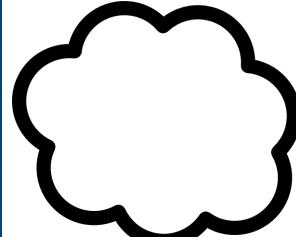
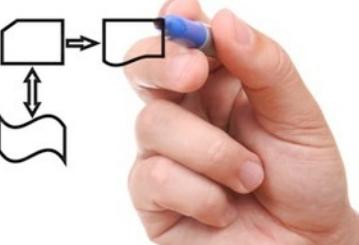


San Jose State University

CMPE 281

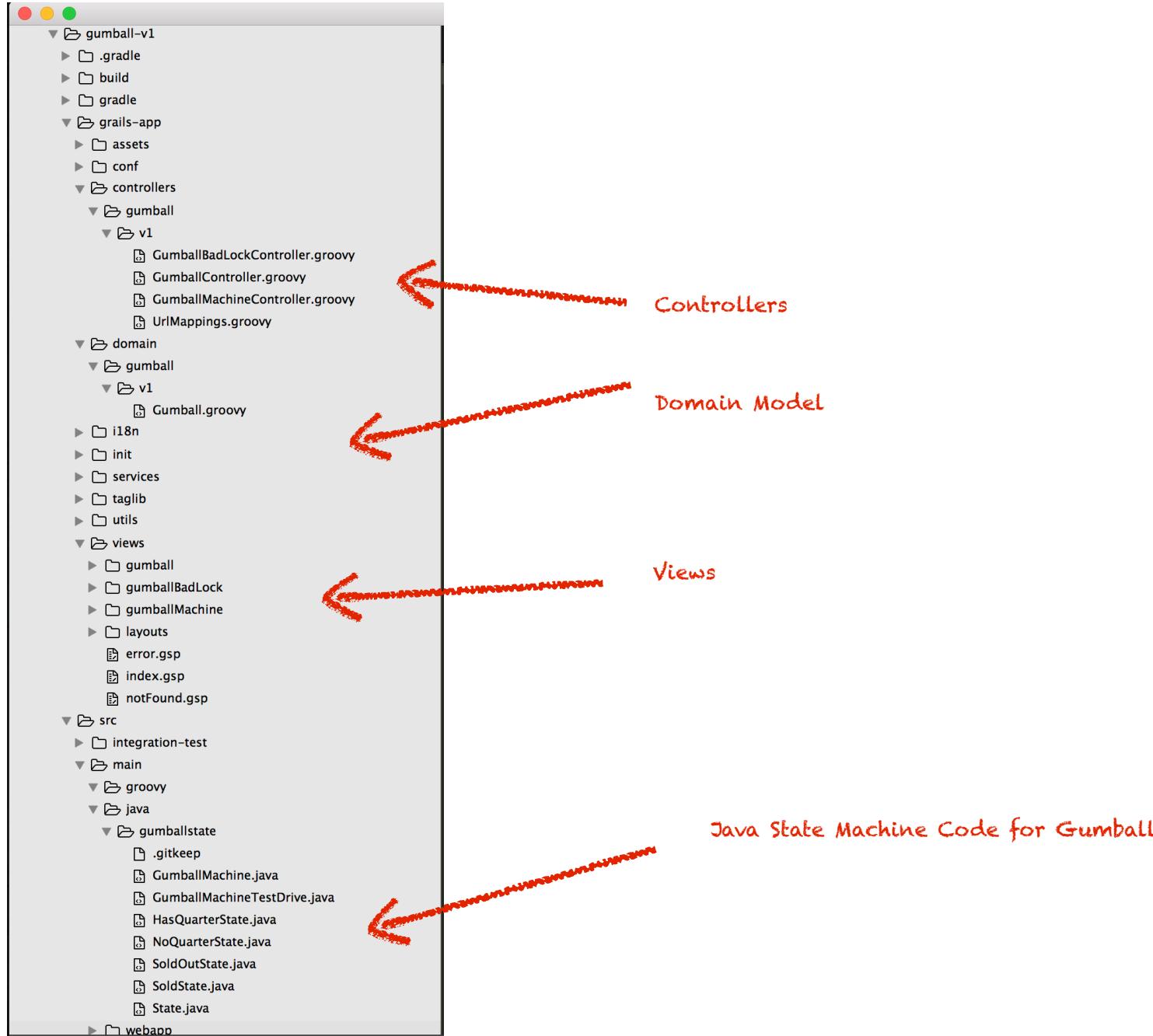
Grails Gumball Lab





Gumball Machine (v1)

Grails Gumball Machine



Running Locally

Welcome to the Gumball Machine

Mighty Gumball, Inc.
Java-enabled Standing Gumball
Model# null
Serial# null
Inventory: 5 gumballs
Machine is waiting for quarter



Insert Quarter Turn Crank

localhost:8080/gumballMachine/index

NOTES ADMIN BOOKS WEB SJSU COURSES LABS EDU CODE API DATA GO JS TOOLS REFS

```
macbox:ios Files pnguyen$ cd
macbox:~ pnguyen$ ls
ADMIN Desktop ECLIPSE Movies PROJECT SANDBOX SUBLIME
Applications Documents GITHUB Music Pictures SCRIPTS SOURCE
Books Downloads Library PHOTOS Public
macbox:~ pnguyen$ ls
ADMIN Desktop ECLIPSE Movies PROJECT SANDBOX SUBLIME
Applications Documents GITHUB Music Pictures SCRIPTS SOURCE
Books Downloads Library PHOTOS Public
macbox:~ pnguyen$ cd SOURCE/
macbox:SOURCE pnguyen$ ls
202 bitbucket data docker
281 cloud design games
macbox:SOURCE pnguyen$ cd 281
macbox:281 pnguyen$ ls
aws bios docker grails models mysql nodejs postman restlet spring xmlrpc
macbox:281 pnguyen$ cd grails/
macbox:grails pnguyen$ ls
README.md gumball-v1 gumball-v3 hellograils
groovy gumball-v2 hellobooks hellorest
macbox:grails pnguyen$ cd gumball-v1
macbox:gumball-v1 pnguyen$ ls
Dockerfile build gradle gradle.bat grailsw
Makefile build.gradle gradle.properties grails-app grails-wrapper.jar grailsw.bat
ZREADME.md docker.sh gradlew src
macbox:gumball-v1 pnguyen$ make run
echo Starting Grails at: http://localhost:8080
Starting Grails at: http://localhost:8080
| Running application...
objc[431]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (0x101c414c0) and /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/libInstrument.dylib (0x101d094e0). One of the two will be used. Which one is undefined.
Grails application running at http://localhost:8080 in environment: development
```

MySQL Config

The screenshot shows a code editor window with the title "application.yml — sjsu". The left pane displays a file tree of a Grails project structure:

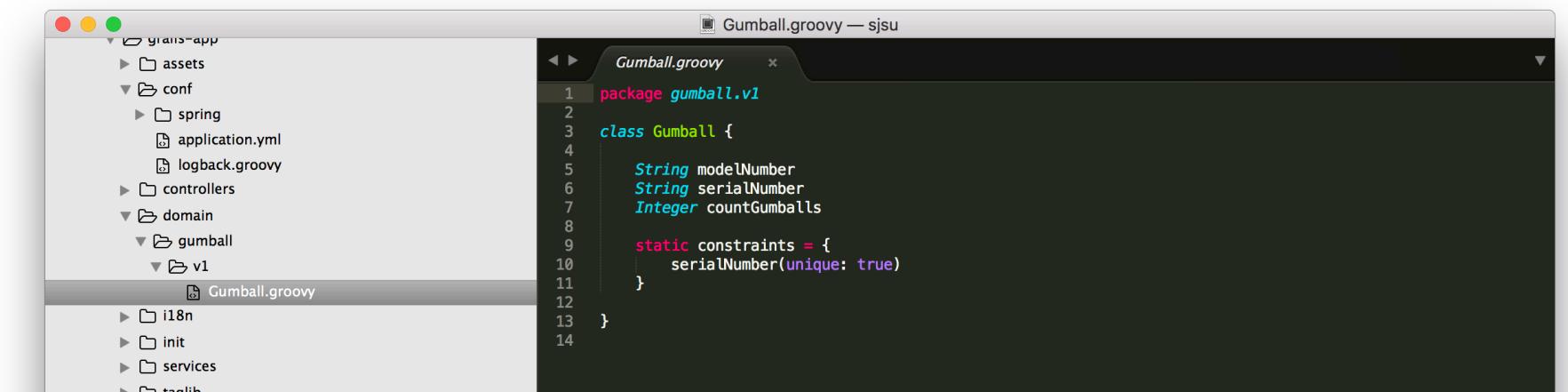
- 202
- 281
- aws
- bios
- docker
- grails
 - groovy
 - gumball-v1
 - .gradle
 - build
 - gradle
 - grails-app
 - assets
 - conf
 - spring
 - application.yml
 - logback.groovy
 - controllers
 - domain
 - i18n
 - init
 - services
 - taglib
 - utils
 - views
 - src
 - .gitignore
 - build.gradle
 - docker.sh
 - Dockerfile
 - gradle.properties
 - gradlew
 - gradlew.bat
 - grails-wrapper.jar
 - grailsw
 - grailsw.bat
 - Makefile
 - ZREADME.md
 - gumball-v2
 - gumball-v3
 - hellobooks
 - hellograils
 - helloworld
 - README.md
 - models

