Position Paper for Workshop on Declarative Models of Distributed Web Applications

Lasse Pajunen

Nokia Research Center

NOKIA
Connecting People

Contents

- Challenges in Designing Web Applications
- Clients Classification in Web Applications
- Modeling Interaction Logic for Web Applications
- Considering Future Work within W3C

Challenges in Designing Web Applications

- On the client side: how to render web pages and make them user friendly with different devices with different and dynamic capabilities?
  - Input controls
  - Screen size
  - Available memory
  - Computing power
  - Network bandwidth
- On the server side: how to design sites and applications that they will make best out of the client and its delivery context?
- In the end: it is the end user experience that matters the most.

Providing Features for Web

- Explicit features are capabilities in service description languages.
  - Layout options on CSS
  - Drawing capabilities of SVG
- Implicit features are capabilities of the system and its implementation. These features are enabled by proper description language design.
  - Speed of HTML rendering
  - Enhancements in CSS layout scaling
  - Security features like application sandbox
- Simplicity and direct execution are important.
  - Authoring technologies should be as simple as possible and there should be as few separate technologies as possible.
  - Indirections make application design process more complicated.
- Web site and application design should be as easy as possible, where the least amount of information is needed by the service designer and the system does as much as possible by default.

Client Classification for User Interface Generation

- Small clients (most restricted)

  - Lacking support for street HTML, CSS, and other similar standards
  - More limitations on memory, computing power, and network bandwidth
  - Limitations on supported use cases
- Normal clients

  - Have support for Full web standards
  - Some limitations on memory, computing power, and network bandwidth
  - Different use contexts require lots of client side adaptation options
- Large clients (least restricted)

  - Have support for Full web standards with possible extensions
  - Do not have limitations on memory, computing power, and network bandwidth
  - Can render most complicated use cases, and benefit of adaptation options
- We must continue making Full web standards more flexible and adaptable to dynamic conditions.

Modeling Interaction Logic for Web Applications

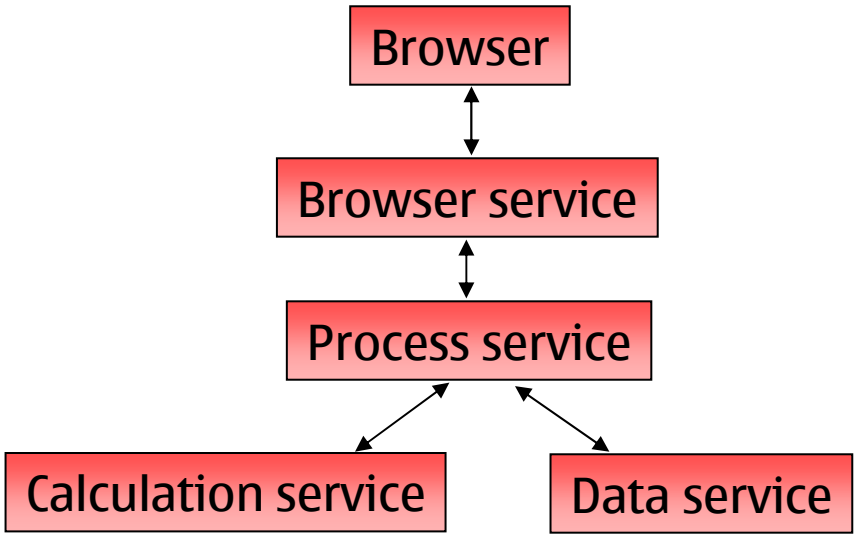
- The usage scenarios for many web applications can be modeled as simple processes.
  - Launch a service
  - Register and select from a personal blog list
  - Comment to the selected blog
  - Log out
- Available options are based on previous actions. They are not independent from each other.
  - You cannot comment before you have registered.
- Modeling plain user interaction is not enough. Connection to computational logic and dynamic data needs to be modeled together with the user interaction.

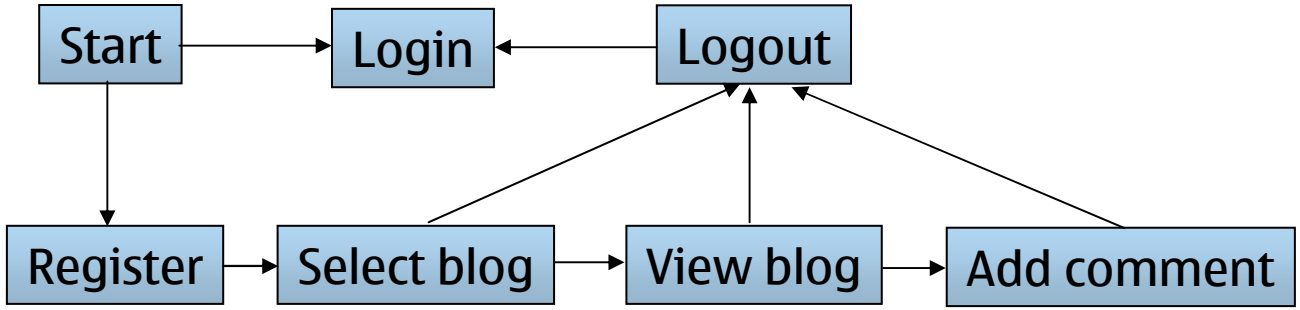
Modeling Web Applications with (X)HTML, WS-BPEL and WSDL

- Web services (especially WSDL technology) allow publishing data and computation services to be used in user interaction processes.
- Modeling browser as a presentation service allows integrating everything together in one generic manner.
  - Input (X)HTML
  - Output form data
- Workflow and service composition technologies (WS-BPEL) can be used to describe executable compositions of data, computation, and presentation services.
- In this kind of model, the process description will
  - Describe the connection to back end data and calculation services,
  - Use the browser as a rendering and interaction service, and
  - Describe the interaction model as the service.

Benefits for Service Authors

- One programming model simplifies design on certain types of applications.
- Considering browser as a service unifies whole application design process to single model.
- Models fits to applications that are long running having explicit steps.





Considering Future Work within W3C

- Understanding different client classes and their evolution is important when planning future standardization efforts.
  - More clients are supporting Full web standards.
  - More client side adaptation options are needed for different and dynamic use contexts.
- Modeling browser as a service would allow leveraging workflow and service composition technologies in a uniform manner.
- To get browser as a service paradigm to work, a page push paradigm is needed.
  - “Server pushes new pages to the client.”
  - This can be implemented using client pull and proxy. A server sends pages to the proxy and client polls the proxy. After retrieving a new page, a client must copy the current state to the new page including form fields and input control states.
- Important choice is to decide which features require platform implementations and which features can be supported by providing best practice guidelines.

Thanks

- Comments or questions?