Google™ 11 I O

High Performance GWT Architecting for Speed

David Chandler, John Labanca

May 10-11, 2011

Hashtags: #io2011 #DevTools

Feedback: http://goo.gl/xZ9da

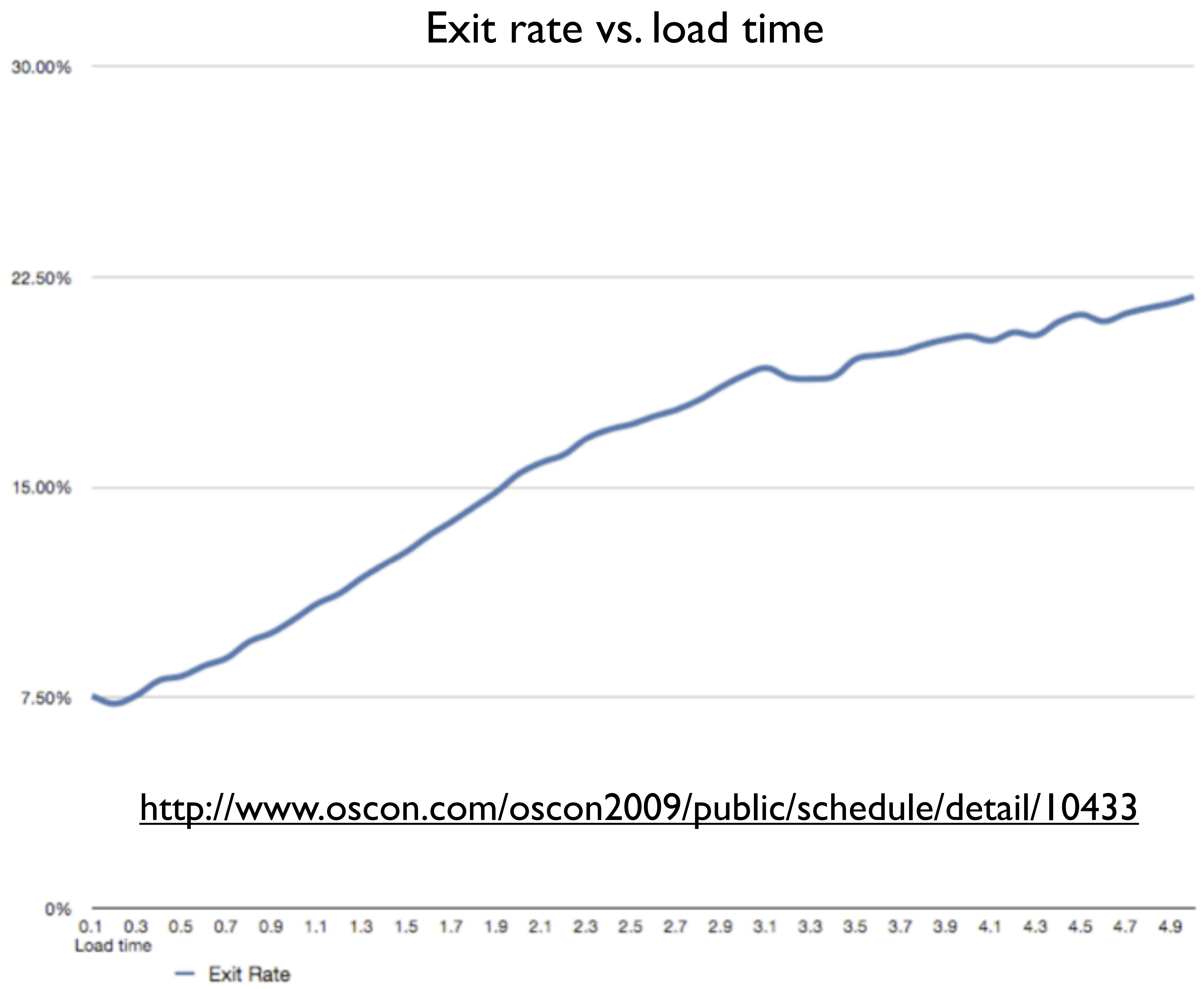
Agenda

- Why does speed matter?
- 5 performance pitfalls
- Cell widgets
- Code splitting with Activities and Places
- Compiler tips

Agenda

- **Why does speed matter?**
- 5 performance pitfalls
- Cell widgets
- Code splitting with Activities and Places
- Compiler tips

Slow site == bad



http://www.oscon.com/oscon2009/public/schedule/detail/10433

Why does speed matter?

- Once upon a time...
  - Google users wanted 30 search results instead of 10
  - Time to first results went from 0.4s to 0.9s (**+0.5s**)
  - First result page searches declined **25%** in 6 weeeks
  - That would be $2.5B drop in revenues!