The right pane contains the content of the "application.yml" file, which is a YAML configuration for a MySQL database connection:

```
scriptlets: html
taglib: none
staticparts: none
endpoints:
jmx:
unique-names: true
hibernate:
cache:
queries: false
use_second_level_cache: true
use_query_cache: false
region.factory_class: org.hibernate.cache.ehcache.SingletonEhCacheRegionFactory
dataSource:
pooled: true
jmxExport: true
driverClassName: org.h2.Driver
username: sa
password:
environments:
development:
dataSource:
dbCreate: create-drop
url: jdbc:h2:mem:devDb;MVCC=TRUE;LOCK_TIMEOUT=10000;DB_CLOSE_ON_EXIT=FALSE
test:
dataSource:
dbCreate: update
driverClassName: com.mysql.jdbc.Driver
dialect: org.hibernate.dialect.MySQLInnoDBDialect
username: admin
password: mysql
url: jdbc:mysql://127.0.0.1:3306/cmpe281
production:
dataSource:
dbCreate: none
url: jdbc:h2:./prodDb;MVCC=TRUE;LOCK_TIMEOUT=10000;DB_CLOSE_ON_EXIT=FALSE
properties:
jmxEnabled: true
initialSize: 5
maxActive: 50
minIdle: 5
maxIdle: 25
maxWait: 10000
maxAge: 600000
timeBetweenEvictionRunsMillis: 5000
minEvictableIdleTimeMillis: 60000
validationQuery: SELECT 1
validationQueryTimeout: 3
validationInterval: 15000
testOnBorrow: true
testWhileIdle: true
testOnReturn: false
jdbcInterceptors: ConnectionState
defaultTransactionIsolation: 2 # TRANSACTION_READ_COMMITTED
```

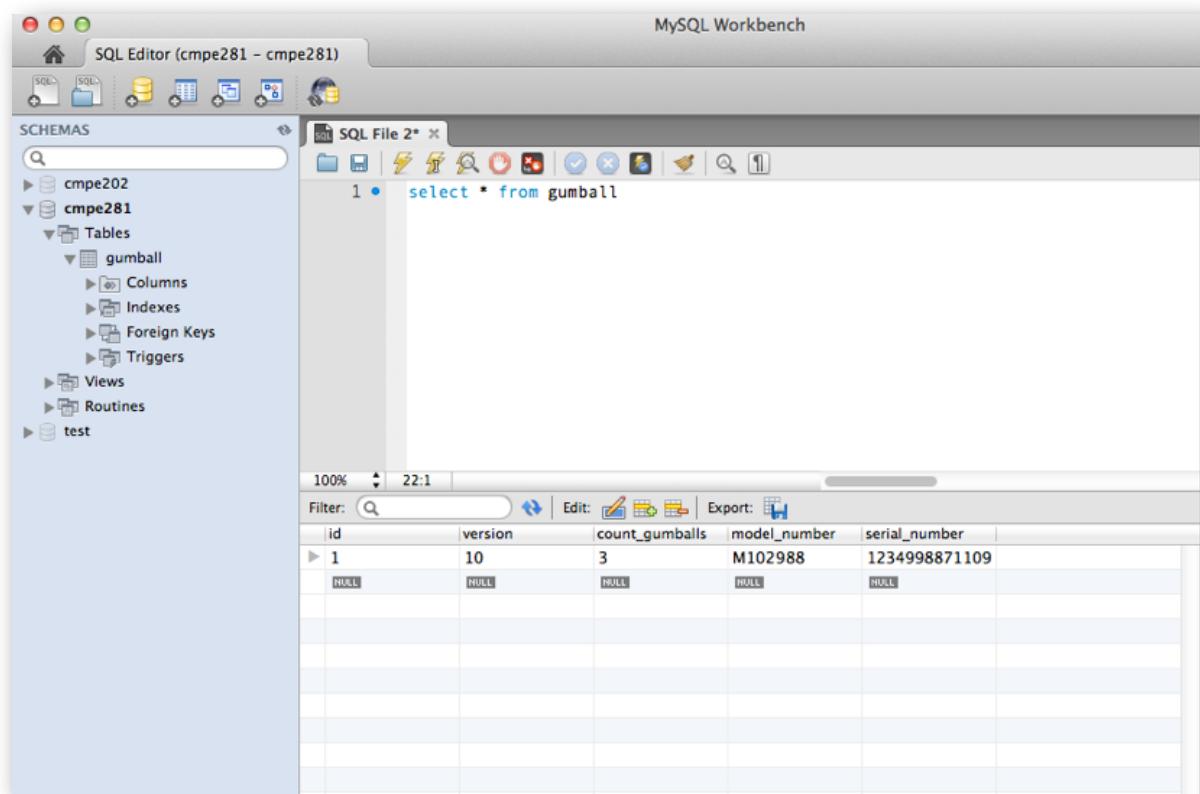
At the bottom of the editor, there are status indicators: "Line 1, Column 1", "Spaces: 4", "YAML", and "YAML".

Domain Model



The screenshot shows a code editor window titled "Gumball.groovy — sjsu". The file contains Groovy code defining a class named Gumball. The code includes fields for modelNumber, serialNumber, and countGumballs, and a static constraints block that makes serialNumber unique.

```
1 package gumball.v1
2
3 class Gumball {
4
5     String modelNumber
6     String serialNumber
7     Integer countGumballs
8
9     static constraints = {
10         serialNumber(unique: true)
11     }
12
13 }
14
```



The screenshot shows the MySQL Workbench interface. On the left, the "SCHEMAS" tree view shows databases cmpe202, cmpe281, and test. The cmpe281 database is selected, and its "Tables" section shows a table named "gumball". The "Columns" tab of the "gumball" table is selected, displaying columns: id, version, count_gumballs, model_number, and serial_number. In the center, the "SQL Editor" tab has the query "select * from gumball" entered. Below the editor, the results pane shows a single row of data:

id	version	count_gumballs	model_number	serial_number
1	10	3	M102988	1234998871109

Domain Model Scaffolding

The screenshot displays a Mac OS X desktop environment with several open windows:

- File Browser:** Shows the directory structure of a Grails application named "grails-app". The "controllers" folder contains a "gumball" folder which has a "v1" folder. Inside "v1", there are four Groovy files: GumballBadLockController.groovy, GumballController.groovy (which is selected), GumballMachineController.groovy, and UrlMappings.groovy. Other folders like "assets" and "domain" are also present.
- Code Editor:** An IDE window titled "GumballController.groovy — sjsu" shows the following Groovy code:

```
1 package gumball.v1
2
3 class GumballController {
4
5     def scaffold = Gumball
6
7 }
```
- Browser:** A Safari window titled "Create Gumball" is open at the URL "localhost:8080/gumball/create". The address bar shows the same URL. The page header includes a back button, a search bar, and a menu bar with various links like "NOTES", "ADMIN", "BOOKS", "WEB", "SJSU", "COURSES", "LABS", "EDU", "CODE", "API", "DATA", "GO", "JS", "TOOLS", and "REFS".
- Grails Application:** Below the browser is the Grails application interface. It features a green header with the "Grails" logo. The main content area is titled "Create Gumball" and contains three form fields:
 - Serial Number ***: An input field with a red border.
 - Model Number ***: An input field with a red border.
 - Count Gumballs ***: An input field with a red border and a dropdown arrow.

