Vývoj spustitelných aplikací v R pomocí balíčku `shiny`

—

Statistický workshop Mariánská (jaro 2019)

Lubomír Štěpánek[1, 2]

[1]Oddělení biomedicínské statistiky
Ústav biofyziky a informatiky
1. lékařská fakulta
Univerzita Karlova v Praze

ČVUT FBMI

[2]Katedra biomedicínské informatiky
Fakulta biomedicínského inženýrství
České vysoké učení technické v Praze

24. března 2019

Obsah

1. Úvod
2. Architektura
3. Ovládací prvky & rendering
4. Reaktivita
5. Dynamizace
6. Deployment
7. Další možnosti
8. Zdroje

Online repozitář k přednášce

githubí repozitář k přednášce ▸ GitHub

https://github.com/LStepanek/shiny_marianska_jaro_2019

Motivace

- možnost jednoduše vytvořit vlastní spustitelnou či webovou aplikaci (jen) pomocí jazyka R a balíčku `shiny`, a to i bez znalostí webařiny
- zároveň možnost aplikaci front-endově libovolně vylepšit při znalosti
  - HTML (HyperText Markup Language)
  - CSS (Cascading Style Sheets)
  - `javascriptu`
  - i dalšího

Motivace

- aplikace může s výhodou využít prakticky libovolnou funkcionalitu R, ale v back-endu i
  - TEX-u
  - SQL
  - C++
  - a mnohých dalších
- unikátní motivací je pak zpřístupnění speciálních výpočetních (třeba statistických) možností v klikací podobě uživatelům bez znalostí R

Motivace

- smyslem je
  - napsat kód v R, který by fungoval i v konzoli, resp. RStudiu,
  - doplnit prvky uživatelského rozhraní,
  - a samostatně spouštět či vystavit online
- základním frameworkem je R-kový balíček `shiny`

Instalace a inicializace balíčku shiny

```
install.packages(
    "shiny",
    dependencies = TRUE,
    repos = "http://cran.us.r-project.org"
)

library("shiny")
```

