WEBAPPS FOR FUN AND PROFIT

Andrew Look,
Software Engineer

About Shopzilla

Shopzilla, Inc. - Online Shopping Network

shopzilla.com bizrate shopzilla publisher program

100M
impressions/day

8,000+
searches per second

20-29M
UV's per Month

100M+
Products

Project Background

- Monitoring web applications in production is difficult

Project Background

- Monitoring web applications in production is difficult
- Difficult to pinpoint the source of
  - Scalability problems
  - Usability problems

Usability

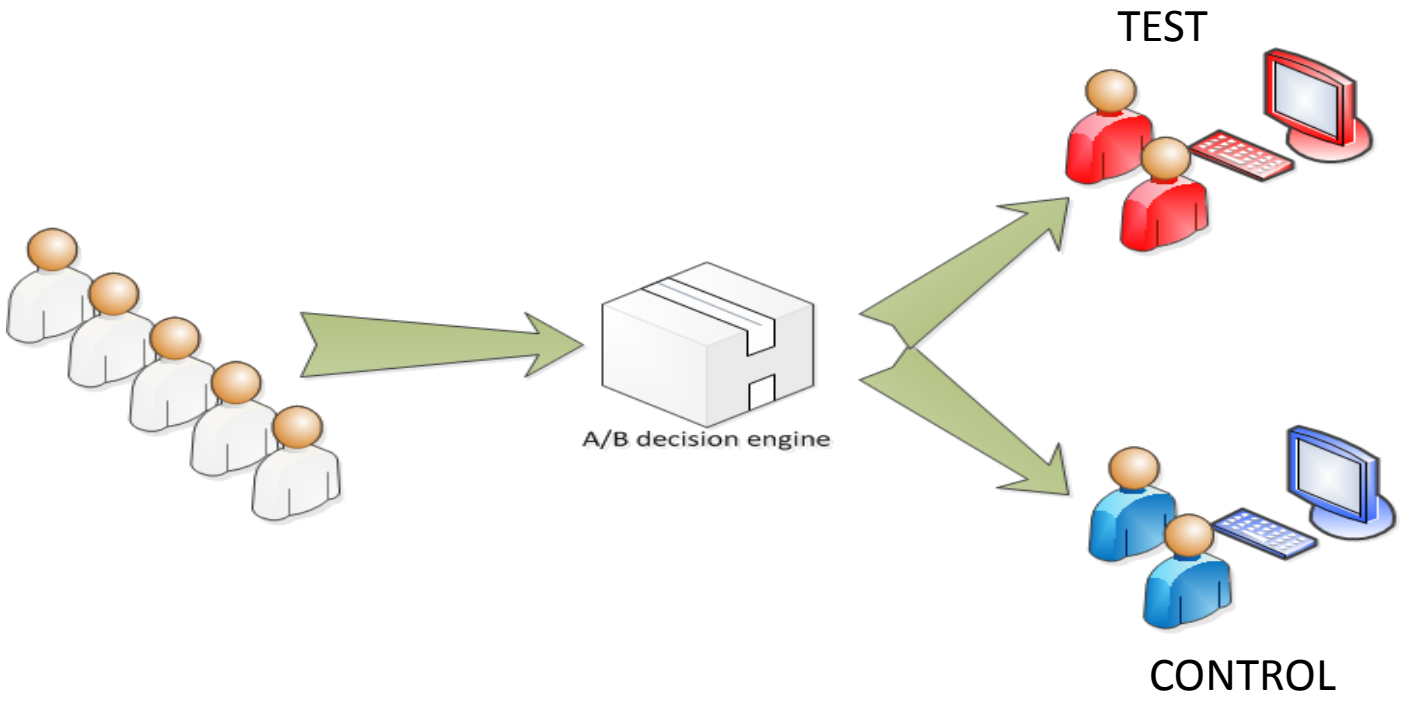
- We constantly develop new features
- What if a new feature…
  - makes the application less user-friendly?
  - prevents us from making money?

Usability

- How can we release these features without losing money?

