Webbutveckling i CMS

Föreläsning 2

NORTH LINK

Angående val av ramverk/bibliotek

- I Avancerad JavaScript använde vi React för att skapa webbapplikationer
- React är bara ett av flera populära frontendbibliotek
- I den här kursen kommer vi skriva en del applikationer och till skillnad från i Avancerad JavaScript är det inget krav att skriva dem i React
- Jag vet att vissa av er kommer att använda andra bibliotek på er LIA och vill kanske öva på använda dem
- På labben och inlämningsuppgiften måste ni använda React, Angular eller Vue. Om ni verkligen vill använda något annat så meddela gärna det i förväg.

Exempelkod

- Precis som i Avancerad JavaScript kommer jag att lägga upp exempel på GitHub

https://github.com/argelius/ec-cms-samples

Mer om Docker

- Vi började kursen med att försöka få igång Docker på allas datorer
- Docker gör det enklare för oss att arbeta tillsammans eftersom det fungerar likadant på alla datorer
- Det är inte så att Docker är nödvändigt för att köra programmen. Exempelvis skulle vi kunna installera Wordpress, Apache och MySQL direkt på våra datorer men ett problem är att förfarandet är olika beroende på vilket operativsystem vi kör

Fler Docker-kommandon

- Under ytan är en Docker-container en Linux-dator. Det innebär t ex att vi skulle kunna köra bash och kolla runt i en container. Detta är väldigt bra för debugging.
- För att köra bash på en container kan vi använda följande kommando

```
docker exec -it CONTAINER_ID bash
```

- Om ni testar detta på en container som är igång kommer ni få upp en terminal

Kopiera filer från Docker

- När vi startar en Docker-container kommer den automatiskt få lite utrymme som den kan skriva filer till.
- Dessa filer kommer att försvinna om containern tas bort. Om vi vill kopiera filer från containern kan vi använda följande kommando

```
docker cp CONTAINER_ID:/path/to/file/on/container
/path/to/destination
```

Starta en stoppad container

- Vi har lärt oss att det går att stoppa en container m.h.a. “docker stop CONTAINER_ID”
- På samma vis kan vi starta en stoppade container med “docker start CONTAINER_ID”

Rensa upp efter Docker

- Docker har en tendens att ta upp väldigt mycket plats på disk när man har använt det ett tag
- Det är bland annat alla containers man har startat och images som laddats ner som tar mycket plats
- För att ta bort alla images och containers som inte används kan man köra följande kommando

```
docker system prune
```

Det finns även lite andra saker i Docker som tar upp mycket plats. Här finns information om hur man tar bort det

https://stackoverflow.com/questions/34658836/docker-is-in-volume-in-use-but-there-arent-any-docker-containers?answertab=votes#tab-top

Cockpit

- Idag ska vi titta djupare på Cockpit och se vad vi kan göra med det
- Cockpit är väldigt flexibelt och vi kan använda det för att skapa många olika sorters sidor
- En tumregel är att om man har mycket innehåll som följer en viss struktur kan ett CMS-verktyg vara ett bra sätt att hantera innehållet
- Några exempel på vad vi skulle kunna använda en CMS till
    - En sida med recept
    - En webbshop med olika varor
    - Ett uppslagsverk om fåglar

Collections i Cockpit

- För att kunna lägga in innehåll i Cockpit använder vi “collections”
- En collection definieras som en lista av fält med olika typer
- När vi har definierat en collection kan vi börja lägga till innehåll och vi gör det genom ett formulär som anpassas efter de fält och fälttyper vi har valt

Typer i Cockpit

- Det finns många olika typer i Cockpit. Några av de vanligare är
  - Text
  - Textarea
  - Image
  - Date
  - Time
  - Select
  - Boolean
- Vissa av typerna går att använda direkt men det finns fall där de behöver lite extra konfiguration som ni kan kolla upp i dokumentationen.

https://github.com/agentejo/cockpit/wiki/Collection-Field-Types

Starta Cockpit med Docker

- Cockpit behöver ingen extern databas så det är väldigt enkelt att starta det med Docker
- Det går att använda Cockpit med en extern databas men om vi inte definierar någon databas kommer Cockpit automatiskt skapa en databas med SQLite
- Ni kan starta Cockpit med följande kommando

```
docker run -d --name cockpit -p 8080:80 agentejo/cockpit
```

Övning - Svampgalleri

Svampsäsongen har börjat. Er uppgift är att skapa ett galleri över svampar i Sverige. Ni kan hitta innehåll på följande sida

http://svampguiden.com/matsvampar/lista/
http://svampguiden.com/giftsvampar/lista/

1. Starta en ny container för Cockpit (ge den t ex namnet “svamp”)
2. Lägg till en ny collection för svampar. Man ska kunna lägga in följande information om svampen.
    a. Namn
    b. Vetenskapligt (latinskt) namn
    c. En kort beskrivning
    d. Smak (använd gärna “Rating”-typen)
    e. En bild
    f. Om den är giftig eller ej (använd gärna “Boolean”-typen)
3. Lägg gärna in ett gäng svampar i din collection så att vi har data att använda
4. Skapa en applikation som hämtar information om svamparna och visar dem i ett galleri