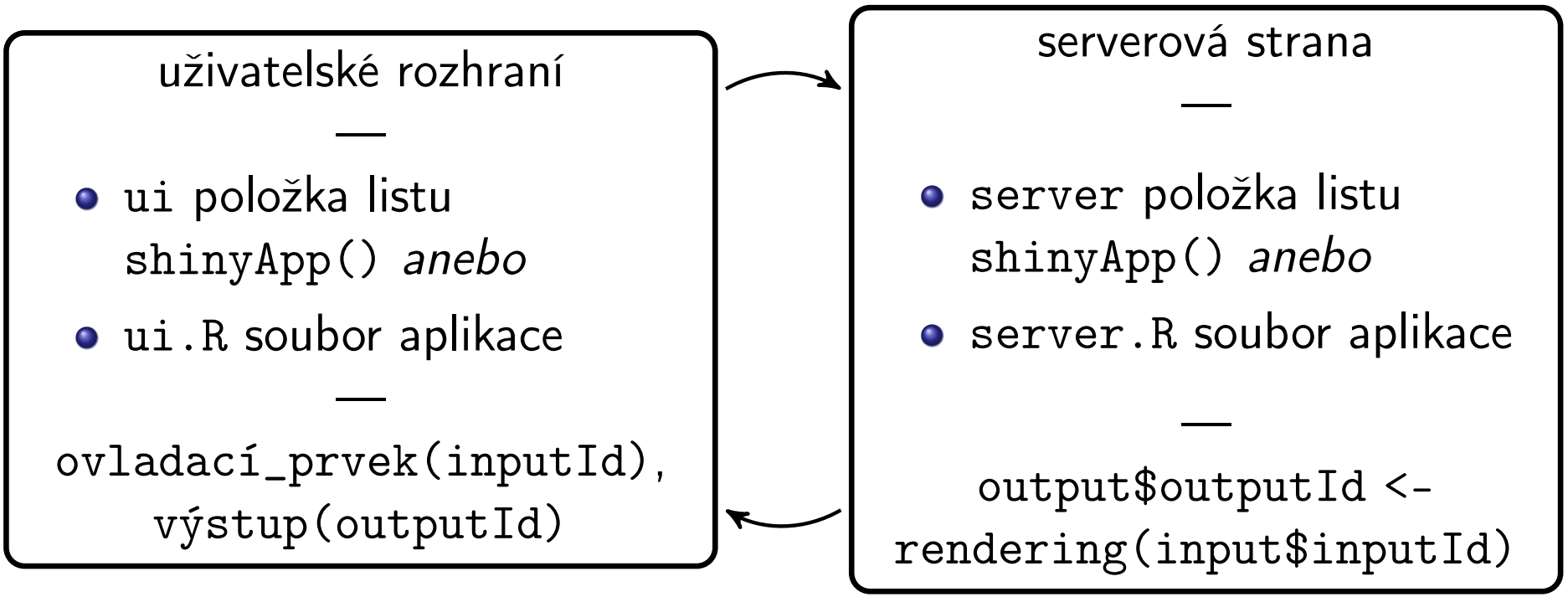
Intermezzo – hexbiny

- pravidelné šestiúhelníkové samolepky fixních definovaných rozměrů
- typické (nejen) pro R komunitu
- více na

http://hexb.in/

laptop Maxwella Ogdena

Základní architektura `shiny` aplikace



Komponenty typické aplikace

- obligátní
  - soubor `*.R` obsahující list `shinyApp()` se složkami `ui` a `R`, tedy

```
shinyApp(
    ui,
    server,
    ...
)
```

  anebo
  - dvojice souborů `ui.R` a `server.R`
- další (ale nepovinné)
  - `global.R`
  - `index.html`
  - libovolné soubory s příponou `.R`
  - složka `www`
    - obrázky, CSS třídy, `javascriptové` funkce atd.
  - cokoliv dalšího, co R, `Shiny` a web zná a „snese“

Složka listu `ui`, resp. samostatný soubor `ui.R`

- user interface (uživatelské rozhraní)
- R-kový kód určující, které prvky a jak budou uživateli zobrazeny a případně je bude moci měnit (vstupy), eventuálně jak na ně bude aplikace reagovat a co bude vracet (výstupy)
- formálně je o funkci `shinyUI()`

```
1 ?shinyUI
```

lze ji ale i vynechat

Typické ui.R

```
shinyUI(fluidPage(
    titlePanel("..."),    # název aplikace
    sidebarLayout(

        # ovládací prvky aplikace (vstupy; levý panel)
        sidebarPanel(
            ## první ovládací prvek,
            ## druhý ovládací prvek,
            ## ...
        ),

        # výstupy; pravý panel
        mainPanel(
            ## první prvek výstupu,
            ## druhý prvek výstupu,
            ## ...
        )

    )
))
```

Složka listu `server`, resp. samostatný soubor `server.R`

- skript s prakticky ryze R-kovým kódem, který obsahuje a definuje všechny aplikací používané funkce a procedury
- v podstatě jde o skript, který by měl jít po drobných úpravách spustit sám o sobě v RStudiu či R-kové konzoli
- eventualitou (a lepší) je mít procedury a funkce v separátních `.R` souborech, které bude `server.R` volat pomocí příkazu `source()` („modulárně“)
- formálně je o funkci `shinyServer()`

```
1 ?shinyServer
```

lze ji ale i vynechat

Typický server.R

```
1  library(shiny)
2
3  shinyServer(function(input, output) {
4
5    # kód první funkce/procedury využívající vstupy "z ui.R"
6    # a generující výstupy "pro ui.R"
7
8    # kód druhé funkce/procedury využívající vstupy "z ui.R"
9    # a generující výstupy "pro ui.R"
10
11   # ...
12
13 })
```

global.R

- nepovinná část aplikace
- může obsahovat
  - „globální“ nastavení,
  - konstanty,
  - error handling,
  - apod.

index.html

- nepovinná část aplikace
- obsahuje plnohodnotné HTML a CSS, které definuje vzhled uživatelského rozhraní
- lze zastoupit pomocí `ui` nebo `ui.R`

Podsložka `www`

- nepovinná část aplikace
- může obsahovat obrázky, CSS styly, javascriptí funkce

„Hello world“ vývojáře v shiny

```
1  ui <- fluidPage(                          # uživatelské rozhraní
2      titlePanel("Ahoj světe!"),
3      sidebarLayout(
4          sidebarPanel(
5              textInput(
6                  inputId = "my_text",
7                  label = "Sem vložte svůj text",
8                  placeholder = "Ahoj světe!"
9              )
10         ),
11         mainPanel(
12             textOutput(outputId = "my_text")
13         )
14     )
15 )
16 server <- function(input, output){ # serverová strana
17     output$my_text <- renderText({input$my_text})
18 }
19 shinyApp(ui, server)                        # spuštění aplikace
```

