Flow with Continuations

Geert Bevin
CTO
Uwyn bvba

gbevin@uwyn.com
http://uwyn.com
http://rifers.org

Who am I

- **Geert Bevin**
- **CTO of Uwyn,** a small custom application development company (http://uwyn.com)
- **founder of RIFE** (http://rifers.org)
- **creator and contributor of many RIFE projects:** RIFE/Crud, RIFE/Jumpstart, RIFE/Continuations, Bamboo (forum), Bla-bla List (RIA todo list), Drone (information bot), Elephant (blog)

Agenda

- **What are continuations?**
- Continuations and the web
- Existing Java tools
- Other notable application domains

Confusing definitions
Confusing name

Confusing definitions
Confusing name

- **“the remaining work to be done”**
- **“the rest of the computation”**
- **“representation of the execution state of a program”**

Confusing definitions
Confusing name

- **Name describes the implementation details**
- **Makes sense if you already understand the concept**
- **Sounds arcane if you know nothing about it**

Let's clarify that

Let's clarify that

... what do they do?

... what do they do?

- **Mark a location in your code**
- **Capture the context**
- **Store the location and the context**

Let's clarify that
... what do they do?

... what's in it for you?

... what's in it for you?

- **Restore the context**
- **Inject new context elements**
- **Resume the execution where you left off**

better name:

Resumable Markers

Concretely

Marks a location in your code (pseudo code)

```
marker = pause()

...

marker.resume()
```

Marks a location in your code (pseudo code)

```
marker = pause()
...
marker.resume()
```

- **Resembles gotos, breaks, continue statements**
  - **label a location**
  - **jump back to it**
- **Except**
  - **captures the context**
  - **can be resumed from anywhere (not limited to scope)**
  - **some implementations even allow an application restart**

Captures the context (pseudo code)

```
var msg = “hello there”
marker = pause()
println(msg)
```

Captures the context (pseudo code)

```
var marker

function hello {
  var msg = “hello there”
  marker = pause()
  println(msg)
}

hello()
print(“I say: ”)
marker.resume()
```

Captures the context (pseudo code)

```
var marker

function hello {
  var msg = “hello there”
  marker = pause()
  println(msg)
}
```

```
hello()
print(“I say: ”)
marker.resume()
```

- **Program output is**

  ```
  I say: hello there
  ```

- **Local variables are automatically captured**
- **Context is automatically restored when resumed**

Stores context and location (pseudo code)

```
function say(arg) {
  marker = pause()
  println(arg)
}
```

Stores context and location (pseudo code)

```
var marker

function say(arg) {
  marker = pause()
  println(arg)
}

say(“this backwards”)
var marker1 = marker
say(“Let's call”)
var marker2 = marker
```

Stores context and location (pseudo code)

```
var marker

function say(arg) {
  marker = pause()
  println(arg)
}

say(“this backwards”)
var marker1 = marker
say(“Let's call”)
var marker2 = marker

marker2.resume()
marker1.resume()
```

Stores context and location (pseudo code)

```
var marker

function say(arg) {
  marker = pause()
  println(arg)
}
```

```
say("this backwards")
var marker1 = marker
say("Let's call")
var marker2 = marker

marker2.resume()
marker1.resume()
```

- **Program output is**

  `Let's call this backwards`

- **Context and location are stored in a variable**
- **Multiple continuations can be active at once**

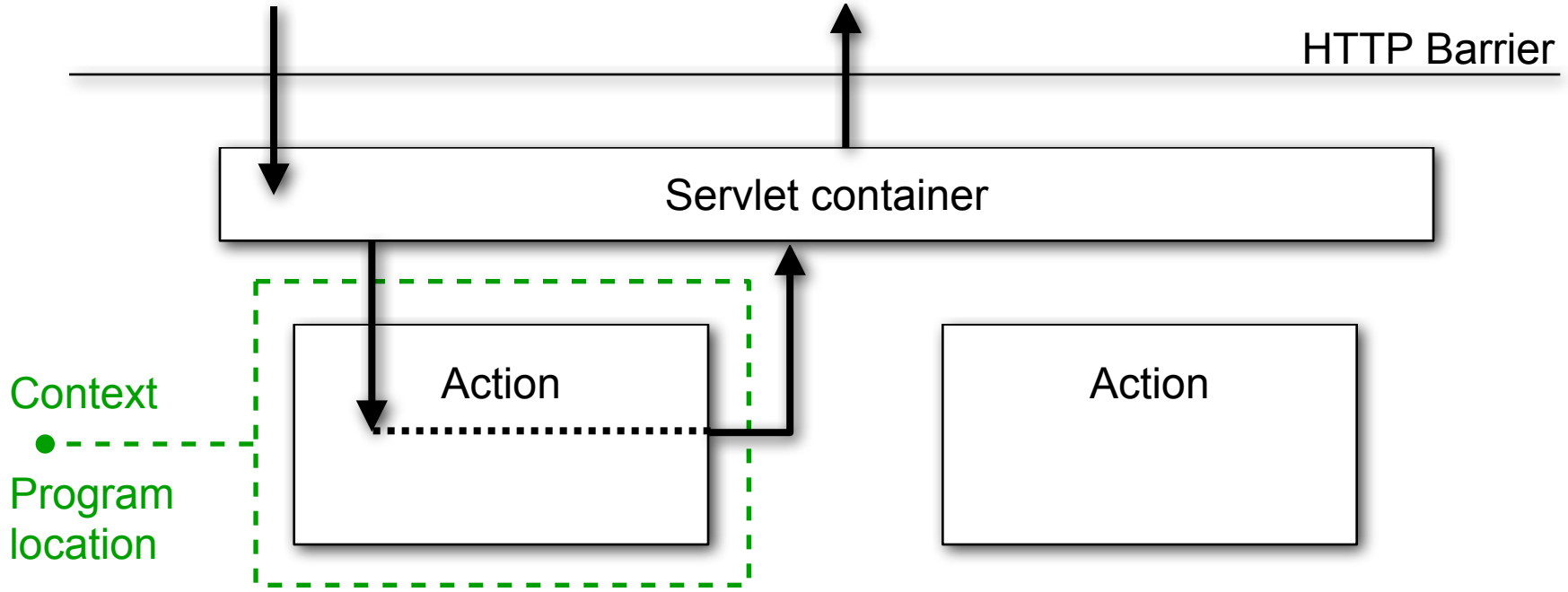
Agenda

- What are continuations?
- **Continuations and the web**
- Existing Java tools
- Other notable application domains