Mer om Cockpits API

- Cockpits API är väldigt kraftfullt och har funktionalitet för att bl.a. filtrera och sortering
- Det finns även stöd för pagination genom att ange “limit” och “skip”
- Tyvärr är det vissa saker som saknas i dokumentationen så ibland får man gräva lite
- En odokumenterad funktion är att man kan filtrera med ett RegEx om man har följande i sin query string

```
&filter[name][$regex]=sopp
```

Övning - Svampgalleri fortsättning

1. Utgå från applikationen vi skapade i förra uppgiften. Nu ska vi använda API:et för att lägga till sortering och filtrering
2. Lägg till en checkbox för som gör att bara ätliga (ej giftiga) svampar visas om den är ikryssad
3. Lägg till en dropdown där man kan välja om man vill sortera efter “namn”, “vetenskapligt” namn eller “smak”
4. Lägg till en textruta så att man kan söka på namn
5. Om du har tid över, lägg gärna till pagination och visa 10 svampar per sida.

Länkar mellan (och inom) collections

- Ofta vill man skapa länkar mellan innehåll i sin CMS
- Till exempel
  - Länkar till liknande artiklar i en blogg eller på en nyhetssida
  - Länk till artikelförfattaren
- I Cockpit kan vi använda typen “Collectionlink” för att skapa länkar

https://github.com/agentejo/cockpit/wiki/Collection-Field-Types#collectionlink

Övning - Förväxlingssvampar

Vid svampplockning är det viktigt att känna till vanliga förväxlingssvampar (svampar som är väldigt lika) till de svampar man försöker plocka.

1. Utgå från svampapplikationen ni redan skapat
2. Lägg till ett nytt fält i er collection för förväxlingssvampar. Kolla upp i dokumentationen hur ni bör konfigurera det.
3. Lägg till funktionalitet i er applikation för att lista förväxlingssvampar

Laboration

- Laborationen finns tillgänglig i PingPong
- I labben ska ni skapa en blogg med Cockpit som backend
- Labben ska lämnas in fredag 6:e september

Inlämning

- Eftersom ni själva bygger upp en struktur i Cockpit kan det bli lite svårt för mig att testa era applikationer om jag inte har tillgång till er databas
- Därför är det viktigt att ni utöver frontendkoden även skickar med er databas
- Fråga gärna om ni är osäkra på vad som behöver skickas med

Struktur

- Ni ska skapa en bloggapplikation som har minst tre olika sidor
  - En huvudsida som visar en lista av artiklar. Varje artikel i listan ska innehålla
    - titel
    - namn på författare
    - datum
  - En sida som visar en artikel. Den här sidan ska innehålla
    - title
    - namn på författare
    - datum
    - artikelns text renderad som HTML (mer om detta senare)
  - En sida som listar alla författare. Varje författare ska ha
    - ett namn
    - en bild
    - en beskrivning

Collections i Cockpit

- För att kunna åstadkomma detta behöver ni två collections i Cockpit: artiklar och författare
- En artikel ska ha följande fält och typer
  - title - Text
  - body - Markdown
  - published_on - Date
  - author - Collectionlink
- En författare ska ha föjande fält och typer
  - name - Text
  - description - Textarea
  - avatar - Image

Gör gärna mer!

- Fälten som listas i föregående slide är minimumkrav. Ni får självklart lägga till fler fält och mer funktionalitet!
- Det finns inga krav och inte heller några begränsningar på layout och design så lägg gärna tid på att formge sidan om ni har lite tid över.

Köra Cockpit med Docker

Använd helst följande kommando för att starta Cockpit

```
docker run -d --name cockpit -p 8080:80 agentejo/cockpit:0.9.2
```

Detta är för att undvika att vi kör med olika versioner av Cockpit. Detta är den senaste versionen av Cockpit som släppts.

När ni är färdiga kan ni kopiera “storage”-katalogen från er Docker-container, zippa den och inkludera den i repot. Detta underlättar när jag rättar uppgiften så att jag inte behöver bygga upp samma struktur manuellt.

Med följande kommando kan ni kopiera katalogen till er arbetskatalog.

```
docker cp cockpit:/var/www/html/storage .
```

Vad är Markdown?

- Ett av kraven är att artikelns innehåll ska skrivas i Markdown
- Markdown är ett markup-språk (ungefär som HTML) för att formatera text
- Det finns inbyggt stöd i Cockpit för att skriva Markdown om man väljer “Markdown” som typ när man lägger till ett nytt fält
- Ni kommer behöva översätta Markdown-texten till HTML i er applikation

Krav

- Frontend-applikationen ska vara en SPA som använder React, Vue eller Angular
- Den ska implementera korrekt routing
- Den ska innehålla minst tre olika sidor: en huvudsida, en sida för varje bloggpost och en sida som listar författare
- Den ska använda Cockpit som backend och hämta data med Cockpits API
- Artiklarna ska skrivas i Markdown och översättas till HTML av klienten
- Ni ska använda “Collectionlink”-typen för att länka författare till artiklar

Tips

- Börja med att göra er bekväma med Cockpit och dess API
- Försök skriva mer simpla applikationer med Cockpit först
- Fråga mig eller varandra om ni är osäkra på något
- Börja i tid!

Deployment

- Om ni har tid över får ni gärna försöka gärna publicera applikationen
- Det går till exempel att köra Docker-containers på Amazon EC2 och det är gratis att köra om man bara har en instans

https://docs.docker.com/machine/examples/aws/