Transformations and the MVC pattern

Pittsburgh-NEH-institute 2022
day 1 session 2 slot 3

How do I transform, what do I transform to?

- Think of this from the perspective of your own project needs

MVC pattern?

- MVC is an architectural pattern. Cf design pattern.
  - So just like any pattern it will be loose in the ends

What?

- MVC pattern divides the application into three logical parts: Model, View, and Controller. hence its acronym name.
- Usually used to describe graphical user interface, nowadays also used for mobile and web apps.
  –

What does the MVC pattern try to solve?

- Let the user control a large and complex data set
- The three parts have their own specific responsibilities
  - This avoids repetition
  - Helps with creating solid structures

Part by part: The model

- The model
  - Maintains the data. E.g. connects with database
  - Responds to model requests (for data)

Part by part: The view

- The view
  - Represents data
  - Creates the user interface

Part by part: The controller

- The controller
  - Tells the model what to do based on the requests from the view
  - Does not handle data logic

Positive effects

- When business logic is separated form the user interface
  - Components are more easily reusable
  - Components can be made, deployed, maintained and tested independently
    - -> TDD, semantic URLs

Negative effects

- Not suitable for small applications (still good for planning)
- High complexity
- Inefficiency of data access in view

Planning tool

- Thinking about programming and organize your files
  - -> translate your ideas into code
- Interaction with other code gets easier
- Returning to the code gets easier as well
- We will certainly return to this later (as some of the pointers indicate)

Alternative approaches

- Microservices / RESTful APIs (no view just JSON data transformed)
- Middleware, mediator, and command patterns

Transformation

- After picking up MVC, at least as a planning tool, we return to the transformation part
  - How do I transform?
  - What do I transform to?
  - Let us start with the latter

What do I transform to?

- Pick a purpose
  - to view:
    - on screen, html
    - for reading offline, ePub, PDF
  - data for components:
    - JSON
    - Other XML
  - other uses:
    - Suitable format
  - transformation from

What are the data needs for each use of transformed data?

- this guides the resulting format and what features are in there
  - Sometimes you need further information for a use than what is available in your data
    - can it be computed from your data?
    - Do you need to add other sources?
    - Can annotations be added to current sources to address the need?

How do I transform?

- extract from data sources (resources)
- combine several sources
- compute (into runtime data or on the fly?)
- pick standard formats (yes, really, we will come back to this)
  - enables possible reuse
  - gives better understanding by more people
  - certainly make it more sustainable

How do I transform?

- With what do I transform?
  - Source is text or encoded text: Xquery, XSLT, explicit API calls to services
  - source is not (written) text:
    - if images of text: OCR and encode minimally to treat it as similar text sources
    - If images, audio, video: convert from surce to runtime/use formats

Testing

- To be safe you should make tests for the transformations too

Continuation follows ...

- Hopefully some food for thought on what to think about for your own project and the resources

Transformations and the MVC pattern

Pittsburgh-NEH-institute 2022
day 1 session 2 slot 3

How do I transform, what do I transform to?

- Think of this from the perspective of your own project needs

MVC pattern?

- MVC is an architectural pattern. Cf design pattern.
  - So just like any pattern it will be loose in the ends

What?

- MVC pattern divides the application into three logical parts: Model, View, and Controller. hence its acronym name.
- Usually used to describe graphical user interface, nowadays also used for mobile and web apps.
  –

What does the MVC pattern try to solve?

- Let the user control a large and complex data set
- The three parts have their own specific responsibilities
  - This avoids repetition
  - Helps with creating solid structures

Part by part: The model

- The model
  - Maintains the data. E.g. connects with database
  - Responds to model requests (for data)

Part by part: The view

- The view
  - Represents data
  - Creates the user interface

Part by part: The controller

- The controller
  - Tells the model what to do based on the requests from the view
  - Does not handle data logic

Positive effects

- When business logic is separated form the user interface
  - Components are more easily reusable
  - Components can be made, deployed, maintained and tested independently
    - -> TDD, semantic URLs

Negative effects

- Not suitable for small applications (still good for planning)
- High complexity
- Inefficiency of data access in view

Planning tool

- Thinking about programming and organize your files
  - -> translate your ideas into code
- Interaction with other code gets easier
- Returning to the code gets easier as well
- We will certainly return to this later (as some of the pointers indicate)

Alternative approaches

- Microservices / RESTful APIs (no view just JSON data transformed)
- Middleware, mediator, and command patterns

Transformation

- After picking up MVC, at least as a planning tool, we return to the transformation part
  - How do I transform?
  - What do I transform to?
  - Let us start with the latter

What do I transform to?

- Pick a purpose
  - to view:
    - on screen, html
    - for reading offline, ePub, PDF
  - data for components:
    - JSON
    - Other XML
  - other uses:
    - Suitable format
  - transformation from

What are the data needs for each use of transformed data?

- this guides the resulting format and what features are in there
  - Sometimes you need further information for a use than what is available in your data
    - can it be computed from your data?
    - Do you need to add other sources?
    - Can annotations be added to current sources to address the need?

How do I transform?

- extract from data sources (resources)
- combine several sources
- compute (into runtime data or on the fly?)
- pick standard formats (yes, really, we will come back to this)
  - enables possible reuse
  - gives better understanding by more people
  - certainly make it more sustainable

How do I transform?

- With what do I transform?
  - Source is text or encoded text: Xquery, XSLT, explicit API calls to services
  - source is not (written) text:
    - if images of text: OCR and encode minimally to treat it as similar text sources
    - If images, audio, video: convert from surce to runtime/use formats

Testing

- To be safe you should make tests for the transformations too

Continuation follows ...

- Hopefully some food for thought on what to think about for your own project and the resources