Web applications have different requirements

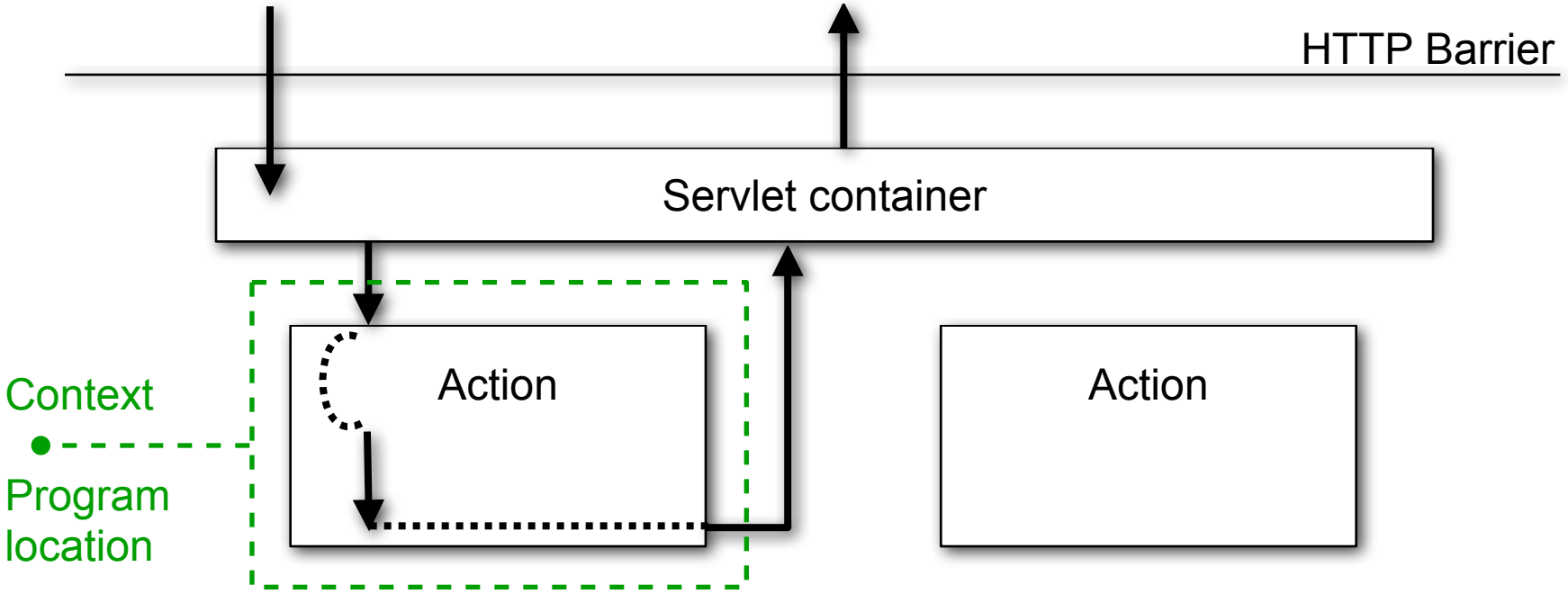
Continuations and the web

- **Context to capture is located in the action**
- **Anything that happens before should be discarded**



Continuations and the web

- **Only a part of the context should be restored**
- **Execution only skipped from a certain point onwards**