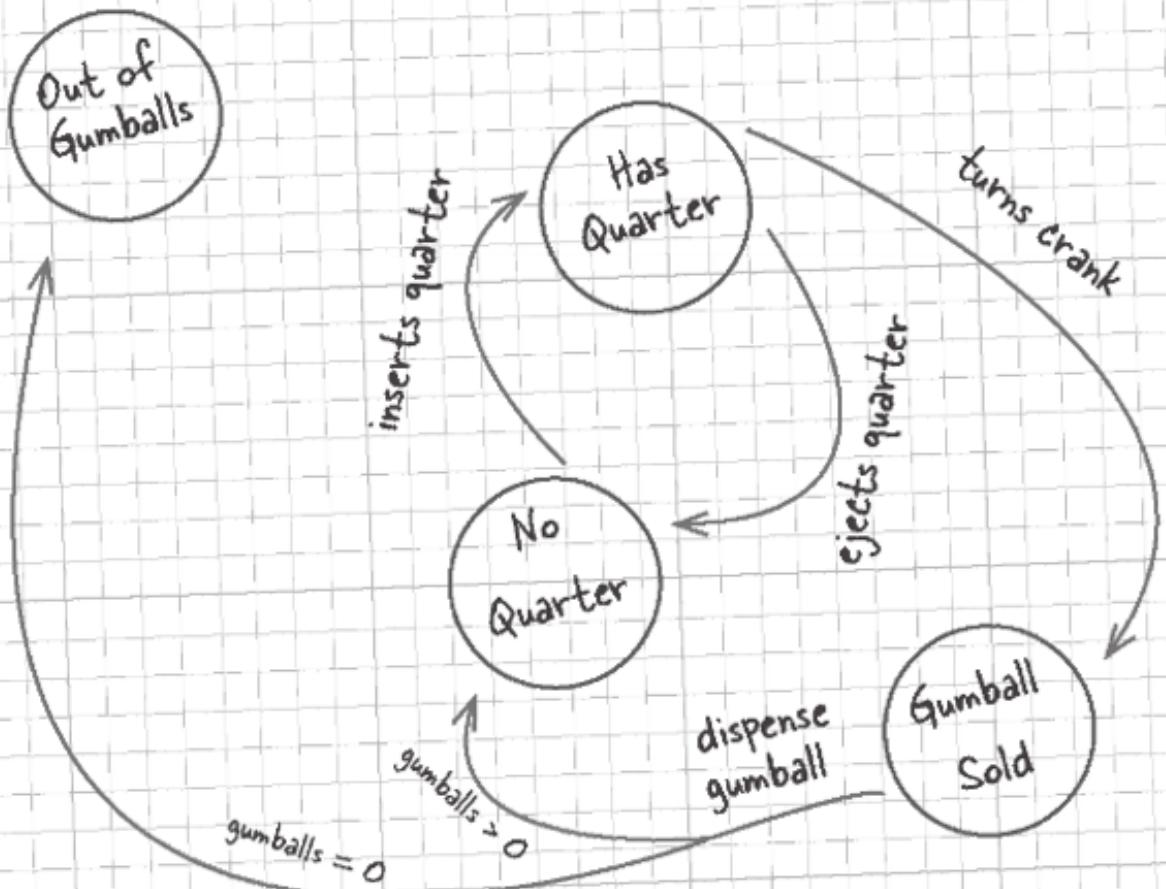


Mighty Gumball, Inc.

Where the Gumball Machine
is Never Half Empty

Here's the way we think the gumball machine controller needs to work. We're hoping you can implement this in Java for us! We may be adding more behavior in the future, so you need to keep the design as flexible and maintainable as possible!

- Mighty Gumball Engineers



Java State Machine

```
First we need to implement the State interface.  
public class NoQuarterState implements State {  
    GumballMachine gumballMachine;  
  
    public NoQuarterState(GumballMachine gumballMachine) {  
        this.gumballMachine = gumballMachine;  
    }  
  
    public void insertQuarter() {  
        System.out.println("You inserted a quarter");  
        gumballMachine.setState(gumballMachine.getHasQuarterState());  
    }  
  
    public void ejectQuarter() {  
        System.out.println("You haven't inserted a quarter");  
    }  
  
    public void turnCrank() {  
        System.out.println("You turned, but there's no quarter");  
    }  
  
    public void dispense() {  
        System.out.println("You need to pay first");  
    }  
}
```

We get passed a reference to the Gumball Machine through the constructor. We're just going to stash this in an instance variable.

If someone inserts a quarter, we print a message saying the quarter was accepted and then change the machine's state to the HasQuarterState.

You'll see how these work in just a sec...

You can't get money back if you never gave it to us!

And, you can't get a gumball if you don't pay us.

We can't be dispensing gumballs without payment.

Java State Machine

```
public class GumballMachine {  
  
    State soldOutState;  
    State noQuarterState;  
    State hasQuarterState;  
    State soldState;  
  
    State state = soldOutState;  
    int count = 0;  
  
    public GumballMachine(int numberGumballs) {  
        soldOutState = new SoldOutState(this);  
        noQuarterState = new NoQuarterState(this);  
        hasQuarterState = new HasQuarterState(this);  
        soldState = new SoldState(this);  
  
        this.count = numberGumballs;  
        if (numberGumballs > 0) {  
            state = noQuarterState;  
        }  
    }  
  
    public void insertQuarter() {  
        state.insertQuarter();  
    }  
  
    public void ejectQuarter() {  
        state.ejectQuarter();  
    }  
  
    public void turnCrank() {  
        state.turnCrank();  
        state.dispense();  
    }  
  
    void setState(State state) {  
        this.state = state;  
    }  
}
```

Here are all the States again...

...and the State instance variable.

The count instance variable holds the count of gumballs – initially the machine is empty.

Our constructor takes the initial number of gumballs and stores it in an instance variable.

It also creates the State instances, one of each.

If there are more than 0 gumballs we set the state to the NoQuarterState.

Now for the actions. These are VERY EASY to implement now. We just delegate to the current state.

Note that we don't need an action method for dispense() in GumballMachine because it's just an internal action; a user can't ask the machine to dispense directly. But we do call dispense() on the State object from the turnCrank() method.

This method allows other objects (like our State objects) to transition the machine to a different state.

Groovy calling Java

The screenshot shows a Mac OS X desktop environment with a terminal window and a code editor window. The code editor is displaying a Java file named `GumballMachine.java`. The terminal window has the command `grails-app` in its title bar and contains the output of a command, likely related to the project structure shown in the code editor.

The code editor displays the following Java code:

```
1 package gumballstate;
2
3 public class GumballMachine {
4
5     State soldOutState;
6     State noQuarterState;
7     State hasQuarterState;
8     State soldState;
9
10    State state = soldOutState;
11    int count = 0;
12
13    String modelNumber ;
14    String serialNumber ;
15
16    public GumballMachine(int numberGumballs) {
17        soldOutState = new SoldOutState(this);
18        noQuarterState = new NoQuarterState(this);
19        hasQuarterState = new HasQuarterState(this);
20        soldState = new SoldState(this);
21
22        this.count = numberGumballs;
23        if (numberGumballs > 0) {
24            state = noQuarterState;
25        }
26    }
27
28    public void insertQuarter() {
29        state.insertQuarter();
30    }
31
32    public void ejectQuarter() {
33        state.ejectQuarter();
34    }
35
36    public void turnCrank() {
37        state.turnCrank();
38        state.dispense();
39    }
40
41    void setState(State state) {
42        this.state = state;
43    }
44
45    void releaseBall() {
46        System.out.println("A gumball comes rolling out the slot...");
47        if (count != 0) {
48            count = count - 1;
49        }
50    }
51
52    int getCount() {
53        return count;
54    }
55
56    void refill(int count) {
57        this.count = count;
58        state = noQuarterState;
59    }
60}
```

The terminal window shows the following output:

```
grails-app
```

Groovy calling Java

The screenshot shows a Mac OS X desktop environment. On the left, a file browser window titled "grails-app" displays the project structure. On the right, a code editor window titled "GumballMachineController.groovy" shows Groovy code interacting with Java classes.

File Browser (grails-app structure):

- assets
- conf
- controllers
 - gumball
 - v1
 - GumballBadLockController.groovy
 - GumballController.groovy
 - GumballMachineController.groovy
 - UrlMappings.groovy
 - domain
 - i18n
 - init
 - services
 - taglib
 - utils
 - views

src
 - integration-test
 - main
 - groovy
 - java
 - gumballstate
 - .gitkeep
 - GumballMachine.java
 - GumballMachineTestDrive.java
 - HasQuarterState.java
 - NoQuarterState.java
 - SoldOutState.java
 - SoldState.java
 - State.java
 - webapp
 - mysql
 - test
 - .gitignore
 - build.gradle
 - docker.sh
 - Dockerfile
 - gradle.properties
 - gradlew
 - gradlew.bat
 - grails-wrapper.jar
 - grailsw
 - grailsw.bat
 - Makefile
 - ZREADME.md

Code Editor (GumballMachineController.groovy):

```
1 package gumball.v1
2
3 import gumballstate.GumballMachine
4
5
6 class GumballMachineController {
7
8     def String machineSerialNum = "1234998871109"
9     def GumballMachine gumballMachine
10
11    def index() {
12
13        def sessionid = session.id
14        println ( "Session ID: $sessionid" );
15
16        if (request.method == "GET") {
17
18            // search db for gumball machine
19            def gumball = Gumball.findBySerialNumber( machineSerialNum )
20            if ( gumball ) {
21
22                // create a default machine
23                gumballMachine = new GumballMachine(gumball.countGumballs)
24                gumballMachine.setModelNumber(gumball.modelNumber)
25                gumballMachine.setSerialNumber(gumball.serialNumber)
26                System.out.println(gumballMachine)
27            } else {
28
29                // create a default machine
30                gumballMachine = new GumballMachine(5);
31                System.out.println(gumballMachine)
32            }
33
34            // save in the session
35            session.machine = gumballMachine
36
37            // report a message to user
38            flash.message = gumballMachine.toString()
39            flash.sessionid = session.id ;
40
41            // display view
42            render(view: "index")
43
44        } else if (request.method == "POST") {
45
46            // dump out request object
47            request.each { key, value ->
48                println( "request: $key = $value")
49            }
50
51            // dump out params
52            params7.each { key, value ->
53                println( "params: $key = $value" )
54            }
55
56            // get machine from session
57            ...
58        }
59    }
60}
```

Bottom status bar: 5 lines, 305 characters selected | Spaces: 4 | Grails Controller (Groovy)

Gumball Machine Controller

The screenshot shows a Mac OS X desktop environment with a terminal window and a code editor.

