ЗАДАЧИ И ПРОБЛЕМИ НА ОБУЧЕНИЕТО ПО WEB-ПРОГРАМИРАНЕ

Николай Чолаков

Използването на Web-приложения както за промишлени цели, така и за лични нужди, непрекъснато нараства. Една от причините за това е, че все повече от домакинствата в развитите страни разполагат с достъп до Интернет. Във Великобритания например към края на 2010 г. достъп до световната мрежа имат над 73% от домакинствата [10].

Друга, по-съществена основна причина, е непрекъснатото развитие и подобряване на възможностите, които Web-приложенията предлагат. Доскоро немислими за онлайн използване услуги като обработка на изображения или обработка на аудио информация в реално време вече са факт. Благодарение на технологии като AJAX Web-приложенията изземват все по-голям дял от функциите, смятани за запазена територия за десктоп-приложенията.

Ето защо подготовката на качествени кадри в областта на Web-технологиите и разработването на Web-приложения придобива все по-голямо значение.

В Интернет може да се намери голямо разнообразие от сайтове, предлагащи свободна информация за основните въпроси, свързани с Web-програмиране и използване на различните технологии и стандарти: [8], [9], [12]. Това многообразие създава измамно впечатление за лекотата, с която начинаещите Web-програмисти могат самостоятелно да разучат стандартите и платформите и да станат добри професионалисти.

Немалка част от студентите, записали се за обучение в информатичните специалности, идват със свой собствен опит в Web-програмирането, придобит най-вече по собствена инициатива. Това, от една страна, улеснява преподавателя, тъй като студентите възприемат по-лесно материята и може да се намали времето за предаване на фактология за сметка на повече реализирани примери и направени обсъждания. От друга страна обаче, някои от самостоятелно придобитите знания са непълни и дори погрешни, вследствие най-вече на лошите практики, инспирирани от PHP [4].

Необходимо е тези знания да бъдат систематизирани и разширени, а вниманието на студентите да бъде насочено към усвояването на принципните въпроси, свързани с разработването на Web-приложения.

1. Познаване на Web-стандартите.

Комплексното овладяване на основните Web-стандарти е задължителна предпоставка за изграждането на един специалист по Web-програмиране. Практиката да се използват интегрирани среди за конструирането на Web-страниците, както и на API библиотеки и фреймуъркове за реализация на логиката на приложенията, поставя познаването на стандартите на заден план. Студентите поставят въпроси от сорта на "Защо трябва да изучаваме стандарта HTML, когато работейки със среди като Dreamweaver, може да ползваме готови шаблони без да пишем директен код?".

Известни са и случаи, при които преподаватели стигат до другата крайност, изисквайки от студентите да конструират HTML документи само с помощта на обикновен текстов редактор като Notepad.

Разбира се, и двете тези са неприемливи. Използването на интегрирани среди не е панацея. Те предлагат отлични възможности за интензифициране на труда на програмиста, но кодът, който генерират, далеч не е оптимален. Това лесно може да се илюстрира на студентите, като им бъде възложено да създадат една проста Web-страница по три различни начина: с обикновен текстов редактор, със специализирана среда за Web-дизайн, и накрая – посредством конвертиране на Microsoft Word документ в HTML формат. Сравняването на HTML кода, генериран по тези три начина, дава много точна представа за нещата. Разликата в размера на кода може да е неколкократна, защото интегрираните среди добавят голямо количество код, който не е пряко свързан с реализацията на желаните функционални възможности, и в крайна сметка е напълно излишен. За сметка на това, в генерирания от средата код липсват задължителни елементи от структурата на HTML документа или някои от Web-стандартите са погрешно използвани.

Необходимо е да се намери разумният компромис между двата подхода. Естествено е да се използват интегрирани среди за разработването на HTML документи. След конструирането им обаче е добре кодът, генериран от средата, трябва да бъде прегледан от програмиста, за да бъде оптимизирано съдържанието му, излишните конструкции – премахнати, а евентуално липсващите или неправилно използвани елементи – допълнени и коригирани. Това е невъзможно при отсъствието на солидно владеене на Web-стандартите от страна на разработчиците.

Овладяването на Web-стандартите предполага и тяхното правилно използване – в съответствие с предназначението на всеки конкретен стан-

дарт [5], [6]. Основна грешка, която се допуска в това отношение, е използването на HTML за стилово оформяне на Web-страниците. Най-често срещани примери за това са използването на таблици за оформяне на разположението на елементите в страницата, както и HTML тагове и атрибути за цвят, шрифт и т.н. Преподавателят е длъжен да разясни причините, поради които тези практики са погрешни, както и да посочи конкретни примери за правилно и неправилно използване на Web-стандартите.

Като илюстрация на изложените принципи е добре да се използват валидатори за различните Web-стандарти, и те да се приложат не само върху примерите, дадени от преподавателя, но и за някои сайтове в мрежата.