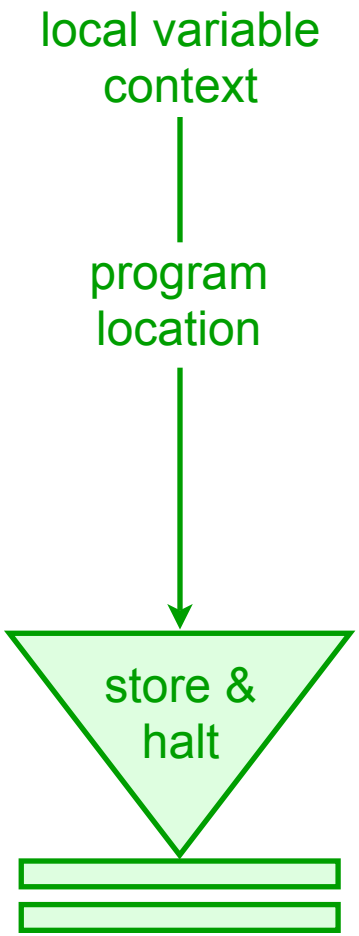
These are called

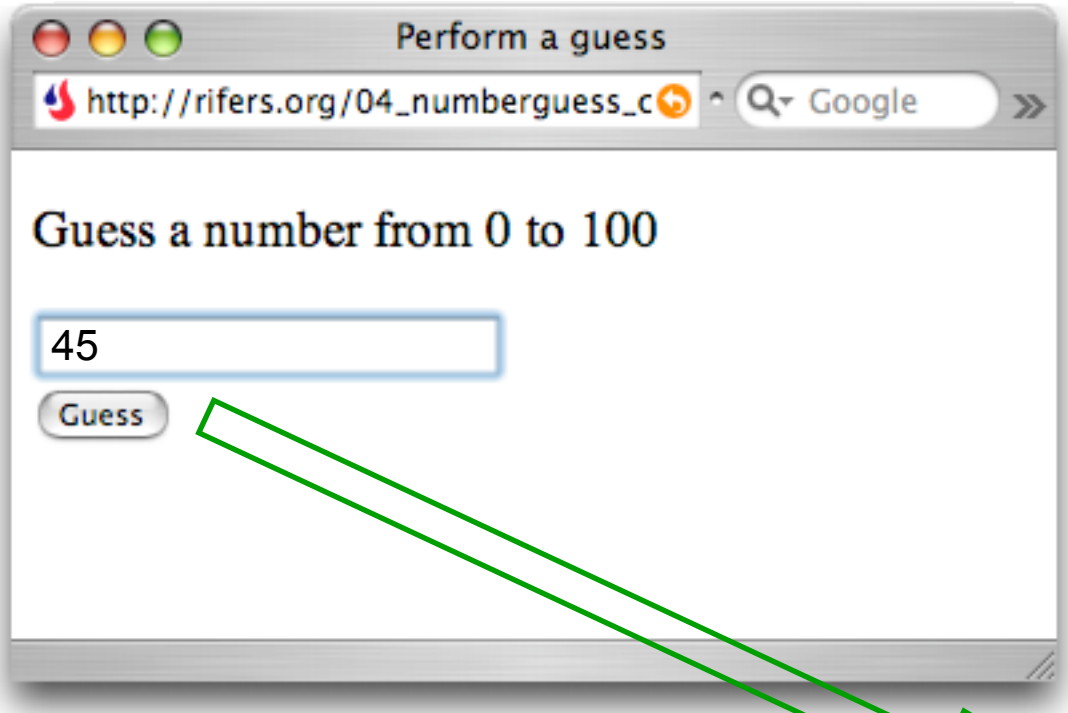
partial continuations

Let's look at a practical example

```
public class Game extends Element {
  private static Random randomNumbers = new Random();
  public void processElement() {

    Template template = getHtmlTemplate("game");
    int answer = 0, guesses = 0, guess = -1;

    answer = randomNumbers.nextInt(101);
    while (guess != answer) {
      print(template);

      pause();

      guess = getParameterInt("guess", -1);
      if (guess < 0 || guess > 100) {
        template.setBlock("warning", "invalid");
        continue;
      }
      guesses++;

      if (answer < guess)      template.setBlock("msg", "lower");
      else if (answer > guess) template.setBlock("msg", "higher");
    }

    template = getHtmlTemplate("success");
    template.setValue("answer", answer);
    template.setValue("guesses", guesses);
    print(template);
  }
}
```

```
public class Game extends Element {
  private static Random randomNumbers = new Random();
  public void processElement() {
    Template template = getHtmlTemplate("game");
    int answer = 0, guesses = 0, guess = -1;
    answer = randomNumbers.nextInt(101);
    while (guess != answer) {
      print(template);
      pause();
      guess = getParameterInt("guess", -1);
      if (guess < 0 || guess > 100) {
        template.setBlock("warning", "invalid");
        continue;
      }
      guesses++;

      if (answer < guess)      template.setBlock("msg", "lower");
      else if (answer > guess) template.setBlock("msg", "higher");
    }

    template = getHtmlTemplate("success");
    template.setValue("answer", answer);
    template.setValue("guesses", guesses);
    print(template);
  }
}
```





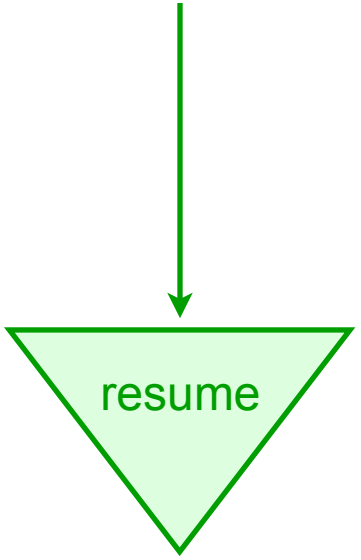
control goes to user

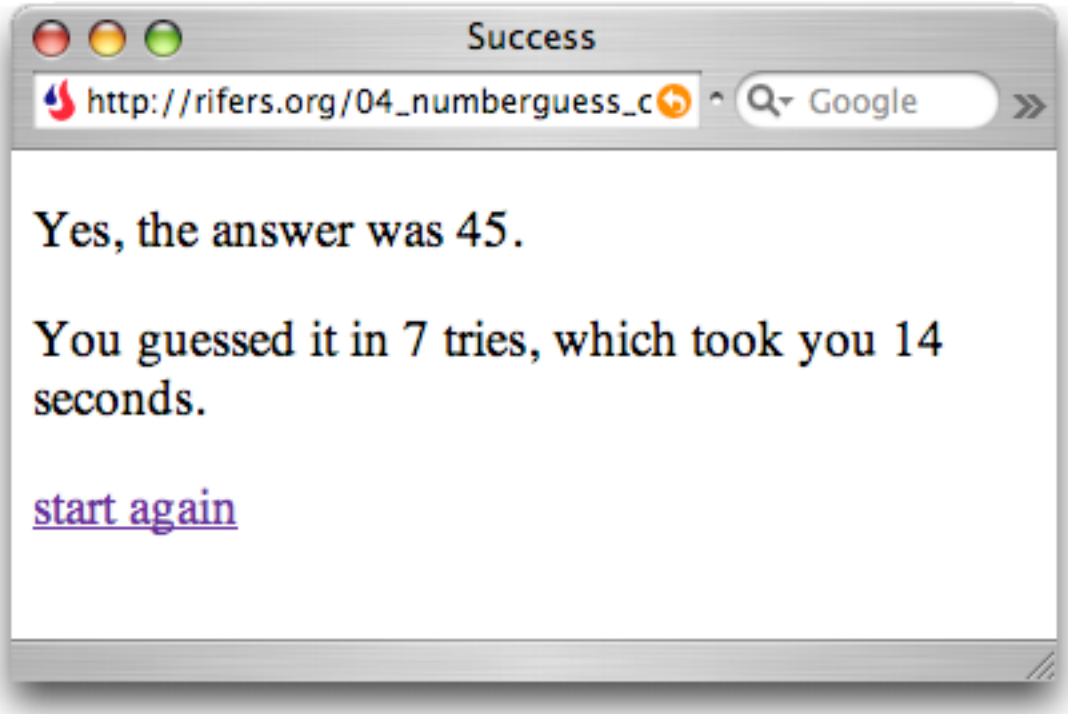
submission from user

```
public class Game extends Element {
  private static Random randomNumbers = new Random();
  public void processElement() {
    Template template = getHtmlTemplate("game");
    int answer = 0, guesses = 0, guess = -1;

    answer = randomNumbers.nextInt(101);
    while (guess != answer) {
      print(template);

      pause();

      guess = getParameterInt("guess", -1);
      if (guess < 0 || guess > 100) {
        template.setBlock("warning", "invalid");
        continue;
      }
      guesses++;

      if (answer < guess)      template.setBlock("msg", "lower");
      else if (answer > guess) template.setBlock("msg", "higher");
    }

    template = getHtmlTemplate("success");
    template.setValue("answer", answer);
    template.setValue("guesses", guesses);
    print(template);
  }
}
```