Terminal:

```
grails-app
├── assets
├── conf
└── controllers
    └── gumball
        └── v1
            ├── GumballBadLockController.groovy
            ├── GumballController.groovy
            ├── GumballMachineController.groovy
            └── UrlMappings.groovy
```

Code Editor:

The code editor displays the file `GumballMachineController.groovy`. The code implements a controller for a gumball machine, handling GET and POST requests to either return a default machine or dump the request object and parameters respectively. It also saves the machine to the session and renders a view.

```
package gumball.v1

import gumballstate.GumballMachine

class GumballMachineController {

    def String machineSerialNum = "1234998871109"
    def GumballMachine gumballMachine

    def index() {
        def sessionid = session.id
        println( "Session ID: $sessionid" )

        if (request.method == "GET") {
            // search db for gumball machine
            def gumball = Gumball.findBySerialNumber( machineSerialNum )
            if (gumball) {
                // create a default machine
                gumballMachine = new GumballMachine(gumball.countGumballs)
                gumballMachine.setModelNumber(gumball.modelNumber)
                gumballMachine.setSerialNumber(gumball.serialNumber)
                System.out.println(gumballMachine)
            } else {
                // create a default machine
                gumballMachine = new GumballMachine(5);
                System.out.println(gumballMachine)
            }

            // save in the session
            session.machine = gumballMachine

            // report a message to user
            flash.message = gumballMachine.toString()
            flash.sessionid = session.id ;

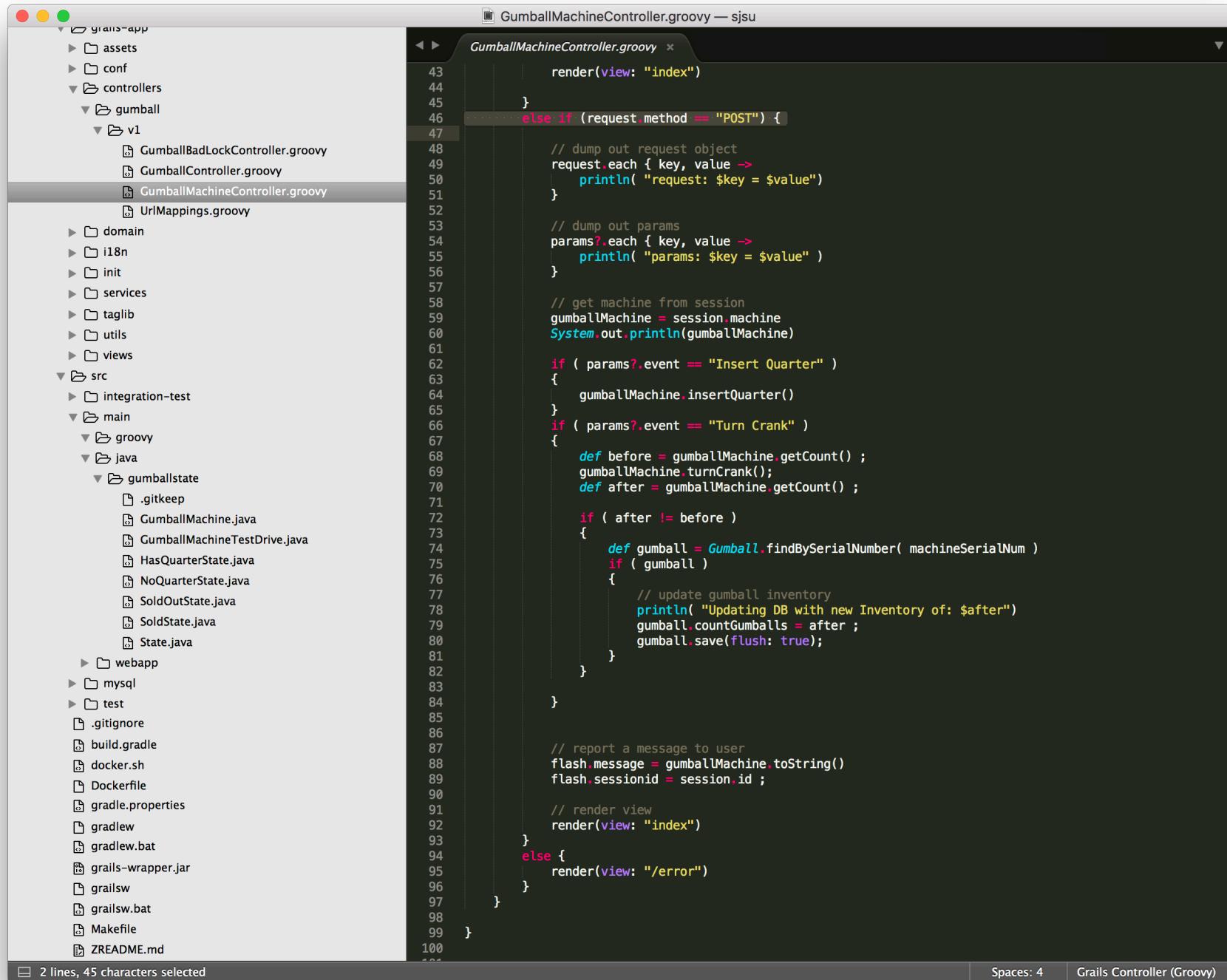
            // display view
            render(view: "index")
        }
        else if (request.method == "POST") {
            // dump out request object
            request.each { key, value ->
                println( "request: $key = $value" )
            }

            // dump out params
            params?.each { key, value ->
                println( "params: $key = $value" )
            }
        }
    }

    // get machine from session
}
```

Bottom status bar: 2 lines, 39 characters selected | Spaces: 4 | Grails Controller (Groovy)

Gumball Machine Controller



```
Grails Application Structure:
grails-app
  - assets
  - conf
  - controllers
    - gumball
      - v1
        - GumballBadLockController.groovy
        - GumballController.groovy
        - GumballMachineController.groovy
        - UrlMappings.groovy
      - domain
      - i18n
      - init
      - services
      - taglib
      - utils
      - views
    - src
      - integration-test
      - main
        - groovy
        - java
          - gumballstate
            - .gitkeep
            - GumballMachine.java
            - GumballMachineTestDrive.java
            - HasQuarterState.java
            - NoQuarterState.java
            - SoldOutState.java
            - SoldState.java
            - State.java
        - webapp
      - mysql
      - test
      - .gitignore
      - build.gradle
      - docker.sh
      - Dockerfile
      - gradle.properties
      - gradlew
      - gradlew.bat
      - grails-wrapper.jar
      - grailsw
      - grailsw.bat
      - Makefile
      - ZREADME.md

Code Editor (GumballMachineController.groovy):


```

43 render(view: "index")
44
45 }
46 else if (request.method == "POST") {
47
48 // dump out request object
49 request.each { key, value -
50 println("request: $key = $value")
51 }
52
53 // dump out params
54 params?.each { key, value -
55 println("params: $key = $value")
56 }
57
58 // get machine from session
59 gumballMachine = session.machine
60 System.out.println(gumballMachine)
61
62 if (params?.event == "Insert Quarter")
63 {
64 gumballMachine.insertQuarter()
65 }
66 if (params?.event == "Turn Crank")
67 {
68 def before = gumballMachine getCount() ;
69 gumballMachine.turnCrank();
70 def after = gumballMachine.getCount() ;
71
72 if (after != before)
73 {
74 def gumball = Gumball.findBySerialNumber(machineSerialNum)
75 if (gumball)
76 {
77 // update gumball inventory
78 println("Updating DB with new Inventory of: $after")
79 gumball.countGumballs = after ;
80 gumball.save(flush: true);
81 }
82 }
83
84 }
85
86 // report a message to user
87 flash.message = gumballMachine.toString()
88 flash.sessionid = session.id ;
89
90 // render view
91 render(view: "index")
92 }
93 else {
94 render(view: "/error")
95 }
96 }
97
98
99 }
```


```

Gumball Machine View (GSP)

The screenshot shows a code editor window with a dark theme. On the left is a file tree for a Grails application named 'gumball'. The 'views' directory contains 'index.gsp', which is the current file being edited. The code in 'index.gsp' is a Groovy Server Page (GSP) template.

```
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <title>Welcome to the Gumball Machine</title>
</head>
<body>
    <h1 align="center">Welcome to the Gumball Machine</h1>
    <!-- Session ID: ${flash.sessionid} -->
    <!-- FORM SECTION -->
    <form name="form1" method="post" action="">
        <p>
            <div align="center">
                <textarea name="message" cols="50" rows="10" readonly id="message">
                    ${flash.message}
                </textarea>
            </div>
        </p>
        <p align="center"></p>
        <p align="center">
            <input type="submit" name="event" id="btnInsertQuarter" value="Insert Quarter">
            &nbsp;&nbsp;&nbsp;
            <input type="submit" name="event" id="btnTurnCrank" value="Turn Crank">
        </p>
    </form>
    <!-- END FORM SECTION -->
</body>
</html>
```

The file tree on the left includes:

- assets
- conf
- controllers
 - gumball
 - v1
 - GumballBadLockController.groovy
 - GumballController.groovy
 - GumballMachineController.groovy
 - UrlMappings.groovy
 - domain
 - i18n
 - init
 - services
 - taglib
 - utils
 - views
 - gumball
 - gumballBadLock
 - gumballMachine
 - index.gsp
 - layouts
 - error.gsp
 - index.gsp
 - notFound.gsp
- src
 - .gitignore
 - build.gradle
 - docker.sh
 - Dockerfile
 - gradle.properties
 - gradlew
 - gradlew.bat
 - grails-wrapper.jar
 - grailsw
 - grailsw.bat
 - Makefile
 - ZREADME.md
- gumball-v2
- gumball-v3
- hellobooks
- hellograils
- helorest
 - README.md
- models
- mysql