Agenda

- Why does speed matter?
- **5 performance pitfalls**
- Cell widgets
- Code splitting with Activities and Places
- Compiler tips

#1 Don’t lose the user at startup

- HTTP requests are the slowest thing you can do in the browser
- Use ClientBundle to minimize trips for images, CSS
- Prefetch data needed at load time
  - Use dynamic host page (JSP, etc.) and write JS variables into the page
  - Read them with JSNI or the GWT Dictionary class
  - http://code.google.com/webtoolkit/articles/dynamic_host_page.html

#2 Don’t lock up the browser

- JavaScript is single threaded
- Use Scheduler.scheduleDeferred()
  - Runs after browser event loop
  - Keeps thread free to respond to events
  - Sometimes required as workaround to focus / layout issues
- For repetitive UI work
  - scheduleFixedPeriod() instead of a for loop
  - scheduleFinally()
    - executes before repaint / event loop
    - good for coalescing (ex: 5 RPC calls, only last one matters to UI)
    - can use to combine DOM operations to reduce flicker

#3 Don’t make two trips when one will do

- Every server trip adds latency
- With GWT-RPC, batch requests using Command pattern
  - Smart dispatcher can collate multiple calls to the same service (common at startup time)
  - http://turbomanage.wordpress.com/2010/07/12/caching-batching-dispatcher-for-gwt-dispatch/
  - http://turbomanage.wordpress.com/2010/07/16/dispatchqueue/
- RequestFactory can batch requests
  - within a service (GWT 2.3) and across services (GWT 2.4, see RequestContext.append())
  - requestContext.method1().to(new Receiver<T>(){...});
  - requestContext.method2().to(new Receiver<T>(){...});
  - requestContext.fire(new Receiver<Void>(){...}); //called only 1x

#4 Watch out for RPC type explosion

- GWT-RPC supports polymorphism
- Generates serializer / deserializer for each subtype
- List<Foo> as RPC argument or return type
  - Results in ArrayList, LinkedList, Stack, Vector, ...
  - Slows down compilation
- GWT-RPC recommendations
  - Prefer concrete types (ArrayList) to interfaces (List)
  - Limit use of polymorphism with GWT-RPC
  - Can blacklist RPC types (see issue 4438)
  - Consider RequestFactory instead
    - Will support polymorphism in a way that doesn't cause type explosions

#5 Don’t use a Widget when HTML will do

- Widgets have overhead
- Use UiBinder to replace Widgets with HTML
  - when don’t need to respond to events
  - or when events can be caught by a parent Widget
  - caution: can’t add Widgets to HTML elements, so leaf Widgets require a parent Widget hierarchy to the top
  - new LayoutPanels more efficient than previous panels
    - Layout mostly delegated to browser
    - Less use of tables (except TabLayoutPanel)
- For lists, tables, and trees
  - Use the new Cell widgets

Agenda

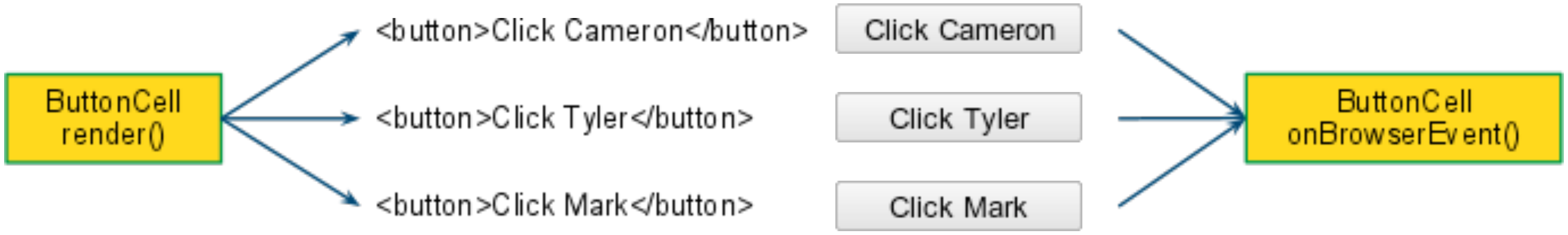
- Why does speed matter?
- 5 performance pitfalls
- **Cell widgets**
- Code splitting with Activities and Places
- Compiler tips

Cell Widgets

- What is a Cell?
- CellTable Overview
- CellTable Examples

Cell

- Cells are Widget flyweights
  - Render content as HTML strings
  - Handle events for multiple DOM instances
- Benefits
  - Decrease overhead versus widget
  - Render data sets as a single HTML string


ButtonCell
render()
<button>Click Cameron</button>
<button>Click Tyler</button>
<button>Click Mark</button>
Click Cameron
Click Tyler
Click Mark
ButtonCell
onBrowserEvent()

CellTable

- Render large data sets as a single HTML string
- Features
  - Paging / data push
  - Multiple row selection
  - Column sorting
  - Fixed column widths using natural layout
  - Keyboard navigation
- Planned Features
  - Fixed headers with scrollable data area
  - Fully customizable structure
    - Child rows, colspans, rowspans