Usability: A/B testing

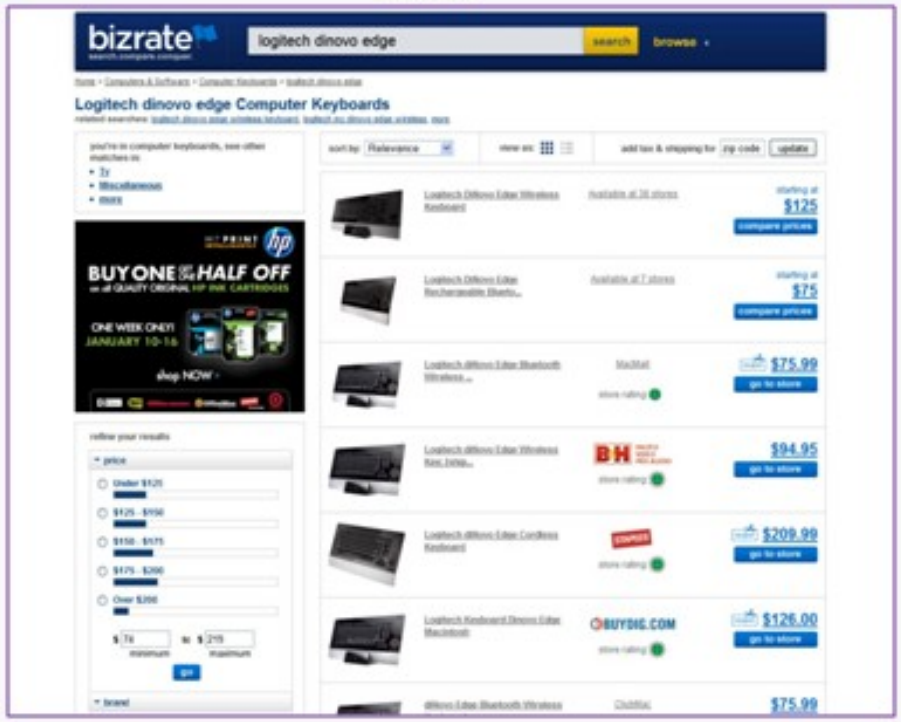
- How can we release these features without losing money?
  - Show the feature to a small “test group,” fraction of users
  - Compare the performance to the “control group”



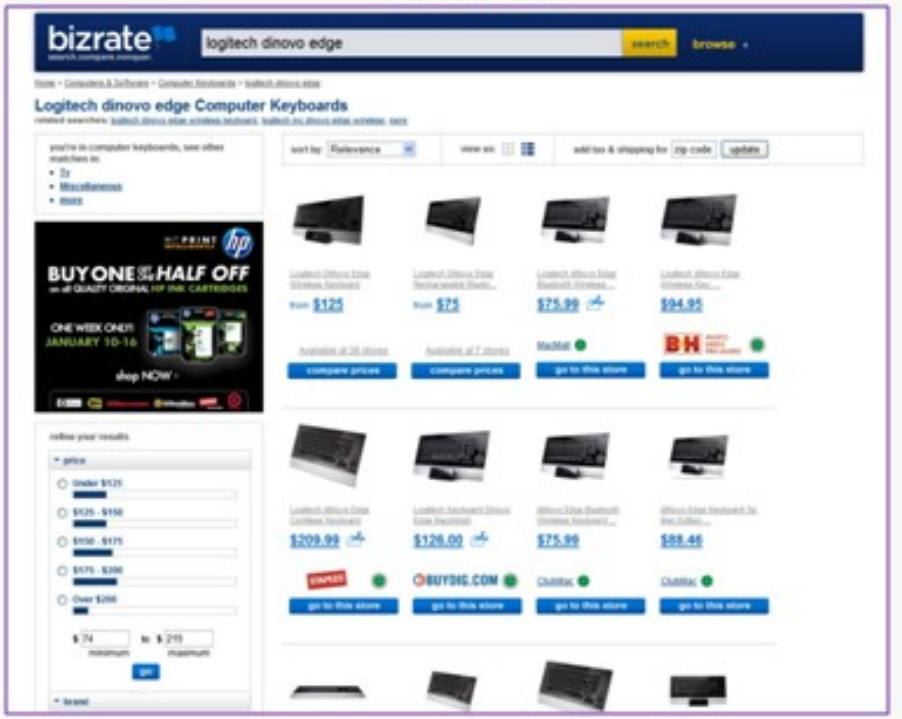
Usability: A/B testing

- What does an A/B test look like in production?

Control



Grid View Test



Usability: A/B testing

- ~~What does an A/B test look like in production?~~
- What can the CODE look like…?

```
//perspectiveConfigValues exposes values that have
//been turned on for particular test session ranges
boolean enableTest = perspectiveConfigValues.get("qvalue.test”);
if(!enableTest ) {
    SwapDetector detector = new SwapDetector(qValueAsKeywordHelper)
    keywordSwapped = detector.swapKeywordIfNecessary(command, request);
    if (keywordSwapped) {
        hotSearchCommand.setUsingSearchEngineQValue(true);
    }
}
```

