Teaching Old Services New Tricks: Adding HATEOAS Support as an Afterthought

Olga Liskin, Leif Singer, Kurt Schneider
{olga.liskin, leif.singer, kurt.schneider}
@inf.uni-hannover.de

28.03.2011
Content

- Motivation
- Main Goals
- Service Example
- Terminology
- Notation for Application Models
- Conception of Wrapper Component
- Comparison
- Conclusions & Outlook
Motivation

- Problem with web service communication: Client needs to know exactly, what a request has to look like
- Many sources for errors
  - Coding errors
  - Invalid requests
- Idea: Server includes request-information in response messages
  - Which requests allowed next
  - What they look like
- HATEOAS
- But: not many services conform to this principle
Motivation

Example:

Service response without HATEOAS:

HTTP/1.1 200 OK
Last-Modified: Wed, 01 Dec 2010 17:36:02 GMT
Content-Type: application/xml
<task>
  <id>208</id>
  <name>create GUI</name>
  <status>inprogress</status>
  <parentStoryId>04</parentStoryId>
</task>

What can the client do next?

No idea. The client has to know by itself.
Motivation

Example:

Service response with HATEOAS:

HTTP/1.1 200 OK
Last-Modified: Wed, 01 Dec 2010 17:36:02 GMT
Content-Type: application/xml
Link: <stories/04/tasks/208/finish>; rel="finish"
Link: <stories/04/tasks/208/block>; rel="block"
<task>
  <id>208</id>
  <name>create GUI</name>
  <status>inprogress</status>
  <parentStory>/stories/04</parentStory>
</task>

...and also indicate this

Control elements directly present in response message.

Now client can see following requests and how to make them.
Main Goals

- Create HATEOAS support using *state charts*
  - Automatically generate a wrapper

- Prerequisite: Clear way to model network-based applications
  - State Charts
  - Terminology
Service Example

Backlog service: support agile development projects

- **Backlog**
  - contains story cards

- **Story Card**
  - One particular topic
  - Different states
    - „defined“, „in progress“, „blocked“,…
    - Depend on included tasks

- **Task**
  - Story Card divided into tasks
  - Different states
    - Changed by user
1 Modelling Network-Based Applications
Terminology

• Valid requests depend on current application state
• What exactly is application state?

• Relevant terminology:
  – application
    • „representation of the business-aware functionality of a system“ (Fielding)
  – resource state
    • Values of a resource‘s attributes
  – application state
    • Requests, responses and the processing of those
    • E.g.: „process detail view of a story card“
Terminology

**New idea:**

- Equivalence classes of application states
  - Combine "similar" states
  - Determined by *resource class* and *resource state*
State Chart

- States
  - Equivalence classes of application states
  - Composite States combine states with same resource class
State Chart

• Transitions
  – Possible change of application state
    • triggered by new client request
  – Target state reached in case of success
• Transitions between Simple States

![State Chart Diagram]
State Chart

- Transitions between Composite States
  - More precise information about control flow necessary
  - Outgoing transition:
    - Transition possible from all sub-states
  - Incoming transition:
    - Client requests a resource of particular class
    - Sub-state can only be determined at runtime
    - choice-pseudostate
State Chart

- Simplified notation
  - Remove end vertex
  - Remove choice pseudostate construct
State Chart

• Use State Chart as a static *map*
2 Constructing A Wrapper
Wrapper - Conception

• Top-Down view
  – Wrapper at first as black-box

• Insert wrapper between client and server
  – Request forwarded
  – Response enriched with transitions
    • But not changed further
Main process

1. Forward request
2. Calculate application state
3. Look up transitions
4. Construct links

Enrich response

<<datastore>>
Transition model

HATEOAS-
Response
Wrapper - Concrete Process Steps

Full process cycle

HTTP/1.1 200 OK
Last-Modified: Wed, 01 Dec 2010
Content-Type: application/xml

<task>
  <id>04</id>
  <name>Update Database</name>
  <status>defined</status>
  <parentStory>08</parentStory>
</task>

Local-name("[1]) = task
/task/status/text() = defined
/task/id/text() = 04
/task/parentStory/text() = 08

URI-Templates

HTTP/1.1 200 OK
Last-Modified: Wed, 01 Dec 2010
Content-Type: application/xml

Link: </stories/08/tasks/04/block>;rel="Task.block"
Link: </stories/08/tasks/04/begin>;rel="Task.begin"
Link: </stories/08/tasks>; rel="Task.allTasks"
Link: </stories/08>; rel="belongs_to_story"

/task/id/text() = 04
/task/parentStory/text() = 08

Task.block: </stories/08/tasks/04/block>
Task.begin: </stories/08/tasks/04/begin>
Task.allTasks: </stories/08/tasks>
Belongs_To_Story: </stories/08>
Comparison

- Implementation of backlog service
- Automated generation of wrapper (from transition model)
- Develop 2 clients to test concepts
  - With wrapper <-> without wrapper

```
List links =
    response.getHeaders().get("Link");
for(String link : links){
    if(link.contains("story.start")){
        String uri = extractUri(link);
        this.startButton.setUri(uri);
        this.startButton.setEnabled(true);
    }
}
```

```
if(story.getStatus().equals(StoryStatus_DEFINED)){
    this.startButton.setEnabled(true);
    this.startButton.setUri("/stories/
        story.getId() + "/start");
    this.blockButton.setEnabled(true);
    this.blockButton.setUri("/stories/
        story.getId() + "/block"");
}
```
Conclusions

- Creation of theoretical concepts
- Development of wrapper process
- Check the concept
  - Is working
  - Generic
  - Improves development and maintainance of clients
- Errors in model lead to invalid/missing links
- Input data is complex
Outlook

• Further research on input of data
  – Project environment
  – Interface descriptions (e.g. WADL) as additional source

• Allow changes of whole service API
  – Can improve more REST aspects

• Use state charts for creation of services (not only wrappers)
Content

• Motivation
• Main Goals
• Service Example
• Terminology
• Notation for Application Models
• Conception of Wrapper Component
• Comparison
• Conclusions & Outlook

Questions?