CellTable

http://goo.gl/akoJL

Creating a CellTable

1. Create a CellTable widget
2. Add columns

Creating a CellTable

```
CellTable<Contact> table = new CellTable<Contact>();

// Add a text column to show the name.
TextColumn<Contact> nameColumn = new TextColumn<Contact>() {
  @Override public String getValue(Contact object) {
    return object.name;
  }
};
table.addColumn(nameColumn, "Name");

// Add a date column to show the birthday.
DateCell dateCell = new DateCell();
Column<Contact, Date> dateColumn = new Coumn<Contact, Date>(dateCell) {
  @Override public Date getValue(Contact object) {
    return object.birthday;
  }
};
table.addColumn(dateColumn, "Birthday");
```

Google 11 I O

Populating a CellTable

Static Data

```
List<Contact> myData = getMyData();
cellTable.setRowData(myData);
```

Populating a CellTable

```
// Create a data provider.
AsyncDataProvider<Contact> dataProvider = new
    AsyncDataProvider<Contact>(){
  @Override
  protected void onRangeChanged(HasData<Contact> display) {
    final Range range = display.getVisibleRange();
    service.requestRows(range, new AsyncCallback<List<Contact>>() {
      public void onSuccess(List<Contact> result) {
        updateRowData(range.getStart(), result);
      }
    });
  }
}

// Connect the table to the data provider.
dataProvider.addDataDisplay(cellTable);
```

Updating with a CellTable

```
dateColumn.setFieldUpdater(new FieldUpdater<Contact, Date>() {
  public void update(final int index, final Contact contact,
                     final Date newBirthday) {
    // Commit the change on the server.
    service.updateContact(contact, newBirthday,
      new AsyncCallback<Void>() {
        public void onSuccess() {
          // Update the local cache and redraw.
          contact.setBirthday(newBirthday);
          cellTable.redraw();
        }
    }
  }
}
```

Agenda

- Why does speed matter?
- 5 performance pitfalls
- Cell widgets
- **Code splitting with Activities and Places**
- Compiler tips

Activities and Places

- Introduced in GWT 2.1
- Helps you manage history / bookmarks / back button
- What does it have to do with MVP?
  - Strictly speaking, not a thing
  - But many MVP frameworks offer place / history mgmt along with Presenter, View concepts
- Demo trunk/samples/expenses

Place

- *Place* represents a bookmarkable state
- *PlaceController* makes back button / bookmarks work like users expect
- *PlaceTokenizers* map to / from String tokens on URL

Place

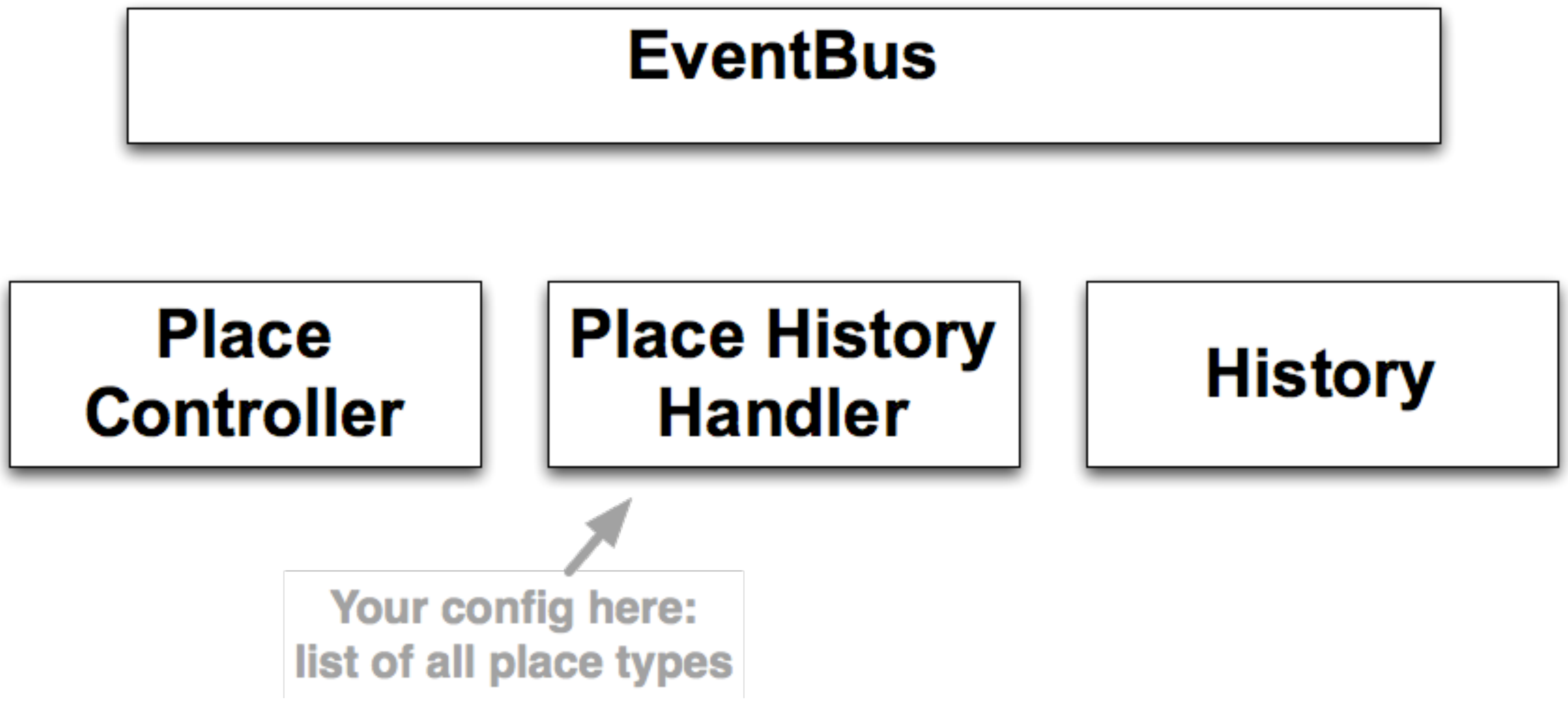
```
public class EditListPlace extends Place {
  private String token;

  public EditListPlace(String token) {
    this.token = token;
  }
  public String getToken() {
    return token;
  }
  public static class Tokenizer implements PlaceTokenizer<EditListPlace> {
    public EditListPlace getPlace(String token) {
      return new EditListPlace(token);
    }
    public String getToken(EditListPlace place) {
      return place.getToken();
    }
  }
}
```