Usability: A/B testing

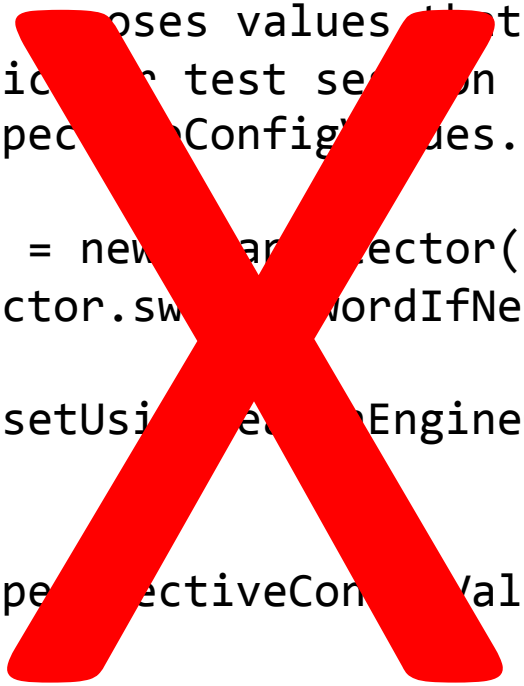
- ~~What does an A/B test look like in production?~~
- What can the CODE look like…?

```
//perspectiveConfigValues exposes values that have
//been turned on for particular test session ranges
boolean enableTest = perspectiveConfigValues.get("qvalue.test”);
if(!enableTest ) {
    SwapDetector detector = new SwapDetector(qValueAsKeywordHelper)
    keywordSwapped = detector.swapKeywordIfNecessary(command, request);
    if (keywordSwapped) {
        hotSearchCommand.setUsingSearchEngineQValue(true);
    }
}
boolean enableViewTest = perspectiveConfigValues.get(”view.test”);
if(!enableViewTest ) {
    enableGridView();
} else {
...
```

Usability: A/B testing

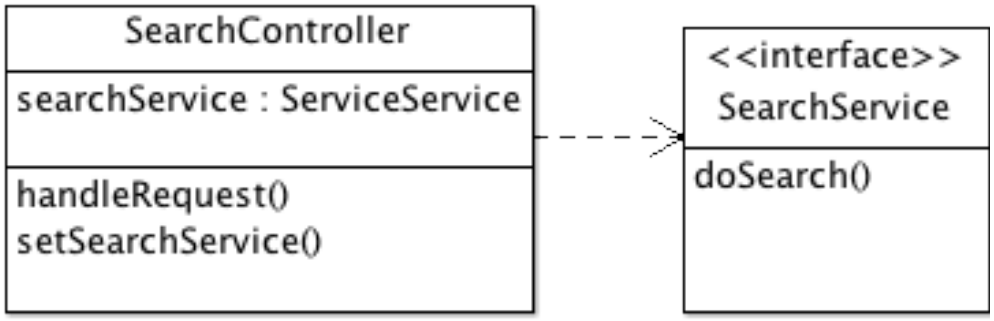
- ~~What does an A/B test look like in production?~~
- What can the CODE look like…?

```
//perspectiveConfigValues  oses values t have
//been turned on for partic  test se n ranges
boolean enableTest = perspec Config ues.get("qvalue.test”);
if(!enableTest ) {
    SwapDetector detector = new ar ector(qValueAsKeywordHelper)
    keywordSwapped = detector.sw wordIfNecessary(command, request);
    if (keywordSwapped) {
        hotSearchCommand.setUsi e EngineQValue(true);
    }
}
boolean enableViewTest = pe ectiveCon Values.get(”view.test”);
if(!enableViewTest ) {
    enableGridView();
} else {
...
```



