Why are we here?
[Live Demo of the „Orbitz Bug“:

1. Visit orbitz.com in a web browser

2. Open multiple flights in multiple windows]

3. Try to book a flight. Regardless of which flight was selected in your window, the flight that will be booked will always be the flight from the most recently opened window (even if it was closed in the meantime)
We want to fix such bugs!
The Web Server

Web Server

Client
The Web Server

- Internal storage
- Consists of key/value pairs
- Represented by $\sigma$: Key $\rightarrow$ Value
- Describes server state

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The Web Server

- Internal storage
- Consists of key/value pairs
- Represented by $\sigma$: Key $\rightarrow$ Value
- Describes server state

- Scripts (dynamic pages, forms)

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
</tbody>
</table>

... ...
The Web Server

- Internal storage
- Consists of key/value pairs
- Represented by $\sigma$: Key $\rightarrow$ Value
- Describes server state

Script:

```php
<?php
   for i in flights
       display(i)
   end>
```
The Web Server

- Internal storage
- Consists of key/value pairs
- Represented by $\sigma$: Key $\rightarrow$ Value
- Describes server state

- Scripts (dynamic pages, forms)
- Lookup function $P$: URL $\rightarrow$ Form

Web Server

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The Client

Web Server

flight 815
dest. SF

Scripts

Client
The Client

• Active page
The Client

- Active page
- Browser cache
- All previously seen forms

Web Server

flight 815
dest. SF

Scripts

Current form

... ... ...
<html>
<form action='book-flight.html'>
  <input name='flight' type='text' value='815'>
  <input name='destination' type='text' value='San Francisco'>
  <input name='submit' type='submit' value='Submit!'>
</form>
</html>
<html>
<form action='book-flight.html'>
  <input name='flight' type='text' value='815'>
  <input name='destination' type='text' value='San Francisco'>
  <input name='submit' type='submit' value='Submit!'>
</form>
</html>

Representation:

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>San Francisco</td>
</tr>
</tbody>
</table>
Supported Actions

- Users may do any of the following at any time:
  - Enter data into the current form
  - Switch to a cached page (e.g., click on the back button)
  - Submit a form
What happens, when...

- Users enter form data:
  - The key/value vector of the form is modified to store the updated value
What happens, when...

• Users enter form data:
  - The key/value vector of the form is modified to store the updated value
  - The updated form is added to the browser cache

URL: book-flight.html

<table>
<thead>
<tr>
<th>flight</th>
<th>816</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination</td>
<td>San Francisco</td>
</tr>
</tbody>
</table>

Current
What happens, when...

• Users switch to some form:
  - The new form is set as the client’s „current page“ (but only if it’s found in the cache)
What happens, when...

- Users submit a form?
Form Submissions

Web Server

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Scripts

Client

Current form

... ...

... ...

... ...
Form Submissions

- Server computes the new form

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Scripts

Web Server

Client

Current form

...   ...   ...
Form Submissions

- Server state (storage) is updated

Web Server

Client

Scripts

Current form

• Server state (storage) is updated

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
</tbody>
</table>
Form Submissions

- Client’s „current page“ is set to the new form
Form Submissions

- New form is added to client’s browser cache

Web Server

- flight 815
- dest. SF
- ...
- ...

Scripts

Client

Current form

... ...

... ...

... ...

... ...
Attention, Mini-Test!

How does switching work again? Explain.
Attention, Mini-Test!

How does switching work again? Explain.

„Rewriting“ describes the transition directly and precisely:

\[ <s, <f_0, \overrightarrow{f}>> \]
\[ \Rightarrow <s, <f_1, \overrightarrow{f}>> \]
where \( f_1 \in f \)
Scripting Language

- Use identifiers, variables
- Create functions
- Apply functions
- Create new forms
- Extract values from forms (via keys)
- Basic I/O (Server storage read/write)
Scripting Language

- Use identifiers, variables
- Create functions
- Apply functions
- Create new forms
- Extract values from forms (via keys)
- Basic I/O (Server storage read/write)
Modelling the Bug

<table>
<thead>
<tr>
<th>Scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>flight</td>
</tr>
<tr>
<td>dest.</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Web Pages

Show Flights
## Modelling the Bug

### Web Pages
- Flight 1 Details
- Show Flights

### Web Server
- Scripts

<table>
<thead>
<tr>
<th>flight</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---
Modelling the Bug

Show Flights

Flight Details

Web Pages

Web Server

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
</tbody>
</table>

Scripts
Modelling the Bug

Show Flights

Web Pages

Web Server

Flight 1 Details

Flight 2 Details

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Scripts
Modelling the Bug

Web Pages

Show Flights

Flight 1 Details

Flight 2 Details

Web Server

<table>
<thead>
<tr>
<th>flight</th>
<th>1632</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Scripts
Modelling the Bug

Web Pages

Show Flights

Flight 1 Details

Flight 2 Details

Book Flight

Web Server

<table>
<thead>
<tr>
<th>flight</th>
<th>1632</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Scripts
Modelling the Bug

Show Flights

Flight 1 Details

Flight 2 Details

Book Flight

Web Pages

Web Server

<table>
<thead>
<tr>
<th>flight</th>
<th>1632</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Scripts
Explaining the Bug

- Obviously, submitting "outdated" forms causes undesired behaviour
- The HTTP Observer Problem: Server cannot "push" updates to the client (as in MVC)
  ➡ At least produce warnings when detecting outdated requests
Detecting outdated requests

- Server needs a notion of time:
  - Model as number of submits
- Storage records time of last write for each field

<table>
<thead>
<tr>
<th></th>
<th>flight</th>
<th>815</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| time | 4 |

Scripts
Detecting outdated requests

- Introduce „carrier sets“ into forms:
  All locations accessed by this script

- Each form stores its creation time

URL: book-flight.html

<table>
<thead>
<tr>
<th>time</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>flight</td>
<td>815</td>
</tr>
<tr>
<td>destination</td>
<td>San Francisco</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>flight</th>
<th>815</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest.</td>
<td>SF</td>
<td>1</td>
</tr>
</tbody>
</table>
| ... | ... | ...
| time | 4 | Scripts |
Detecting outdated requests

- Whenever a form is submitted, check its carrier set against current storage state and compare time stamps.
- The carrier set represents the assumptions the script made while working.
- If any location from this set was overwritten, script assumptions may have been violated.
Thank you!

- We have built a comprehensive, yet simple model of web interactions

- Three basic semantic rules suffice to describe all possible user actions:
  - „switch“
  - „fill-out“
  - „submit“

- Any questions?
References


- Daniel R. Licata, Shriram Krishnamurthi: "Verifying Interactive Web Programs" (2005)
Addendum: Fun with Types

- Make forms typed!
- Enables static checks for common bugs, like trying to access form data that never got submitted
- Also enables us to give some other safety guarantees
- But: How to keep track of types in a dynamic setting?
Incremental Type Checking

- Uses constraints along with regular type judgements
- Constraints are introduced by creating forms:
  - The successor url of any form must contain a program that takes as input exactly the data (type) of that form
- Otherwise, forms behave essentially like records
Consistency

- Consistency is achieved by checking that all types registered for some form at a particular URL are equivalent.

- Since type constraints can be introduced by the regular type system as well as by additional constraints, this is not always the case.

- If the types are not consistent, refuse to execute the script.