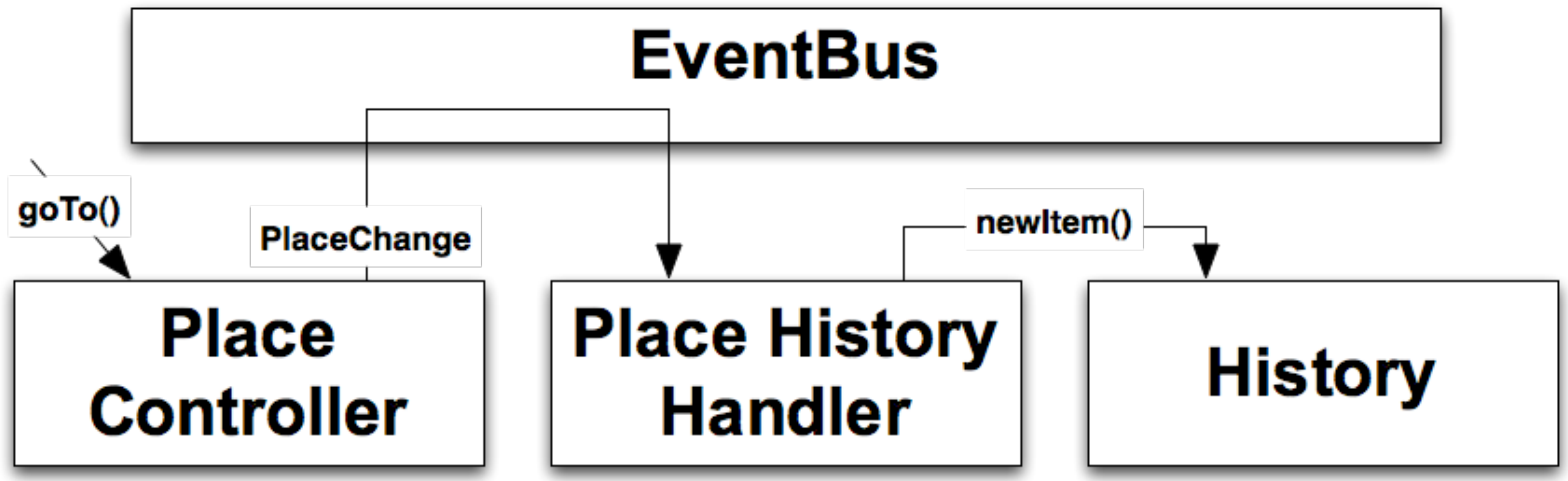
PlaceHistoryMapper

```
/**
 * PlaceHistoryMapper interface is used to attach all places which the
 * PlaceHistoryHandler should be aware of. This is done via the @WithTokenizers
 * annotation or by extending PlaceHistoryMapperWithFactory and creating a
 * separate TokenizerFactory.
 */
@WithTokenizers({ ListsPlace.Tokenizer.class, EditListPlace.Tokenizer.class })
public interface AppPlaceHistoryMapper extends PlaceHistoryMapper
{
}
```