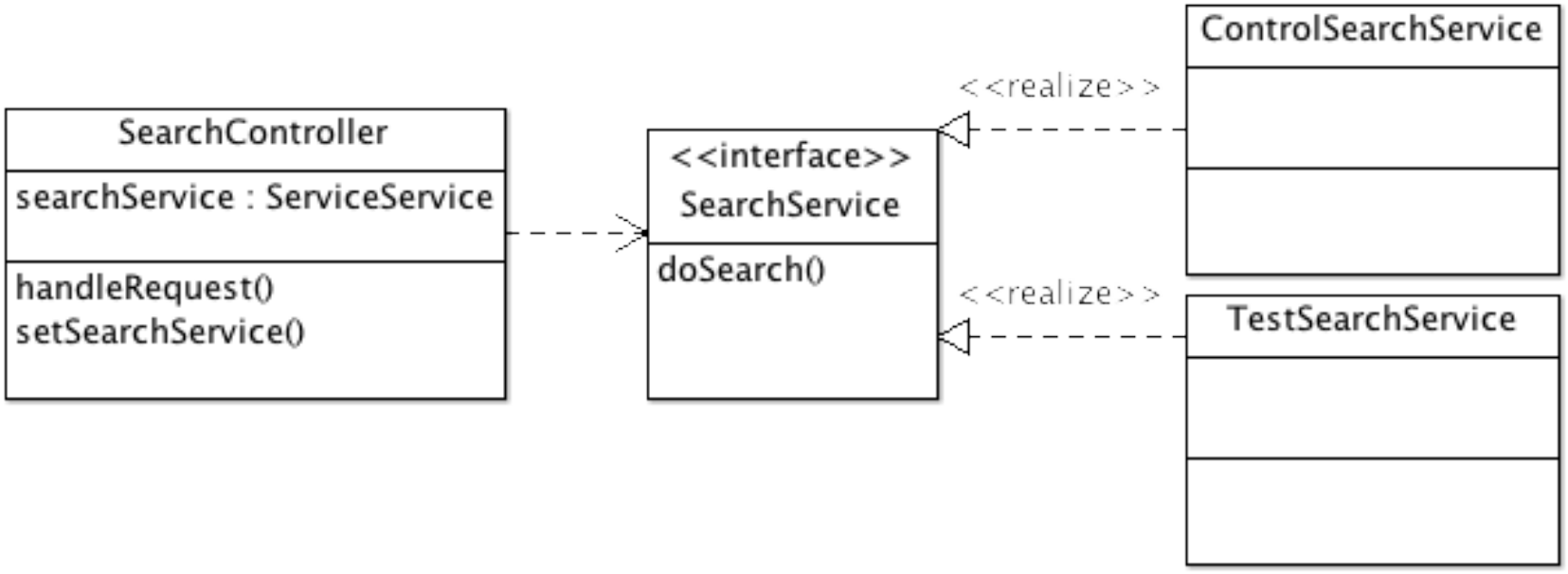
Usability: A/B testing

- Instead, let's take advantage of Object Oriented programming!
- Create an interface to abstract varying behavior