7 lines, 60 characters selected

Spaces: 4

Grails Server Page (GSP)

Using the “Sessions” and “Flash” Object

The screenshot shows a file tree on the left and a code editor on the right. The file tree includes assets, conf, controllers (with gumball and v1 subfolders containing GumballBadLockController.groovy, GumballController.groovy, and GumballMachineController.groovy), domain, i18n, init, services, taglib, utils, views (with gumball, gumballBadLock, and gumballMachine subfolders containing index.gsp, error.gsp, index.gsp, and notFound.gsp), src, .gitignore, build.gradle, docker.sh, Dockerfile, gradle.properties, gradlew, gradlew.bat, grails-wrapper.jar, grailsw, grailsw.bat, Makefile, and ZREADME.md. The GumballMachineController.groovy file is open in the editor.

```
1 package gumball.v1
2
3 import gumballstate.GumballMachine
4
5
6 class GumballMachineController {
7
8     def String machineSerialNum = "1234998871109"
9     def GumballMachine gumballMachine
10
11    def index() {
12        def sessionid = session.id
13        println( "Session ID: $sessionid" );
14
15        if (request.method == "GET") {
16
17            // search db for gumball machine
18            def gumball = Gumball.findBySerialNumber( machineSerialNum )
19            if (gumball) {
20
21                // create a default machine
22                gumballMachine = new GumballMachine(gumball.countGumballs)
23                gumballMachine.setModelNumber(gumball.modelNumber)
24                gumballMachine.setSerialNumber(gumball.serialNumber)
25                System.out.println(gumballMachine)
26            }
27            else {
28                // create a default machine
29                gumballMachine = new GumballMachine(5);
30                System.out.println(gumballMachine)
31            }
32
33            // save in the session
34            session.machine = gumballMachine
35
36            // report a message to user
37            flash.message = gumballMachine.toString()
38            flash.sessionid = session.id ;
39
40            // display view
41            render(view: "index")
42
43        }
44        else if (request.method == "POST") {
45
46            // dump out request object
47            request.each { key, value ->
48                println( "request: $key = $value" )
49            }
50
51            // dump out params
52            params?.each { key, value ->
53                println( "params: $key = $value" )
54            }
55
56            // get machine from session
57            session.machine
58
59        }
60    }
61}
```

The code editor has two sections highlighted with orange boxes. The first box highlights the line `def sessionid = session.id` and the subsequent `println` statement. The second box highlights the `flash.message` assignment and the `flash.sessionid` assignment. The status bar at the bottom indicates "Line 1, Column 1" and "Spaces: 4".

Problem with Sessions

Welcome to the Gumball Machine

Mighty Gumball, Inc.
Java-enabled Standing Gumball
Model# null
Serial# null
Inventory: 3 gumballs
Machine is waiting for quarter

Insert Quarter Turn Crank

Welcome to the Gumball Machine

Mighty Gumball, Inc.
Java-enabled Standing Gumball
Model# null
Serial# null
Inventory: 5 gumballs
Machine is waiting for quarter

Insert Quarter Turn Crank

Problem with Locks

The screenshot shows an IDE interface with a file tree on the left and a code editor on the right. The file tree shows a project structure with various packages like assets, conf, controllers, domain, i18n, init, services, taglib, utils, views, layouts, and models. The 'controllers' package contains several files: GumballBadLockController.groovy, GumballController.groovy, GumballMachineController.groovy, and UrlMappings.groovy. The 'views' package contains gumball, gumballBadLock, and gumballMachine sub-directories with index.gsp files. The 'src' directory contains .gitignore, build.gradle, docker.sh, Dockerfile, gradle.properties, gradlew, gradlew.bat, grails-wrapper.jar, grailsw, grailsw.bat, Makefile, and ZREADME.md. The 'models' and 'mysql' directories are also present.

The code editor displays the content of `GumballBadLockController.groovy`. A specific section of the code is highlighted with a red rectangular box:

```
    }
    else if (request.method == "POST") {
        // dump out request object
        request.each { key, value ->
            println( "request: $key = $value")
        }

        // dump out params
        params?.each { key, value ->
            println( "params: $key = $value" )
        }

        // get machine from session
        gumballMachine = session.machine
        System.out.println(gumballMachine)

        if ( params?.event == "Insert Quarter" )
        {
            gumballMachine.insertQuarter()
        }
        if ( params?.event == "Turn Crank" )
        {
            def before = gumballMachine.getCount() ;
            gumballMachine.turnCrank();
            def after = gumballMachine.getCount();

            if ( after != before )
            {
                def gumball = Gumball.findByIdSerialNumber( machineSerialNum )
                gumball.lock() // pessimistic lock
                Thread.currentThread().sleep(4000)
                if ( gumball )
                {
                    // update gumball inventory
                    println( "Updating DB with new Inventory of: $after" )
                    gumball.countGumballs = after ;
                    gumball.save(flush: true); // default optimistic lock
                }
            }
        }
    }

    // report a message to user
    flash.message = gumballMachine.toString()
    flash.sessionid = session.id ;

    // render view
    render(view: "index")
}
else {
    render(view: "/error")
}
```

The highlighted code block is located between lines 66 and 87, specifically within the `if (params?.event == "Turn Crank")` block. It demonstrates a race condition where two threads might try to update the same database record simultaneously, leading to inconsistent results.

At the bottom of the code editor, there is a status bar with the text "Line 1, Column 1" and "Spaces: 4".



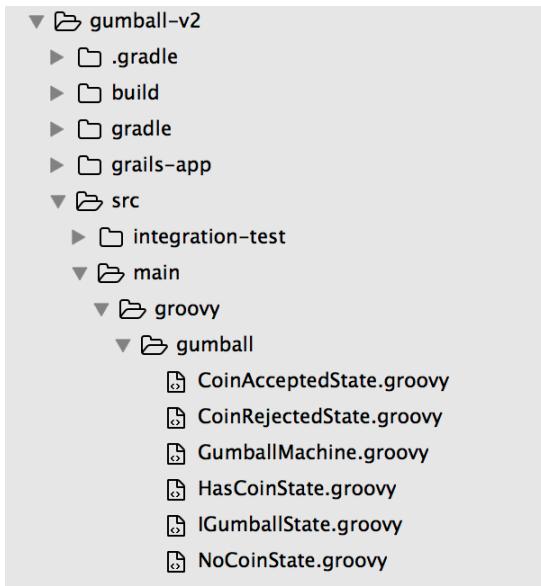
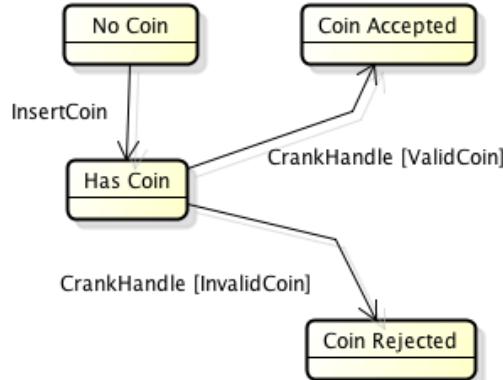
Gumball Machine (v2)

Refactoring's (from v1 to v2):

1. Gumball State Machine Responsible for Application State Only
2. Domain Object Persistent State moved to GORM (i.e. Inventory Count)

Refactoring's (from v1 to v2):

1. Gumball State Machine Responsible for Application State Only
2. Domain Object Persistent State moved to GORM (i.e. Inventory Count)



```
GumballMachine.groovy
```

```
1 package gumball
2
3 class GumballMachine {
4
5     def String _model_number
6     def String _serial_number
7     def IGumballState _current_state
8
9     NoCoinState _no_coin = new NoCoinState(this)
10    CoinAcceptedState _coin_accepted = new CoinAcceptedState(this)
11    CoinRejectedState _coin_rejected = new CoinRejectedState(this)
12    HasCoinState _has_coin = new HasCoinState(this)
13
14    GumballMachine(model, serial)
15    {
16        _current_state = _no_coin
17        _model_number = model
18        _serial_number = serial
19    }
20
21    def insertCoin()
22    {
23        _current_state.insertCoin()
24    }
25
26    def crankHandle()
27    {
28        _current_state.crankHandle()
29    }
30
31    def setNoCoin() { _current_state = _no_coin }
32    def setCoinAccepted() { _current_state = _coin_accepted }
33    def setCoinRejected() { _current_state = _coin_rejected }
34    def setHasCoin() { _current_state = _has_coin }
35    def String getCurrentState() { return _current_state.getClass().getName() }
36
37    def setCurrentState(state)
38    {
39        if (state.equals("gumball.NoCoinState")) { setNoCoin() ; }
40        if (state.equals("gumball.CoinAcceptedState")) { setCoinAccepted() ; }
41        if (state.equals("gumball.CoinRejectedState")) { setCoinRejected() ; }
42        if (state.equals("gumball.HasCoinState")) { setHasCoin() ; }
43    }
44
45
46    def String getAbout() {
47        return """
48
49        Mighty Gumball, Inc.
50        Groovy-Enabled Standing Gumball
51        Model# ${_model_number}
52        Serial# ${_serial_number}
53
54
55        Gumball Machine: ${this.toString()}
56        Current State: ${_current_state.toString()}
57
58    """
```

The code defines the GumballMachine class. It maintains a current state (either NoCoinState, CoinAcceptedState, CoinRejectedState, or HasCoinState) and provides methods to insert a coin, crank the handle, and change the state. It also includes a toString method that returns the machine's model number, serial number, and current state.