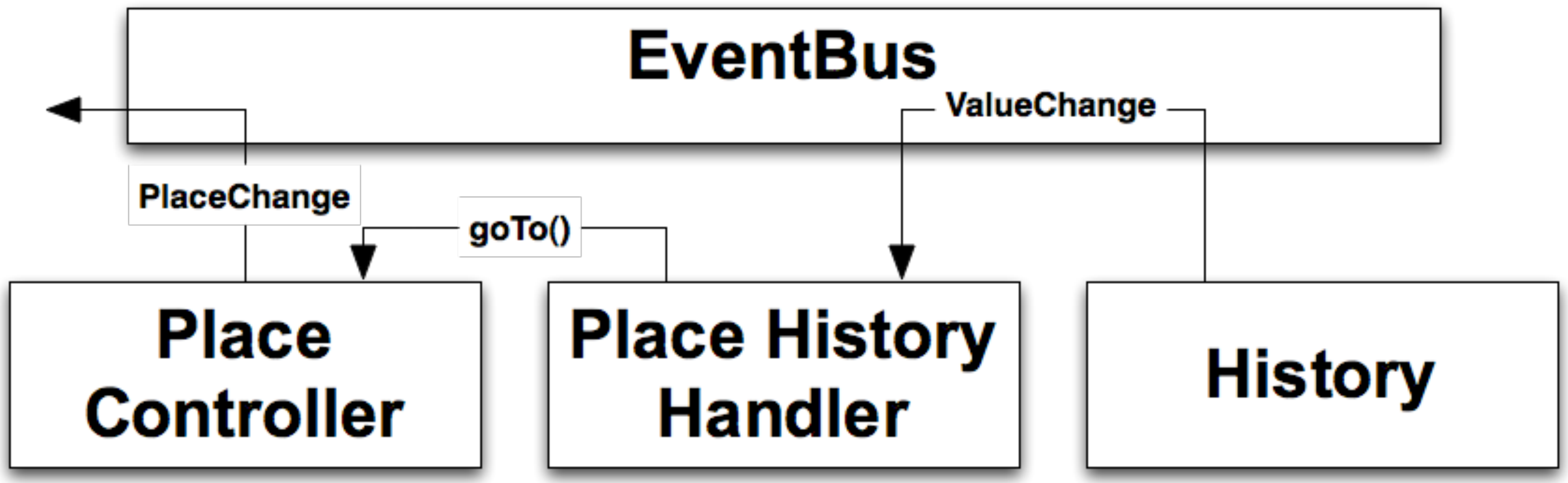
Places: moving parts



Places: Go to


EventBus
goTo()
PlaceChange
newItem()
Place Controller
Place History Handler
History

Places: Back and forth



Activity

- Something the user is doing
- “wake up, set up, show up”
- Can automatically warn users before leaving
- Started / stopped by *ActivityManager* (per panel)
- Instantiates view (or obtains from factory)
- Can be a presenter, but higher level
- Can be associated with a Place

Activity

```
public class EditListActivity extends AbstractActivity
{
   private EventBus eventBus;

   public EditListActivity(EditListPlace editListPlace)
   {
      this.itemListToken = editListPlace.getToken();
   }

   @Override
   public void start(final AcceptsOneWidget panel, EventBus eventBus)
   {
      this.eventBus = eventBus;
      panel.setWidget(new EditListView());
   }
}
```

ActivityMapper

```java
public class AppActivityMapper implements ActivityMapper {

   @Override
   public Activity getActivity(Place place) {
      if (place instanceof EditListPlace) {
         return new EditListActivity((EditListPlace) place);
      }
      if (place instanceof ListsPlace)
      {
         return new ListsActivity();
      }
      return null;
   }
}
```

