shiny

shiny는 R로 웹 어플리케이션을 만들 수 있게 해주는 프레임워크이다.

shiny는 ui라고 불리는 화면, server라고 불리는 데이터 처리 및 관리, 그 안에 입출력과 render와 배포로 구성되어 있다.

[설치]

패키지를 설치하는 방법은 다음과 같다.

```
# cran에서 설치
install.packages("shiny")
```

[shiny 앱 예시]

shiny는 개발이 어려운 만큼 생태계 활성화를 위해 다양한 예시를 쉽게 접할 수 있게 준비되어 있다.

```
library(shiny)
runExample()
```

[예시 실행]

runExample() 함수로 리스트 내의 예시들을 실행해 볼 수 있다.

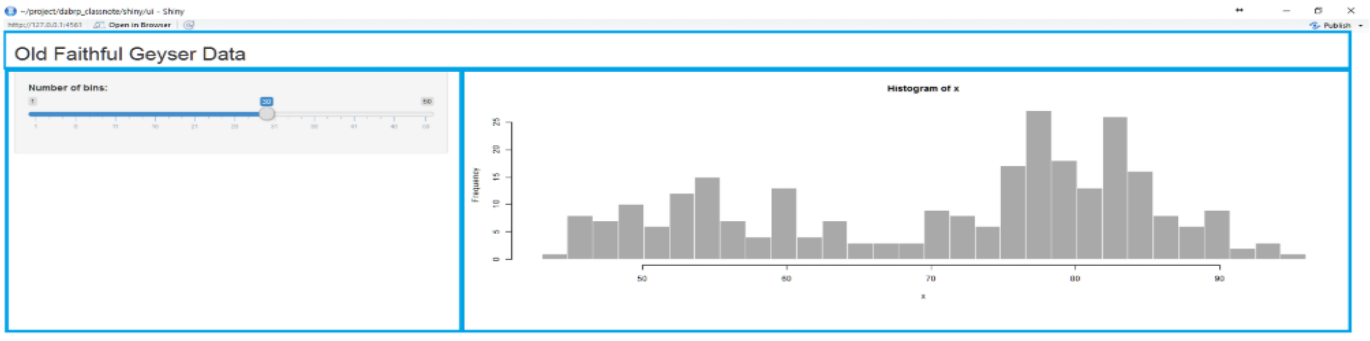
```
runExample("01_hello")
```

[shiny의 화면]

ui라고 말하는 화면은 실제로 사용자가 보는 화면을 뜻한다.

shiny에서는 크게 titlePanel과 sidebarPanel, mainPanal의 세 가지로 구성되어 있다.

이렇게 지정함으로써 html에 대해 잘 몰라도 사용할 수 있게 shiny가 구성되어 있다.



[shiny의 서버]

shiny에서의 server는 실제 서버가 브라우저와 통신하는 과정 전체를 간단하게 만들어주는 역할을 한다.

화면에서 사용자가 동작하는 것에 대해서 받을 수 있는 input을 서버에서 데이터나 그림 조작에 사용을 하고 웹 기술이 이해할 수 있게 render하여 output으로 화면에 다시 보내주는 형태로 shiny가 동작한다.

[shiny의 입출력]

shiny는 다른 R 패키지와는 다르게 강제하는 변수가 3개 있다. 그것은 input, output, session 이다.
각각 객체로써 존재하고 이번에는 input과 output으로 데이터를 ui와 server가 교환하는 방법을 소개한다.
각각의 객체는 *Input 함수와 *Output 함수로 데이터를 입력 받아 저장한다. *Output 함수는 render* 함수로 선언된다.

[shiny의 배포]

shiny는 shiny-server를 통해서 동작한다. shiny-server는 shiny appoi 동작할 수 있는 서버를 의미한다.
오픈소스와 기업용 솔루션이 모두 준비되어 있으며 실습에는 http://www.shinyapps.io/를 이용해 보겠다.