local variable context

program location



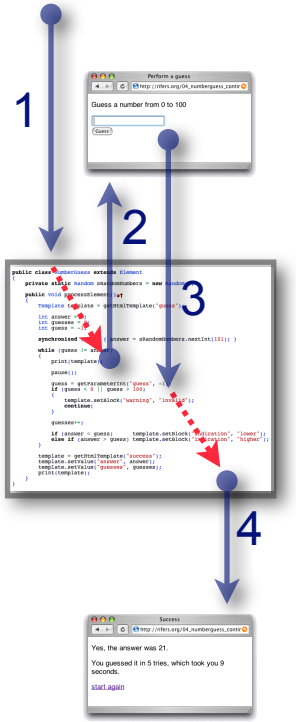


result goes to user

How does it compare?

How does it compare?

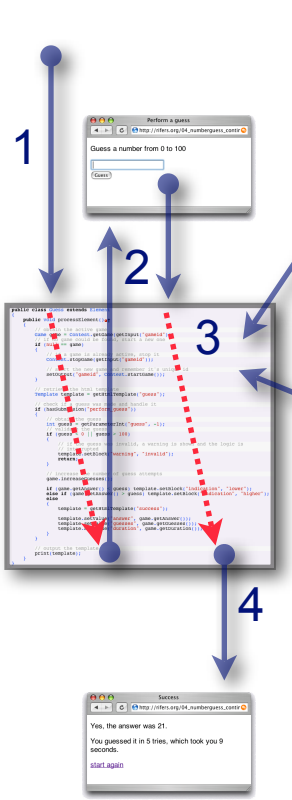
Continuations



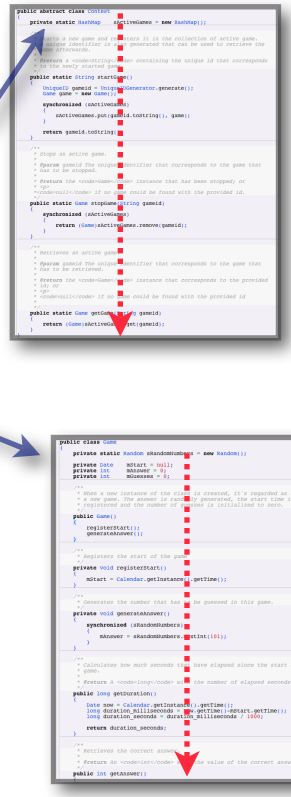
1. process request
2. display form
3. process reply
4. display result

executed 1x
internal automatic state

Traditionally



executed 2x



external manual state

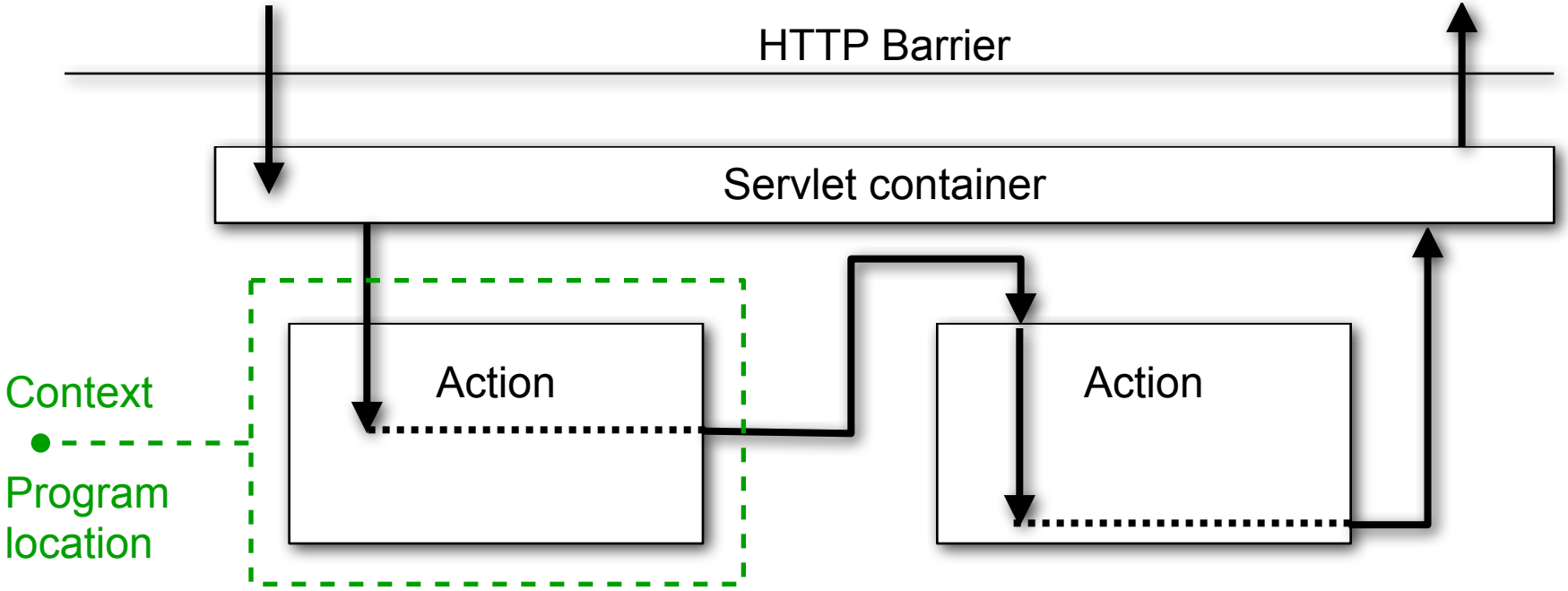
Summarized

- **Natural linear control flow**
- **Centralized and automatic state handling**
- **Facilitates creation of complex web application flows**