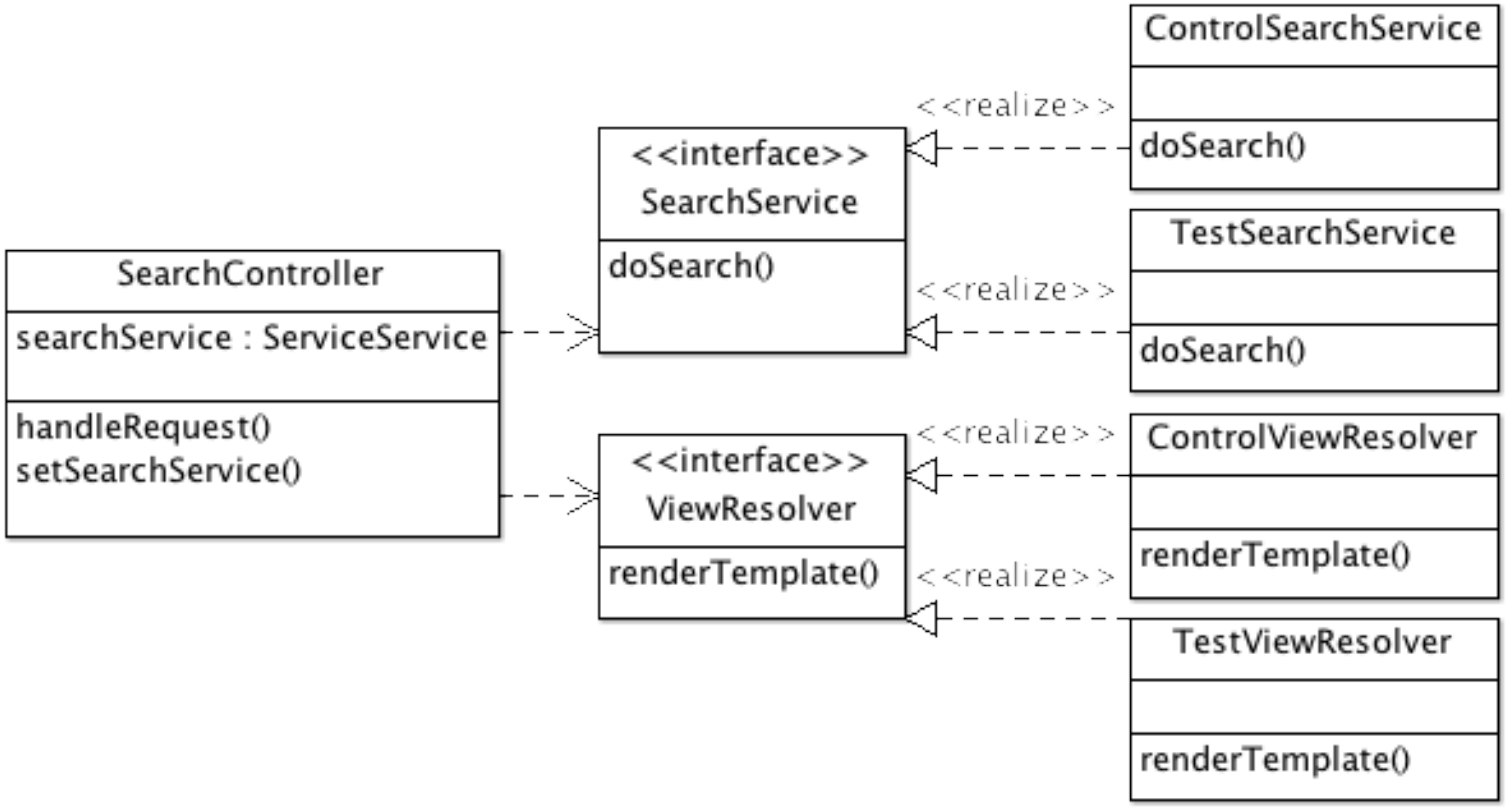
Usability: A/B testing

- Instead, let's take advantage of Object Oriented programming!
- Create an interface to abstract varying behavior



Usability: A/B testing

- Remove old tests by switching up our configuration: no code changes
- Add new tests without polluting business logic



Usability: Project

- Develop an abstract class for these testing interfaces to extend
- Make these interfaces request-aware
- Encapsulate logic of splitting requests in these testing interfaces
- Make them configurable, to switch them on/off without code changes
- Make them responsible for logging, so we can analyze the results
  - Then the application can focus on its logic
  - Without bending over backwards to support loads of tests

Usability

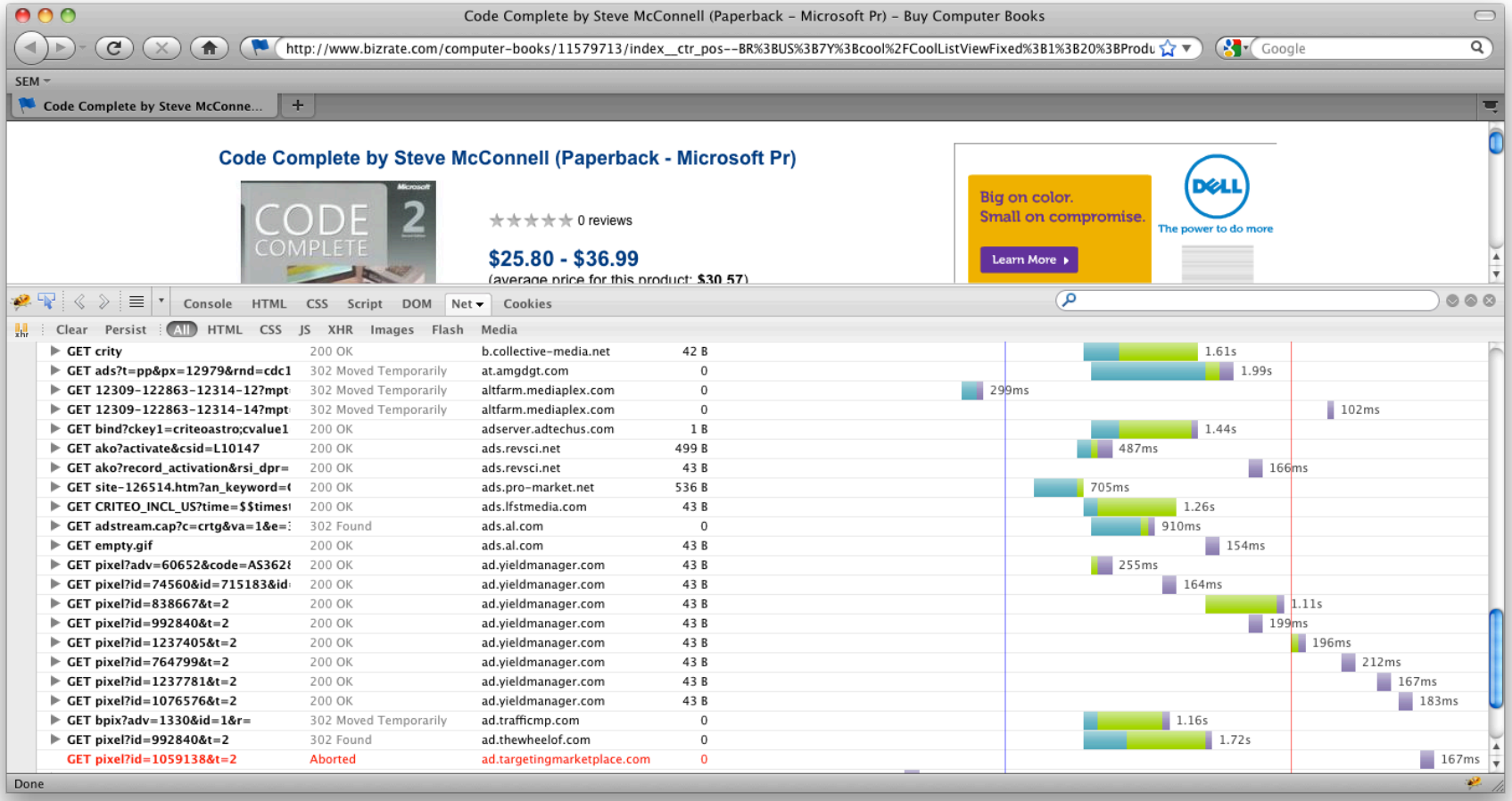
- Now that we know which features to develop…