„Hello world“ vývojáře v shiny

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/hello_world

Gramatika aplikace na straně `ui` či `ui.R`

- stavebnicový systém
- kód pro úroveň vstupů (obvykle levý panel)
- `typ_ovládacího_prvku(...)`

```
    typ_ovládacího_prvku(
      inputId = "id_vstupu",
      argumenty,
      ...
    )
```

- `typVýstupu(...)`

```
    typVýstupu(
      outputId = "id_vystupu"
    )
```

Gramatika aplikace na straně `server` či `server.R`

- taktéž stavebnicový systém
- `vhodnéRenderováníVýstupu({...})`

```
output$id_vystupu <- vhodnéRenderováníVýstupu({

  # R-ková procedura či funkce beroucí jako vstupy
  input$id_vstupu

  # a vracející vhodný výstup

})
```

Slovník aplikace

- dvojice správných typů výstupu `server.R` pro vstupy `ui.R`

typVýstupu	vhodnéRenderováníVýstupu
`verbatimTextOutput(...)`	`renderPrint({...})`
`textOutput(...)`	`renderText({...})`
`tableOutput(...)`	`renderTable({...})`
`plotOutput(...)`	`renderPlot({...})`
`uiOutput(...)`	`renderUI({...})`
...	...

- kompletní seznam prvků použitelných v `ui` a `server` je zde

https://shiny.rstudio.com/reference/shiny/latest/

- renderovací proceduru k očekávanému výstupu lze najít v nápovědě

```
1 ?verbatimTextOutput
```

Textový vstup

- kód na straně `ui` či `ui.R`

```
1     textInput(
2         inputId,                  # ID vstupu
3         label,                    # statický popisek
4         value = "",               # iniciální hodnota
5         width = NULL,             # šířka prvku
6         placeholder = NULL        # vepsaný příklad vstupu
7     )
```

- kód na straně `server` či `server.R`

```
1     input$inputId                 # takto je kódován jako proměnná
```

- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/r_kalkulator

Intermezzo – `r_kalkulator`

- Pokud v kódu aplikace `r_kalkulator` změním příkaz `renderText()` na `renderPrint()`, který další příkaz musím změnit, aby aplikace vracela výsledek?

Intermezzo – `r_kalkulator`

- Pokud v kódu aplikace `r_kalkulator` změním příkaz `renderText()` na `renderPrint()`, který další příkaz musím změnit, aby aplikace vracela výsledek?
- Spočítejte pomocí kalkulátoru hodnotu výrazu $\boldsymbol{x}^T\boldsymbol{y}$, je-li $\boldsymbol{x} = (1, 3, 5, 2, 6)^T$ a $\boldsymbol{y} = (14, 31, 75, 34, 21)^T$.

Intermezzo – `r_kalkulator`

- Pokud v kódu aplikace `r_kalkulator` změním příkaz `renderText()` na `renderPrint()`, který další příkaz musím změnit, aby aplikace vracela výsledek?
- Spočítejte pomocí kalkulátoru hodnotu výrazu $\boldsymbol{x}^T\boldsymbol{y}$, je-li $\boldsymbol{x} = (1, 3, 5, 2, 6)^T$ a $\boldsymbol{y} = (14, 31, 75, 34, 21)^T$.
- Najděte součet všech lichých přirozených čísel menších než 100.

Intermezzo – `r_kalkulator`

- Pokud v kódu aplikace `r_kalkulator` změním příkaz `renderText()` na `renderPrint()`, který další příkaz musím změnit, aby aplikace vracela výsledek?
- Spočítejte pomocí kalkulátoru hodnotu výrazu $\boldsymbol{x}^T\boldsymbol{y}$, je-li $\boldsymbol{x} = (1, 3, 5, 2, 6)^T$ a $\boldsymbol{y} = (14, 31, 75, 34, 21)^T$.
- Najděte součet všech lichých přirozených čísel menších než 100.
- Je číslo 4717 prvočíslem?