Call/answer continuations

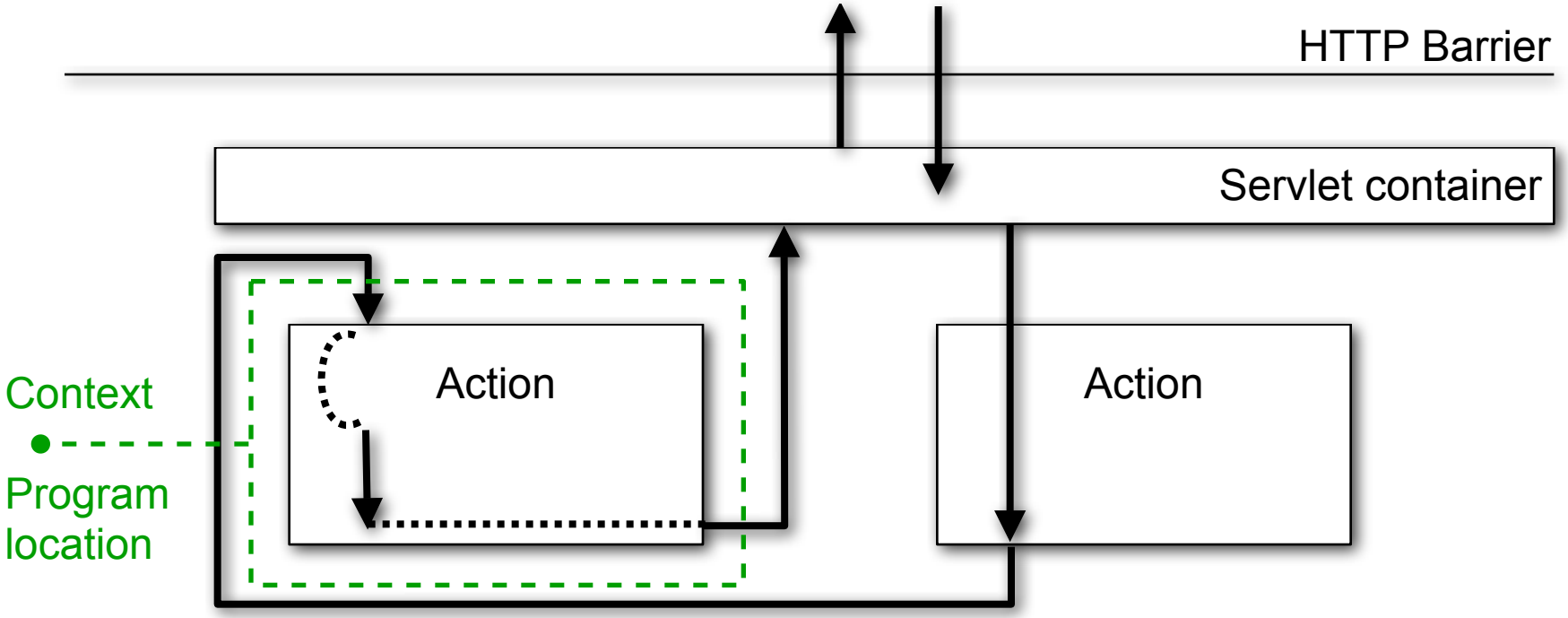
Call/answer continuations

- **Request doesn't have to return after** `pause()`
- **You can call another action to interact with the user**



Call/answer continuations

- **Next request goes to last action**
- **Resumes the original action and provides an answer**



Let's look at another practical example

Data modification page

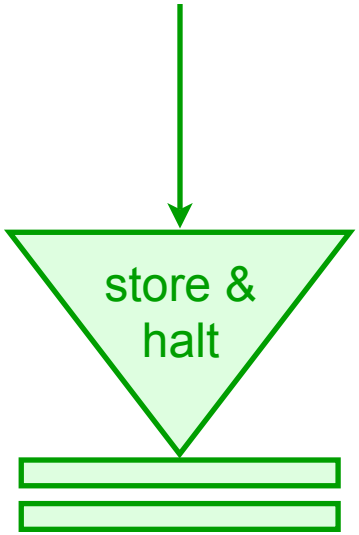
```
public class Delete extends Element {
  public void processElement() {

    Template template = getHtmlTemplate("pub.delete");

    print(template);

    pause();

    if (hasSubmission("delete")) {

      Boolean answer = (Boolean)call("dialog");

      if (answer.booleanValue()) {
        template.setBlock("content", "deleted");
      } else {
        template.setBlock("content", "preserved");
      }
    }
    print(template);
  }
}
```