Онлайн валидатор на HTML код например може да се ползва на сайта на организацията Word Wide Web Consortium (W3C) [7], която се занимава с разработването и поддръжката на Web-стандартите. Интересно и показателно за студентите е да се направят експерименти с този или друг подобен валидатор и да се анализират и коментират резултатите.

В Таблица 1 са представени резултатите от HTML валидацията на някои от най-посещаваните Web-сайтове по света и у нас, подредени според своята популярност към края на 2010 г.

Таблица 1. Резултати от валидиране на HTML за популярни сайтове

Web-Сайт	Грешки	Предупреждения
http://www.facebook.com	19	0
http://www.google.com/	36	2
http://www.youtube.com	71	2
http://en.wikipedia.org	2	0
http://www.mozilla.com	0	0
http://vbox7.com	44	60
http://abv.bg	114	287
http://www.dir.bg	286	302
http://www.data.bg	68	13
http://www.jobs.bg	1637	42

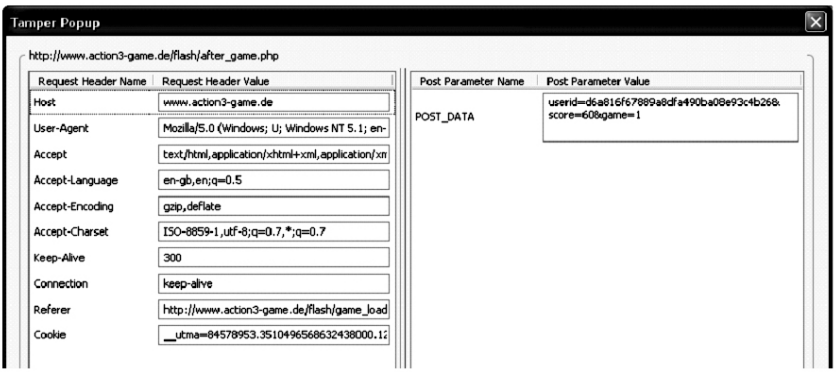
Анализът на резултатите показва, че единствено сайтът на Mozilla преминава валидацията без никакви проблеми, което говори само по себе си за професионализма на неговия екип. Почти без грешки е и

валидацията на Wikipedia, докато при развлекателните сайтове, които безспорно се ползват с най-голяма популярност, отношението на разработчиците е доста по-небрежно. А какво да се каже за най-популярните български сайтове – техните резултати са красноречиви сами по себе си. Причината за това не е в липсата на квалифицирани специалисти у нас, а по-скоро в непрофесионалния подход към изграждането на тези сайтове.

Освен на различните стандарти за конструиране и оформление на Web-страниците, в курсовете по Web-програмиране трябва да се обърне особено внимание на протоколите, по които се осъществява комуникацията в Web-средата, и най-вече на HTTP (HyperText transfer Protocol).

Протоколът HTTP дефинира правилата за комуникация между клиента и сървъра в Web-пространството. В този смисъл той има определяща роля за функционирането на Web-приложенията изобщо, и доброто познаване на неговата структура и форматите на заявките и отговорите е задължително.

За разлика от Web-стандартите като HTML, където структурните елементи и тяхното действие могат да се проследят визуално, HTTP форматите остават скрити. Затова при разглеждането на стандартите за HTTP заявка и отговор е добре да се използват сердства с помощта на които тези елементи могат да се визуализират и да се правят експерименти с тях. Подходящи за целта са дебъгерите на заявки, един от които е TamperData. Този инструмент се инсталира като плъгин на Mozilla и позволява трасиране на HTTP заявките, с визуално представяне на всички хедъри и данни.



Фиг. 1. Дебъгер на заявки TamperData

За съжаление, освен че са много полезни при разработването, настройката и поддръжката на Web-приложения, софтуерните системи от този тип са и основен инструмент в арсенала на хакерите и значително улесняват пробивите в сигурността на недобре защитените приложения [3].

По подобен начин стоят нещата и при другите протоколи, използвани при работата на Web-приложенията. Почти няма реални системи, които да не предлагат функционални възможности за изпращане на електронна поща например. Срещат се и такива, които предлагат автоматизирано управление на входящата електронна поща. Изграждането на ефективни приложения, осигуряващи такива услуги, изисква детайлното познаване на протоколите за изпращане на електронна поща и за комуникация със мейл сървъри и управление на пощенски кутии (IMAP, POP3). Тук отново постигането на добри резултати в обучението изисква онагледяване на физическите формати, поддържани от тези протоколи, не просто като хипотетичен код, а конкретно за примерите, с които се работи в дадения курс.

2. Съвместимост на приложенията с Интернет браузърите

Потребителите на Web-средата използват сравнително широк спектър от браузъри. За съжаление Web-стандартите имат препоръчителен, а не задължителен характер, и начинът, по който всеки от браузърите ги интерпретира, е различен. Задачата да се създават приложения, страниците на които да изглеждат и функционират по един и същи начин във всички браузъри често пъти няма решение. Въпреки това разработчикът е длъжен да вземе всички мерки, за да осигури максимална съвместимост и универсалност на приложението си.