[shiny의 최소 코드]

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){ }
shinyApp(ui=ui,server=server)
```

[ui 만들기]

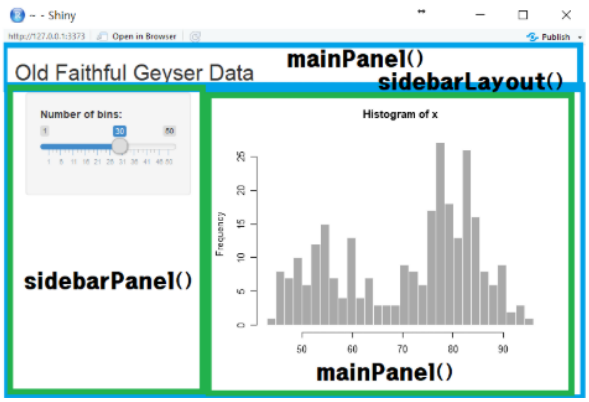
ui는 화면의 위치를 나누는 함수와 입력을 받는 요소, 출력을 보여주는 요소로 구성되어 있다.
*는 여러 이름이 있다는 뜻이다.

```
# *Page(), *Panel() 등의 함수로 위치를 나누고 모양을 결정
ui <- fluidPage(
  # *Input() 함수로 입력을 받는 요소들을 배치
  # *Output() 함수로 결과물을 출력하여 배치
)
```

[*Panel() 사용]

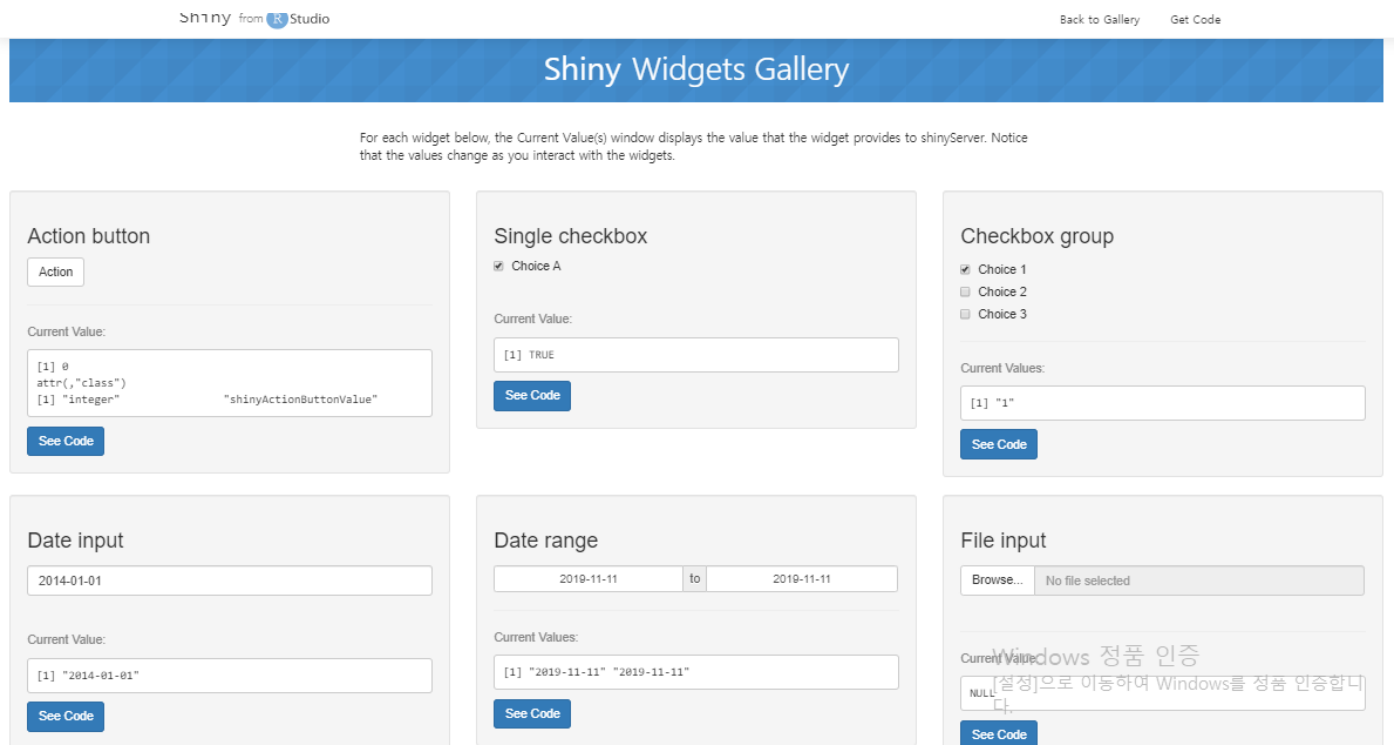
*Panel() 함수는 기본적으로 3가지(titlePanel(), sidebarPanel(), mainPanel())를 제공하며 구조는 아래와 같다.