Refactoring's (from v1 to v2):

1. Gumball State Machine Responsible for Application State Only
2. Domain Object Persistent State moved to GORM (i.e. Inventory Count)

```
        }
    else if (request.method == "POST") {

        // dump out request object
        request.each { key, value ->
            println( "request: $key = $value")
        }

        // dump out params
        params?.each { key, value ->
            println( "params: $key = $value" )
        }

        // get machine from session
        gumballMachine = session.machine

        System.out.println(gumballMachine.getAbout())

        if ( params?.event == "Insert Quarter" )
        {
            gumballMachine.insertCoin()
        }
        if ( params?.event == "Turn Crank" )
        {
            gumballMachine.crankHandle();

            if ( gumballMachine.getCurrentState().equals("gumball.CoinAcceptedState") )
            {
                def gumball = Gumball.findByIdSerialNumber( machineSerialNum )
                if ( gumball )
                {
                    // update gumball inventory
                    // gumball.lock() // pessimistic lock
                    if ( gumball.countGumballs > 0)
                        gumball.countGumballs--;
                    gumball.save(flush: true); // default optimistic lock
                }
            }
        }
    }
```

The screenshot shows a Mac OS X desktop with a web browser window titled "GORM for Hibernate". The address bar contains the URL "gorm.grails.org/6.0.x/hibernate/manual/#locking". The browser's menu bar includes "File", "Edit", "Search", "View", "Window", and "Help". Below the menu bar is a toolbar with icons for back, forward, search, and other functions. The main content area displays the "6.9. Pessimistic and Optimistic Locking" section of the manual. It includes a sub-section "6.9.1. Optimistic Locking" and a note about the default configuration for optimistic locking. A code snippet is shown for retrieving an airport object and printing its version. Another code snippet demonstrates how to handle an OptimisticLockingFailureException. At the bottom, there is a search bar with the query "Locking" and options for "Highlight All", "Match Case", and "Whole Words".

6.9. Pessimistic and Optimistic Locking

6.9.1. Optimistic Locking

By default GORM classes are configured for optimistic locking. Optimistic locking is a feature of Hibernate which involves storing a version value in a special `version` column in the database that is incremented after each update.

The `version` column gets read into a `version` property that contains the current versioned state of persistent instance which you can access:

```
def airport = Airport.get(10)  
println airport.version
```

When you perform updates Hibernate will automatically check the version property against the version column in the database and if they differ will throw a [StaleObjectException](#). This will roll back the transaction if one is active.

This is useful as it allows a certain level of atomicity without resorting to pessimistic locking that has an inherit performance penalty. The downside is that you have to deal with this exception if you have highly concurrent writes. This requires flushing the session:

```
def airport = Airport.get(10)  
  
try {  
    airport.name = "Heathrow"  
    airport.save(flush: true)  
}  
catch (org.springframework.dao.OptimisticLockingFailureException e) {  
    // deal with exception  
}
```

The way you deal with the exception depends on the application. You could attempt a programmatic merge of the data or go back to the user and ask them to resolve the conflict.

Alternatively, if it becomes a problem you can resort to pessimistic locking.

The screenshot shows a web browser window titled "GORM for Hibernate". The address bar contains the URL "gorm.grails.org/6.0.x/hibernate/manual/#locking". The page content is about pessimistic locking in GORM.

6.9.2. Pessimistic Locking

Pessimistic locking is equivalent to doing a SQL "SELECT * FOR UPDATE" statement and locking a row in the database. This has the implication that other read operations will be blocking until the lock is released.

In GORM pessimistic locking is performed on an existing instance with the [lock\(\)](#) method:

```
def airport = Airport.get(10)
airport.lock() // lock for update
airport.name = "Heathrow"
airport.save()
```

GORM will automatically deal with releasing the lock for you once the transaction has been committed.

However, in the above case what we are doing is "upgrading" from a regular SELECT to a SELECT.FOR UPDATE and another thread could still have updated the record in between the call to `get()` and the call to `lock()`.

To get around this problem you can use the static [lock\(id\)](#) method that takes an id just like [get\(id\)](#):

```
def airport = Airport.lock(10) // lock for update
airport.name = "Heathrow"
airport.save()
```

In this case only SELECT..FOR UPDATE is issued.

As well as the [lock\(id\)](#) method you can also obtain a pessimistic locking using queries. For example using a dynamic finder:

```
def airport = Airport.findByName("Heathrow", [lock: true])
```

Or using criteria:

```
def airport = Airport.createCriteria().get {
    eq('name', 'Heathrow')
    lock true
```

At the bottom of the browser window, there is a search bar with the text "Locking" and navigation buttons for "Highlight All", "Match Case", and "Whole Words".

The screenshot shows a file browser interface on the left and a code editor on the right. The file browser lists several project modules and their contents:

- OPEN FILES
- FOLDERS
 - 202
 - 281
 - aws
 - bios
 - docker
 - grails
 - groovy
 - gumball-v1
 - gumball-v2
 - .gradle
 - build
 - gradle
 - grails-app
 - assets
 - conf
 - controllers
 - gumball
 - domain
 - i18n
 - init
 - services
 - taglib
 - utils
 - views
 - src
 - integration-test
 - main
 - groovy
 - gumball
 - CoinAcceptedState.groovy
 - CoinRejectedState.groovy
 - GumballMachine.groovy
 - HasCoinState.groovy
 - IGumballState.groovy
 - NoCoinState.groovy
 - webapp
 - test
 - .gitignore
 - .build.gradle

OPEN FILES

FOLDERS

```

202
281
aws
bios
docker
grails
  groovy
  gumball-v1
  gumball-v2
    .gradle
    build
    gradle
    grails-app
      assets
      conf
      controllers
        gumball
        v2
          GumballMachineController.groovy
          GumballSecuredController.groovy
          GumballStatelessController.groovy
          UrlMappings.groovy
      domain
      i18n
      init
      services
      taglib
      utils
      views
src
  integration-test
  main
    groovy
      gumball
        CoinAcceptedState.groovy
        CoinRejectedState.groovy
        GumballMachine.groovy
        HasCoinState.groovy
        IGumballState.groovy
        NoCoinState.groovy
    webapp
test
.gitignore
.build.gradle

```

GumballMachineController.groovy — sjsu

```

35   render(view: "index")
36
37 }
38 else if (request.method == "POST") {
39
40   // dump out request object
41   request.each { key, value ->
42     println( "request: $key = $value" )
43   }
44
45   // dump out params
46   params?.each { key, value ->
47     println( "params: $key = $value" )
48   }
49
50   // get machine from session
51   gumballMachine = session.machine
52
53   System.out.println(gumballMachine.getAbout())
54
55   if ( params?.event == "Insert Quarter" )
56   {
57     gumballMachine.insertCoin()
58   }
59   if ( params?.event == "Turn Crank" )
60   {
61     gumballMachine.crankHandle();
62
63     if ( gumballMachine.getCurrentState().equals("gumball.CoinAcceptedState") )
64     {
65       def gumball = Gumball.findBySerialNumber( machineSerialNum )
66       if ( gumball )
67       {
68         // update gumball inventory
69         // gumball.lock() // pessimistic lock
70         if ( gumball.countGumballs > 0 )
71           gumball.countGumballs--;
72         gumball.save(flush: true); // default optimistic lock
73       }
74     }
75   }
76
77   // report a message to user
78   flash.message = gumballMachine.getAbout()
79
80   // render view
81   render(view: "index")
82 }
83 else {
84   render(view: "/error")
85 }
86
87 }
88
89 }
90
91
92

```

Line 1, Column 1

Spaces: 4 Grails Controller (Groovy)

Critical Section:

1. Search/Find Record
2. Read Count & Decrement by one
3. Save (lock released if pessimistic, or version check if optimistic)

Welcome to the Gumball Machi... +

localhost:8080/gumballMachine/index

Search

NOTES ADMIN BOOKS WEB SJSU COURSES LABS EDU CODE API DATA GO JS TOOLS REFS

Welcome to the Gumball Machine

Mighty Gumball, Inc.
Groovy-Enabled Standing Gumball
Model# M102988
Serial# 1234998871109

Gumball Machine: gumball.GumballMachine@6b2059b8
Current State: gumball.NoCoinState@7c7016e1



Insert Quarter Turn Crank



Gumball Machine (v2)

Refactoring's (from v1 to v2):

3. Remove Server Side “Session” State
4. Add “Secured” Client State Validation

Refactoring's (from v1 to v2):