На първо място това е стриктното придържане към изискванията за оформление и използване на Web-стандартите. Всяко отклонение от стандарта, колкото и незначително да е то, създава предпоставки за различно поведение на приложението в някои от браузърите. Тъй като поддръжката на основните стандарти е най-непълна в Internet Explorer, не е чудно, че най-често именно при него се наблюдават различия в изгледа и функционирането на Web-приложенията. Всъщност в повечето случаи разделението между браузърите е на принципа "Internet Explorer срещу всички останали". Разбира се, има изключения от това правило, но те най-често се дължат на бъгове във версиите на различните браузъри.

Наложително е разработчиците на Web-приложения да познават особеностите на съвременните браузъри и да следят динамиката на тяхната

популярност в мрежата. Много трудно е за едно реално приложение, особено ако предлага по-сложни функции с динамични ефекти и т.н., да се осигури еднаква поддържка във всички съществуващи в мрежата браузъри. Поне за основните, най-популярните от тях обаче това е задължително.

В Таблица 2 може да се види статистика за най-популярните браузъри в Интернет за последните десет години.

Таблица 2. Популярност на интернет браузърите по години [11]

2011	**Internet Explorer**	**Firefox**	**Chrome**	**Safari**	**Opera**
Март	25.80%	42.20%	25.00%	4.00%	2.50%
2010	**Internet Explorer**	**Firefox**	**Chrome**	**Safari**	**Opera**
Декември	27.50%	43.50%	22.40%	3.80%	2.20%
2009	**Internet Explorer**	**Firefox**	**Chrome**	**Safari**	**Opera**
Декември	37.20%	46.40%	9.80%	3.60%	2.30%
2008	**Internet Explorer**	**Firefox**	**Chrome**	**Safari**	**Opera**
Декември	46.00%	44.40%	3.60%	2.70%	2.40%
2007	**Internet Explorer**	**Firefox**	**Mozilla**	**Safari**	**Opera**
Ноември	56.00%	36.30%	1.20%	1.80%	1.60%
2006	**Internet Explorer**	**Firefox**	**Mozilla**	**Netscape**	**Opera**
Ноември	60.60%	29.90%	2.50%	0.20%	1.50%
2005	**Internet Explorer**	**Firefox**	**Mozilla**	**Netscape**	**Opera**
Ноември	68.90%	23.60%	2.80%	0.40%	1.50%
2004	**Internet Explorer**		**Mozilla**	**Netscape**	**Opera**
Ноември	76.20%		16.50%	1.70%	1.60%
2003	**Internet Explorer**		**Mozilla**	**Netscape**	**Opera**
Ноември	84.90%		7.20%	2.60%	1.90%
2002	**Internet Explorer**	**AOL**		**Netscape**	
Ноември	83.40%	5.20%		8.00%	

В таблицата не се включват браузъри, чиято популярност е под 0,5%. Анализът на изнесените данни позволява ясно да се очертаят основните тенденции. Браузърът Internet Explorer на Microsoft, който в миналото притежаваше монопол върху интернет услугите, е с постоянно намаляваща популярност. Причините за това са съвсем обективни – недостатъчно добра поддръжка на Web-стандартите и пробиви в сигур-

ността. За сметка на това различните версии на браузъра Mozilla, който е с отворен код, добиват все по-голяма популярност, за да се стигне до ситуацията в наши дни, когато те доминират на пазара. В същото време на пазара излизат нови продукти, като Google Chrome, и тяхната популярност бързо нараства.

Детекцията на браузъра може да се осъществи както от страна на клиента, така и от страна на сървъра. Добре е преподавателят да даде примери и за двата подхода, като за да има ефект от тези примери, те трябва да се тестват под различни браузъри.

Всъщност при тестването и валидирането на вече разработеното Web-приложение, а и още в процеса на разработка, всяка страница задължително се тества с различни браузъри, най-малко с първите пет в списъка по популярност. При това е добре да се използват различни версии за всеки браузър. За тази цел могат да се използват както оригиналните браузъри, така и някои от популярните средства, симулиращи действието на различните версии. В някои случаи последното е наложително, защото Internet Explorer например не позволява едновременното съществуване на свои различни версии в една инсталация на Windows.

Изводът, който може да се направи тук е, че спектърът използвани браузъри от интернет потребителите се променя относително динамично. Необходимо е тенденциите в това развитие да се следят непрекъснато и разработваните Web-приложения да осигуряват поддръжка на всички популярни браузъри.

3. Разделяне на логиката от представянето

Принципът за разделяне на логиката на приложението от представянето на съдържанието е фундаментален елемент от обучението по програмиране изобщо. Всъщност той не е заложен като отделна тема в курсовете по Web-програмиране, но поради изключителната му важност трябва да му бъде отделено специално внимание.

