Owner: Joanna Wallace

Collaborators: Weronika, Linn

Reviewer: David - This is good!

Hangout: <http://youtu.be/Ucl46TncWdk>

Pivotal Tracker: <https://www.pivotaltracker.com/story/show/84548384>

17. Answer the following:

a. Describe each of the four different HTTP methods and explain when/how they are used.

The four different HTTP methods are GET, POST, PUT, and DELETE.

**- GET:** Requests a representation of the specified resource. Requests using GET should only [retrieve data](http://en.wikipedia.org/wiki/Data_retrieval) and should have no other effect.

**- POST:** Used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the URI. POST is designed to allow a uniform method to cover the following functions:

* Annotation of existing resources
* Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles
* Providing a block of data, such as the result of submitting a form, to a data-handling process
* Extending a database through an append operation.

**- PUT:** Requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified. If the URI does not point to an existing resource, then the server can create the resource with that URI.

**- DELETE:** Requests that the origin server delete the resource identified by the URI.

Web browsers implement GET (for following and link) and POST (for submitting forms). To compensate, Rails’ routing mechanism lets browsers use POST for requests that normally would require PUT or DELETE. Rails annotates the Web forms associated with such request so that when the request is submitted, Rails can recognize it as special and can internally change the HTTP method “seen” by the controller to PUT or DELETE as appropriate. The result is that the Rails programmer can operate under the assumption that PUT and DELETE are actually supported, even though browsers don’t implement them. The advantage is that the same set of routes and controller methods can be used to handle either requests coming from another service in a SOA.

b. What is REST and how is it implemented?

REST (REpresentational State Transfer) is an abstraction of the architecture of the World Wide Web; more precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

The basic architectural properties of REST include performance, scalability, simplicity of interfaces, component modifiability, communication visibility, component portability, and reliability.

One can characterize web services as “RESTful” if they conform to the following constraints are implemented:

- Client-Server: A uniform interface separates clients from servers. This improves the portability of client code and makes the servers simpler and more scalable.

- Stateless: The client-server communication is further constrained by no client context being stored on the server between requests.

- Cacheable: Clients can cache responses, which helps eliminate some client-server interactions.

- Layered system: A client cannot tell whether it is connected directly to the end server, or to an intermediary along the way.

- Code on demand: Servers can temporarily extend or customize the functionality of a client by the transfer of executable code.

- Uniform interface: Simplifies and decouples the architecture, which enables each part to evolve independently.

c. Why is REST such an important organizing principle in Web applications?

REST is such an important organizing principle in Web applications because it simplifies architecture without losing functionality. It improves portability and scalability and eliminates unnecessary client-server interactions. For example, due to the way the routes are organized in REST, every http request has all the necessary information.

d. How are the HTML tag attributes id and class used?

The id must be unique, whereas the classes don’t need to be unique. CSS uses ids and classes as selectors.

e. How are CSS selectors used?

CSS selectors allow you to select and manipulate HTML elements. CSS selectors are used to “find” (or select) HTML elements based on their id, classes, types, attributes, and values of attributes.

Owner: Joanna Wallace

Collaborators: Linn, Kelsey

Reviewer: Shari - Looks great!

Hangout: <http://youtu.be/Ucl46TncWdk>

Pivotal Tracker: <https://www.pivotaltracker.com/story/show/84548308>

31. Scrum, like Extreme Programming, is another 'flavor' of Agile.

a. Describe the activity for which this methodology is named that is performed daily by Scrum team members

The activity for which this methodology is named is the daily Scrum stand up meeting. These meetings are designed to be at the same time daily and last about 15 minutes. There are three questions that attendees must answer which are: “What did I accomplish yesterday?,” “What will I do today?,” and “What obstacles are impeding my progress?.” These meetings are intended as a communication vehicle for team members and not as a status update for management. The structure of these meetings promotes follow-up conversation between team members and identifies issues along the way.

b. Define and describe a sprint. What does it correspond to in the Extreme Programming process model?

A sprint is the unit of development in Scrum. The sprint is restricted to a specific time duration that is fixed in advance. For Agile project development the duration of a sprint is usually between one and four weeks, but two weeks is the most common.

The sprints each begin with a planning meeting where the spring backlog is defined and the tasks for the sprint are identified and estimated. The sprints end with a review meeting where progress is reviewed and demonstrated to stakeholders.