3. Remove Server Side “Session” State
4. Add “Secured” Client State Validation

```
GumballStatelessController.groovy — sjsu
1 package gumball.v2
2
3 import gumball.GumballMachine
4
5 class GumballStatelessController {
6
7     def String machineSerialNum = "1234998871109"
8     def GumballMachine gumballMachine
9
10    def index() {
11
12        String VCAP_SERVICES = System.getenv('VCAP_SERVICES')
13
14        if (request.method == "GET") {
15
16            // search db for gumball machine
17            def gumball = Gumball.findBySerialNumber( machineSerialNum )
18            if ( gumball ) {
19
20                // create a default machine
21                gumballMachine = new GumballMachine(gumball.modelNumber, gumball.serialNumber)
22                System.out.println(gumballMachine.getAbout())
23            }
24            else {
25
26                flash.message = "Error! Gumball Machine Not Found!"
27                render(view: "index")
28            }
29
30            // don't save in the session
31            // session.machine = gumballMachine
32
33            // send machine state to client (instead)
34            flash.state = gumballMachine.getCurrentState();
35            flash.model = gumball.modelNumber;
36            flash.serial = gumball.serialNumber;
37
38            // report a message to user
39            flash.message = gumballMachine.getAbout()
40
41            // display view
42            render(view: "index")
43
44        }
45        else if (request.method == "POST") {
46
47            // dump out request object
48            request.each { key, value ->
49                println( "request: $key = $value" )
50            }
51
52            // dump out params
53            params?.each { key, value ->
54                println( "params: $key = $value" )
55            }
56        }
57    }
58
59}
```

*WebUI sends State
(of State Machine)
to Browser via HTML
“hidden fields”*

```
<!-- FORM SECTION -->
<form name="form1" method="post" action="">
    <p>
        <input type="hidden" name="state" id="state" value="${flash.state}" />
        <input type="hidden" name="model" id="model" value="${flash.model}" />
        <input type="hidden" name="serial" id="serial" value="${flash.serial}" />
    </p>
    <div align="center">
        <textarea name="message" cols="50" rows="15" readonly id="message">
            ${flash.message}
        </textarea>
    </div>
```

Refactoring's (from v1 to v2):

3. Remove Server Side “Session” State
4. Add “Secured” Client State Validation

```
}

else if (request.method == "POST") {

    // dump out request object
    request.each { key, value ->
        println( "request: $key = $value")
    }

    // dump out params
    params?.each { key, value ->
        println( "params: $key = $value" )
    }

    // don't get machine from session
    // gumballMachine = session.machine

    // restore machine to client state (instead)
    def state = params?.state
    def modelNum = params?.model
    def serialNum = params?.serial
    gumballMachine = new GumballMachine(modelNum, serialNum) ;
    gumballMachine.setCurrentState(state) ;

    System.out.println(gumballMachine.getAbout())
}
```

Upon receiving each request, must reset State Machine to previous State tracked by Client Browser “hidden field”.

Welcome to the Gumball Machi... +

localhost:8080/gumballStateless/index

NOTES ADMIN BOOKS WEB SJSU COURSES LABS EDU CODE API DATA GO JS TOOLS REFS

Welcome to the Gumball Machine

Mighty Gumball, Inc.
Groovy-Enabled Standing Gumball
Model# M102988
Serial# 1234998871109

Gumball Machine: gumball.GumballMachine@30b67365
Current State: gumball.NoCoinState@492f4991



Insert Quarter Turn Crank

view-source:http://localhost:8080/gumballStateless/index

```
1
2
3
4 <html>
5 <head>
6   <title>Welcome to the Gumball Machine (New Improved Version)</title>
7 </head>
8
9 <body>
10 <h1 align="center">Welcome to the Gumball Machine</h1>
11
12 <!-- FORM SECTION -->
13 <form name="form1" method="post" action="">
14
15   <input type="hidden" name="state" id="state" value="gumball.NoCoinState" />
16   <input type="hidden" name="model" id="model" value="M102988" />
17   <input type="hidden" name="serial" id="serial" value="1234998871109" />
18
19   <div align="center">
20     <textarea name="message" cols="50" rows="15" readonly id="message">
21
22 -----
23 Mighty Gumball, Inc.
24 Groovy-Enabled Standing Gumball
25 Model# M102988
26 Serial# 1234998871109
27
28 Gumball Machine: gumball.GumballMachine@30b67365
29 Current State: gumball.NoCoinState@492f4991
30
31   </textarea>
32 </div>
33 </p>
34 <p align="center"></p>
35
36 <p align="center">
37   <input type="submit" name="event" id="btnInsertQuarter" value="Insert Quarter">
38   &ampnbsp&ampnbsp&ampnbsp&ampnbsp
39   <input type="submit" name="event" id="btnTurnCrank" value="Turn Crank">
40 </p>
41 </form>
42 <!-- END FORM SECTION -->
43
44 </body>
45 </html>
```

Refactoring's (from v1 to v2):

3. Remove Server Side “Session” State
4. Add “Secured” Client State Validation

For Demo Only. Do not include Credentials in Source Code / GitHub for Production Systems

```
GumballSecuredController.groovy — sjsu
1 package gumball.v2
2
3 import gumball.GumballMachine
4 import javax.crypto.Mac;
5 import javax.crypto.spec.SecretKeySpec;
6 import java.security.InvalidKeyException;
7
8 class GumballSecuredController {
9
10    def String machineSerialNum = "1234998871109"
11    def String secretKey = "kwRg54x2Go91Ed149jFENRM12Mp711QI"
12    def GumballMachine gumballMachine
13    def String msg
14    def hash
15
16    // REF: http://www.jokecamp.com/blog/examples-of-creating-base64-hashes-using-hmac-sha256-in-different-languages/#groovy
17    def hmac_sha256(String secretKey, String data) {
18        try {
19            Mac mac = Mac.getInstance("HmacSHA256")
20            SecretKeySpec secretKeySpec = new SecretKeySpec(secretKey.getBytes(), "HmacSHA256")
21            mac.init(secretKeySpec)
22            byte[] digest = mac.doFinal(data.getBytes())
23            return digest
24        } catch (InvalidKeyException e) {
25            throw new RuntimeException("Invalid key exception while converting to HMac SHA256")
26        }
27    }
28
29    def index() {
30
31        String VCAP_SERVICES = System.getenv('VCAP_SERVICES')
32
33        if (request.method == "GET") {
34
35            // search db for gumball machine
36            def gumball = Gumball.findBySerialNumber( machineSerialNum )
37            if ( gumball ) {
38
39                // create a default machine
40                gumballMachine = new GumballMachine(gumball.modelNumber, gumball.serialNumber)
41                System.out.println(gumballMachine.getAbout())
42            }
43            else {
44
45                flash.message = "Error! Gumball Machine Not Found!"
46                render(view: "index")
47            }
48
49            // don't save in the session
50            // session.machine = gumballMachine
51
52            // send machine state to client (instead)
53            flash.state = gumballMachine.getCurrentState();
54            flash.model = gumball.modelNumber ;
55            flash.serialNumber = gumball.serialNumber
56        }
57    }
58}
```

Use HMAC SHA256 Hash

Refactoring's (from v1 to v2):

3. Remove Server Side “Session” State
4. Add “Secured” Client State Validation

```
if (request.method == "GET") {  
  
    // search db for gumball machine  
    def gumball = Gumball.findBySerialNumber( machineSerialNum )  
    if ( gumball )  
    {  
        // create a default machine  
        gumballMachine = new GumballMachine(gumball.modelNumber, gumball.serialNumber)  
        System.out.println(gumballMachine.getAbout())  
    }  
    else  
    {  
        flash.message = "Error! Gumball Machine Not Found!"  
        render(view: "index")  
    }  
  
    // don't save in the session  
    // session.machine = gumballMachine  
  
    // send machine state to client (instead)  
    flash.state = gumballMachine.getCurrentState() ;  
    flash.model = gumball.modelNumber ;  
    flash.serial = gumball.serialNumber ;  
  
    // get system date  
    flash.ts = System.currentTimeMillis().toString();  
    msg = flash.state + "|" + flash.model + "|" + flash.serial + "|" + flash.ts + "|" + secretKey  
    hash = hmac_sha256(secretKey, msg)  
    println ( "MSG: " + msg )  
    println ( "HASH: " + hash.encodeBase64() )  
    flash.hash = hash.encodeBase64()  
  
    // report a message to user  
    flash.message = gumballMachine.getAbout()  
  
    // display view  
    render(view: "index")  
}
```

On GET, generate “Clear Text String” and Send to Browser Timestamp and Hash

```
<!-- FORM SECTION -->  
<form name="form1" method="post" action="">  
    <p>  
        <input type="hidden" name="state" id="state" value="${flash.state}" />  
        <input type="hidden" name="model" id="model" value="${flash.model}" />  
        <input type="hidden" name="serial" id="serial" value="${flash.serial}" />  
        <input type="hidden" name="ts" id="ts" value="${flash.ts}" />  
        <input type="hidden" name="hash" id="hash" value="${flash.hash}" />  
  
    <div align="center">  
        <textarea name="message" cols="50" rows="15" readonly id="message">  
            ${flash.message}  
        </textarea>  
    </div>
```

Refactoring's (from v1 to v2):

3. Remove Server Side “Session” State
4. Add “Secured” Client State Validation

```
// restore machine to client state (instead)
def state = params?.state
def modelNum = params?.model
def serialNum = params?.serial
def tsString = params?.ts

// check hash and time stamp
def long ts = Long.parseLong( tsString )
def long cts = System.currentTimeMillis()
def long diff = cts - ts
println ( diff/1000 ) // seconds
def hash1 = params?.hash
def String msg = state + "|" + modelNum + "|" + serialNum + "|" + tsString + "|" + secretKey
def hashBytes = hmac_sha256(secretKey, msg)
def hash2 = hashBytes.encodeBase64().toString()
println ( "MSG: " + msg )
println ( "HASH1: " + hash1 )
println ( "HASH2: " + hash2 )

def invalidTS = ((diff/1000) > 120)
def invalidHASH = (hash1 != hash2)
println( "invalid ts: " + invalidTS )
println( "invalid hash: " + invalidHASH )
```

On POST, validate Timestamp and Hash.