Студентите, които вече имат известен опит в разработването на по-сложни проекти, не се нуждаят от убеждаване в необходимостта от разделяне на логиката от представянето. Не така стоят нещата обаче при тези, които все още не са напреднали достатъчно. Този проблем се проявява особено силно, ако използваната платформа е PHP, понеже нейните основни средства по никакъв начин не стимулират спазването на принципа. Много е лесно в един PHP скрипт да се струпа цялото решение на дадена задача, което отново поражда въпроси от типа "А защо ни е необ-

ходимо това ?", въпреки че на тях е било отговаряно нееднократно в курсовете, предшестващи този по Web-програмиране.

Тук преподавателят е улеснен от факта, че при разработването на Web-приложения смисълът и ефектът от спазването на това разделение могат много добре да се подчертаят и да се илюстрират съвсем явно. Причината за това е в използването на различни езици и стандарти за оформяне на логиката и интерфейса на Web-приложенията.

Разделението между статичния и динамичния код следва да бъде демонстрирано от преподавателя с примери, като една и съща задача бъде структурирана по двата начина – с разделение и без, и бъдат коментирани резултатите. Много подходящ инструментариум за построяването на такива примери са JSP страниците и Beans класовете в JAVA.

На следващото ниво вече е необходимо да се работи по класическата схема Model – View – Controller (MVC), като се наблегне върху основните функции на компонентите. Моделът (Model), който дефинира средствата за обработка и съхранение на данните, изгледът (View), който генерира представянето на данните пред потребителя, и контролерът (Controller), който управлява взаимодействието между модела и изгледа. Подходящи средства за проста, но ефективна илюстрация на тази програмна архитектура предоставя JAVA платформата – JavaBean клас за реализация на модела, JSP страница за изгледа и сървлет клас в ролята на контролер.

Разбира се, най-добрата илюстрация за мощността, ефективността и гъвкавостта на MVC архитектурата са разработените MVC фреймуъркове, като Struts за езика JAVA или Zend Framework за PHP. В курсовете по Web-програмиране е задължително да бъдат включени поне основни примери за използването на такива програмни среди. За съжаление ограниченият хорариум не позволява пълноценното разглеждане на определен фреймуърк в рамките на един такъв курс. Възможни решения на този проблем са включването на специализирани дисциплини като факултативни курсове, или изнасянето на тези дисциплини в магистърските програми.

4. Оптимизиране на кода и на управлението на ресурси.

Фактът, че Web-приложенията работят в мрежова среда, и имат на практика неограничено множество потенциални клиенти, налага много икономичното използване на сървърните ресурси – най-вече процесорно време и оперативна памет. Важно е студентите да осъзнаят, че допуснатите

пропуски в това отношение могат да бъдат фатални за целостта на приложението.

Често наглед незначителни фактори могат да имат критично значение. Много показателен в това отношение е следващият пример на езика JAVA. При обработката на стрингове постоянно се налага слепването на няколко стринга в един, и в JAVA това може да стане с предефинираната операция "+"

```
String mess = "Налични в" + store + "склад" +
                result.getRows() + "стоки";
```

Този начин за конкатенация на стрингове обаче е изцяло погрешен. Проблемът идва оттам, че класът String в JAVA не позволява обектите да бъдат променяни. Ако се налага един String обект да бъде променен, то той се унищожава и се създава наново, инициализиран с вече променената стойност. По тази причина, за да изпълни този оператор, виртуалната машина ще трябва да унищожи и създаде наново четири обекта – по един за всяка операция по промяна на стринга. В едно десктоп приложение това едва ли би имало сериозно значение, но в Web-приложенията, където този фрагмент от кода може да бъде изпълняван едновременно от десетки и дори стотици клиенти, такова разхищение на ресурси е недопустимо. Правилното решение в случая е да се използва класът StringBuffer, който позволява свободно манипулиране на стрингове, а след окончателното му оформяне стрингът да бъде конвертиран обратно до обект от клас String:

```
StringBuffer sb = new StringBuffer ("Налични в");
sb.append (store);
sb.append ("склад");
sb.append (result.getRows());
sb.append ("стоки");
String mess = sb.toString();
```

Друг важен момент е навременното освобождаване на заетите от приложението системни ресурси – заделена памет, отворени файлове, отворени конекции към бази от данни, заети портове и т.н. Лошата практика тук е да се разчита на факта, че сървърът ще освободи автоматично заетите ресурси след завършването на конкретния скрипт. Дори и това

да е така, разработчикът не бива да разчита друг да свърши неговата работа. Възможно е по една или друга причина скриптът да не завърши коректно и това да наруши нормалния ход на работа на приложението като цяло. Възможно е също така при по-нататъшното развитие на логиката на приложението конструкцията, която заделя определени системни ресурси, да се окаже в рамките на многократно повтарящ се фрагмент от кода. Тогава, ако тази конструкция отваря конекция към база от данни или заделя ново количество памет, е съвсем реална опасността съответният системен ресурс – конекции или памет – да се окаже изчерпан далеч преди края на скрипта, а оттам да се стигне и до неговото прекъсване.