ActivityMapper idioms

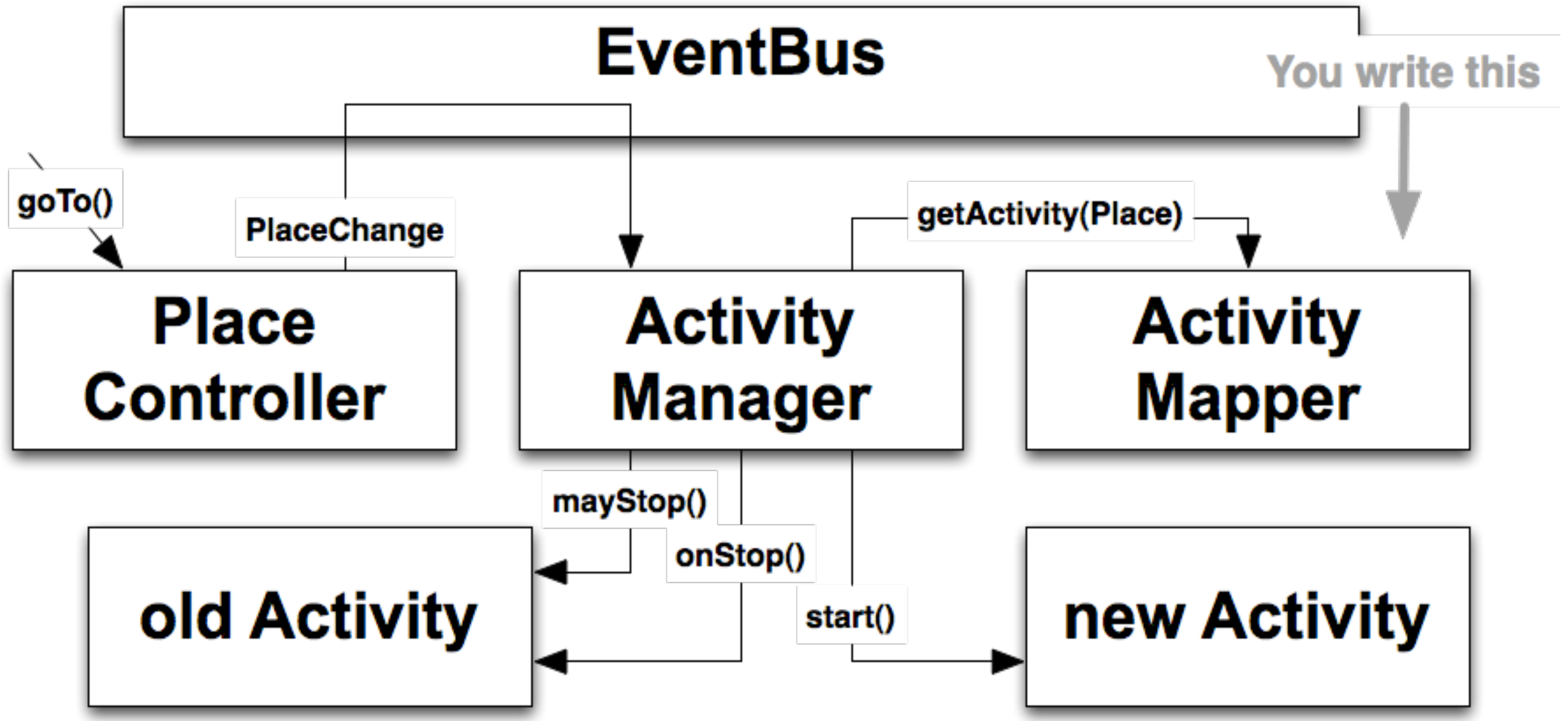
Disposable Activity, reusable view (makes for clean code)

```
if (place instanceof FooPlace) {
  return new FooActivity(theOnlyFooView);
}
```

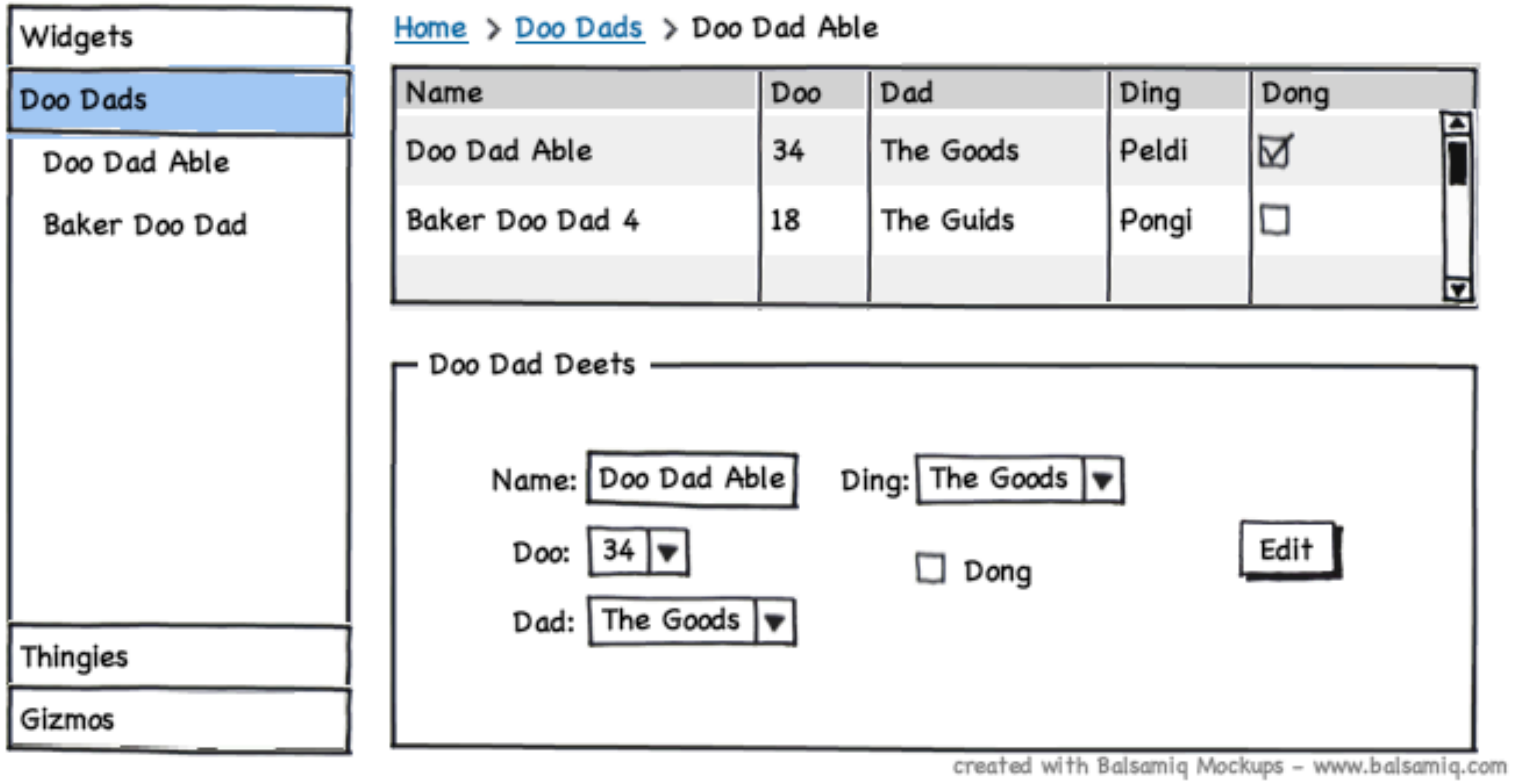
Singleton Activity (Activity cleanup required, little perf benefit to reuse)

```
if (place instanceof FooPlace) {
  theOnlyFooActivity.update((FooPlace) place);
  return theOnlyFooActivity;
}
```