```
fluidPage(
  titlePanel(),
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  )
)
```



[*Input() 함수]

모양을 이쁘게 잡았다면, 이제 입력을 받을 도구들과 출력물을 배치할 차례이다. 입력을 받을 도구는 여기 준비되어 있다.



*Input()은 sidebarPanel()에 위치

```
fluidPage(
  titlePanel("test"),
  sidebarLayout(
    sidebarPanel(
      sliderInput(
          inputId = "slider1"
        , label = "Slider"
        , min = 0
        , max = 100
        , value = 50
        )
    ),
    mainPanel("mainPanel Area")
  )
)
```

여러개는 순차적으로 아래로 배치

```
fluidPage(
  titlePanel("test"),
  sidebarLayout(
```

```
    sidebarPanel(
      sliderInput(
           inputId = "slider1"
        , label = "Slider"
        , min = 0
        , max = 100
        , value = 50
        ),
      dateInput(
           inputId = "date"
        , label = "Date input"
        , value = "2014-01-01"
        ),
      fileInput(
           inputId = "file"
        , label = "File input"
        )
    ),
    mainPanel("mainPanel Area")
```

[*outPut() 함수]

input에서 입력값을 받아 서버가 처리한 결과물을 보여주는 곳으로 아래와 같은 함수들과 추가 패키지들이 제공하는 *Output() 함수가 있다.

Function	Inserts
dataTableOutput()	인터렉티브 테이블
htmlOutput()	HTML 문서
imageOUtput()	그림
plotOutput()	차트
tableOutput()	테이블
textOutput()	글자
verbatimTextOutput()	코드 글자

*Output()은 mainPanel()에 위치

```
fluidPage(
  titlePanel("test"),
  sidebarLayout(
    sidebarPanel(
```

```
      sliderInput(
          inputId = "slider1"
        , label = "Slider"
        , min = 0
        , max = 100
        , value = 50
      )
    ),
    mainPanel(
      plotOutput(
        outputId = "plot1"
      )
    )
  )
)
```

[각 변수의 Id]

Input() 함수와 Output() 함수는 사용할 변수에 대한 Id를 입력받습니다. 전부 글자형(character)로 구성된다.
inputId는 서버에서 input$inputId의 형태로 사용하며 아래의 경우 input$slider1 이다.
outputId는 서버에서 만든 결과물을 전달 받기 위해서 사용하며 서버에서 저장할 때는 output$outputId로 사용한다. 아래의 경우 output$plot1 이다.

```
# inputId
sliderInput(inputId = "slider1", label = "Slider", min = 0, max = 100, value = 50)
# outputId
plotOutput(outputId = "plot1")
```

[server 만들기]

server는 아까 Input() 함수로 입력받은 데이터를 사용하기 위한 input 변수와 Output() 함수로 출력하기 위한 결과물을 저장하는 output 변수, 마지막으로 출력 결과물인 R객체를 web의 세상에서 사용할 수 있는 상태로 output 변수에 저장하는 render*({}) 함수로 구성된다.