Един добър пример за това как е правилно да се конструира логиката на управление на заделени ресурси може да се даде с помощта на конструкцията try-catch-finally в езика JAVA [1]:

```
Socket quoteSocket = null;
try {
   quoteSocket = new Socket(serverName,S_PORT);
    . . .
}
catch(IOException ex) {
   System.out.println(" \nFAILED I/O" + ex);
}
finally {
    try {
       if( mySocket != null ) mySocket.close();
    }
   catch(IOException ex) {
       System.out.println("\nFAILED I/O" + ex);
    }
}
```

Обектът от клас Socket се създава в try блока, при което отвореният сокет заема порта с номер S_PORT. При това положение, ако сокетът не бъде затворен нормално, портът ще остане зает, и няма да е достъпен за по-нататъшно използване в програмата. За да се предотврати такава ситуация сокетът се затваря във finally секцията, която се изпълнява винаги – независимо дали е възникнало изключение, или не.

5. Осигуряване на преносимост на приложенията

Платформено независимите софтуерни решения като JAVA и PHP за разработване на Web-приложения имат предимството да осигуряват преносимост на приложенията между различни архитектури и операционни системи. Съвсем нормално е програмистът да разработва своето JAVA приложение под Windows например, след което да го публикува и хоства на Linux сървър.

Използването на платформено независим инструментариум само по себе си не може да гарантира платформена независимост на приложенията. Разработчикът трябва да се съобразява и с редица допълнителни фактори, за да постигне реална преносимост на своето приложение.

На първо място, приложението не трябва да разчита на странични средства, които от своя страна не са платформено независими. Стандартът на езика JAVA например позволява използването на методи, писани на друг език (Native methods), най-често на ASSEMBLER. Кодът на тези методи обаче е ориентиран към конкретна платформа и те не могат да бъдат универсално използвани. Друг уместен пример в тази сфера е използването на CRON услугата на Linux за реализация на задачи в реално време посредством езика PHP. Приложение, разработено по този начин, не може да бъде хоствано под Windows, което нарушава неговата преносимост.

Всяко реално Web-приложение по време на своето изпълнение разчита на определени статични данни, които могат да се използват на много места и по различен начин. Това са акаунти за достъп да бази от данни, линкове към Web-сървиси, имейл-адреси, различни константи, определящи граници за даден параметър – например максимален размер на файл за качване и т.н. Някои от тези параметри се променят в процеса на експлоатация на приложението, други трябва да се коригират, ако приложението бъде пренесено на друг сървър, или свързано към друга база от данни. Затова те трябва да бъдат обособени в отделен конфигурационен файл, който да може да се редактира свободно, без това да изисква рекомпилиране или друга намеса в структурата на приложението.

Стандартите за структуриране, именуване и разположение на конфигурационните файлове са различни в различните платформи за Web-програмиране. Преподавателят трябва да подготви адекватни за използваната в курса платформа примери и да изисква използването на такива файлове дори и за простите задачи, които се решават на семинарните занятия.

Друг, много важен фактор за преносимостта на приложенията, са използваните методи за адресация на външни ресурси, независимо дали става въпрос за скриптове, стилови файлове, графични изображения или друг тип ресурси. Често срещана грешка в това отношение е използването на абсолютни адреси, което прави приложението работоспособно само в рамките на една конкретна конфигурация. Тук може да се даде пример с конструирането на HTML документ посредством интегрирана среда. Ако със средствата на средата не се работи правилно, например при добавяне на графични изображения в генерирания HTML код може да се видят пътища, обвързани с локалната файлова система и директорията, от която е зареден файлът с изображението:

```
C:\Documents and Settings\All Users\Desktop\logo.jpg
```

Естествено използването на такъв път би направило включването на това изображение в страницата невъзможно.

Правилният подход към адресиране на външни ресурси е да се използва относителна адресация спрямо текущата директория. По такъв начин се гарантира независимост на приложението от конкретната директория, в която е разположено, и то запазва работоспособността си при промяна на тази директория. Това е съвсем нормална ситуация при реорганизация на сървъра, или при промяна на домейна на самото приложение.

Ето един подходящ пример: включване на външен в HTML документ, при положение, че CSS файлът с име "main.css" се намира в поддиректория с името "styles", която е на едно ниво с текушата директория:

```
<link rel="stylesheet" href="../styles/main.css">
```

В някои случаи все пак се налага използването на абсолютна адресация. Пример за това може да се даде при разработването на скрипт, който ще бъде викан от команден промпт. При такова извикване не се знае от коя директория ще стане то, поради което, ако скриптът ползва външни ресурси, трябва да бъде осигурен абсолютен път до тях. Ако този път бъде директно описан, преносимостта отново се изгубва:

```
define("ROOT", "/var/www/workplace.com/admin");
require_once(ROOT.'/inc/config.inc.php');
```

Правилният начин да се изгради тази конструкция е с използването на средства за детектиране на текущата директория и пътя до нея:

```
define('ROOT',preg_replace("`pages$`","",dirname(__FILE__)));
require_once(ROOT.'/inc/config.inc.php');
```

Спазването на посочените по-горе мерки трябва да бъде насърчавано, тъй като така се осигурява необходимата гъвкавост на Web-приложенията и значително се опростява тяхната поддръжка и експлоатация.