Welcome to the Gumball Machine

Mighty Gumball, Inc.
Groovy-Enabled Standing Gumball
Model# M102988
Serial# 1234998871109

Gumball Machine: gumball.GumballMachine@5e2451ae
Current State: gumball.NoCoinState@29866ela



Insert Quarter Turn Crank

```
1
2
3
4 <html>
5 <head>
6   <title>Welcome to the Gumball Machine (New Improved Version)</title>
7 </head>
8
9 <body>
10 <h1 align="center">Welcome to the Gumball Machine</h1>
11
12 <!-- FORM SECTION -->
13 <form name="form1" method="post" action="">
14   <p>
15     <input type="hidden" name="state" id="state" value="gumball.NoCoinState" />
16     <input type="hidden" name="model" id="model" value="M102988" />
17     <input type="hidden" name="serial" id="serial" value="1234998871109" />
18     <input type="hidden" name="ts" id="ts" value="1487408567581" />
19     <input type="hidden" name="hash" id="hash" value="RpXciFt7HDccY1Eplxgk7wXMmFvq25nf/DT5hm8Y=" />
20
21   <div align="center">
22     <textarea name="message" cols="50" rows="15" readonly id="message">
23
24
25 Mighty Gumball, Inc.
26 Groovy-Enabled Standing Gumball
27 Model# M102988
28 Serial# 1234998871109
29
30 Gumball Machine: gumball.GumballMachine@5e2451ae
31 Current State: gumball.NoCoinState@29866ela
32
33   </textarea>
34 </div>
35 </p>
36 <p align="center"></p>
37
38 <p align="center">
39   <input type="submit" name="event" id="btnInsertQuarter" value="Insert Quarter">
40   &ampnbsp&ampnbsp&ampnbsp
41   <input type="submit" name="event" id="btnTurnCrank" value="Turn Crank">
42 </p>
43
44 <!-- END FORM SECTION -->
45
46 </body>
47 </html>
```



Gumball (v3) - REST Service

Gumball REST Service (v3):

3. Transaction Object (Domain Model Changes)
4. REST Order Status and Place Order API

Gumball REST Service (v3):

1. Transaction Object (Domain Model Changes)

2. REST Order Status and Place Order API

```
◀ ▶ Gumball.groovy *
```

```
1 package gumball.v3
2
3 class Gumball {
4
5     String modelNumber
6     String serialNumber
7     Integer countGumballs
8
9     static constraints = {
10        serialNumber(unique: true)
11    }
12 }
13
```

```
◀ ▶ GumballOrder.groovy *
```

```
1 package gumball.v3
2
3 class GumballOrder {
4
5     String orderstatus ;
6
7     static mapping = {
8         id generator: 'increment'
9     }
10
11 }
12
```

Why?

*More Scalable and Friendly to
Sharding to always Create
Orders (i.e. No Read/Write Locks?).*

Gumball REST Service (v3):

1. Transaction Object (Domain Model Changes)

2. REST Order Status and Place Order API

```
UrlMappings.groovy *
```

```
1 package gumball.v3
2
3 class UrlMappings {
4
5     static mappings = {
6         delete "/$controller/$id(.format)?"(action:"delete")
7         get "/$controller(.format)?"(action:"index")
8         get "/$controller/$id(.format)?"(action:"show")
9         post "/$controller(.format)?"(action:"save")
10        put "/$controller/$id(.format)?"(action:"update")
11        patch "/$controller/$id(.format)?"(action:"patch")
12
13
14        "/order/$id?"(controller: "GumballRest", parseRequest: true) {
15            action = [GET: "orderStatus", POST: "placeOrder"]
16        }
17
18        get "/gumball"(controller: 'GumballRest', action:'machineStatus')
19
20        "/"(controller: 'application', action:'index')
21        "500"(view: '/error')
22        "404"(view: '/notFound')
23    }
24 }
25 }
```

*Get Order Status API (GET with Order ID)
Place Order API (POST)*

Gumball REST Service (v3):

1. Transaction Object
(Domain Model Changes)
2. REST Order Status and Place Order API

*Find GumballOrder.
If Found, Send back
in JSON Format*

```
// HTTP GET -- Get Order Status
def orderStatus() {

    // dump out request object
    request.each { key, value ->
        println( "request: $key = $value")
    }

    // dump out params
    params?.each { key, value ->
        println( "params: $key = $value" )
    }

    if ( params.id ) {
        def ord = GumballOrder.findById(params.id)
        if ( ord ) {
            render ord as JSON
        } else {
            render(contentType: "application/json") {
                Error( message: "Order ${params.id} Not Found." )
            }
        }
    } else {
        render(contentType: "application/json") {
            Error( message: "Order Number Missing." )
        }
    }
}
```

Gumball REST Service (v3):

1. Transaction Object
(Domain Model Changes)
2. REST Order Status and Place Order API

*Check Inventory Count
If Low, Create a “Backorder”
Else, Create an Order and
Update Inventory QTY.*

```
// HTTP POST -- Create a new Order
def placeOrder() {

    // dump out request object
    request.each { key, value ->
        println( "request: $key = $value")
    }

    // dump out params
    params?.each { key, value ->
        println( "params: $key = $value" )
    }

    def gumball = Gumball.findByIdSerialNumber( machineSerialNum )
    if ( gumball )
    {
        if ( gumball.countGumballs > 0 )
        {
            gumball.lock() // pessimistic lock
            gumball.countGumballs--
            gumball.save(flush: true);
            def newOrder = new GumballOrder(orderstatus: "Order Placed");
            newOrder.save(flush: true) ;
            render newOrder as JSON ;
        }
        else
        {
            def newOrder = new GumballOrder(orderstatus: "Backed Ordered") ;
            newOrder.save(flush: true) ;
            render newOrder as JSON ;
        }
    }
    else {
        render(contentType: "application/json") {
            Error( message: "System Error." )
        }
    }
}
```

Postman

Runner Import +

Builder Team Library

Gumball Machine Stat X + No Environment

Paul Nguyen... IN SYNC

Filter History Collections All Me Team

POSTMAN 41 requests

PROJECT 12 requests

RESTAPIS 129 requests

Grails Books

Grails REST

Gumball REST

GET Gumball Order Status (Not Found)

GET Gumball Machine Status (Inventory)

POST Gumball Place New Order

GET Gumball Order Status

Gumball Restlet

Hello Restlet

Quickpoll

Restlet Tutorial

Service Test

Starbucks API

Starbucks Kong

Starbucks Test

WORKSPACE 144 requests

Gumball Machine Status (Inventory)

GET localhost:8080/gumball/ Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

Type No Auth

Body Cookies Headers (4) Tests Status: 200 OK Time: 239 ms Size: 254 B

Pretty Raw Preview JSON

Save Response

```
1 [ { "id": 1, "countGumballs": 993, "modelNumber": "M102988", "serialNumber": "1234998871109" } ]
```

Postman

Runner Import +

Builder Team Library

Gumball Order Status X +

No Environment

Paul Nguyen...

Filter

History Collections

All Me Team

POSTMAN 41 requests

PROJECT 12 requests

RESTAPIS 129 requests

Grails Books

Grails REST

Gumball REST

GET Gumball Order Status (Not Found)

GET Gumball Machine Status (Inventory)

POST Gumball Place New Order

GET Gumball Order Status

Gumball Restlet

Hello Restlet

Quickpoll

Restlet Tutorial

Service Test

Starbucks API

Starbucks Kong

Starbucks Test

WORKSPACE 144 requests

Gumball Order Status (Not Found)

GET localhost:8080/order/1234 Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

Type No Auth

Body Cookies Headers (4) Tests Status: 200 OK Time: 54 ms Size: 208 B

Pretty Raw Preview JSON Save Response

```
1 [ { "Error": { "message": "Order 1234 Not Found." } }
```

The screenshot shows the Postman application interface. The left sidebar displays a tree view of collections and projects. The main workspace shows a collection named "Gumball Place New Order" with a single POST request to "localhost:8080/order/". The request body is set to "raw" and "JSON (application/json)". The response body is displayed as a JSON object:

```
1 {  
2   "id": 9,  
3   "orderstatus": "Order Placed"  
4 }
```

Postman

Runner Import +

Builder Team Library

Gumball Order Status IN SYNC Paul Guy... No Environment

Filter History Collections All Me Team

POSTMAN 41 requests

PROJECT 12 requests

RESTAPIS 129 requests

Grails Books

Grails REST

Gumball REST

GET Gumball Order Status (Not Found)

GET Gumball Machine Status (Inventory)

POST Gumball Place New Order

GET Gumball Order Status

Gumball Restlet

Hello Restlet

Quickpoll

Restlet Tutorial

Service Test

Starbucks API

Starbucks Kong

Starbucks Test

WORKSPACE 144 requests

Gumball Order Status

GET localhost:8080/order/9 Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

Type No Auth

Body Cookies Headers (4) Tests Status: 200 OK Time: 20 ms Size: 208 B

Pretty Raw Preview JSON

1 [2 "id": 9, 3 "orderstatus": "Order Placed" 4]