Slider

- kód na straně `ui` či `ui.R`

```
    sliderInput(
        inputId,                # ID vstupu
        label,                  # statický popisek
        min,                    # minimální hodnota slideru
        max,                    # maximální hodnota slideru
        value,                  # iniciální hodnota slideru
        step = NULL,            # krok slideru
        ...                     # další argumenty
    )
```

- kód na straně `server` či `server.R`

```
    input$inputId               # takto je kódován jako proměnná
```

- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/histogram_1

Numerický vstup

- kód na straně `ui` či `ui.R`

```
1     numericInput(
2         inputId,                  # ID vstupu
3         label,                    # statický popisek
4         min,                      # minimální hodnota
5         max,                      # maximální hodnota
6         value,                    # iniciální hodnota
7         step = NULL,              # krok slideru
8         ...                       # další argumenty
9     )
```

- kód na straně `server` či `server.R`

```
1     input$inputId                 # takto je kódován jako proměnná
```

- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/histogram_2

Roletka

- uživatel může zvolit právě jednu z jejích možností (podobně jako u radiobuttonu)
- kód na straně `ui` či `ui.R`

```
    selectInput(
        inputId,                    # ID vstupu
        label,                      # statický popisek
        choices,                    # vektor možností s popisky
        selected,                   # iniciálně vybraná hodnota
        ...                         # další argumenty
    )
```

- kód na straně `server` či `server.R`

```
    input$inputId                   # takto je kódován jako proměnná
```

- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/histogram_3

Checkbox

- hraje roli typického přepínače (`TRUE` / `FALSE`)
- kód na straně `ui` či `ui.R`

```
    checkboxInput(
        inputId,                # ID vstupu
        label,                  # statický popisek
        value = FALSE,          # iniciálně vybraná hodnota
        ...                     # další argumenty
    )
```

- kód na straně `server` či `server.R`

```
    input$inputId               # takto je kódován jako proměnná
```

- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/histogram_4

Radiobutton

- uživatel může zvolit právě jednu z jeho možností
- kód na straně `ui` či `ui.R`

```
    radioButtons(
        inputId,                    # ID vstupu
        label,                      # statický popisek
        choices,                    # vektor možností s popisky
        selected,                   # iniciálně vybraná hodnota
        ...                         # další argumenty
    )
```

- kód na straně `server` či `server.R`

```
    input$inputId                   # takto je kódován jako proměnná
```

- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/histogram_5

Multiple checkbox

- uživatel může zvolit jednu až všechny jeho možnosti
- kód na straně `ui` či `ui.R`

```
checkboxInput(
    inputId,                    # ID vstupu
    label,                      # statický popisek
    choices,                    # vektor možností s popisky
    selected,                   # iniciálně vybraná hodnota
    ...                         # další argumenty
)
```

- kód na straně `server` či `server.R`

```
input$inputId                   # takto je kódován jako proměnná
```

- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/mtcars_app

Vstup typu datum

- uživatel volí v embedovaném okně jednu hodnotu typu `as.Date`
- kód na straně `ui` či `ui.R`

```
    dateInput(
        inputId,                    # ID vstupu
        label,                      # statický popisek
        value,                      # iniciální datum
        min, max                    # minimální, maximální hodnota
        format,                     # defaultně "yyyy-mm-dd"
        language,                   # jazyk ("cs" pro češtinu)
        ...                         # další argumenty
    )
```

- na straně `server` či `server.R` jako `input$inputId`
- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/calendar_app

Intermezzo – vstup typu datum

- Použijte aplikaci `calendar_app` k nalezení, kolika započatých dní se dožil Albert Einstein, jestliže žil mezi 14. březnem 1879 a 18. dubnem 1955.
  HINT: Zamyslete se, zda se nebude hodit do aplikace doplnit ještě druhý kalendářový vstup.

Externě nahrávání vlastních dat

- umožní uživateli nahrát flat-file (i jiná vlastní data)
- kód na straně `ui` či `ui.R`

```
    fileInput(
        inputId,                # ID vstupu
        label,                  # statický popisek
        multiple = FALSE,       # nahrání více souborů najednou
        accept,                 # vektor podporovaných přípon
        ...                     # další argumenty
    )
```

- na straně `server` či `server.R` jako `input$inputId`
- příklad použití v aplikaci

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/file_upload

Chování `shiny` aplikace při změně vstupů

- `shiny` aplikace se může chovat rozdílně v závislosti na okamžité změně vstupních hodnot
- defaultně se při jakékoliv změně vstupních hodnot ihned přepočítávají a aktualizují výstupy
  - to je někdy nevýhodné (a uživatelsky nepříjemné)
- reaktivitu aplikace lze však zrelaxovat ovládacími prvky či „meziukládáním“ počítaných hodnot do reaktivních objektů
  - „meziukládání“ je obecně šikovné
- našimi přáteli jsou
  - `submitButton()`, `isolate()`
  - `reactive({})`, `reactiveValues()`

submitButton()