Scalability

- Now that we know which features to develop…
- How do we make it available to millions of people?

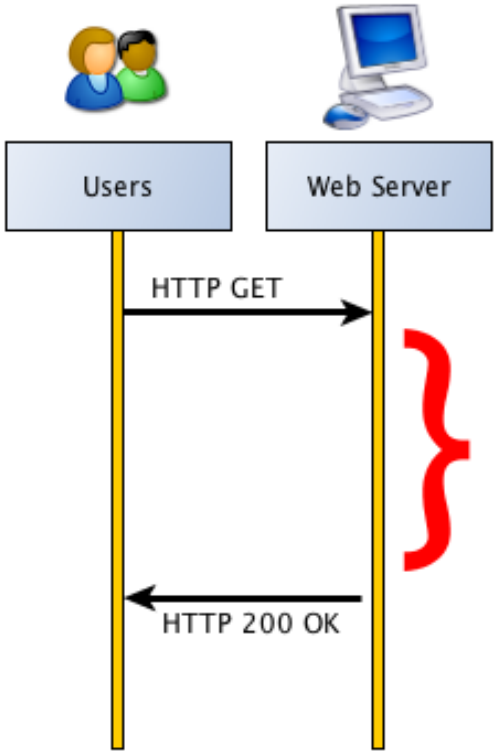
Scalability

- It’s easy to see:
  - How long it takes the client to render the server’s response



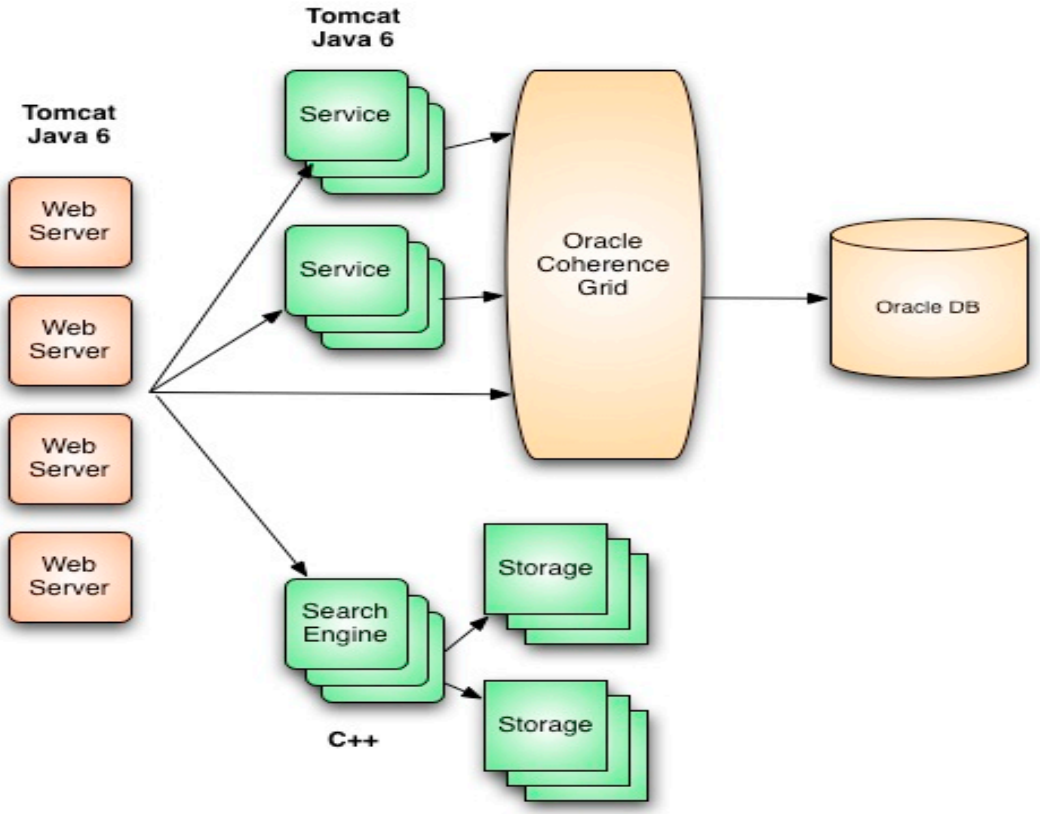
Scalability

- It's easy to see:
  - How long it takes the client to render the server's response
  - How long it takes the server to process a request