6. Сигурност и защита на Web-приложенията

Уязвимостта на Web-приложенията от потенциални атаки във враждебната Интернет-среда е най-сериозният проблем при тяхната експлоатация. Лайтмотив в курсовете по Web-програмиране следва да бъде създаването у студентите на навик постоянно да мислят за уязвимите места в приложенията и за мерките, които трябва да се вземат за осигуряване на адекватна защита.

Опасностите, на които Web-приложенията са постоянно изложени, са много сериозни. По-голямата част от тях се проявяват на местата, където има предаване и обработка на данни, въвеждани от потребителите. Това са слабите места, през които хакерите най-често провеждат своите атаки. Последствията от тях могат да са най-различни, и често пъти фатални – източване на финансови средства, уволнение от работа, значителни загуби и дори фалити на компании, въвличане в сериозни икономически, политически и международни скандали.

Основен принцип в стратегията за обработване на данни, въведени от потребителя е, че на потребителя не трябва да се вярва, а да се приема, че той има потенциално враждебни намерения. Този принцип важи не само за действията на външни, т.е. нерегистрирани потребители, но също така и за всеки момент от използването на системата от регистрираните потребители. На всяко място, където потребителят би могъл да се намеси, трябва да се предвидят възможните (волни или неволни) потенциално опасни действия, и да се вземат необходимите предпазни мерки.

Това изисква Web-разработчиците много добре да познават разпространените методи за атакуване на приложенията, както и адекватните мерки за предотвратяване на тези атаки.

В курсовете по Web-програмиране на тези проблеми е необходимо да се посвети отделна тема. Първата основна задача в нея е да се изяснят същината и методите на действие на основните форми за атака срещу Web-

приложенията: враждебни клиентски скриптове (cross-site scripting, XSS), инжектиране на SQL код (SQL injection), фалшифициране на заявки (Phishing), автоматизирани програми за атакуване на форми (ботове) и т.н.

Най-добре е това да стане примери, подобни на следния пример за атака от тип „Инжектиране на SQL". Нека имаме заявката:

```
"SELECT * FROM users WHERE user='".$user.
                    "' AND pass='".$pass."'";
```

Ако променливата $user получава стойността си директно от потребителя и той е въвел

```
' OR user != '' OR user = ''
```

то на практика ще се формира SQL заявката

```
SELECT * FROM users WHERE user='' OR user != ''
                     OR user = '' AND pass='xxx'
```

която ще даде положителен резултат, независимо какво е въведено като парола и ще осигури не оторизиран достъп на потребителя.

Необходимо е също така да се подчертаят заплахите, които са следствие на различните форми на атака, и включват:

- Кражба (изтичане) на информация и лични данни;
- Извършване на неоторизирани действия от името на жертвата;
- Злоупотреба с функциите на атакуваното приложение.
- Получаване на административен достъп до данните и дори цялостно компрометиране на базата данни;

Втората основна задача, свързана с темата за сигурността на Web-приложенията, е да се разгледат мерките за защита и предотвратяване на атаките, отново с конкретни примери за всяка мярка:

- Филтриране на входния поток с ескейпиране на апострофи и кавички, за предотвратяване на активното им действие. При това е най-добре да се използват средствата, които предоставя сървърът за бази от данни [2] или самият фреймуърк, с който се работи. Ескейпирането на входния поток в PHP например, ако не се използва специален фреймуърк, е добре да става с функцията mysql_real_escape_string(), която прави обръщение към MySQL сървъра за филтриране на подадения стринг.

• Ограничаване и контрол на дължината на полетата във формите до необходимия минимум, за да не може да се въвежда лесно допълнителен код.

• Валидиране на въведеното от потребителя – например, ако се очаква числова величина, да се провери дали е въведено число, или дори въведеното да се преобразува принудително към числова величина и т.н. Тук е по-добре да се използва принципът на "белия списък" (whitelist), отколкото този на "черния списък" (blacklist), т.е. проверката да допуска само приемливото съдържание, а всичко останало да се игнорира, вместо да се правят опити да се блокира нежеланото съдържание.

• Използване на параметризирани заявки. При тях заявката е предварително подготвена, оставени са единствено места за стойностите на параметрите, идващи от потребителя. Тук филтрирането на въведените данни се прави автоматично. Ето един пример, написан на JAVA, с използването на специализирания клас PreparedStatement:

```
PreparedStatement pstm = conn.prepareStatement(
      "SELECT * FROM users WHERE username=? AND
password=?");
pstm.setString(1, username);
pstm.setString(2, password);
```

• Ограничаване на правата на DB потребителите, с които работят Web-приложенията. В никакъв случай не бива да се допуска Web-приложенията да използват за достъп да базите данни потребители с административни права. Трябва да се създаде и използва специален потребител, със силно ограничени правомощия, позволяващи му да изпълнява само необходимите за приложението действия с базата данни.