- nejjednodušším řešením, jak aplikaci zabránit v „nechtěném“ přepočítání hodnot, je použití `submitButton()`
- k prvnímu výpočtu resp. přepočtu dojde jen a tehdy, je-li stisknuto tlačítko `submitButton()`
- jako příklad porovnejme aplikace `r_kalkulator` s ne- a zakomentovaným řádkem 28, tedy

```
1 ||      submitButton(text = "Spočítej!")
```

resp.

```
1 ||      #submitButton(text = "Spočítej!")
```

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/r_kalkulator

`isolate()`

- dalším řešením je možnost „zafixovat“ výstup tak, že je sestaven na základě vstupních hodnot před jejich změnou a k jeho přepočítání dojde až např. po nějaké akci
- hodnoty výstupu lze „zafixovat“ pomocí `isolate()`
- příkladem budiž modifikace aplikace `histogram_1`

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/isolate

`reactive({})`

- slouží k „meziuložení“ objektu v rámci `shiny` aplikace tak, že lze tento objekt opakovaně volat v libovolných částech `server` či `server.R`
- hodnoty objektu jsou po „meziuložení“ neměnné
- příkladem buď aplikace `histogram_6`

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/histogram_6

`reactiveValues()`

- slouží k „meziuložení“ objektu v rámci `shiny` aplikace tak, že lze tento objekt opakovaně volat v libovolných částech `server` či `server.R`
- hodnoty objektu lze i po „meziuložení“ opakovaně updatovat se zachováním jeho dosavadních hodnot
- příkladem buď aplikace `konvergence`

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/konvergence

Příklad komplexnější aplikace pracující s reaktivitou

- aplikace `Conwayova_hra_života`
- publikována online na

  http://shiny.statest.cz:3838/Conwayova_hra_zivota/

- a na GitHubu

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/.../Conway_Game_of_Life

Dynamický rendering aplikací v R

- smyslem je uživateli nabízet jen ty ovládací prvky nebo vlastnosti aplikace, které mají v daný okamžik smysl
- obvykle se zajišťuje kombinací funkcí `observe()` a `observeEvent({})` a logických výroků či událostí

Funkce `observeEvent({})` a `observe()`

- funkce `observeEvent({})` reaguje na stisknutí `actionButton` tlačítka
- příkladem je aplikace

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/konvergence

- funkce `observe()` vyhodnocuje kombinaci logických výroků tlačítka
- příkladem je aplikace

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/file_upload_2

Desktopové spuštění aplikace v RStudiu či R konzoli

- příkaz `runApp(launch.browser = TRUE)`
- příkaz `shinyApp(ui, server)`
- zelený trojúhelníček vpravo nahoře („Run app“)

Desktopové spuštění aplikace bez RStudia (a bez R konzole)

- spouštěcí (R-kový) soubor je na

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/.../spust_aplikaci

- vhodné dodržet architekturu `ui.R` a `server.R` (nikoliv jen jediný soubor `*.R` s aplikací)
- nutné spárovat příponu `.myRscript` s programem, který bude aplikaci spouštět, tedy `spust_aplikaci.myRscript`
- před prvním spuštěním se po poklepání na tento soubor otevře dialog pro nastavení výchozího spouštěcího programu; v lokální nabídce přes *Další možnosti* a *Najít jinou aplikaci v tomto počítači* vyberme nástroj `Rscript` ve složce *bin* složky R
- obvyklá cesta k nástroji vypadá například takto

C:/Program Files/R/R-3.3.0/bin/Rscript

- pokud postup nezafunguje, je nutná editace registrů

Online vypublikování aplikace

- aplikaci lze vystavit zdarma online na web

https://www.shinyapps.io/

- nutná registrace, zdarma může běžet maximálně pět aplikací maximálně 25 hodin měsíčně
- link aplikace je obvykle neatraktivní, obsahuje doménu `shinyapps.io`