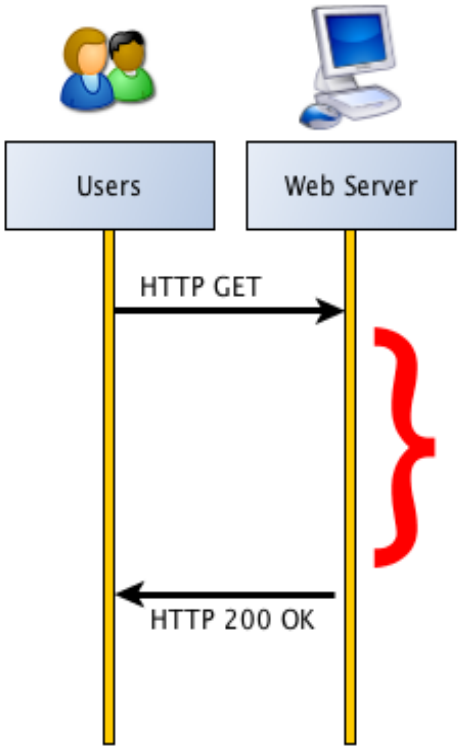
Scalability

- We would like to see
  - Why the server is taking so long



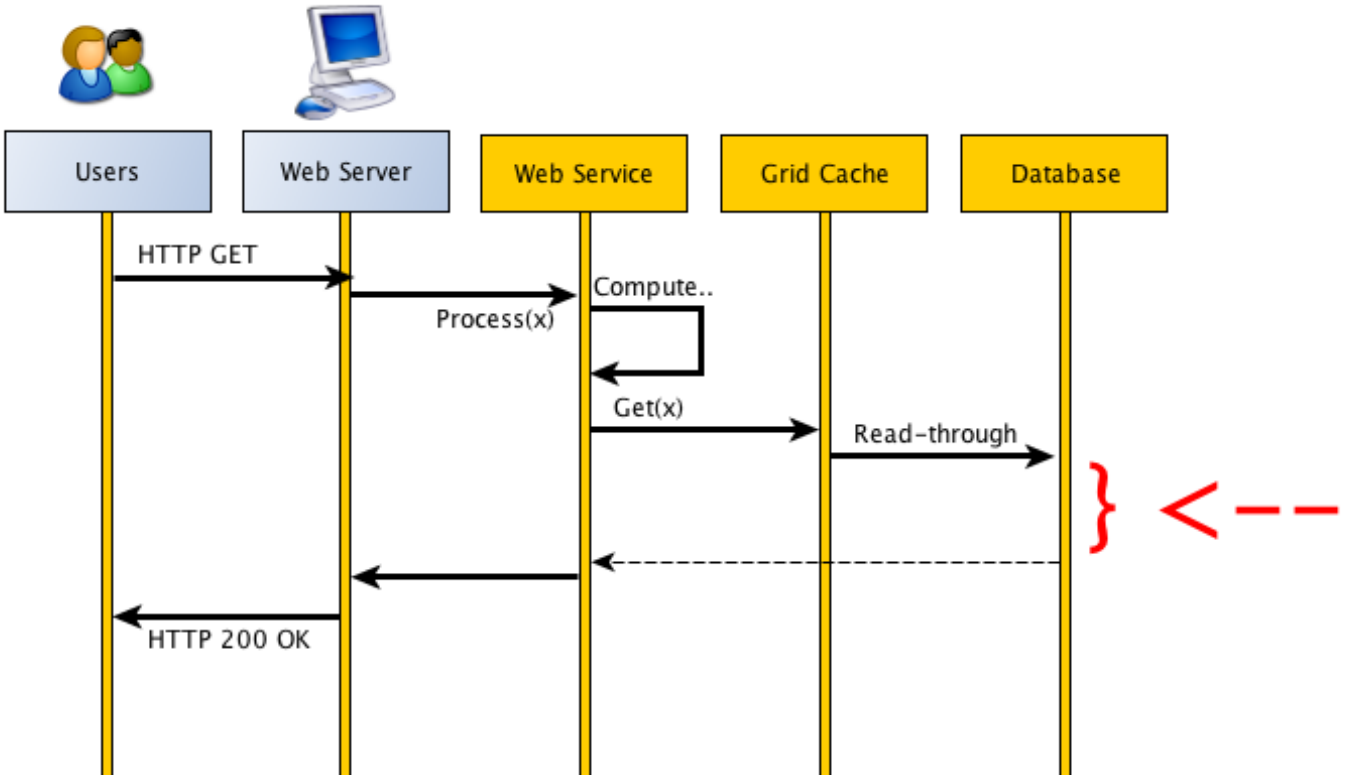
Scalability

- We would like to see
  - Why the server is taking so long



Scalability

- We would like to see
  - Why the server is taking so long
  - At the same fine granularity that we can look into client-side latency



Scalability: The Project

- We want a GRANULAR view into into application performance
- How?

