

Objectives

- Review: databases, SQL, JDBC
- Software Engineering Tools
 - Maven
 - Issue Tracking: Jira
- Project Organization

Review: Databases, SQL, JDBC

- What do databases do for us?
- What are tables made up of?
- What language do we use to query and update relational databases?
- What is the syntax for the **SELECT** statement?
- What is a *primary key* vs a *foreign key*?
- How are SQL and JDBC related?

Review: Guest Speaker Nika

- Recording of her talk is on Sakai
- What questions do you have after Nika's presentation?

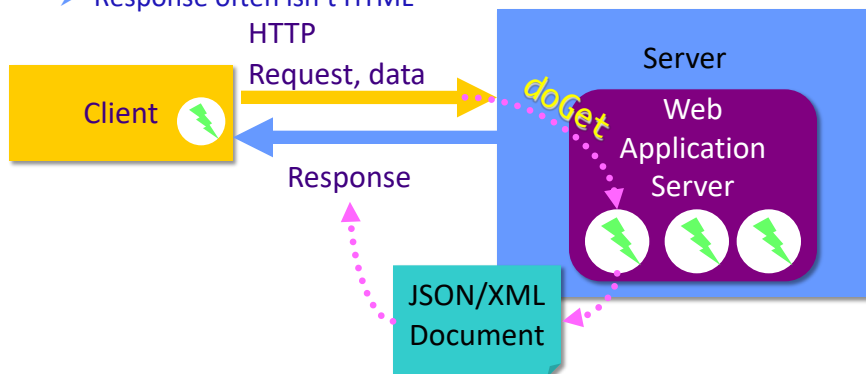
May 6, 2019

Sprenkle - CSCI335

3

REST APIs

- Typical use: access data/information over HTTP
- Similar to typical web flow
- Differences:
 - Client often isn't a user/browser (e.g., called by code)
 - Response often isn't HTML



May 6, 2019

Sprenkle - CSCI335

4

REST API Example: AGP: Individual Graffito

GET /graffito/{agpId}/json

Download the individual graffito in JSON format.

Response Class (Status 200)

OK

Model Example Value

```
{
  "app": {
    "agpId": "string",
    "caption": "string",
    "csl": "string",
    "commentary": "string",
    "contentTranslation": "string",
    "contributors": "string",
    "figuralInfo": {
      "descriptionInEnglish": "string",
      "descriptionInLatin": "string"
    }
  }
}
```

Response Content Type application/json;charset=UTF-8

Parameters

Parameter	Value	Description	Parameter Type	Data Type
agpId	(required)	agpId	path	string

Example API call to get the graffito's data in JSON form

May 6, 2019

Sprenkle - CSCI335

5

JQuery

- JavaScript library
- From jQuery site: "Getting started with jQuery can be easy or challenging, depending on your experience with JavaScript, HTML, CSS, and programming concepts in general."

```
$( "a" ).click(function( event ) {
    alert( "Thanks for visiting!" );
});
```

```
$( "p" ).click(function(){
    $(this).hide();
});
```

May 6, 2019

Sprenkle - CSCI335

6

Git Repositories Organization

- Our projects [will] have a private and public repository
- Configuration of projects contain private/sensitive information
 - Server location
 - User names and passwords
- Our solution
 - Keep our work private until reach a checkpoint to make public
 - Put placeholders into the config files

May 6, 2019

Sprenkle - CSCI335

7

PROJECTS

May 6, 2019

Sprenkle - CSCI335

8

Projects: Next Steps

- Respond to clients' feedback
- Implement high-priority functionality
- Understand code base



Apache Maven™

- Maven: Yiddish word meaning *accumulator of knowledge*
- For building and managing any Java-based project
 - Uses a **project object model (POM)**
- Goal: download and build a project quickly
- Can be used as standalone tool or within Eclipse (what we'll use)

<http://maven.apache.org/>

Maven Philosophy: Convention Over Configuration

- Maven's location assumptions:
 - source code: `${basedir}/src/main/java`
 - Resources: `${basedir}/src/main/resources`
 - Tests: `${basedir}/src/test`
- Other assumptions:
 - Produce a JAR file in `${basedir}/target`
 - Compile byte code to `${basedir}/target/classes`

How does this convention philosophy help us?

Maven Philosophy: Convention Over Configuration

- Benefit: reduces effort
 - Put source in the correct directory
 - Maven handles the rest
- Beyond location conventions...
- **Core plugins** apply a common set of conventions for compiling source code, packaging distributions, generating web sites, and many other processes

Consequences of Convention Over Configuration

- Users may feel forced to use a particular methodology or approach
- Most defaults can be customized
- Can create custom plugins for your requirements

Maven Build Lifecycle

- Defined by a list of *build phases*
- Example build phases
 - `compile` - compile the source code of the project
 - `test` - test the compiled source code using a suitable unit testing framework
 - `package` - take the compiled code and package it in its distributable format, such as a JAR
- When execute a phase, executes life cycle's previous phases first, in order
 - E.g., calling `package` would execute `compile` and then `test`

May 6, 2019

Sprenkle - CSCI335

15

Maven Build Lifecycle

- 3 built-in build lifecycles
 - `default` lifecycle handles project deployment
 - `clean` lifecycle handles project cleaning
 - `site` lifecycle handles the creation of project's site documentation

May 6, 2019

Sprenkle - CSCI335

16

Maven Repository

<https://mvnrepository.com/>

- Holds build artifacts and dependencies
- Typically: looking for a stable release
 - rc = release candidate (*not* what you want)

Adding a Dependency

Maven

Add a dependency to `junit:junit` in `test` scope. (Note: 4.12 is the latest stable version as of the latest edit on this page.)