Data modification page

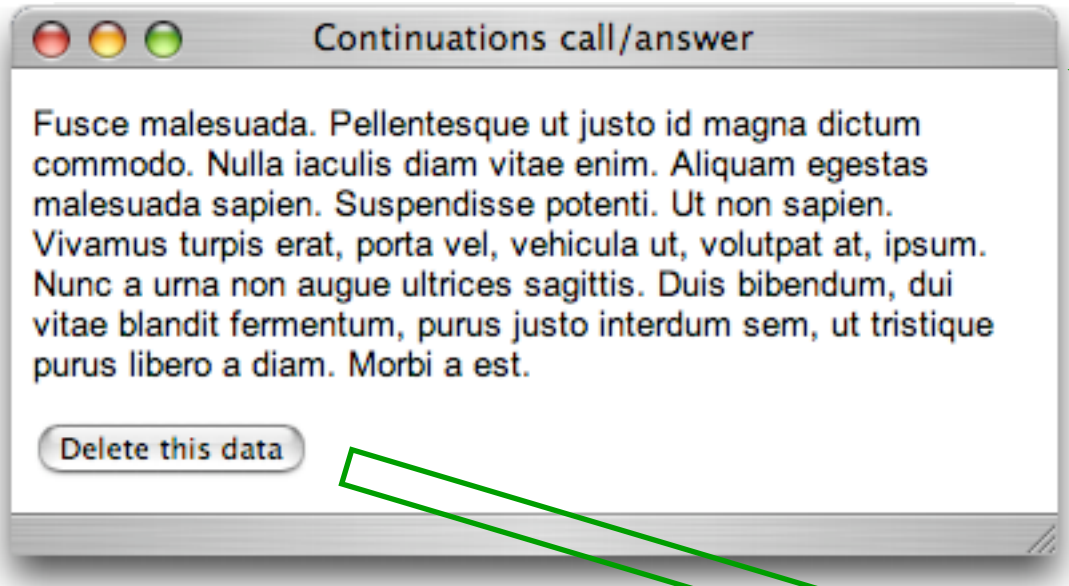
```
public class Delete extends Element {
  public void processElement() {
    Template template = getHtmlTemplate("pub.delete");
    print(template);
    pause();
    if (hasSubmission("delete")) {

      Boolean answer = (Boolean)call("dialog");

      if (answer.booleanValue()) {
        template.setBlock("content", "deleted");
      } else {
        template.setBlock("content", "preserved");
      }
    }
    print(template);
  }
}
```

local variable context

program location



Data modification page



control goes to user

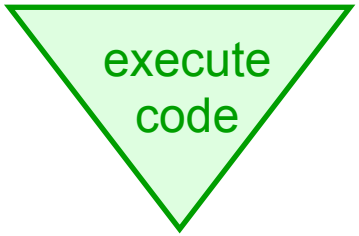
submission from user

Data modification page

```
public class Delete extends Element {
  public void processElement() {
    Template template = getHtmlTemplate("pub.delete");
    print(template);
    pause();
    if (hasSubmission("delete")) {

      Boolean answer = (Boolean)call("dialog");

      if (answer.booleanValue()) {
        template.setBlock("content", "deleted");
      } else {
        template.setBlock("content", "preserved");
      }
    }
    print(template);
  }
}
```

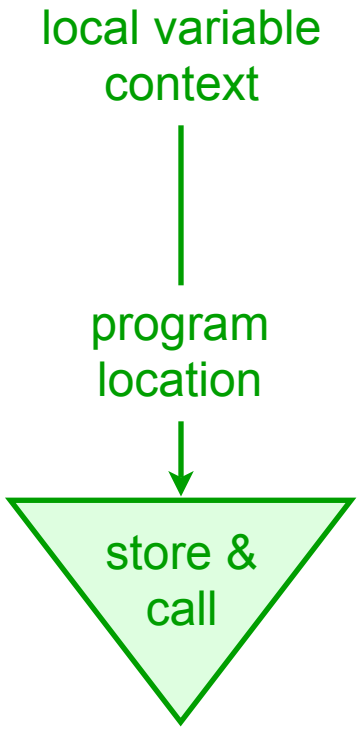
Data modification page

```
public class Delete extends Element {
  public void processElement() {

    Template template = getHtmlTemplate("pub.delete");

    print(template);

    pause();

    if (hasSubmission("delete")) {

      Boolean answer = (Boolean)call("dialog");

      if (answer.booleanValue()) {
        template.setBlock("content", "deleted");
      } else {
        template.setBlock("content", "preserved");
      }
    }
    print(template);
  }
}
```



Data modification page

```
public class Delete extends Element {
  public void processElement() {
    Template template = getHtmlTemplate("pub.delete");
    print(template);

    pause();

    if (hasSubmission("delete")) {
      Boolean answer = (Boolean)call("dialog");
      if (answer.booleanValue()) {
        template.setBlock("content", "deleted");
      } else {
        template.setBlock("content", "preserved");
      }
    }
    print(template);
  }
}
```



Confirmation dialog

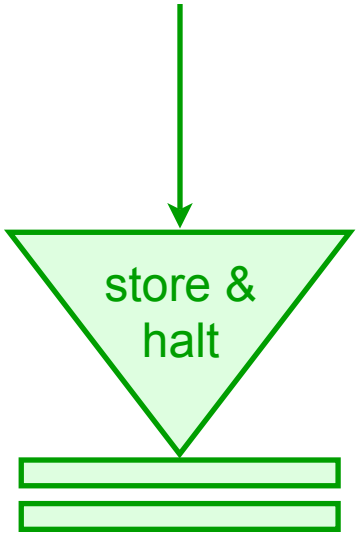
```
public class Dialog extends Element {
  public void processElement() {

    Template template = getHtmlTemplate("pub.dialog");

    print(template);

    pause();


    if (getParameter("answer", "no").equals("yes")) {

      answer(true);

    } else {

      answer(false);

    }
  }
}
```