Scalability: The Project

- We want a GRANULAR view into into application performance
- How?
  - By building a coding construct to let us to set “performance breakpoints”
  - By setting these breakpoints throughout our architecture stack
    - Web application
    - Web services
    - Cache accesses
    - Database accesses
  - By displaying these as a Gantt chart for finding bottlenecks

Project Details

Choose:

A. A/B Testing
- Create two different features
- Build a framework to run & analyze an A/B test

B. Performance
- Create a feature with different performance implications
- Build a framework to set performance breakpoints
- Display the performance chart in the browser as a debugging mechanism

Project Details

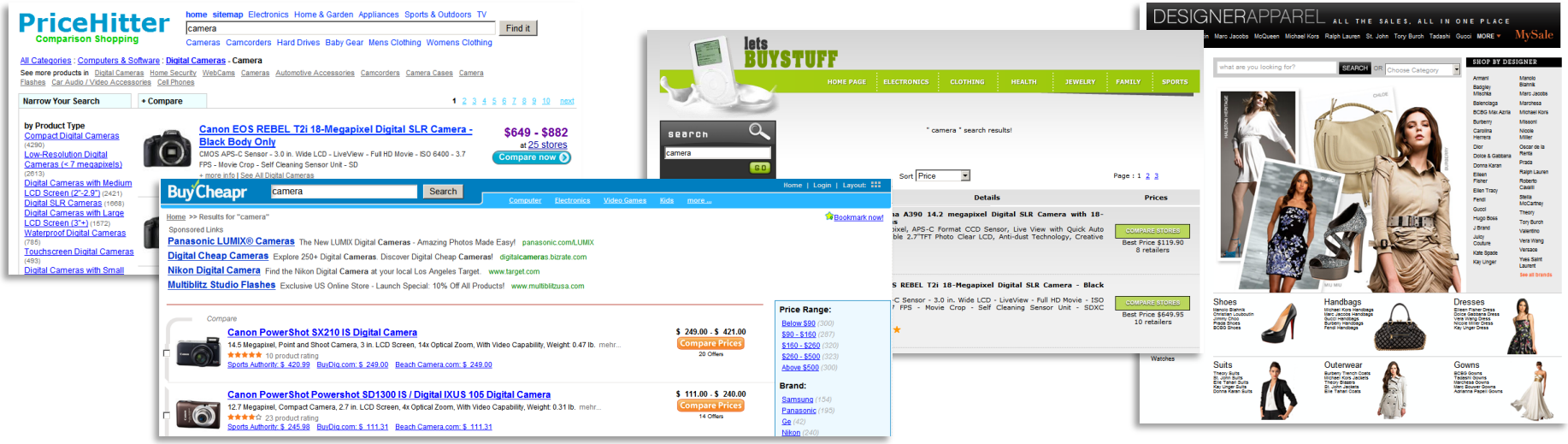
- Build a Java web application to optimize!
  - Open-source Shopzilla Catalog API Client library – get paid if your app gets hits!
  - Example webapp to serve as skeleton code
- Put it into production!
- Deploy it to the cloud – Amazon Elastic Beanstalk

Project Details

- Shopzilla Catalog API
  - The Shopzilla, Inc. Catalog API provides access to the Shopzilla, Inc. inventory of catalogued products, merchant offers, and merchant ratings & reviews content through a query-based RESTful web service, responses formatted in XML.
  - Managed through the Shopzilla Publisher Program, a CPC affiliate marketing program, publishers are able to monetize using this API. There are two versions available, each providing logic native to its corresponding Shopzilla, Inc. property: Bizrate.com or Beso.com.
    - Bizrate API: Provides access to the entire Shopzilla, Inc. offer universe, exposing content from 8,000+ merchants. You would use this version to create a full comparison shopping site.
    - Beso API: Provides the opportunity to create a style-based experience with a curated set of offers focused in soft goods categories. You would use this version to create a fashion-oriented shopping site.

Project Details

- Example Integrations
  - BuyCheapr http://www.buycheapr.com/us/
  - PriceHitter http://www.pricehitter.com/
  - Let's Buy Stuff http://letsbuystuff.com/
  - Designer Apparel http://www.designerapparel.com/



- More Info: http://www.programmableweb.com/api/shopzilla