Vlastní R-kový server

- možné nainstalovat na vlastní web pomocí návodu

https://www.rstudio.com/products/rstudio/download-server/

- je třeba se orientovat v Linuxu
- je vhodné používat běžnou linuxovou distribuci (např. Ubuntu, Fedora, „Debian“)
- nutné mít možnost root SSH přístupu k serveru, což běžný provider neumožní
- anebo je možné mít virtuální server na Amazon Web Services (AWS), více zde

https://aws.amazon.com/blogs/big-data/running-r-on-aws/

Příklad aplikace s pokročilejším formátováním a funkcionalitami

- aplikace `adventni_kalendar_2018`
- na aplikaci si ukážeme další možnosti formátování a vlastností
- publikována online na

  http://shiny.statest.cz:3838/adventni_kalendar_2018/

- a na GitHubu

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/.../adventni_kalendar_2018

Počítadlo návštěv

- založeno na permanentním souboru `counter.RData` obsahující dosavadní počet návštěv (jako `integer`)
- kód v `server.R` pak vypadá

```
1   output$counter <- renderText({
2       if(!file.exists("counter.Rdata")){
3           counter <- 0
4       }else{
5           load(file = "counter.Rdata")
6       }
7
8       counter <- counter + 1
9
10      save(counter, file = "counter.Rdata")
11      paste("Hits:", counter, sep = "")
12
13  })
```

Javascript v shiny aplikaci

- javascript zajišťuje uživatelsky komfortní prostředí aplikace, je překládán přímo prohlížečem
- lze vložit vlastní javascriptí funkcionalitu pomocí kódu do ui.R

```
    tags$head(
        tags$script(
            type = "text/javascript",
            src = "*.js"
        )
    )
```

- eventuálně lze využít více „R“ přístup a zavolat shinyjs balíček pomocí

```
    library("shinyjs")
```

a inicializovat ho

```
    shinyjs::useShinyjs()
```

Kaskádové styly (CSS) v shiny aplikaci

- kaskádové styly (CSS) umožňují pokročilejší formátování aplikace, než na které stačí HTML
- lze je vložit pomocí kódu do `ui.R`

```
    tags$head(
        tags$link(rel = "stylesheet",
                  type = "text/css",
                  href = "style.css")
    )
```

Busy indikátor

- založen na třídě `style.css`, javascriptové funkci `busy.js` (obě by měly být ve složce `www`) a GIFu `free_busy_indicator.gif`
- kód v `ui.R` pak vypadá

```
1     tags$head(
2         tags$link(rel = "stylesheet",
3                   type = "text/css",
4                   href = "style.css"),
5         tags$script(type = "text/javascript",
6                   src = "busy.js")
7
8     ),
9     div(class = "busy",
10        p("Application is busy..."),
11        img(src = "free_busy_indicator.gif", height =
              50, width = 50)
12    )
```

data.table

- jQuery plug-in pro hezčí a funkční formátování výstupních tabulek v `shiny` aplikaci
- na straně `ui.R` je namísto `tableOutput()` třeba použít `dataTableOutput()`
- na straně `server.R` je namísto `renderTable({})` třeba použít `renderDataTable({})`

▸ GitHub

github.com/LStepanek/shiny_marianska_jaro_2019/tree/master/mtcars_app_2

Další možnosti

- renderování TEX-ové matematické notace v `shiny` (pomocí MathJaX)
- formátování (header, footer)
- pop-up vyskakovací hlášky, tooltips
- Google Analytics

Kde získat templáty aditivních funkcionalit?

- sledovat přední vývojáře R-kových aplikací (jmenovitě Dean Attali)
- sledovat jejich githubí účty a streamy z useR! konferencí

Příklady existujících aplikací

- aplikace `statisticke_natroje`

  http://shiny.statest.cz:3838/statisticke_nastroje/

- aplikace `Conwayova_hra_zivota`

  http://shiny.statest.cz:3838/Conwayova_hra_zivota/

- aplikace `ShinyItemAnalysis` (Patrícia Martinková et al.)

  www.shinyitemanalysis.org

Literatura

Chris Beeley. *Web Application Development with R using Shiny*. City: Packt Publishing, 2013. ISBN: 1783284471.

Hadley Wickham. *Advanced R*. Boca Raton, FL: CRC Press, 2015. ISBN: 978-1466586963.

Karel Zvára. *Základy statistiky v prostředí R*. Praha, Česká republika: Karolinum, 2013. ISBN: 978-80-246-2245-3.

Děkuji za pozornost!

lubomir.stepanek@lf1.cuni.cz
lubomir.stepanek@fbmi.cvut.cz

▸ GitHub

https://github.com/LStepanek/shiny_marianska_jaro_2019