How we built a Shiny App used by 700 users?

Data Science team story

useR! Brussels 2017

@olga_mie
HELLO!

@olga_mie
@AppsilonDS
700 users Shiny App

PROJECT STORY

1. Shiny Application for 700 end users supporting their decision making

2. Team only Data Scientists

3. Prototype the next day, working demo after 2 weeks

4. Solution delivered for BCG
“...during the first week [of UAT], we got **overwhelmingly positive** feedback and **good results**...”

John Dannberg, The Boston Consulting Group
700 users Shiny App

CHALLENGES

UI

UX

SCALE
BEAUTIFUL UI
Client's info
Name: John Smiths
City: Warsaw, Poland
Client since: 07/2014
Our rating: ★★★★★
Monthly spendings: $2500 approx.

Calls history
Show: 10 entries
Search:

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Consultant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2016-12-08</td>
<td>Connection problem</td>
<td>Kate Lees</td>
</tr>
<tr>
<td>2 2016-12-22</td>
<td>New plan</td>
<td>You</td>
</tr>
<tr>
<td>3 2016-12-23</td>
<td>Resume services</td>
<td>Mike Bradley</td>
</tr>
<tr>
<td>4 2017-01-01</td>
<td>Suspension of service</td>
<td>Kate Lees</td>
</tr>
</tbody>
</table>

Convert to 'Pay as you go' plan
Estimated reduction
33% cost reduction based on historical data

Offer health insurance
Upselling opportunity
Clients of following profile are 2 times more likely to accept our health insurance plan

Sell credit card for spouse
Upselling opportunity
Accounts used by family members are likely to accept an additional card and increase number of transactions.

Current call
05:37
Voice analysis: Interested
Score client: ★★★★★
Churn risk - your feelings: ★★★★★

What is it?

- R package available on CRAN and GH

  ```
  install.packages('shiny.semantic')
  devtools::install_github('Appsilon/shiny.semantic')
  ```

- Package for Semantic UI components
  
  - alternative to currently available Bootstrap

- domain specific language wrapping HTML tags

What it does?

- downloads and imports Semantic UI CSS classes
- creates the abstraction enabling the user to define Shiny (text) inputs
- contains ready to use pre-defined more complex elements as R functions
USER EXPERIENCE
Filtering in the backend

**FAST DATA LOOKUPS**

---

**Fast Lookups: Indexing using** `data.table` **25x faster than** `dplyr::filter`

```
library(dplyr)

benchmark({
    key_to_lookup <- select_random_key()
    time(data %>% filter(col1 == key_to_lookup))
})
```

```
##    min    max   mean
## 0.0960 0.2440 0.1124
```

```
library(data.table)

time(setkey(data_table, col1)) ## 5.645

benchmark({
    key_to_lookup <- select_random_key()
    time(data_table[.(key_to_lookup), nomatch = 0L])
})
```

```
##     min     max    mean
## 0.00200 0.01200 0.00455
```

*data has 10 M rows.*
Fast custom search with server side API

- Server-side solution sends only matching results

Why not use shiny selectize?
- Better UI element
- Flexible functionality of searching
- Custom search algorithm
Use `shiny::registerDataObj` to create API from R.

```r
register_search <- function(session, data, search_query) {
  session$registerDataObj("search_api", data, function(data, request) {
    response <- jsonlite::toJSON(list(
      success = TRUE,
      results = search_query(data, request)
    ))
    shiny::httpResponse(200, 'application/json',
                      enc2utf8(response))
  })
}
```
Rendering hidden output

- Output hidden in modals or accordions

- Default:
  
  ```
  outputOptions(x, name, suspendWhenHidden = FALSE)
  ```

- Solution: `suspendWhenHidden = TRUE`
Render on demand

- **reactiveValue** breaks the reactivity chain if a value doesn’t update
- force the rendering of the component using **reactive trigger**
- concept introduced by Joe Cheng, the author of Shiny
- button that triggers reactivity, but programmatically

```r
make_trigger <- function() {
  simple_value <- shiny::reactiveValues(a = FALSE)
  list{
    depend = function() {
      simple_value$a
      invisible()
    },
    trigger = function() {
      simple_value$a <-
      shiny::isolate(ifelse(simple_value$a, FALSE, TRUE))
    }
  }
}
```
SCALE UP
Each Shiny Server has the same configuration, and we run the same app in each container.

We move authorization to a separate layer of our stack.

Improves the distribution of workloads across multiple servers.
Our Architecture

COMPATIBLE WITH:
- Amazon EC2
- Microsoft Azure
- On-Premise
Data Science teams can efficiently scale production ready Shiny Apps with a UI/UX focus, quickly and aesthetically.
QUESTIONS?

olga_mie

appsilondatascience.com

olga@appsilondatascience.com