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

- Several different ways dependency can be added to `pom.xml` file
 - Can copy the XML code provided on the Maven repository site

Adding a Dependency

Maven

Add a dependency to `junit:junit` in `test` scope. (Note: 4.12 is the latest stable version as of the latest edit on this page.)

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

- After adding a repository, give Eclipse some time to work it out
 - Eclipse will also download dependencies
- View the Maven Dependencies in Eclipse

May 6, 2019

Sprenkle - CSCI335

19

Updating a Dependency

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

- Changing versions is easy
 - Will see errors in POM if causes conflicts between dependencies
 - Will see compilation errors if upgraded version causes issues in your current code

May 6, 2019

Sprenkle - CSCI335

20

SPRING

May 6, 2019

Sprenkle - CSCI335

21

Spring Framework

- Open-source, powerful, and flexible framework focused on building Java applications
- Spring handles a lot of the common, “boilerplate” code so that you can focus on the logic for your application
 - Example: connecting to a database is fairly routine; just need to know which database, the user name, password, ...
- Dependence on conventions over configuration
- Multiple components, e.g., WebMVC, Boot, ...

May 6, 2019

Sprenkle - CSCI335

22

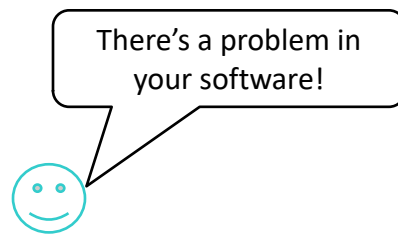
ISSUE TRACKING

May 6, 2019

Sprenkle - CSCI335

23

Problem Life Cycle



user

** Amount of happiness may vary*

vendor



1. reproduces the problem
2. isolates the circumstances
3. locates and fixes the *defect*
4. delivers the fix to the user

May 6, 2019

Sprenkle - CSCI335

Zeller 24

What's a Problem?

- A *problem* is a questionable property of a program run
 - It becomes a *failure* if it's incorrect...
 - ...a *request* for enhancement if missing...
 - ... and a *feature* if normal behavior

It's not a bug, it's a feature!

Challenges

- How do I organize the life cycle?
- Which problems are currently *open*?
 - Haven't been diagnosed, fixed
- Which are the most severe problems?
- Did similar problems occur in the past?

Problem Report

- A problem comes to life with a **problem report**
- Includes all information vendor needs to fix problem
- Also known as **change request** or **bug report**

May 6, 2019

Sprenkle - CSCI335

Zeller 27

Example Problem Report

```
From: me@dot.com  
To: you@there.org  
Subject: Crash  
Your program crashed.  
(core dumped)
```

- Core dump: recorded state of the working memory of a computer program at a specific time, generally when the program has terminated abnormally (crashed)
- Email content similar to students' emails to me when they want to know why something went wrong in their program


What does the report tell you?

May 6, 2019


Sprenkle - CSCI335

Zeller 28

Example Problem Report #2

```
From: me@dot.com  
To: you@there.org  
Subject: Re: Crash  
Sorry, here's the core  
 <core, 14MB>
```

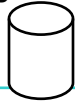
Example Problem Report #3

```
From: me@dot.com  
To: you@there.org  
Subject: Re: Crash  
You may need this too,  
just in case  
 <data, 148GB>
```

- What's the problem with these problem reports?

Example Problem Report #3

```
From: me@dot.com  
To: you@there.org  
Subject: Re: Crash  
You may need this too,  
just in case
```



```
<data, 148GB>
```

- What's the problem with the problem reports?
 - Limited information about what the problem is, what caused it
 - Information is across 3 emails

May 6, 2019

Sprenkle - CSCI335

Zeller 31

What To Report

- The product release
- The operating environment
- The problem history
- A one-line summary
- Expected and experienced behavior

May 6, 2019

Sprenkle - CSCI335

Zeller 32

Product Release

- Typically, some *version number* or otherwise unique identifier
 - Required to reproduce the problem

Perfect Publishing Program 1.1 (Build 7E47)

- Generalize: Does the problem occur only in this release?

Operating Environment

- Typically, *version information* about the operating system
- Can be simple (“Windows 10”) or complex (“Ubuntu Linux 16.04.1LTS with the following packages...”)
- Generalize: In which environments does the problem occur?

Problem History

- Steps needed to *reproduce* the problem
 1. Create “bug.ppp”
 2. Print on the default printer...
- If the problem cannot be reproduced, it is unlikely to be fixed
- Simplify: Which steps are relevant?