Using Places and Activities together



Strategies



Widgets

Doo Dads

Doo Dad Able

Baker Doo Dad

Thingies

Gizmos

Home > Doo Dads > Doo Dad Able

Name	Doo	Dad	Ding	Dong
Doo Dad Able	34	The Goods	Peldi	☑
Baker Doo Dad 4	18	The Guids	Pongi	☐

Doo Dad Deets

Name: Doo Dad Able

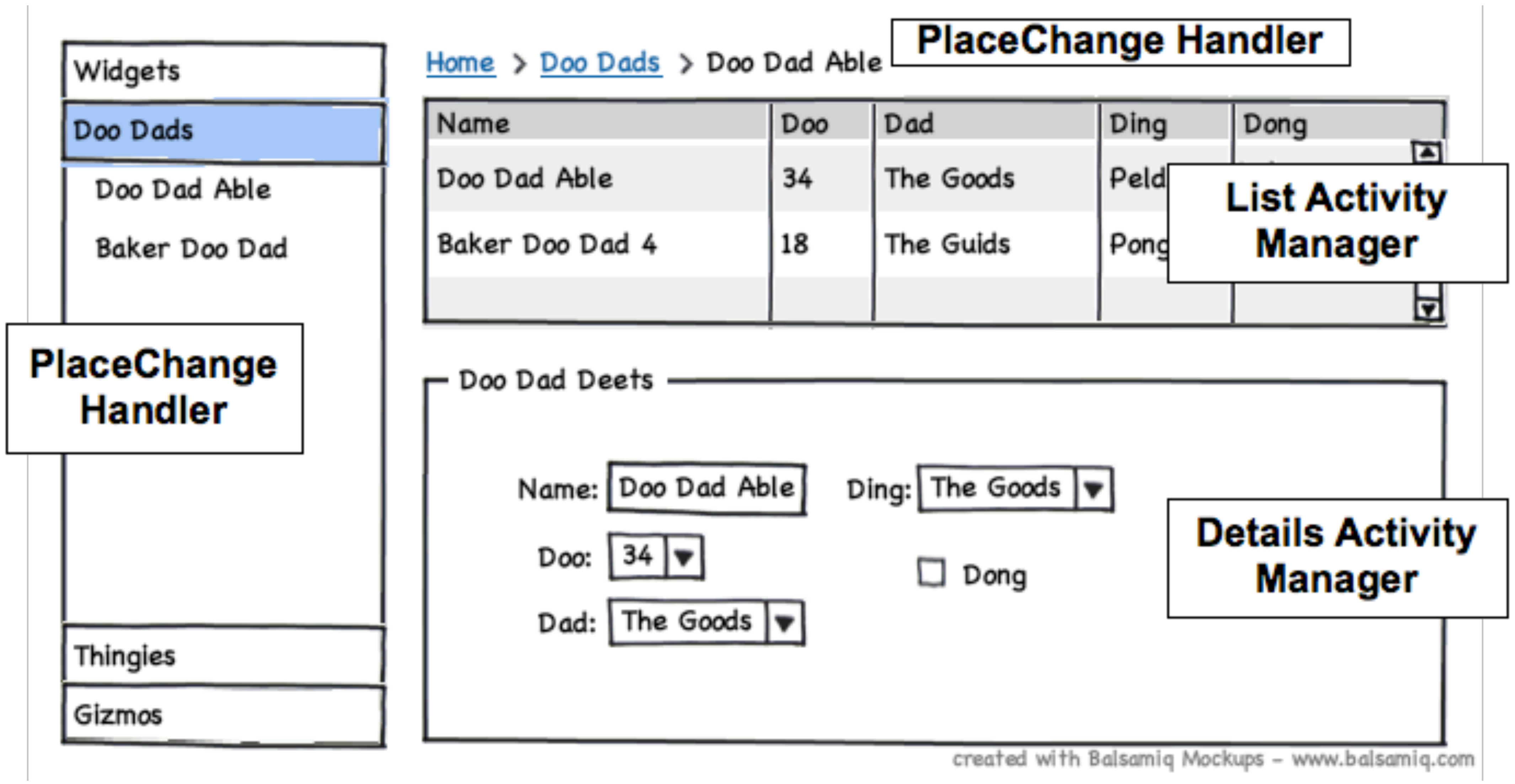
Doo: 34

Dad: The Goods

Ding: The Goods

☐ Dong

Strategies



Widgets
Doo Dads
Doo Dad Able
Baker Doo Dad
Thingies
Gizmos
Home > Doo Dads > Doo Dad Able
PlaceChange Handler
Name
Doo
Dad
Ding
Dong
Doo Dad Able
34
The Goods
Peld
Baker Doo Dad 4
18
The Guids
Pong
List Activity Manager
PlaceChange Handler
Doo Dad Deets
Name: Doo Dad Able
Ding: The Goods
Doo: 34
Dong
Dad: The Goods
Details Activity Manager
created with Balsamiq Mockups - www.balsamiq.com

Strategies

- How to update multiple regions in response to Place change?
- Each region has its own
  - ActivityManager
  - ActivityMapper
- onPlaceChange
  - all ActivityManagers get notified
  - activityMapper.getActivity(Place place) gets called for each ActivityManager
  - resulting Activities each update their regions

Paradigms

- Places are disposable
- Activities may be, too
- Views
  - could be re-created in response to each Place/Activity change
  - but more efficient to construct once and
  - obtain from a factory (or DI) in the Activity
- Significant performance benefit to reusing views, especially complex ones

Code splitting

- Allows you to defer code download until needed

```
GWT.runAsync(new RunAsyncCallback() {
  @Override
  public void onSuccess() {
   // Deferred code goes here
  }
  @Override
  public void onFailure(Throwable reason) {
    // TODO Auto-generated method stub
  }
});
```

- See also GWT's AsyncProxy
- -compileReport
  – Look in /extras dir for soycReport (Story Of Your Compile)

Code splitting with Activities and Places

- An Activity is a natural split point
  - easy to understand
  - not too big, not too small
  - well proven on Google projects
- With GIN
  - Use AsyncProvider to create your activities
  - one possibility: getActivity(Place p) returns activityAsyncProvider.get()
  - GIN generates the runAsync call for you
- Without GIN