Scrum emphasizes getting a working product by the end of a sprint that has been integrated, tested, and documented, and maybe even ready to be shipped out.

Sprints correspond to Extreme Programming because the time duration of a sprint is supposed to be optimized to be as short as it can be without harming performance. Extreme Programming tries to make time increments as short as possible while still remaining productive.

c. Define and describe Scrum master.

The Scrum Master facilitates the Scrum. This person is responsible for removing impediments that are in the way of the team being able to deliver the product or meet goals. The Scrum Master is the person between the team and distracting influences. The Scrum Master ensures that:

- The Scrum process is used as intended.

- The rules of Scrum are being followed.

- The team is being challenged to continuously improve.

- Key meetings are held on time and professionally.

The Scrum master is not a team lead or project manager because he or she does not have additional people responsibilities.

d. Define and describe Product Owner.

The Product Owner represents the stakeholders and can be seen as the voice of the customer. This person is responsible for ensuring that the team delivers value to the business. The Product Owner assists the team in writing and ranking user stories and then adding them to the product backlog. Scrum teams have one Product Owner who can be part of the development, but cannot also have the role of the Scrum Master.

The Product Owner must have very good communication skills and be able to keep the project on track. The Product Owner must also be able to empathize with the development team and understand the limitations of the team.

Some responsibilities of the Product Owner may include:

- Demonstrating solutions to stakeholders

- Announcing releases

- Communicating status of the team

- Organize milestone reviews

- Keep open communication with stakeholders

- Negotiate priorities, scope, funding, and schedule

- Make sure the Product Backlog is clear and accessible

e. Define and describe a Two-Pizza team, and discuss the pros and cons of its suggested size.

A Two-Pizza team is usually around 4-6 people and absolutely no more than 10. The team is kept in the single digits because the more people on a team, the more communication links there are between people on the team. This can become problematic for larger teams. The number of links is calculated by (n(n-1))/2 where n is the number of people. A team with 6 people has only 15 links whereas a team with 12 people has 66 links. The reasoning behind the Two-Pizza team is to improve individual and team productivity and minimize the cost of coordinating, communicating, and relating with teammates.

Pros

- Eliminates overconfidence from large team size.

- Prevents team members from feeling lost or disconnected from large team.

- Close-knit feeling of small group.

- Eliminates need to talk to many group members to find answers.

- Shorter more concise meetings.

Cons

- Less people inherently means there is a chance for less variety of skillsets between team members.

- The possibility of not enough members to complete certain tasks by specific deadlines.

Owner: Joanna Wallace

39. Lessons Learned (Everyone must answer!): Most organizations carry out a postmortem (i.e., after delivery and acceptance) analysis of a project. In it, they assess all aspects of the project including products, process and resources with the intent of identifying both what well in the project and what areas should be improved upon in future projects. Before a formal postmortem, each team member does a self-assessment, called a Lessons Learned document. Write your own sections of our Lessons Learned document for the class by summarizing:

a. What were the most important (to you, personally) concepts, principles, skills, tools, etc. that you learned/mastered throughout the semester.

I really enjoyed learning all the new tools and programming languages. I had never used ruby or haml before so I found those tutorials and homeworks very interesting. It was fun to work on an agile project and see how real Scrum meetings really work.

b. Note any concepts, principles, skills, tools, etc. that you would have liked to learn, but that we didn't cover.

I can’t really think of anything specific. I didn’t know much about software engineering before this course, so I was happy to get an introduction to rails and other tools!

c. List all everything that you feel you did especially well this semester (master specific tools, languages, skills, etc.; contribute to the project in a specific way, etc., learn and implement specific things quickly, etc.).

I think after being immersed in Agile and seeing how an entire project works I have a great understanding of Agile development. I also have an improved understanding of Github!

d. List anything that that you feel that you didn't do as well, or that didn't go as well as you hoped (getting a list of these items will help me to better schedule material for future semesters).

I think it is hard to get people into agile projects on agile ventures when we aren’t committed to staying on the project after this class ends. Its hard to only work on a project for a few weeks. I also think it was stressful at the beginning of the class to join all the tools (such as pivotal, slack, agile ventures, and piazza) and I sometimes thought I was missing something because I forgot to check all the websites. If we could go over the logistics of each website a little more at the beginning of the course that would be helpful.

e. THANK YOU!

Thank you!