May 6, 2019

Sprenkle - CSCI335

Zeller 35

Expected Behavior

- What should have happened according to the user:

The program should have printed the document.

- Reality check: What is the understanding of the user?

May 6, 2019

Sprenkle - CSCI335

Zeller 36

Observed Behavior

- The *symptoms* of the problem — in contrast to the expected behavior

The program crashed with the following information:

*** STACK DUMP OF CRASH (LemonyOS)

```
Back chain  ISA  Caller
00000000    SPC  0BA8E574
03EADF80    SPC  0B742428
03EADF30    SPC  0B50FDDC  PrintThePage+072FC
SnicketPC unmapped memory exception at
          0B512BD0 PrintThePage+05F50
```

A One-Line Summary

- Captures the essence of the problem

PPP 1.1 crashes when printing

If we're developing a large software application,
as good as we may be, we're going to have bugs...

A lot of them....

Managing Problems

- **Alternative #1: *A Problem File***
 - Only one person at a time can work on it
 - History of earlier (fixed) problems is lost
 - *Does not scale*
- **Alternative #2: *A Problem Database***
 - Examples: Bugzilla, JIRA

Classifying Problems

- Severity
- Priority
- Identifier
- Comments
- Notification

Problem Severity

- **Enhancement.** A desired feature
- **Trivial.** Cosmetic problem
- **Minor.** Problem with easy workaround
- **Normal.** “Standard” problem
- **Major.** Major loss of function
- **Critical.** Crashes, loss of data or memory
- **Showstopper.** Blocks development

Priority

- Every new problem is assigned a *priority*
- The higher the priority, the sooner the problem will be addressed
- Priority is *independent* from severity
- Prioritizing problems is the main tool to control development and problem solving

Identity

- Every new problem is assigned an *identifier*
 - Also known as PR—problem report—number or bug number
- The identifier is referenced in all documents during the debugging process

Subject: PR #3427 is fixed?

- Included in your commit comments

Comments

- A developer can attach *comments* to a problem:

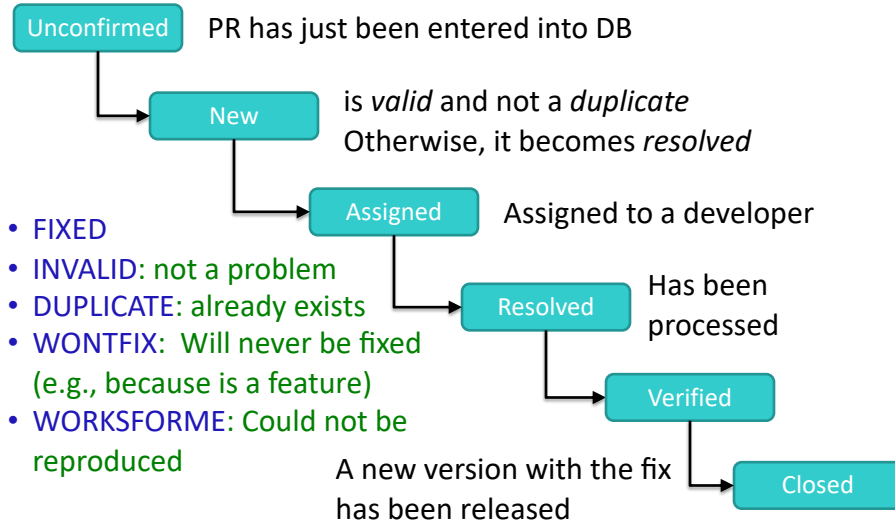
```
I have a patch for this. It's  
just an uninitialized variable,  
but I still need a review.
```

- Comments may also include files, documents, etc.

Notification

- Developers and users can attach an e-mail address to a problem report
- They will be notified every time the report changes

Simplified Problem Lifecycle



May 6, 2019

Sprenkle - CSCI335

Zeller 47

Management

- Who enters problem reports?
- Who classifies problem reports?
- Who sets priorities?
- Who takes care of the problem?
- Who closes issues?

May 6, 2019

Sprenkle - CSCI335

Zeller 48

Summary

- Reports about problems encountered in the field are stored in a *problem database*
- A problem report must contain everything relevant to reproduce the problem
- It is helpful to set up a standard set of items that users must provide (product release, operating environment...)

Problem Reports Summary

- An effective problem report...
 - is well-structured
 - is reproducible
 - has a descriptive one-line summary
 - is as simple and general as possible
 - is neutral and stays with the facts

Issue Tracking Summary


- A typical problem life cycle starts with an *unconfirmed* status
- It ends with a *closed* status and a specific resolution (such as fixed or worksforme)

May 6, 2019

Sprenkle - CSCI335

Zeller 51

Issue Tracking Tools

- Bugzilla
- Jira 
- TRAC (+ wiki)

May 6, 2019

Sprenkle - CSCI335

52

Using Jira

- Add Requirements/Features/Bugs to JIRA
 - assign to team members
- Creates TODO lists
- Can mark when you've completed task, including the git commit code

Project Organization Discussion

To Do

- HW – Read Don't Make Me Think
- Work on High-priority functionality
 - [Add issues to Jira](#)
- Exam: Thursday a.m.