  - Activity start() method is a good place for the runAsync block, basic idea is to proxy the method through GWT's AsyncProxy or similar
  - work in progress, watch issue 5129, see also http://goo.gl/s59w4, http://goo.gl/2881K

Agenda

- Why does speed matter?
- 5 performance pitfalls
- Cell widgets
- Code splitting with Activities and Places
- **Compiler tips**

Compile faster

- The problem: large GWT projects can take several minutes to compile
- -draftCompile
  - Skip optimizations (not for production)
- Set only one user-agent in gwt.xml
  - no need for all permutations during development
  - `<set-property name="user.agent" value="safari"/>`
- Reminder: avoid RPC type explosion

Compile faster: the numbers

- Spirodraw app (12 classes, no RPC)

	All browsers	Safari only
-compileReport	49.9s	30.5s
Standard compile	43.3s	27.7s
-draftCompile	35.2s	24.1s

Shrink JS (compiler flags)

- -XdisableClassMetadata
  - Disables some java.lang.Class methods (e.g. getName())
- -XdisableCastChecking
  - Disables run-time checking of cast operations
- Careful!
  - if you were using the features you disable, you'll get JS exceptions
  - compiler will not warn you
  - instanceof will still work
- -compileReport (SOYC)
  - "story of your compile" in /extra dir

Shrink JS (gwt.xml params)

```
<set-property name="compiler.stackMode" value="strip"/>
```

Removes client-side stack trace info (can reduce size up to 15%)

```
<set-configuration-property name="compiler.enum.obfuscate.names" value="true"/>
```

(only use if you’re not using enums as String values)

```
<set-configuration-property name="CssResource.obfuscationPrefix" value="empty"/>
```

See also

- GWT FAQ
- CompilerParameters.gwt.xml

Shrink JS: the numbers

- Spirodraw app, minimal casting, 1 enum

Compiler options	Bytes	Percent
NONE	283,187	Shrinkage
-XdisableClassMetadata	276,218	2.5%
-XdisableCastChecking	280,196	1.1%
Stack stripping	272,518	3.8%
Enum obfuscation	282,233	0.3%
ALL	261,705	7.6%

Summary

- Why does speed matter?
- 5 performance pitfalls
- Cell widgets
- Code splitting with Activities and Places
- Compiler tips

Thank you!

http://code.google.com

Hashtags: #io2011 #DevTools

Feedback: http://goo.gl/xZ9da