• Проверка за адреса на извикване на заявката (Referrer header). По такъв начин се елиминират заявки, идващи от адрес, различен от оригиналния адрес на потребителя, започнал сесията.

• Искане на допълнително потвърждение за най-критичните операции – с повторно въвеждане на парола, или с помощта на captcha (код, стойността на който е записана в графично изображение, откъдето потребителят го вижда, за да го въведе във формата).

• Ограничаване на периода на валидност на бисквитките. По такъв начин ако потребителят е напуснал сайта без да изпълни logout, сесията ще изтече по-бързо и шансовете за атака стават значително по-малки;

• Криптиране на данните при предаване през заявки, особено когато се касае за пароли, идентификационни номера, номера на банкови сметки и

друга поверителна информация. Криптирането трабва да се извършва с помощта на необратим алгоритъм като SHA1 или MD5, а ако трябва предаваният параметър да се сравни с друга стойност – например при проверка на въведена парола, то същият алгоритъм се прилага и върху другата величина и се сравняват криптираните стойности. Всъщност най-добре е поверителните данни да се съхраняват в базата в криптиран вид, за да не станат директно достъпни при пробив в сигурността на самата база от данни.

Важен момент при разглеждането на мерките за защита от потенциални атаки е да се подчертае, и да се демонстрира с подходящи примери, че тези мерки трябва да се прилагат на две нива: на клиента и на сървъра. Ако проверките се правят само на сървъра, това е неудобно за потребителя. Например, ако сървърът върне няколко пъти формата за регистрация поради неволни грешки, които биха могли да се открият още на клиента, т.е. преди изпращането на формата, коректният потребител ще се почувства ощетен и ненужно ще изгуби време. От друга страна, не бива да се разчита само на проверки от страна на клиента, защото те могат да бъдат заобиколени. В крайна сметка на студентите трябва да стане ясно, че мерки за защита се вземат на всички възможни нива, но определящи са тези на сървъра.

Действията, които се предприемат за защита на Web-приложенията от потенциални опасности, трябва да бъдат адекватни на съответните заплахи, и същевременно да бъдат комплексни. Частичните, половинчати мерки не осигуряват достатъчна защита. И може би най-важният фактор за успех в борбата със заплахите са сериозното отношение и непрекъснатото внимание към този проблем.

7. Обработка на грешки, регистриране на събития и архивиране

Грешките, възникващи при експлоатацията на Web-приложенията, са неизбежни. Колкото и добре да е написано, тествано и валидирано приложението, в процеса на неговата работа се намесват редица странични фактори: недостъпни в даден момент ресурси (мрежа, портове, файлове и директории), некоректни потребители, неволни операторски грешки и т.н. По тази причина е важно да има ясна стратегия за реакция при възникване на грешка. Недопустимо е да се стигне до ситуация, в която приложението да изгуби контрол, защото последиците може да бъдат много сериозни.

Например, ако злонамерен потребител успее да предизвика грешка в някоя от SQL заявките на приложението, и тази грешка бъде изведена в потребителския интерфейс, то информацията, съдържаща се в нея (имена на бази данни, таблици, полета, конструкции на заявки) може да бъде от полза при построяването на атаките. Затова не е допустимо потребителят да

вижда каквито и да е системни съобщения за грешка. Обработката на прихванатите от приложението грешки трябва да е еднотипна. Оптималният вариант за това е да има специално създадена страница за представяне на възникнала грешка, и при възникнала такава потребителят да се препраща там. Още по-добре е, ако има отделни страници за всеки тип грешка – преместен ресурс (302), забранен достъп (403), вътрешна грешка на сървъра (500) и т.н.

Различните платформи за Web-програмиране предлагат различни средства за прихващане на грешки. Всички съвременни програмни системи поддържат, в една или друга степен, механизма на изключенията. От друга страна, има средства специфични за конкретната платформа – например функцията set_error_handler( ) в PHP, или директивата error_page в JSP:

```
<%@ page errorPage="/pages/error.jsp" %>
```

При възникване на грешка в JSP страницата, ако тя не се прихваща от логиката в самата страница, потребителят се препраща на указания адрес.

Регистрирането на събития в процеса на работа на програмата (logging) също спомага за справяне с проблемни ситуации. Ако системата регистрира влизането на всеки потребител с неговия адрес, дата и час, страниците, които е посещавал и действията, които е предприемал, при възникване на пробив в сигурноста или загуба на данни, логът може да се проследи и да се локализира причината, както и да се посочи виновникът.