Confirmation dialog

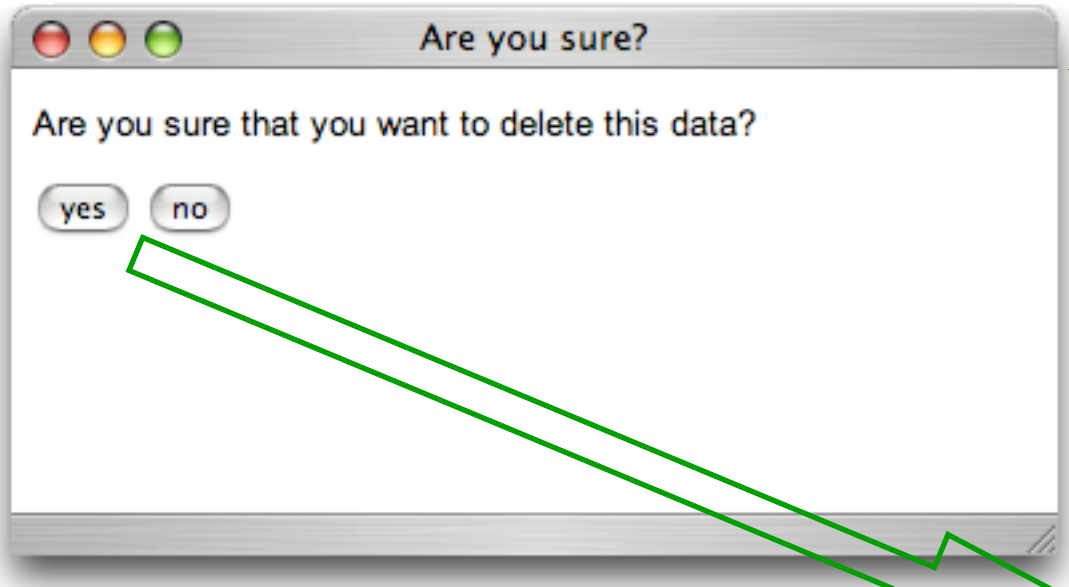
```
public class Dialog extends Element {
  public void processElement() {
    Template template = getHtmlTemplate("pub.dialog");
    print(template);
    pause();

    if (getParameter("answer", "no").equals("yes")) {
      answer(true);
    } else {
      answer(false);
    }
  }
}
```

local variable context

program location



Confirmation dialog

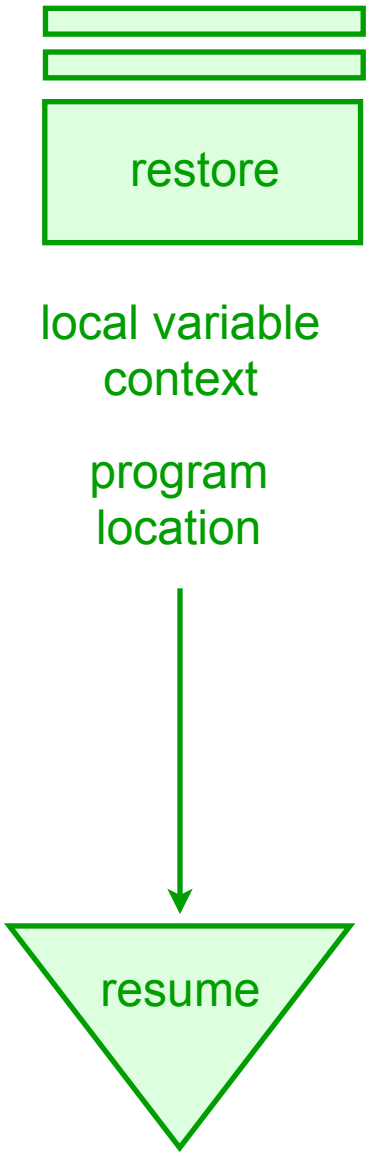


control goes to user

submission from user

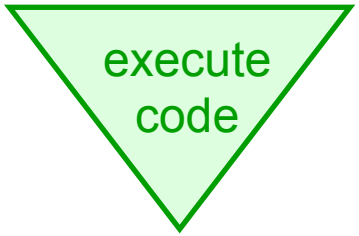
Confirmation dialog

```
public class Dialog extends Element {
  public void processElement() {
    Template template = getHtmlTemplate("pub.dialog");
    print(template);
    pause();

    if (getParameter("answer", "no").equals("yes")) {

      answer(true);

    } else {

      answer(false);

    }
  }
}
```



Confirmation dialog

```
public class Dialog extends Element {
  public void processElement() {

    Template template = getHtmlTemplate("pub.dialog");

    print(template);

    pause();


    if (getParameter("answer", "no").equals("yes")) {

      answer(true);

    } else {

      answer(false);

    }
  }
}
```



Confirmation dialog

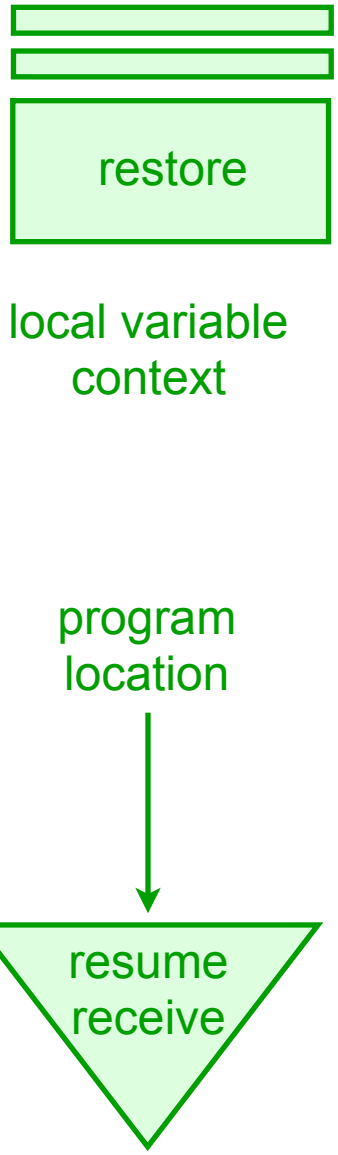
```
public class Dialog extends Element {
  public void processElement() {

    Template template = getHtmlTemplate("pub.dialog");

    print(template);

    pause();


    if (getParameter("answer", "no").equals("yes")) {

      answer(true);

    } else {

      answer(false);

    }
  }
}
```

provide value to the call continuation

answer call

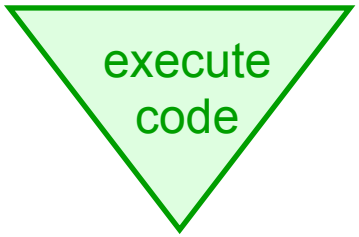
Data modification page

```
public class Delete extends Element {
  public void processElement() {
    Template template = getHtmlTemplate("pub.delete");

    print(template);

    pause();

    if (hasSubmission("delete")) {
      Boolean answer = (Boolean)call("dialog");

      if (answer.booleanValue()) {
        template.setBlock("content", "deleted");
      } else {
        template.setBlock("content", "preserved");
      }
    }
    print(template);
  }
}
```

