



innoQ
© 2012 innoQ Deutschland GmbH



© 2012 innoQ Deutschland GmbH

Noir

© 2012 innoQ Deutschland GmbH

www.webnoir.org

- Integration of Ring/Compojure/Hiccup
- `defpage` for defining routes
- Some middleware preconfigured
  - Parameter parsing, static resources, 404 page etc.
- Helpers for form validation etc.
- Auto reload in dev mode

innoQ

© 2012 innoQ Deutschland GmbH

Demo

© 2012 innoQ Deutschland GmbH

Conclusion

© 2012 innoQ Deutschland GmbH

:-)

- Simple basic concepts
- Easy to use
- Little code (also in libraries)
- Helpful community
- Mature eco system

innoQ

© 2012 innoQ Deutschland GmbH

Thank you!

Philipp Schirmacher
philipp.schirmacher@innoq.com

Stefan Tilkov
stefan.tilkov@innoq.com
@stilkov

We'll take care of it. Personally.

© 2012 innoQ Deutschland GmbH

Backup

innoQ

© 2012 innoQ Deutschland GmbH

Task	Libraries
HTTP Basics	Ring
Routing	Compojure Moustache
HTML	Hiccup Enlive
Persistence	clojure.java.jdbc Korma Monger
Asynchronous Server	Aleph
JavaScript	ClojureScript
Micro Framework	Ringfinger Noir

© 2012 innoQ Deutschland GmbH

Lots of other cool stuff

- Persistent data structures
- Sequences
- Support for concurrent programming
- Destructuring
- List comprehensions
- Metadata
- Optiional type information
- Multimethods
- Pre & Post Conditions
- Records/Protocols
- Extensive core and contrib libraries
- ...

innoQ

© 2011 innoQ Deutschland GmbH