За изграждане на такава система има различни варианти на действие – да се разчита на системни логове или приложението да ползва собствен лог. От своя страна логът може да се прави във файл или в база от данни. Тук обаче трябва да се подчертае фактът, че използването на логове не е безплатна операция – колкото повече събития се отразяват в лога, толкова по-голям е разходът на сървърни ресурси. Затова е добре лог функцията да може да се активира и деактивира с параметри от конфигурационния файл например. Още по-добри резултати може да се постигнат, ако има няколко нива на логинг – само основни събития или по-широк кръг, запис на повече или по-малко детайли.

Освен на разработването на приложенията, в курсовете по Web-програмиране трябва да се обърне внимание и на тяхната експлоатация и поддръжка. Задължение на администратора на приложението е периодично да актуализира архив на базата от данни, както и на самата директория, където е инсталацията на програмата, тъй като освен сорс файловете, много често там се натрупват и файлове с данни – качени снимки, експортирани справки и др.

Дейността на администратора може да бъде сериозно улеснена, ако самото приложение има модул за архивиране, който автоматично да създава пълен архив на базата от данни и на инсталацията на програмата. Тази процедура може да се изпълнява по заявка от администратора или регулярно – по предварително зададен график.

Заключение

Курсовете по Web-програмиране в информатичните специалности обобщават опита, натрупан от обучението по основни дисциплини като програмиране, бази от данни, мрежи и др., и го пренасят в Web-средата. Освен това те имат задачата да подпомогнат усвояването на основните Web-стандарти, платформите за програмиране от страна на сървъра, както и принципите за проектиране, изграждане, внедряване, защита и експлоатация на Web-приложенията.

Предвид широкия спектър от въпроси, които трябва да бъдат разгледани и усвоени, хорариумът на семинарните занятия по тези курсове е крайно недостатъчен. Постигането на добри резултати от студентите предполага сериозна извън аудиторна ангажираност, като най-подходяща тук е формата на курсовия проект. Разработван в продължение на по-голямата част от семестъра, един такъв реален проект позволява практическото отработване, ако не на всички, то поне на по-голямата част от представените тук проблеми. В този процес преподавателят също има отговорна роля – да насочва усилията на студентите в правилна посока и да изисква спазването на разгледаните принципи.

ЛИТЕРАТУРА

1. *Екел, Б.* Да мислим на JAVA. С., СофтПрес, 2009.

2. *Ръсел, Г., А. Кърминг SQL Хакове*: Съвети и инструменти за изслед-ване на вашите данни. С., Зест Прес, 2008.

3. *Чолаков, Н.* Аспекти на сигурността при обработка на потребителски данни в Web приложения. Сборник доклади "Нови предизвикателства в технологиите за програмиране, компютърните алгоритми и обучението във висшите училища, свързано с тях". В. Търново, ПСИТ-гарант, 2010.

4. *Cholakov, N.* On some drawbacks of the PHP platform. International conference on computer systems and technologies CompSysTech'2008. Gabrovo, 2008.

5. *Duckett, J.* Beginning Web Programming with HTML, XHTML, and CSS, 2nd Edition. Wrox Press, NY, 2008.

6. *Van der Vlist, E. D. Ayers, E. Bruchez, J. Fawcett, A. Vernet.* Professional Web 2.0 Programming. Wrox Press, NY, 2006.

7. http://validator.w3.org/

8. http://www.cyberciti.biz/tips/php-perl-mysql-webprogramming-tutorials.html
9. http://www.irt.org/
10. http://www.sbp-romania.com/Blog/2011/01/11/web-vs-desktop-applications-friends-or-foes.aspx
11. http://www.w3schools.com/browsers/browsers_stats.asp
12. http://www.webstepbook.com/supplements.shtml

ЗАДАЧИ И ПРОБЛЕМИ НА ОБУЧЕНИЕТО ПО WEB-ПРОГРАМИРАНЕ

НИКОЛАЙ ЧОЛАКОВ

Резюме

В работата се разглеждат принципните задачи, стоящи пред обучението по Web-програмиране на студенти в информатичните специалности. Аргументирана е необходимостта от доброто познаване и правилното използване на Web-стандартите, разделението на логиката от представянето, оптимизирането на кода на приложенията и управлението на ресурсите. Специално внимание е отделено на мерките, които е необходимо да се вземат за сигурността и защитата на Web-приложенията в процеса на тяхната експлоатация. Анализът е направен с отчитане на особеностите на съвременната Web-среда. Очертани са основните проблеми, съпътстващи работата на преподавателя. Приведени са някои примери, които той може да използва в своята работа.

TASKS AND PROBLEMS OF THE TRAINING IN WEB-PROGRAMMING

NIKOLAI CHOLAKOV

Summary

The paper discusses the principal problems facing the training in Web-programming of students in IT-majors. Substantiated are the necessity of better understanding and proper use of Web-standards, separation of business and presentation logic, optimization of the code of the applications and resource management. Special attention is devoted to measures that need to be taken for security and protection of the Web-applications in the process of their operation. The analysis of the issues involved is made taking into account the peculiarities of modern Web-environment. The main problems concerning the work of the lecturer are outlined. Some examples that can be used in his work are given.