```
server <- function(input, output){
  output$plot1 <- renderPlot({
    plot(input$slider1)
  })
}
```

[shiny의 입출력]

shiny는 웹 기술을 R로 사용할 수 있게 만드는 덕분에 render라는 과정이 필요하다.
그래서 render 환경에서 변수들이 관리되어야 한다. 입출력은 input 변수와 output 변수로 관리한다. 이 두 가지는

render 밖에서 관리되는 변수로 다르게 취급해야 한다.

input : 웹 페이지의 입력을 통해서 들어오는 데이터를 사용하기 위한 변수
output: 웹 페이지에 R 연산 결과물을 전달하기 위해 사용하는 변수

[output$]
output 변수는 list라고 이해하시면 좋을 것 같다. list인 output 변수는 output$뒤에 변수명을 작성함으로써 output 객체에 필요한 결과물을 전달한다. 만약에 화면에 보여줘야할 것의 이름을 sample이라고 하면 output$sample에 필요한 내용을 선언하는 것으로 진행한다.

```
...
output$sample <- randerPlot({ ...plot()... })
...
```

위 코드는 server쪽 코드에서 작성한다. 위 예시는 plot함수로 만들어지는 이미지를 output$sample에 저장하는 것을 뜻한다. 그럼 ui쪽에서 이걸 어디에다 위치하게 하는지를 결정하는 함수에서 사용할 수 있다. 이때는 plotOutput 함수를 사용한다.

plotOutput

```
...
plotOutput("sample")
...
```

함수명이 Output 이고, "로 변수명을 감싸며, output$ 문법을 사용하지 않고 컬럼명인 sample만 사용한다는 점을 주목해 주세요. server쪽 코드에서 output$에 컬럼명의 형태로 저장한 R의 결과물을 ui쪽 코드에서 *Output 함수에서 "로 감싼 글자의 형태로 컬럼명만 작성해서 사용한다.

- render*({})

```
...
output$sample <- randerPlot({
  plot(faithful)
})
...
```

render*({ }) 함수 안에는 plot 함수가 R 문법으로 작성되어 있다.
그리고 render*({ })의 특이한 점은 () 안에 {}가 또 있다는 점이다.
여기서는 ({ }) 안쪽은 R의 세계이고, 그 바깥은 웹의 세계라고 이해한다.

```
...
output$sample <- randerPlot({
  data <- faithful[faithful$eruptions >3, ]
```

```
  plot(data)
})
```

...

R의 세계 내에서 처리하는 것은 계속 사용할 수 있어서 data 변수에 선언한 내용을 plot함수가 사용할 수 있다. 여기서 input$이 활용될 때 반응형으로 작성할 수 있는 것이다.

- render*() 함수

output() 함수에 대응되는 render() 함수가 모두 있다.

Output.func	render.func	Inserts
dataTableOutput()	renderDataTable()	인터렉티브 테이블
imageOUtput()	renderImage()	그림
plotOutput()	renderPlot()	차트
tableOutput()	renderTable()	테이블
textOutput()	renderText()	글자
verbatimTextOutput()	renderText()	코드 글자

[input$]

intput$은 output$과 같이 웹의 세계에 있는 변수이다. 그래서 *Input() 함수들을 통해서 input$의 컬럼 이름으로 웹 페이지 내에서 얻을 수 있는 데이터를 R의 세계에서 사용할 수 있게 해줍니다. 우선 input$에 웹의 세계의 데이터를 R의 세계로 가져와 보겠다.

sliderInput("sdata1", "슬라이더 입력:", min=50, max=150, value=100)

위의 코드는 sliderInput()이라는 *Input() 패밀리 함수를 통해서 input$sdata1이라는 곳에 웹 데이터를 저장하는 것을 뜻한다. *Input() 패밀리 함수는 같은 규칙을 가지는데 입력형태명Input()의 함수명을 가지고, 첫 번째 인자는 input$ 뒤로 붙을 컬럼명, 두번째 인자는 화면에 보여줄 글자를 의미한다. 나머지 인자는 입력형태에 따라 다양하게 달라진다.