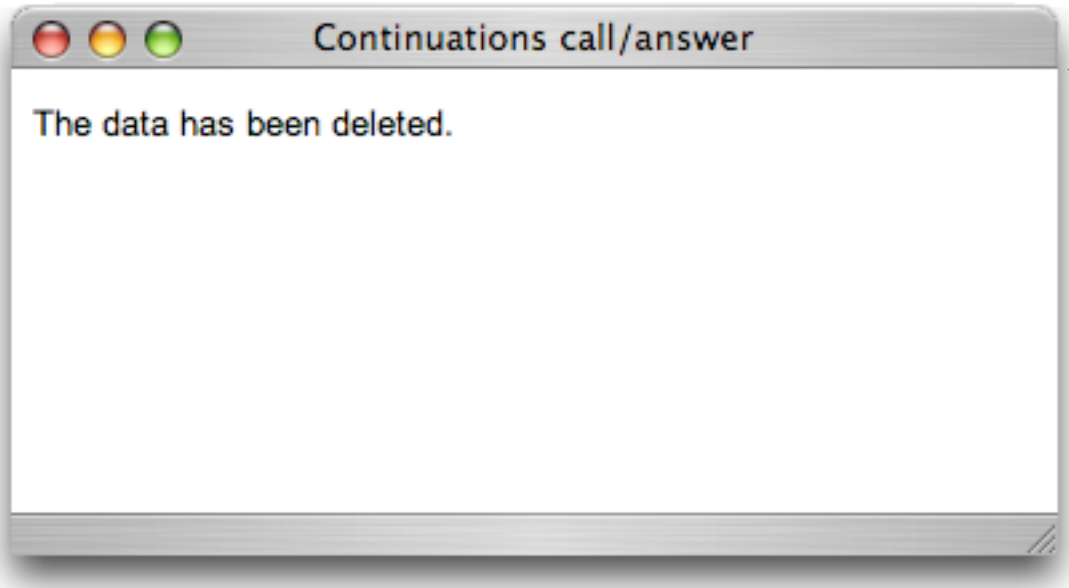


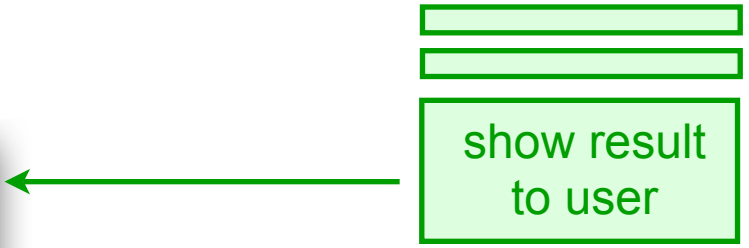
Data modification page

```
public class Delete extends Element {
  public void processElement() {

    Template template = getHtmlTemplate("pub.delete");

    print(template);

    pause();

    if (hasSubmission("delete")) {

      Boolean answer = (Boolean)call("dialog");

      if (answer.booleanValue()) {
        template.setBlock("content", "deleted");
      } else {
        template.setBlock("content", "preserved");
      }
    }
    print(template);
  }
}
```



Data modification page





What would you use this for?

What would you use this for?

- **Islands of functionality**
  - **Wizards**
  - **Setup and configuration**
  - **Questionnaires**
- **Sidesteps**
  - **Confirmation dialogs**
  - **Additional multi-step information**

So what are the downsides?

Downsides to web continuations

- **Cloning/serialization of variable context can be expensive**
- **State is handled by the server-side**
- **Tiny restrictions for the developer**
- **Careful with resource management** (transactions, locks, connections)

Agenda

- What are continuations?
- Continuations and the web
- **Existing Java tools**
- Other notable application domains

Existing Java tools

- **RIFE** : http://rifers.org
  - **Full-stack component framework to quickly and consistently develop and maintain Java web applications**
  - **Integrated web continuations**
  - **Write them in pure Java**
  - **Support for regular continuations and call/answer continuations**
  - **Stable and production-proven**
  - **Provides reusable continuations library for others** (eg. WebWork)

Existing Java tools

- **Cocoon** : http://cocoon.apache.org
  - **Web development framework built around the concepts of separation of concerns** (making sure people can interact and collaborate on a project, without stepping on each other toes) **and component-based web development.**
  - **Web continuations through Cocoon Flowscript**
  - **Stable version uses Rhino (JavaScript)**
  - **Seperate project JavaFlow provides continuations in pure Java** (http://jakarta.apache.org/commons/sandbox/javaflow/)

Existing Java tools

- **JauVM** : http://jauvm.blogspot.com
  - **JVM bytecode interpreter in Java that manages its own stack**
  - **Support for continuations through its “secondary” JVM**
  - **Uses JDK 1.5 annotations to mark “special” methods**
  - **Seems to not handle partial continuations**
  - **Documentation is scarce**

Existing Java tools

- **Dalma** : https://dalma.dev.java.net
  - **Continuations-based workflow engine**
  - **Integrates JavaFlow**
  - **Workflow in pure Java without XML**

Agenda

- What are continuations?
- Continuations and the web
- Existing Java tools
- **Other notable application domains**

Other notable application domains

- **Request thread parking**
  - **Introduced with Jetty 6** (http://jetty.mortbay.org/jetty6/)
  - **Suspend the request**
  - **Free the current thread**
  - **Perfect for Ajax**
  - **Instead of continuous polling,**
    resume continuation when new data is available

Other notable application domains

- **Discrete event simulation**
  - **No wait states for threads**
  - **No blocking while waiting for an event**
  - **Capture the state with a continuation**
  - **Reactivate processes when the approperiate conditions occur**

Questions